

Meet my Expectations: On the Interplay of Trustworthiness and Deep Learning Optimization

Dissertation

der Fakultät für Mathematik und Informatik
der Universität des Saarlandes
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)



vorgelegt von
Adrián Javaloy Bornás, M. Sc.

Saarbrücken
2024

Gedruckt mit Genehmigung der Fakultät für Mathematik und Informatik der Universität des Saarlandes.

Tag des Kolloquiums: 11. März 2025

Dekan: Prof. Dr. Roland Speicher

Prüfungsausschuss:

Vorsitz:	Prof. Dr. Martina Maggio
1. Berichterstatter:	Prof. Dr. Isabel Valera
2. Berichterstatter:	Prof. Dr. Jilles Vreeken
3. Berichterstatter:	Prof. Dr. Benjamin Bloem-Reddy
Akademischer Beisitzer:	Dr. Kavya Gupta

Dedicated to Alfonso Bornás Bayonas.

Your family will never forget you.

Disclaimer. This style is a modified version of Felix Dangel's thesis template, which in turn uses Federico Marotta's kaobook template based on Ken Arroyo Ohori's doctoral thesis, and I am grateful to all of them for their openness to share their \LaTeX codes. Otherwise, this thesis would not look nearly the way it does.

Acknowledgments

It has been a week since I defended my thesis in front of an incredible crowd and no, I have not written the acknowledgments until now. My procrastination has been so apparent that it made it into my PhD hat (thanks), but I always struggle with this type of writing: I want it to be as perfect and sincere as humanly possible, since that is the least the people that have supported me these years deserve. So here we go.

First and foremost, I want to express my deepest gratitude to Isabel Valera. When I joined the lab back in 2019 I knew nothing, I was a “monkey with a gun” as I jokingly used to say and, while I still have many things to improve upon, it is unquestionable that I have gone a long way in every aspect since then. Not only Isabel has taught me how to be a great researcher, but she has also taught me how to be a better person, and I am lucky enough to have had her along my side all these years not only as my advisor and collaborator, but also as my friend, and I hope it keeps being this way for a long time. Thank you for *all the patience* you have had with me, and for adopting me as an early member of your lab and a foster member of your family. I am really proud of what we have done and, if I had the opportunity, I would undoubtedly repeat this journey with you.

I am also grateful to those incredible scientists that I somehow fooled into agreeing on being part of my examination committee: Jilles Vreeken, Benjamin Bloem-Reddy, Martina Maggio, and Kavya Gupta. While I could have done better during the presentation Q&A, I can assure you that I enjoyed the questions, that I love and have been following your works for quite some time, and that I fondly hope to collaborate with all of you in the near future (I definitely have ideas to discuss when the occasion comes!).

I also would like to briefly acknowledge the work of the administration staff that has made my paperwork bearable all these years and which usually goes unnoticed. Starting from Sabrina and Mara Cascianelli back in Tübingen, to Mona, Asel and, especially, Natalia Weis in Saarbrücken. Probably they would have blocked my (often silly) emails if it were not part of their duties, but I am really grateful anyways.

The work I present in this dissertation (and the one I unfortunately had to leave out) would have not been possible without all the incredible people I was lucky enough to work with. Starting with my Master advisor, Ginés García-Mateos, who agreed to work in Deep Learning with me despite both of us not knowing what a neural network was, and who got me into computer science (and competitive programming) when I was only fifteen and performed poorly at the local competitions. I also want to thank Luigi Gresele, Giancarlo Fissore, Bernhard Schölkopf, and Aapo Hyvärinen, which gave me the opportunity to help on what it became my first PhD publication, Pablo Sánchez-Martín, with whom I published two beautiful works, Maryam Meghdadi, whose help was invaluable to make impartiality blocks work, and Amit Levi, as I was lucky to pick his brain to work on proofs I would have never been able to develop myself. Finally, I am thankful to the Machine Learning lab for bearing with me for so long: Amir, Ayan, Batuhan, Deborah, Jonas, Kavya, Nektarios, and the newer members who do not have to cope with me. I also enjoyed supervising and interacting with all the students that worked with us: Raj, Bao, Georgi, Sonya, Dilan, and those that I might be missing.

During this journey I met many people, some of which I am happy to call friends to this day, despite not being as much in contact as I definitely would love to. My journey in Tübingen would have not been as memorable without Pablo Moreno, Annalisa, Antonio, Luigi and Sabrina, Diego and Sofia, Julius and Ina, Federico, Ju Young, Diego Fioravanti, Felix, Martina, Daniela, or José Carlos, to name a few. Similarly, Saarbrücken would have been just and only work had I not met Jonas, Luis, Maribel, or Yaiza, who gave me a fresh view outside the laboratory. This paragraph would be incomplete if I did not mention those who left us and who I would never forget, Alfonso (*¡Ay!*, *Abuelo...*) and Otto, as well as those who have come to take over the world after us and who definitely make me hopeful: Carmen, Viola, Tomás, Adrián, Golfa, Nena, and Nano.

Finally, I am eternally grateful to my family who have supported me all these years despite having to accept living thousands of kilometers away from me, especially to my parents, Aurora María and Francisco, my little old kitty, Mica, as well as to my sister, Marina. *Gracias por todo mamá, papá, y de nada, Marina.*

Thank you!

Adrián Javaloy Bornás
Edinburgh, March 18, 2025

Abstract

You cannot optimize what you do not account for.

Deep learning (DL) has emerged as a powerful and general-purpose framework for learning from data which, as a result of the enormous amount of data and parallel computing available these days, has been capable of obtaining exceptional outcomes in a wide range of real-world applications. However, as a consequence of its fast adoption, there exist an increasing urge to develop *trustworthy* models for those applications where high stakes are in play, *i.e.*, where significant errors can entail severe consequences on individuals. In this dissertation, we argue that a key aspect of model trustworthiness is our *perception* of control over the model, *i.e.*, whether the model meets the pre-existing expectations that we place on it, and thus whether we can accurately predict its behaviour to our future requests.

In this thesis, we focus on the role that DL optimization has on training models matching ours expectations. Intuitively, although there may exist valid parametrizations within the parameter space, the optimization process usually has no mechanisms to prefer these optima over seemingly equal optima which do not fulfil our expectations. In order to incorporate our preferences into the training pipeline, we reinterpret them as optimization constraints, and revisit the concept of inductive bias as a way to guide the optimization towards low-error solutions that meet our preconceived expectations. Naturally, our expectations change according to the tasks we seek to solve with a model and, consequently, we explore in this thesis three families of models, in increasing order of the complexity of the expectations we place on the models.

In the first part, we focus on multitask learning (MTL), where we seek to simultaneously solve multiple tasks. In this context, we argue that the recent trend of casting MTL problems as multi-objective optimization (MOO) problems conceals a fundamental obstacle: contrary to our expectations, tasks are usually *not comparable*, leading to difficulties designing loss objectives, comparing models and, ultimately, optimizing our DL models. To address these issues, we advocate for the importance of clearly defining an objective function to optimize *a priori*, justify the use of ranking-based statistics to compare MTL models—grounded on probability theory and no-preference MOO methods—and introduce novel metrics and algorithms to measure and manipulate the interactions between task gradients in order to improve the optimization process.

In the second part, we shift to probabilistic generative models (PGMs), whose aim is to model the data distribution to later solve different tasks (*e.g.*, data generation, or missing-data imputation). PGMs not only comprehend the same set of expectations as MTL models do, but also carry additional constraints (*e.g.*, densities need to integrate to one) and expectations (*e.g.*, the information from each modality should be equally valid to infer any other data modality). We propose two approaches to meet these expectations: first, a preprocessing algorithm whose aim is to make each modality log-likelihood have a similar optimization landscape; and second, building strong connections between MTL and PGMs, an in-processing algorithm that leverages existing MTL methods to aggregate modality gradients on different parts of the network that are prone to comparability issues. We empirically show that the proposed methods are effective for modelling the data modalities more uniformly, significantly increasing the overall model performance.

In the third part, we consider causal generative models (CGMs), where the goal is now to learn the underlying *causal* mechanisms generating our data. In addition to those tasks solved by PGMs, CGMs can also be used to perform *causal inference*—*i.e.*, to respond *what-if* questions—which entails an additional set of causal expectations that we place on the models. Remarkably, we present in this dissertation a novel family of causal models, and prove that not only are they identifiable given a causal ordering between variables (*i.e.*, that we can recover them from observational data), but also that any other causal model—under a fairly mild set of assumptions—can also be reduced to an equivalent member of this family. Using the newly-developed theory, we introduce a new family of CGMs, causal normalizing flows (Causal NFs), which meet our causal expectations by design, and are the first-of-their-kind DL models to accurately perform causal inference while providing strong theoretical guarantees on their causal capabilities.

By carefully studying the (often implicit) expectations that we place on different families of DL models, this dissertation shows how we can incorporate this external knowledge into the training process, designing effective inductive biases that guide the optimization process towards optima that better match these expectations. As a result, we are able to consistently produce more reliable DL models, which we perceive to be more in control of, and which we can confidently use to better assist us in real-world scenarios.

Zusammenfassung

Deep Learning (DL) hat sich als leistungsfähige und universelle Methode für das Lernen aus Daten herauskristallisiert, der aufgrund der enormen Datenmengen und der heutzutage verfügbaren parallelen Datenverarbeitung in der Lage ist, in einer Vielzahl von realen Anwendungen außergewöhnliche Ergebnisse zu erzielen. Als Folge der schnellen Verbreitung besteht jedoch ein zunehmender Bedarf an der Entwicklung vertrauenswürdiger Modelle für Anwendungen, bei denen viel auf dem Spiel steht, d. h. bei denen signifikante Fehler schwerwiegende Folgen für den Einzelnen haben können. In dieser Dissertation argumentieren wir, dass ein Schlüsselaspekt der Vertrauenswürdigkeit eines Modells darin besteht, dass wir die Kontrolle über das Modell wahrnehmen, d. h. ob das Modell die Erwartungen erfüllt, die wir an es stellen, und ob wir daher sein Verhalten in Bezug auf unsere zukünftigen Anfragen genau vorhersagen können.

In dieser Arbeit konzentrieren wir uns auf die Rolle der DL-Optimierung beim Training von Modellen, welche unseren Erwartungen entsprechen. Intuitiv kann es zwar gültige Parametrisierungen innerhalb des Parameterraums geben, aber der Optimierungsprozess hat normalerweise keine Mechanismen, um diese Optima gegenüber scheinbar gleichwertigen Optima, die unsere Erwartungen nicht erfüllen, zu bevorzugen. Um unsere Präferenzen in die Trainingspipeline zu integrieren, interpretieren wir sie als Optimierungsbedingungen neu und greifen das Konzept der induktiven Verzerrung (inductive bias) wieder auf, um die Optimierung auf Lösungen mit geringen Fehlern zu lenken, die unseren vorgefassten Erwartungen entsprechen. Natürlich ändern sich unsere Erwartungen je nach den Aufgaben, die wir mit einem Modell zu lösen versuchen, und daher untersuchen wir in dieser Arbeit drei Modellfamilien, die in der Reihenfolge der Komplexität unserer Erwartungen an die Modelle aufgeführt werden.

Im ersten Teil konzentrieren wir uns auf das Multitasking-Lernen (MTL), bei dem wir versuchen, mehrere Aufgaben gleichzeitig zu lösen. In diesem Zusammenhang argumentieren wir, dass der jüngste Trend, MTL-Probleme als Mehrzieloptimierungsprobleme (MOO) zu betrachten, ein grundlegendes Hindernis birgt: Entgegen unseren Erwartungen sind die Aufgaben in der Regel nicht vergleichbar, was zu Schwierigkeiten beim Entwurf von Zielfunktionen, beim Vergleich von Modellen und letztlich bei der Optimierung unserer DL-Modelle führt. Um diese Probleme anzugehen, plädieren wir für die Wichtigkeit einer klaren Definition einer Zielfunktion, die a-priori optimiert werden soll, rechtfertigen die Verwendung von rangbasierten Statistiken zum Vergleich von MTL-Modellen - basierend auf der Wahrscheinlichkeitstheorie und MOO-Methoden ohne Präferenz - und stellen neuartige Metriken und Algorithmen vor, um die Interaktionen zwischen Aufgabengradienten zu messen und zu manipulieren, um den Optimierungsprozess zu verbessern.

Im zweiten Teil wenden wir uns probabilistischen generativen Modellen (PGMs) zu, deren Ziel es ist, die Datenverteilung zu modellieren, um später verschiedene Aufgaben zu lösen (zum Beispiel Datengenerierung oder Imputation fehlender Daten). An PGMs werden nicht nur die gleichen Erwartungen wie an MTL-Modelle gestellt, sondern auch zusätzliche Einschränkungen (z. B. müssen Dichten zu eins integriert werden) und Erwartungen (z. B. sollten die Informationen aus jeder Modalität gleichermaßen gültig sein, um auf jede andere Datenmodalität schließen zu können). Wir schlagen zwei Ansätze vor, um diese Erwartungen zu erfüllen: erstens einen Vorverarbeitungsalgorithmus, der darauf abzielt, dass die Log-Likelihood jeder Modalität eine ähnliche Optimierungslandschaft aufweist; und zweitens einen In-Processing-Algorithmus, der starke Verbindungen zwischen MTL und PGMs herstellt und bestehende MTL-Algorithmen nutzt, um Modalitätsgradienten auf verschiedenen Teilen des Netzwerks zu aggregieren, die anfällig für Vergleichbarkeitsprobleme sind. Wir zeigen empirisch, dass die vorgeschlagenen Methoden wirksam sind, um die Datenmodalitäten einheitlicher zu modellieren, was die Gesamtleistung des Modells deutlich erhöht.

Im dritten Teil betrachten wir kausale generative Modelle (CGMs), bei denen das Ziel nun darin besteht, die zugrunde liegenden kausalen Mechanismen zu erlernen, die unsere Daten erzeugen. Zusätzlich zu den Aufgaben, die von PGMs gelöst werden, können CGMs auch zur Durchführung von Kausalschlüssen verwendet werden, d. h. zur Beantwortung von Was-wäre-wenn-Fragen, was einen zusätzlichen Satz kausaler Erwartungen an die Modelle voraussetzt. Bemerkenswerterweise stellen wir in dieser Dissertation eine

neuartige Familie von Kausalmodellen vor und beweisen, dass sie nicht nur unter der Voraussetzung einer kausalen Ordnung zwischen den Variablen identifizierbar sind (d. h., dass wir sie aus Beobachtungsdaten wiederherstellen können), sondern auch, dass jedes andere Kausalmodell - unter recht milden Annahmen - ebenfalls auf ein äquivalentes Mitglied dieser Familie reduziert werden kann. Unter Verwendung der neu entwickelten Theorie stellen wir eine neue Familie von CGMs vor, die Causal Normalizing Flows (Causal NFs), die unsere kausalen Erwartungen von vornherein erfüllen und die ersten DL-Modelle ihrer Art sind, die kausale Schlussfolgerungen genau durchführen und gleichzeitig starke theoretische Garantien für ihre kausalen Fähigkeiten bieten.

Durch eine sorgfältige Untersuchung der (oft impliziten) Erwartungen, die wir an verschiedene Familien von DL-Modellen stellen, zeigt diese Dissertation, wie wir dieses externe Wissen einbeziehen und wirksame induktive Verzerrungen entwickeln können, um den Optimierungsprozess in Richtung von Optima zu lenken, die diesen Erwartungen besser entsprechen. Dadurch sind wir in der Lage, durchgängig zuverlässigere DL-Modelle zu erstellen, die wir besser zu beherrschen glauben und die wir vertrauensvoll nutzen können, um uns in realen Szenarien besser zu unterstützen.

Publications

Plagiarism is the sincerest form of flattery.

Oscar Wilde

The following publications are at the core of my PhD research and covered in this dissertation:

- [I] ‘Lipschitz standardization for multivariate learning.’
Javaloy, A., and Valera, I., in arXiv, 2020.
- [II] ‘RotoGrad: Gradient Homogenization in Multitask Learning.’
Javaloy, A., and Valera, I., in ICLR (🔦), 2022.
- [III] ‘Mitigating Modality Collapse in Multimodal VAEs via Impartial Optimization.’
Javaloy, A., Meghdadi, M., and Valera, I., in ICML (🔦), 2022.
- [IV] ‘Causal normalizing flows: from theory to practice.’
Javaloy, A., Sánchez-Martín, P., and Valera, I., in NeurIPS (🎤), 2023.

The following publications originated during my time as a PhD student but are omitted from this thesis:

- [V] ‘Text normalization using encoder–decoder networks based on the causal feature extractor.’
Javaloy, A., and García-Mateos, G., in Applied Sciences, 2020.
- [VI] ‘Preliminary results on different text processing tasks using encoder-decoder networks and the causal feature extractor.’
Javaloy, A., and García-Mateos, G., in Applied Sciences, 2020.
- [VII] ‘Relative gradient optimization of the Jacobian term in unsupervised deep learning.’
Gresele, L., Fissore, G., **Javaloy, A.**, Schölkopf, B., and Hyvärinen, A., in NeurIPS, 2020.
- [VIII] ‘Learnable Graph Convolutional Attention Networks.’
Javaloy, A., Sánchez-Martín, P., Levi, A., and Valera, I., in ICLR (🔦), 2023.

*: Equal contribution; 🎤: Oral; 🔦: Spotlight; 🔄: Another work plagiarized this one after publication.

Table of Contents

Acknowledgments	v
Abstract	vii
Zusammenfassung	ix
Publications	xi
Table of Contents	xiii
Notation	xvii
Acronyms	xix
1. Trustworthy Deep Learning and User Expectations	1
1.1. Trustworthy deep learning as functional predictability	1
1.2. From predictive to causal generative models: a motivating example	2
2. The Role of Optimization in Trustworthy Deep Learning	7
2.1. Weaknesses of deep learning optimization	8
2.2. Expectations and optimization	10
3. Thesis Outline and Contributions	17
3.1. Outline	17
3.2. Main contributions	17
I. MULTITASK LEARNING	21
4. Introduction to Multitask Learning	23
4.1. Historical overview	23
4.2. Problem statement	24
4.3. Network architectures	25
4.4. Multitask learning as multi-objective optimization	26
4.5. Task impartiality	26
4.6. Gradient conflict	28
5. Gradient Homogenization in Multitask Learning	31
5.1. Problem statement	31
5.2. Gradient homogenization	32
5.3. Illustrative examples	36
5.4. Empirical validation	37
5.5. Concluding remarks	41
6. On Task Incomparability and its Effects in Multitask Learning	43
6.1. Motivation and background	43
6.2. How to measure your multitask learning model	47
6.3. Benchmark probing	52
6.4. Empirical validation	58

6.5. Concluding remarks	62
II. PROBABILISTIC GENERATIVE MODELS	65
7. Deep Learning and Probabilistic Modelling	67
7.1. Problem statement	67
7.2. Exponential family	68
7.3. Latent-variable models	69
7.4. Modality collapse and multitask learning	73
8. On Modality Collapse and Data Preprocessing	77
8.1. Problem Statement	78
8.2. Multivariate impartial learning	79
8.3. Data scaling and smoothness	79
8.4. Lipschitz standardization	82
8.5. Empirical evaluation	84
8.6. Concluding remarks	86
9. Mitigating Modality Collapse via Impartial Optimization	89
9.1. Preliminaries	89
9.2. Impartial optimization in multimodal variational autoencoders	90
9.3. Extending our framework	93
9.4. Experiments	97
9.5. Concluding remarks	101
III. CAUSAL GENERATIVE MODELS	103
10. Introduction to Causal Inference in Deep Learning	105
10.1. Correlation does not imply causation	105
10.2. Structural causal models	106
10.3. Causal inference	107
10.4. Problem statement	109
10.5. Existing works	109
11. Causal Identifiability Given a Causal Ordering	111
11.1. Solution characterization	111
11.2. The multiple representations of structural causal models	113
11.3. Causal identifiability	116
11.4. Extension to real-world settings	119
11.5. Reimplementing the do-operator	122
11.6. Concluding remarks	124
12. Effective Deep Causal Inference with Causal Normalizing Flows	125
12.1. Causal normalizing flows	126
12.2. Causal inference queries	129
12.3. Ablation study	130
12.4. Model comparison	133
12.5. Fairness auditing and classification	133
12.6. Concluding remarks	134

IV. EPILOGUE	137
13. Conclusion	139
13.1. Summary and impact	139
13.2. Prospects	141
V. APPENDIX	145
A. Additional Material for Chapter 5	147
A.1. Proofs	147
A.2. Stackelberg games and RotoGrad	147
A.3. Experiments	149
B. Additional Material for Chapter 6	161
B.1. Experimental details	161
C. Additional Material for Chapter 8	165
C.1. Basic properties of L-smoothness	165
C.2. Exponential family	165
C.3. Full description of Lipschitz standardization	167
C.4. L-smoothness after standardizing	170
C.5. Details on the experimental setup	174
C.6. Additional experimental results	176
D. Additional Material for Chapter 9	181
D.1. General algorithm	181
D.2. Analysis by data type	182
D.3. Model descriptions	184
D.4. Experimental details	187
E. Additional Material for Chapters 11 and 12	197
E.1. Interventions in previous works	197
E.2. Experimental details and extra results	198
E.3. Details on the fairness use-case	205
Bibliography	207

Notation

Some of the notation is influenced by Goodfellow et al. [67] and their book ‘Deep Learning.’

Set and number notation

x	A scalar.
\mathbf{x}	A column vector.
\mathbf{X}	A matrix.
\mathbb{X} or \mathcal{X}	A space.
x_i	The i -th entry of the vector \mathbf{x} .
$x_{i:j}$	The elements x_i, x_{i+1}, \dots, x_j of the vector \mathbf{x} , with $i < j$.
x_A	The elements of the vector \mathbf{x} with indexes in A .
$[\mathbf{x}_1 \ \mathbf{x}_2 \ \dots \ \mathbf{x}_D]$	Matrix made by stacking the vectors $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_D$.
$\mathcal{P}(D)$	The set of all possible sets with elements $1, 2, \dots, D$.
\mathbf{I}	Identity matrix.
$\mathbf{0}$ and $\mathbf{1}$	Constant vectors full of, respectively, zeroes and ones.

General operations

$\mathbf{1}_{\{x \in A\}}$	Indicator function on the set A .
$\lfloor x \rfloor$	Integer part of x .
$\log(x)$	Napierian logarithm, <i>i.e.</i> , in base e , of x .
$\text{cos}(\mathbf{x}, \mathbf{y})$	Cosine similarity between \mathbf{x} and \mathbf{y} .
$\mathbf{x} \odot \mathbf{y}$	Hadamard or element-wise product of \mathbf{x} and \mathbf{y} .
$f \circ g$	Function composition of f and g .
$\mathbb{X} \times \mathbb{Y}$	Cartesian product between the spaces \mathbb{X} and \mathbb{Y} .

Real analysis

$\partial_x f(x)$ or $\frac{\partial}{\partial x} f(x)$	Partial derivative of $f(x)$ with respect to x .
$\frac{d}{dx} f(x)$	Total derivative of $f(x)$ with respect to x .
$\nabla_{\mathbf{x}} f(\mathbf{x})$	Jacobian matrix of $f(\mathbf{x})$ with respect to \mathbf{x} .

Complexity theory

$o(\epsilon)$	Little-o of ϵ .
$\mathcal{O}(n)$	Big-O of n .

Relationships

$x = y$	'x equals y.'
$x := y$	'x equals y by definition.'
$x \stackrel{d}{=} y$	'x equals y in distribution.'
$x \stackrel{\text{a.s.}}{=} y$	'x equals y almost surely.'
$x \leftarrow y$	'x gets assigned the value y.'
$x \sim y$	'x relates to y.'
$x \approx y$	'x approximates y.'
$x \propto y$	'x is proportionally equal to y.'
$f \stackrel{\text{a.e.}}{=} g$	'f equals g almost everywhere.'
$A \equiv B$	'A is structurally equivalent to B.' Defined in Chapter 11 .
$A \leq B$	'A is structurally sparser than B.' Defined in Chapter 11 .

Statistics

x	A random scalar variable.
\mathbf{x}	A random vector variable.
\mathbf{X}	A random matrix.
P_x	The distribution function of the variable x .
$P_x(x y; \theta)$	Distribution of x , given y , with parameters θ .
$x \sim P_x$	'x is distributed according to P_x .'
$\mathbf{x} \stackrel{i.i.d.}{\sim} P_x$	'x are independent and identically distributed according to P_x .'
$\mathbb{E}_{P_x}[f(x)]$	Expected value of $f(x)$ with respect to $x \sim P_x$.
$\mathbb{V}_{P_x}[f(x)]$	Variance of $f(x)$ with respect to $x \sim P_x$.
$\text{KL}(p \parallel q)$	Kullback-Leibler divergence between the densities p and q .
$f_{\#}p$	Push-forward measure of p through f .
$\delta_{x \in A}$	Dirac delta measure located in the set A .
$\mathcal{N}(\mu, \sigma)$	Normal (or Gaussian) distribution with mean μ and standard deviation σ .
$\log \mathcal{N}(\mu, \sigma)$	Log-normal distribution with mean μ and standard deviation σ .
$\mathcal{U}(a, b)$	Continuous uniform distribution in the interval $[a, b]$.
$\text{Cat}(\pi)$	Categorical distribution with probabilities π .
$\Gamma(a, b)$	Gamma distribution with parameters a and b .

Acronyms

- ANF** Autoregressive normalizing flow. (Pages 19, 109, 110, 125–129, 131, 132, 134, 140, 205)
- ATE** Average treatment effect. (Pages 130, 131)
- BBVI** Black-box variational inference. (Pages 69, 70, 77, 78, 84, 87, 175)
- BCE** Binary cross-entropy. (Pages 39, 41, 52, 151–153, 162)
- Causal NF** Causal normalizing flow. (Pages vii, 14, 19, 109, 110, 125–135, 140, 143, 198, 201, 205, 206)
- CDF** Cumulative distribution function. (Pages 18, 43, 47–49, 62, 139)
- CGM** Causal generative model. (Pages vii, 2, 5, 6, 11, 17–19, 105, 109, 111, 125, 140)
- DAG** Directed acyclic graph. (Pages 112, 121)
- DL** Deep learning. (Pages vii, 1–3, 6–12, 14, 17–19, 27, 43, 45, 67, 69, 71, 73, 109, 110, 124–126, 139–143)
- DM** Decision maker. (Pages 12, 27, 49, 74)
- DNN** Deep neural network. (Pages 109, 110)
- ELBO** Evidence lower bound. (Pages 69, 70, 78, 90, 95, 185–187, 194)
- GMM** Gaussian mixture model. (Pages 189)
- GNN** Graph neural network. (Pages 110, 133, 197, 202)
- HI-VAE** Heterogeneous-incomplete VAE. (Pages 72, 93, 94, 175, 186, 188)
- ICA** Independent component analysis. (Pages 11, 111, 118)
- IWAE** Importance weighted autoencoder. (Pages 90, 185, 194)
- KL** Kullback-Leibler divergence. (Pages 69, 70, 132)
- KR** Knöthe-Rosenblatt. (Pages 117, 118, 120–122)
- LVM** Latent variable model. (Pages 78, 79, 89)
- MAF** Masked autoregressive flow. (Pages 130, 132, 201)
- ML** Machine learning. (Pages 8, 14, 23, 24, 67, 69, 79)
- MLE** Maximum likelihood estimation. (Pages 78, 81, 126, 127, 129, 135, 170)
- MOO** Multi-objective optimization. (Pages vii, 12, 17, 18, 26, 27, 32, 43, 47–50, 62, 74, 141, 142)
- MSE** Mean squared error. (Pages 38, 52, 60, 85, 86, 97, 150, 151, 154, 155, 162)
- MTL** Multitask learning. (Pages vii, 2–4, 12–14, 17, 18, 23–33, 37, 38, 40, 41, 43–48, 50, 52, 53, 55, 57, 58, 60–63, 67, 73–75, 89, 92–94, 101, 139, 141, 142, 150, 161, 189, 191)
- NF** Normalizing flow. (Pages 109, 126, 130, 131, 133)
- NLL** Negative log-likelihood. (Pages 52, 151, 162)
- NSF** Neural spline flow. (Pages 132, 134, 205)
- PGM** Probabilistic generative model. (Pages vii, 2, 4–6, 17, 18, 28, 67, 73–75, 77, 78, 84, 90, 105–108, 125, 126, 139)
- RMSE** Root mean squared error. (Pages 44, 130)
- R. V.** Random variable. (Pages 48, 49, 67, 68, 70, 75, 78, 81–83, 89, 96, 98, 105, 106, 109, 113, 119, 166, 182)
- SCC** Strongly connected component. (Pages 121, 122, 205)
- SCM** Structural causal model. (Pages 19, 106–118, 120, 122–134, 140, 197–199, 201, 203)
- SGD** Stochastic gradient descent. (Pages 8, 11)
- SIWAE** Self-importance weighted autoencoder. (Pages 194, 195)
- SPL** Semantic probabilistic layer. (Pages 14)
- STL** Single task learning. (Pages 23, 44, 53, 55, 58, 60, 154, 155, 159, 163)
- TMI** Triangular monotonically increasing. (Pages 19, 117, 118, 120–122, 124, 125, 127, 129, 132, 134)
- VAE** Variational autoencoder. (Pages 71–73, 75, 77, 84, 86, 89–101, 109, 140, 175, 176, 184, 185, 189, 192, 193)

Trustworthy Deep Learning and User Expectations

1.

El privilegio de ser mayoría
te hace responsable sobre la minoría.

Mafalda; Haw Haw!

The impact of **deep learning (DL)** in recent years is undeniable. With the establishment of the Internet, the massive use of personal devices (*e.g.*, smartphones), and the vast amount of parallel computing offered by modern processing units, we have created a perfect storm where data-hungry methods, such as those from DL, are able to truly shine. The impact is such that Christopher M. Bishop, a renowned figure in the field, starts his latest book [13] with a chapter titled ‘The Deep Learning Revolution’, stating that ‘DL has emerged as an exceptionally powerful and general-purpose framework for learning from data.’

DL is indeed an extremely general-purpose framework when successfully applied, but we should not underestimate the amount of manual work that goes into designing DL models, and the difficulties encountered in the way. One noteworthy example is the use of *live* training, which has become the norm to train extremely large models released to the general public, such as DALL·E [43]. When training such models (see Figure 1.1) a human expert supervises the optimization process, recovering the model parameters when the training diverges, and resuming it with a different set of hyperparameters chosen based on their personal expertise. Unfortunately, examples like this one are rarely released to the public.

While live training is a clear reflection of how brittle DL optimization can be, even when successfully trained, DL models can still behave in mysterious and unexpected ways (*e.g.*, in path routing applications, occasionally providing invalid routes that form a disconnected path). This becomes specially relevant when DL is employed in *high-stake applications*, such as medical, planning, or industrial applications, where significant errors from the model can entail severe consequences in the real world. As a result, there exists an *increasing urge* to improve the **trustworthiness** of DL models, so that we can safely rely on them.

The end goal of this dissertation is to set grounds for more trustworthy DL models, for which we will delve in the importance that DL optimization can have towards this goal. In the following, we describe what we understand by trustworthy here, and motivate our perspective by showing how different expectations are placed on different families of DL models depending on the complexity of their attributed tasks.

1.1 Trustworthy DL as functional predictability

In this section, we try to convey the idea that, perhaps counter-intuitively, we must be able to predict the functionality of a model in order to *trust* it. To this end, let us first paraphrase an example extracted from Hyvärinen [79, Chapter 11] on a system which we naturally trust: *our own body*.

1.1 Functional predictability	1
1.2 From predictive to causal generative models	2

[13] Bishop and Bishop (2024), ‘Deep Learning: Foundations and Concepts.’

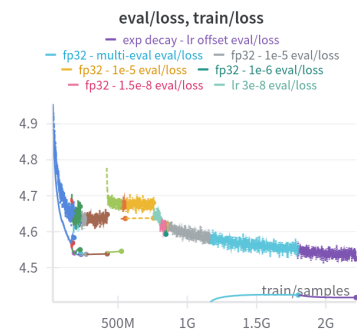



Figure 1.1: Logs of the live training of DALL·E. Each colour represents a different set of hyperparameters manually set by a human in-the-loop.

[43] Dayma (2022), *DALL·E Mega - Training Journal* [🔗](#)

[79] Hyvärinen (2022), ‘Painful intelligence: What AI can tell us about human suffering.’ [🔗](#)

It is undeniable that we, as humans, generally *feel* in control of our bodies, and that we can do things as natural as raising an arm. We *trust* our bodies: if we need to run an errand, we blindly trust that our legs will effortlessly give one step after the other to reach our final destination; we *control* our legs. Consequently, a common belief is that there exists a central executive unit that controls our bodies. However, there is a more interesting take: what if we do not have control, but we instead *perceive* that we have it because we can *predict* the received outcome? That is, ‘I trust my legs, because whenever I want them to walk, they comply.’

1: *I.e.*, the prediction error.

[155] Petkova and Ehrsson (2008), ‘If I Were You: Perceptual Illusion of Body Swapping.’ 

This is not always the case (*e.g.*, my knee may hurt at a certain moment), but as long as the *difference* between outcomes and expectations¹ stays relatively low, we still feel that we are in control. One extreme example of this control perception is the *rubber hand illusion* [155]. In this experiment, a subject’s arm is hidden from plain sight, and a rubber one is attached to the subject, with the shoulder joint covered by a piece of cloth. Then, a series of stimuli (*e.g.*, pinching, touching, or caressing with a feather) are simultaneously performed on the real and rubber arms. When these actions are conducted for enough time, the subject ‘learns to trust’ the rubber arm, as the expected feelings meet the subject’s visual perception, to the point that when new actions are now performed *only* on the rubber arm, they are truly ‘felt’ by the subject for a moment.

In this dissertation, we argue that we can apply a similar logic to DL models: if, when I request the model to solve a certain task, the outcome *always* meets my preconceived expectations, up to a certain tolerable error, then I will start considering the model *trustworthy*, as the predictability of its functionality will make me *feel* that I am in control of the model. This is up to this date, unfortunately, a far-fetched desire, if we take into account the weaknesses and expectations that we will introduce later in [Sections 1.2](#) and [2.1](#). To make DL more trustworthy, we need to make DL optimization more robust, and ensure that *the optima we find during training do indeed match our expectations in practice*.

1.2 From predictive to causal generative models: a motivating example

1.2.1 MTL approach	3
1.2.2 Probabilistic approach	4
1.2.3 Causal approach	5

The set of expectations that we place on a model change according to the complexity and diversity of the tasks we design it to solve. We now illustrate this idea with a small motivating example, which we tackle with the three DL frameworks discussed in each part of this dissertation, in order of appearance: **multitask learning (MTL)**, **probabilistic generative models (PGMs)**, and **causal generative models (CGMs)**.

Our hypothetical scenario is set at the heart of an important international video-on-demand company, which provides uncountable hours of entertainment to millions of people around the globe. Working remotely in a tiny apartment of New Yorkshire we find Paul, the head manager of the data-centric research department of the company. Paul has been tasked with an extremely important task: he is in charge of overhauling the existing recommendation system of the platform.

The company has experienced a decrease in expected revenue growth and number of active users in the platform—which is inconceivable—and

they thus want you to incorporate the latest DL technology to reverse the situation. To this end, Paul has to design a single DL model (maintenance is expensive), and he has been promised as much historical data as needed for the following set of attributes:

- ▶ User information:
 - 👤 User data: number of subscriptions, current-video topics, *etc.*
 - ♂ Sensitive attributes of the user, *e.g.*: sex, age, *etc.*
 - 📍 Location from which the user is connected to the platform.
- ▶ Recommendations to the platform:
 - 🕒 Total duration of advertisements for the current video.
 - ⏮ Attributes of the incoming suggested video.
- ▶ Predictions on the user's reaction:
 - 👁 Whether the user will finish watching the current video.
 - 📈 User engagement, *i.e.*, whether the user will stay (or return) to watch the suggested videos.

The DL model needs to provide recommendations to the platform to maximize revenue, while keeping the users engaged to ensure future interactions, based on the information of the user and the current video they are watching. To the surprise of Paul, successfully designing such a system can be more complicated than initially expected.

1.2.1 Multitask learning approach

The first attempt of Paul is to use an MTL approach, *i.e.*, to train a DL model that can solve several tasks simultaneously. An schematic of the model can be found in [Figure 1.2](#). Here, the model takes as input information about the user and the current video, and learns during training to simultaneously predict the other quantities, *i.e.*:

$$f_{\theta}(\text{👤}, \text{♂}, \text{📍}) = [\text{⏮}, \text{🕒}, \text{👁}, \text{📈}], \quad (1.1)$$

where f is the MTL model with parameters θ . This first model can just perform one fixed task: take the user information as input, and provide all the recommendations and predictions.

Consequently, the additional expectations that Paul places on this model are simple, and regard the interaction between tasks. First, Paul needs to decide what trade-off he is comfortable doing, as the model might not be able to perfectly predict all targets always. In this case, Paul takes an uninformed standpoint and decides that *all tasks are equally important*, *i.e.*, the model should be *impartial* towards learning one task over another. A second and more subtle expectation that Paul places on the model is that, during training, the model can tell whether one task is being learnt more than another, *i.e.*, that the tasks are *comparable*.

Unfortunately, even if the model meets Paul's initial expectations, its functionality soon falls short for a number of scenarios. For example:

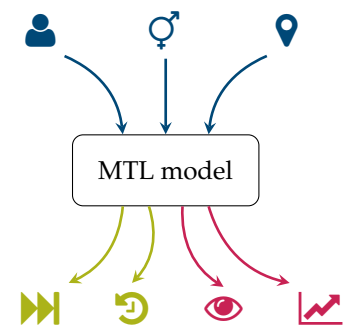


Figure 1.2: Illustrative sketch of Paul's MTL approach. User information works as the input of the model, and the rest of variables compose the output its output.

- The platform has a queue functionality, for which the user can append videos to watch after the current one. Therefore, the model should be flexible enough to predict the rest of outcomes, having observed the attributes of the next ‘suggested’ video.
- In order to understand and improve the recommendation system, data scientists would like to consult the likelihood of different outcomes and find out, *e.g.*, the average duration of the shown advertisements, or whether there exists a correlation between this same variable and the user location.

The company quickly urges Paul to find a different approach, increasing the range of queries that the model can answer.

1.2.2 Probabilistic generative approach

After considering different approaches, Paul comes up with the idea of using probabilistic generative modelling. This is a major change in the way he was tackling the problem in two different aspects:

1. Instead of predicting some variables using the rest as the input, he will now train a model that will be able to generate all variables at the same time, imitating the observed data distribution.
2. PGMs are generally more flexible than MTL models, allowing for different types of conditional and marginal queries.

2: PGMs will be fully covered in [Part II](#).

While we gloss over the details for now,² [Figure 1.3](#) illustrates the new model, which now takes all variables as input *and* output. In mathematical terms, this means that the model now models the joint distribution of the data, *i.e.*, the model is of the form:

$$p_{\phi}(\text{user}, \text{gender}, \text{location}, \text{video_id}, \text{duration}, \text{view_count}, \text{click_count}), \quad (1.2)$$

where p_{ϕ} denotes the joint density of the data, parametrized by ϕ .

In order to generate new data, the model relies on a *latent variable* \mathbf{z} which ties together the dependencies between all observed variables, *i.e.*, the joint density in [Equation 1.2](#) is actually of the form:

$$\int p_{\phi}(\text{user}, \text{gender}, \text{location}, \text{video_id}, \text{duration}, \text{view_count}, \text{click_count} | \mathbf{z}) p_{\phi}(\mathbf{z}) d\mathbf{z}, \quad (1.3)$$

and, in order to conditionally generate new data, the model also learns the posterior distribution of \mathbf{z} as

$$p_{\theta}(\mathbf{z} | \text{user}, \text{gender}, \text{location}, \text{video_id}, \text{duration}, \text{view_count}, \text{click_count}), \quad (1.4)$$

where θ is another set of parameters and, for simplicity, we can safely assume that the model can adapt to any of the conditional variables in the equation above to be missing.

With the increased complexity of the model, the expectations placed on it become more complex as well. Just as before, Paul expects the model to be *impartial* towards learning to model one variable over another, as he has no preference in that regard and, again, this implies that Paul expects the model to be able to compare these variables during training to meet the first expectation. Now, additionally, Paul expects the model



Figure 1.3: Illustrative sketch of Paul's PGM approach. The model learns the joint distribution and can generate all observed variables.

to also be well-behaved when it comes to the new conditional queries, *i.e.*, the model predictions should be equally good *irrespective* of which variables are used for conditioning.

Despite the improvements made to the model, the list of scenarios for which the model should provide a compelling solution keeps increasing. Specifically, Paul has found that the new PGM cannot be satisfactorily applied in the following scenarios:

1. In order to further increase revenue, the company has been testing case-specific algorithms only for the duration of advertisements. Unfortunately, while the model can use this variable as input through conditioning, the new predictions given by the system largely differ from those observed in reality during AB testing.
2. The company has also been warned by the authorities that their recommendations has been shown to be biased against some protected groups. To avoid legal problems, Paul needs to *make sure* that the recommendations made by the model are not using any information from the sensitive attributes of the users.

Unfortunately, a probabilistic approach does not seem enough to overcome these further issues and, once again, Paul needs to go back to the drawing board and come up with a better approach.

1.2.3 Causal approach

After doing his homework, Paul realizes that the conditional predictions given by the previous model did not match the real experiments since these were not conditional, but *interventional* queries. In short,³ there exist a cause-and-effect relationship between variables which the PGM does not take into account. As a result, the answers provided by the PGM only reflect the passive observational world, but do not actually simulate the downstream effect that externally changing a variable would have. In mathematical terms, this is expressed as

$$\begin{array}{c}
 \text{Probability of observing } \text{👁} \dots \\
 \downarrow \\
 P(\text{👁} \mid \text{👤}) \neq P(\text{👁} \mid \text{do}(\text{👤})). \quad (1.5) \\
 \begin{array}{cc}
 \uparrow & \uparrow \\
 \text{... given that I observed} & \text{... given that I forced} \\
 \text{👤 having this value.} & \text{👤 to have this value.}
 \end{array}
 \end{array}$$

To take causality into account, and thus perform accurate simulations, Paul replaces the previous probabilistic model by a causal one. The new model attempts to imitate the underlying causal data-generating process governing the observational data. For this reason, in addition to training data, Paul needs to provide the CGM with a *causal graph* during training, *i.e.*, a graph describing the cause-and-effect relationships between variables. Figure 1.4 illustrates the new CGM, where the arrows indicate the variable-generation order, *i.e.*, that the joint distribution is modelled according to the following factorization:

$$\begin{aligned}
 p_{\theta}(\text{👤}, \text{♀}, \text{📍}, \text{📺}, \text{👤}, \text{👁}, \text{📈}) &= p_{\theta}(\text{👤}) p_{\theta}(\text{♀}) p_{\theta}(\text{📍}) \\
 &\quad p_{\theta}(\text{📺} \mid \text{👤}, \text{📍}) p_{\theta}(\text{👤} \mid \text{👤}, \text{📍}) \\
 &\quad p_{\theta}(\text{👁} \mid \text{👤}, \text{♀}, \text{👤}) p_{\theta}(\text{📈} \mid \text{👤}, \text{♀}, \text{👤}, \text{📺}, \text{👁}). \quad (1.6)
 \end{aligned}$$

3: We will go into detail in [Part III](#).

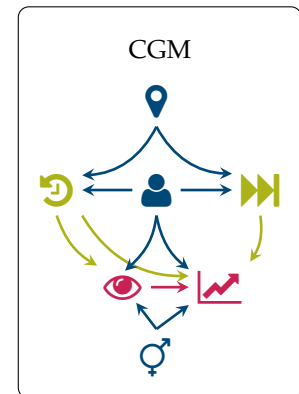


Figure 1.4: Illustrative sketch of Paul's CGM approach. The model learns the underlying data-generating process, respecting the cause-effect relationships between variables.

Not only this model can answer all the queries that the previous models could, but now it can also answer *causal inference* queries, *i.e.*, questions about *what-if* scenarios of how the world would have been, had something else happened. Additionally, structural constraints are trivial to ensure, *e.g.*, if the model follows the causal graph provided, the sensitive attributes are then ensured not to cause the recommendations, yet they can still be used to predict the user's behaviour.

Unfortunately, CGMs not only undergo the same set of expectations as the previous PGMs do, but they also need to meet causal expectations. Namely, they need to make sure the data is generated according to the causal graph provided, *e.g.*, the model needs to make sure that the location, 📍, does not directly affect the user engagement variable, 📈, but only indirectly through the recommendation variables, 🎵 and 🎬.

This is a significantly more challenging expectation to meet, especially if Paul's aim is to use a single DL model that can perfectly learn the underlying causal system. In **Part III** of this dissertation, we will introduce a novel family of models which will be able to meet these expectations, while providing theoretical guarantees that they can actually learn the causal data-generating process.

The Role of Optimization in Trustworthy Deep Learning

2.

Por allí viene Durruti con las tablas de la ley,
pá que sepan los obreros que no hay patria, dios, ni rey.

Chicho Sánchez Ferlosio; Romancero de Durruti

2.1 Weaknesses of DL optimization	8
2.2 Expectations and optimization	10

In the previous chapter, we argued on the importance of having **deep learning (DL)** models that are **trustworthy**, *i.e.*, models for which we feel in control as they comply with our expectations on how they should behave. This way, every real-world application (and, especially, high-stake ones) could safely benefit from the many advantages of DL models. To near this goal, we consider in this thesis the role that optimization has on producing trustworthy models and, as this chapter is focused on DL optimization, we need to first introduce some initial formulation.

In general, we can define any DL model as the result of a constrained optimization process that tries to solve a problem of the following form, defined over a training population $\mathbf{x} \sim P_{\mathbf{x}}$:

$$\underset{\theta \in \Theta}{\text{minimize}} \quad \mathbb{E}_{\mathbf{x} \sim P_{\mathbf{x}}} [L(\mathbf{x}, \theta)], \quad (2.1)$$

$$\begin{aligned} \text{s.t.} \quad & g(\mathbf{x}, \theta) = 0 \\ & \forall \mathbf{x} \text{ with } p(\mathbf{x}) > 0, \end{aligned} \quad (2.2)$$

i.e., we search for the optimal parameters θ , in the parameter space Θ , that minimize the average *objective* L , while fulfilling the *constraints* described by g for every *plausible* element of the population (*i.e.*, $p(\mathbf{x}) > 0$). To make this optimization process amenable, the gold-standard in DL is to use *unconstrained* first-order optimization. That is, an iterative process which updates the model parameters using gradient information:¹

$$\overset{\text{new parameters}}{\theta^{t+1}} = \overset{\text{current parameters}}{\theta^t} - \overset{\text{step size}}{\alpha^t} \overset{\text{direction in } \Theta \text{ to follow}}{\nabla_{\theta} \mathbb{E}_{\mathbf{x} \sim P_{\mathbf{x}}} [L(\mathbf{x}, \theta)]}. \quad (2.3)$$

1: We have opted here for a simplified notation that does not include all possible algorithms, since the goal is to provide an intuition on the role of the gradient in the model parameters.

In this chapter, we argue that the lack of robustness in DL, meaning that they do not meet our expectations during deployment, can be understood as the result of converging to ‘undersired’ local optima. One of the main arguments supporting this view is that, using an universal-approximation point-of-view [76], any large-enough DL model should be able to reach the optima that meet our expectations. However, during training, the optimization algorithm has no mechanisms to prefer these optima over other ones whose final properties do not suit our needs.

The main methodological contribution of this dissertation is to **interpret our expectations as optimization constraints** (*i.e.*, as g in Equation 2.2), and therefore to argue that the optimization process should be aware of these constraints during training. Put differently, **you cannot optimize what you do not account for**. To incorporate our expectations to the optimization process, we revisit in this chapter the concept of inductive bias, and reinterpret it as such a mechanism to inform the optimization process about the expectations placed on the model behaviour.

[76] Hornik, Stinchcombe and White (1989), ‘Multilayer feedforward networks are universal approximators.’

2.1 Weaknesses of DL optimization

Despite DL models being powerful when properly trained, their optimization can be difficult, tedious, and brittle. We present now a number of examples to illustrate the existing difficulties on training DL models, connecting the lack of robustness of DL models at deploy time, with the use of uninformed optimization processes that reach undesired local optima which do not meet our expectations. This idea defies contemporary practices, where optimization problems are often downplayed, as practitioners try to be overcome them with sheer brute-force using, *e.g.*, over-parametrized models, post-hoc model-selection approaches, tailored hand-crafted training processes, or mere trial-and-error.

[163] Robbins and Monro (1951), ‘A stochastic approximation method.’

[67] Goodfellow, Bengio and Courville (2016), ‘Deep Learning.’

[66] Glorot and Bengio (2010), ‘Understanding the difficulty of training deep feedforward neural networks.’

[150] Pascanu, Mikolov and Bengio (2013), ‘On the difficulty of training recurrent neural networks.’

Training divergence. Stochastic optimization processes as the one in Equation 2.3 converge under fairly mild assumptions [163]. However, this is often not the case in practice and, while we do not cover training divergence in this thesis, we consider it important to stress the numerical issues of training DL models even to this date. One big factor is the model architecture: when composed of many layers, it can lead to vanishing and exploding gradients, where the gradients become too small or large, respectively, leading to uninformative or unstable gradient directions [67, Chapter 8.2.5]. Although there exist practical solutions (*e.g.*, by using tailored weight-initialization techniques [66], or by clipping the norm of the gradients [150]), it is still a common practice to use live training with extremely large models (see Figure 1.1).

Unfortunately, even if the training does converge, we have in general no guarantees on the properties that the DL model with that particular parametrization will have. That is, while we know that the desired optima exist in our parameter space (*e.g.*, by using universal-approximator arguments [76]), it is often the case that the optimization ends up in *undesired local optima* with adverse consequences. Some relevant examples of this undesired behaviour are the following:

Generalization problems. Another fundamental issue of DL is the lack of generalization, *i.e.*, obtaining DL models whose performance does not transfer well to slightly different scenarios from the ones they were trained on, opposite of what we would generally expect. In other words, these models cannot *generalize* to different environments. While this is ubiquitous in all **machine learning (ML)**, the case of DL is particularly interesting as classical arguments, such as the bias *v.s.* variance trade-off, seem to generally not apply to these settings [9].

Instead, over-parametrized DL models can actually generalize quite well if trained accordingly, *i.e.*, if they find the correct local optima. While there are post-hoc techniques to select such models (*e.g.*, cross-validation), we can as well use inductive biases to guide the optimization algorithm towards optima with better generalization capabilities. One particularly interesting example is that of **stochastic gradient descent (SGD)**, for which recent work found that it carries *implicit inductive biases* favouring solutions with flatter local optima [190]. Intuitively, solutions with flatter neighbourhoods are more robust (and, therefore, generalize better) since small changes in the input provide similar outputs.

[9] Belkin, Hsu, Ma and Mandal (2019), ‘Reconciling modern machine-learning practice and the classical bias–variance trade-off.’

[190] Smith, Dherin, Barrett and De (2021), ‘On the Origin of Implicit Regularization in Stochastic Gradient Descent.’

There are, however, other notions of generalization that are harder to address. For example, during optimization, DL models solely focus on solving one task given the training data, which can make them pick up on *spurious correlations* that are simple and predictive of the target at hand, but that do not transfer at all. One common example is that of image classification. Say that you train a DL model to classify different animals and, since cows are usually pictured in a green meadow, one simple way of quickly achieving high accuracy is to predict the presence of a cow if the picture contains lots of green. However, there is no need to say that a cow is not defined by the surrounding scenery, and so it is futile to rely on observing a green meadow if we want to learn *what* is a cow, and *how* to predict if there is one in any given picture.

In contrast, if the model is properly trained using, *e.g.*, external causal knowledge describing the relationship between a cow and its surroundings to help us guide the optimization, it is possible to teach the model robust features that generalize well when presented with similar data, yet recollected from different environments. We will explore this idea in **Part III** of the dissertation.



Figure 2.1: Two images of: **(top)** a cow; and **(bottom)** something that we were told is a cow. Images were taken from tinyurl.com/clearly-a-cow and tinyurl.com/maybe-a-cow.

Incomparable objectives. Let us imagine a multitask setup similar to that we presented in **Subsection 1.2.1**, where we aim to learn two tasks with different loss objectives, L_1 and L_2 , respectively. Let us further imagine that the first loss, in general, takes values orders of magnitude larger than those from the second one, $L_1 \gg L_2$. To optimize the DL model, we combine both losses by adding them up, *i.e.*:

$$L(\mathbf{x}, \boldsymbol{\theta}) := L_1(\mathbf{x}, \boldsymbol{\theta}) + L_2(\mathbf{x}, \boldsymbol{\theta}). \quad (2.4)$$

As we will discuss later in **Part I**, a common side effect of aggregating losses this way is that the optimization lands on local optima that learn to solve the first task, but that largely disregard solving the second one. The mathematical intuition is that the differences in magnitude are generally transferred to the task gradients, *i.e.*,

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x} \sim P_{\mathbf{x}}} [L(\mathbf{x}, \boldsymbol{\theta})] = \underbrace{\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x} \sim P_{\mathbf{x}}} [L_1(\mathbf{x}, \boldsymbol{\theta})]}_{\gg} + \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x} \sim P_{\mathbf{x}}} [L_2(\mathbf{x}, \boldsymbol{\theta})], \quad (2.5)$$

and, as a result, the optimization prioritizes learning the first task.² Once the model has learned the first task such that its gradient norm converges to zero, it is fairly difficult to start learning the second task, as the first one will pull the optimization back to the local optima upon straying away. In this scenario, even though we have converged, we have no guarantees on the performance of the DL model *on individual tasks*, not meeting our expectations on the trade-off between tasks.

2: Remember the update rule we introduced in **Equation 2.3**.

Inconsistent trade-off solutions. Unfortunately, we may have no guarantees on the performance of individual tasks even if the task objectives were comparable. Following on the previous example, assume now that both losses are *comparable*, *i.e.*, $L_1 \approx L_2$. Say that you train three identical models with different random initializations using the loss in **Equation 2.5**, and that they follow the training trajectories depicted in

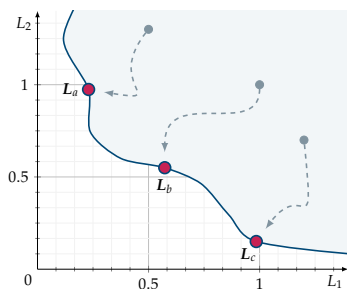


Figure 2.2: Illustrative example of how different initializations lead to different trade-off solutions. Dashed lines represent the trajectories followed during optimization, which end up in three trade-off optimal solutions.

Figure 2.2, obtaining three different loss vectors:

$$L_a = [0.23 \ 0.974] \quad L_b = [0.573 \ 0.55] \quad L_c = [0.985 \ 0.15]$$

Why did this happen? In short, different model initializations start with different initial losses, which can create an artificial imbalance between objectives, just as in the previous example. Combined with the local, stochastic nature of DL optimization, we have that the models ended up in different *non-dominated* solutions. Interestingly, note that all three loss vectors approximately sum to 1.15, *i.e.*, we could consider all three of them as quantitatively similar solutions, yet they exhibit extremely different results when tasks are considered individually.

Again, the optimization process is unaware of a more important question: which model do *you* prefer? If we took an impartial point-of-view as in [Subsection 1.2.1](#), L_b clearly looks like the most impartial solution. In that case, while *we* can tell which solution comes closest to meet our expectations,³ we often leave the optimization blind to this known expectation, and thus *we have no control* on the trade-off solution that the optimization will converge to.

3: And we could choose it *a posteriori*, again, brute-forcing the problem.

2.2 Expectations and optimization

2.2.1 Specificity	11
2.2.2 Taxonomy	12

[16] Caruana (1993), ‘Multitask Learning: A Knowledge-Based Source of Inductive Bias.’

We now formally revisit the concept of inductive bias, and argue that it can be used to incorporate our expectations in the optimization process, helping us guide the model towards more preferable local optima. However, we should ask first: *what* is an inductive bias?

The traditional definition of inductive bias is vague and leaves room for interpretation. For example, Caruana defined it as follows [16]:

“An inductive bias is anything that causes an inductive learner to *prefer* some hypotheses over others.”

In other words, an inductive bias is a (or, rather, any) mechanism that introduces some *external knowledge* in the model or its training process, such that it *biases* the optimization towards more preferable solutions.


Perfectly unbiased models. If we agree on qualifying DL as a general-purpose framework: why would we then want to add inductive *biases*? The truth is that, even if we disregard for a moment the weaknesses discussed in [Section 2.1](#), the idea of having a *truly* unbiased model sounds great at first, but it is actually undesirable. In words of T. M. Mitchell, an unbiased model renders ‘nearly useless’ in practice [133], as it needs to consider as plausible any hypothesis that agrees with the observational data, irrespectively of how exotic this hypothesis is.

[133] Mitchell (1980), ‘The need for biases in learning generalizations.’

Moreover, we should realize that there are no unbiased systems: *any conceivable algorithm makes assumptions*, either explicit or implicit, and biases our models towards a subset of solutions. One notorious example is the explicit use of the (piecewise) differentiable assumption, *i.e.*, in DL we always assume that the solution to our problem is a differentiable function, since the networks we construct are of such a kind. Another

example is the aforementioned implicit bias of SGD, which regularizes the network towards solutions with flatter, more uniform optima [190].

In other cases, it is unrealistic to guide the optimization towards preferred solutions solely with data. Expanding on the causality example from Section 2.1, it is well-known that the underlying causal system generating the input data cannot be learnt solely from observational data, but that stronger assumptions are indeed required. This result can be proved as a direct consequence of the results in Part III.

[190] Smith, Dherin, Barrett and De (2021), ‘On the Origin of Implicit Regularization in Stochastic Gradient Descent.’


2.2.1 Expectations and their specificity

Does this mean that, in order to make DL more efficient and trustworthy, we always need to sacrifice generality? No, or at least not to a large degree. For example, when we standardize a dataset, everyone would agree that we do not make the model using this dataset less general, yet by applying this transformation we ease the work of many common algorithms which *expect* the data to be standardized. As mentioned before, every algorithm and user places expectations, and it is our *awareness* about these that can serve us as sources of external knowledge to design effective inductive biases. More specifically, in this dissertation we classify expectations within the following levels of specificity:

Optimization-specific. Some expectations do not affect the generality of the methods in use, as they can be related to the algorithms we apply, and not to the problem at hand. For example, if we attempt to solve an **independent component analysis (ICA)** problem, a common expectation is that the data has been whitened beforehand and, in other cases, that the data has been centred as well. In this cases, however, these expectations can be easily met by preprocessing the input dataset.

In other cases, implementing an inductive bias is not as straight-forward. Take as an example the multitask setup presented in Section 2.1, where we had two tasks objectives that were not comparable. In that case, the underlying expectation that we did not meet is that the optimizer was able to compare both objectives, such that we could aggregate their gradient information in an informed manner and meet the expected trade-off solution. We explore different ways of incorporating this expectation as an inductive bias in Parts I and II of this dissertation.

Problem-specific. Sometimes, our expectations regard the type of solutions we seek within a problem family. These expectations involve the nature of the problem at hand, and typically exploit specific symmetries and structure present in any valid problem solution.

A famous example is that of translational symmetries in image classification, which gave birth to the architectural inductive bias that we now know as convolutional layers [101]. In this example, if we were to detect a cow in a given picture, the expectation we place is that the model should be invariant to the position of the cow, *i.e.*, it should produce the same output if the cow appears in the left- or right-side of the image.

Another example, this one introduced in Part III of this dissertation, is that of causal consistency in **causal generative models (CGMs)**. In this

[101] LeCun, Bottou, Bengio and Haffner (1998), ‘Gradient-based learning applied to document recognition.’


case, we know that any valid solution of the problem needs to be causally consistent *w.r.t.* the true causal system, *i.e.*, it needs not to exploit any spurious correlation. Similar to the case of convolutional networks, we implement this inductive bias by forcing the network structure to match the causal dependencies between variables.


Case-specific. Nevertheless, we often have specific expectations related to the current instance of the problem we are trying to solve. This is the most tailored knowledge that we can exploit and, while hurting the generality of the DL model for *other* problem instances, they can be *essential* if we want to solve the one instance we have.

A clear example, presented in [Section 2.1](#), is that of finding the correct trade-off solution.⁴ In said example, we saw that the model could select different solutions with identical loss sums, depending on the initial network initialization (recall [Figure 2.2](#)), and that it was us who should select which of them we preferred. Here, our *preference* on the trade-off to commit between the two objectives is a case-specific expectation, which can serve as an inductive bias to find solutions meeting this expectation. We will implement several of these biases in [Parts I and II](#), and denote their related external knowledge as *impartiality* expectations.

Other better-known examples can be found in the fairness literature, where it is usually possible to threshold the ‘amount of fairness’ required by the **decision maker (DM)** for that specific use-case [\[229\]](#), as it can entail a trade-off with other measures such as differential privacy [\[39\]](#).

4: This is, as we will see in [Part I](#), an instance of **multi-objective optimization (MOO)** where we select a specific scalarization function.

[\[229\]](#) Zafar, Valera, Gomez-Rodriguez and Gummadi (2017), ‘Fairness Constraints: Mechanisms for Fair Classification.’ 

[\[39\]](#) Cummings, Gupta, Kimpara and Morgenstern (2019), ‘On the Compatibility of Privacy and Fairness.’ 

2.2.2 Inductive bias taxonomy

At the beginning of this section, we defined an inductive bias as *anything*. How can we, then, classify different types of inductive biases? Instead of focusing on how these biases are *implemented*, in this dissertation we propose to focus on the way they *change the constrained optimization problem* we initially posed in [Equation 2.1](#). This way, we can get a better understanding on the effect that different inductive biases have on the optimization process. More specifically, in this dissertation we classify inductive biases as follows:

New objective. Sometimes, inductive biases can be as simple as changing the objective function we optimize. This new objective can enhance the optimization by incorporating our expectations into the problem formulation, helping us guide the optimization towards that part of the parameter space that contains more favourable local optima. Whilst this perspective is new, there are indeed many common examples of such inductive biases in the literature:

As we discuss later in [Chapter 4](#), **multitask learning (MTL)** was first introduced in 1993 as an inductive bias for learning related tasks better. Here, the intuition is that information from one task can be used by the model to learn another task and, to this end, the objective in [Equation 2.1](#) is substituted by the sum of the task losses, *i.e.*:

$$\mathbb{E}_{\mathbf{x} \sim P_{\mathbf{x}}} \left[L(\mathbf{x}, \boldsymbol{\theta}) \right] \quad \Rightarrow \quad \mathbb{E}_{\mathbf{x} \sim P_{\mathbf{x}}} \left[\sum_{k=1}^K L_k(\mathbf{y}_k, \mathbf{x}, \boldsymbol{\theta}) \right], \quad (2.6)$$

where \mathbf{y}_k is the target for the k -th task, as MTL is usually a supervised problem. One way of understanding this bias is to use a probabilistic perspective. If we consider the target of each task as a random variable, $\mathbf{y}_k \sim P_{\mathbf{y}_k}$, and the input \mathbf{x} as a common random covariate, $\mathbf{x} \sim P_{\mathbf{x}}$, then each objective above could be interpreted⁵ as the (negative) conditional log-likelihood of the target prediction given the input, *i.e.*,

$$L_k(\mathbf{y}_k, \mathbf{x}, \theta) \equiv -\log p_{\theta}(\mathbf{y}_k | \mathbf{x}). \quad (2.7)$$

If we agree on this interpretation, then the sum of task losses can be seen as the (negative) *joint* likelihood of the targets given the input, since⁶

$$\log p_{\theta}(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_K) = \mathbb{E}_{\mathbf{x} \sim P_{\mathbf{x}}} \left[\sum_{k=1}^K \log p_{\theta}(\mathbf{y}_k | \mathbf{x}) \right], \quad (2.8)$$

which is equivalent to minus Equation 2.6 given Equation 2.7. From this perspective, solving the MTL problem is equivalent to learning a joint distribution, while solving K individual problems is equivalent to learning K marginal distributions. Clearly, the former is a more expressive objective, as it takes into account all the dependencies between target variables, and exploits this information to make better predictions.

A more tangible example can be adapted from the partial information decomposition literature [71], which studies how information can be decomposed in three types: **i)** exclusive information, which solely describes one variable; **ii)** redundant information, which appears in several variables; and **iii)** synergetic information, which can *only* be obtained by jointly observing the variables. For this example, assume a toy experiment where we have two binary targets, whose XOR output works as our input \mathbf{x} , as described in Table 2.1. In this case, the best we could do to predict each \mathbf{y}_k individually would be to predict zero or one at random. However, if we predict both targets simultaneously, we know that both should be equal if \mathbf{x} is zero, and that they should be opposite otherwise. In other words, the *synergetic information* of the target variables helps the model make better prediction when *jointly* predicted.

A different example, this time from the fairness literature, is that of class resampling [87]. Here, we assume that there exists a mismatch between the distribution that we are interested in learning and the observed one, which is imbalanced across classes (*e.g.*, male *v.s.* females). In class resampling, the population is divided by classes (*i.e.*, stratified), and the frequency at which a member of each population is sampled is equalized, so that all classes are equally represented in the dataset. In mathematical terms, if we could write the initial population as a mixture, *i.e.*, $p(\mathbf{x}) = \sum_{d=1}^D \alpha_d p_d(\mathbf{x})$ with $\sum_d \alpha_d = 1$, then the re-balanced population would have a density of the form $\tilde{p}(\mathbf{x}) = \frac{1}{D} \sum_{d=1}^D p_d(\mathbf{x})$, and we would re-write the initial objective of Equation 2.1 as

$$\mathbb{E}_{\mathbf{x} \sim P_{\mathbf{x}}} [L(\mathbf{x}, \theta)] \implies \mathbb{E}_{\mathbf{x} \sim \tilde{P}_{\mathbf{x}}} [L(\mathbf{x}, \theta)], \quad (2.9)$$

where we change the objective by changing the data distribution.

Hard constraints. Inductive biases can also encode expectations regarding the constraints g in Equation 2.2 that the model should fulfil.

5: With a bit of care, we gloss over the details here for brevity.

6: We assume here a mean-field factorization of the targets given \mathbf{x} .


[71] Gutknecht, Wibral and Makkeh (2021), ‘Bits and pieces: understanding information decomposition from part-whole relationships and formal logic.’

Table 2.1: Truth table that the input variable and targets of the XOR example can take. See accompanying text.

\mathbf{y}_1	\mathbf{y}_2	\mathbf{x}
0	0	0
0	1	1
1	0	1
1	1	0

[87] Kamiran and Calders (2012), ‘Data preprocessing techniques for classification without discrimination.’

Depending on the specific case, we sometimes can design inductive biases that help the optimization completely disregard those solutions that do not meet these expectations, typically through the network design.

[2] Ahmed, Teso, Chang, Broeck and Vergari (2022), ‘Semantic Probabilistic Layers for Neuro-Symbolic Learning.’ 

One example from the probabilistic ML literature is that of **semantic probabilistic layers (SPLs)** [2]. In this setting, the SPL encodes domain-specific constraints as a density defined over the points that fulfil these constraints, $\delta_{\text{SPL}}(\mathbf{x}, \mathbf{y})$, which multiplies the conditional distribution learned by the DL model, $p_{\theta}(\mathbf{y} \mid \mathbf{x})$, resulting in a probabilistic model whose domain is by design those points that satisfy the constraints described by the user, *i.e.*, $\tilde{p}_{\theta}(\mathbf{y} \mid \mathbf{x}) \propto \delta_{\text{SPL}}(\mathbf{x}, \mathbf{y}) \cdot p_{\theta}(\mathbf{y} \mid \mathbf{x})$.

[101] LeCun, Bottou, Bengio and Haffner (1998), ‘Gradient-based learning applied to document recognition.’


Two other clear examples are the architectural inductive biases related to problem-specific expectations that we introduced in the examples from the previous subsection. In the case of translational invariance in image classification, convolutional neural networks [101] were introduced as architectures which, *by design*, will always be translation invariant. Similarly, in **Part III** we introduce **causal normalizing flows (Causal NFs)** whose design is such that they will always be causally consistent with respect to the provided causal graph, *i.e.*, they will meet our causal expectations and have no spurious correlations. In short, this is done by ensuring that the information flows through the network the same way as we expect it to flow, by dropping all connections which could potentially break this constraint.


Soft constraints. Other times, implementing inductive biases that enforce our expectations to be always satisfied is not as straight-forward. We can, however, design biases that guide the optimization towards those solutions that meet our expectations, by providing information during training of what parameters directions to follow.

This is the case for any constraint that is introduced into the optimization pipeline using Lagrange multipliers, *i.e.*, by re-writing the problem in **Equation 2.1** as

$$\mathbb{E}_{\mathbf{x} \sim P_{\mathbf{x}}} [L(\mathbf{x}, \theta)] \implies \mathbb{E}_{\mathbf{x} \sim P_{\mathbf{x}}} [L(\mathbf{x}, \theta) + \lambda g(\mathbf{x}, \theta)], \quad (2.10)$$

with λ being a positive value that regulates the ‘importance’ we give to the model meeting our expectations. This type of inductive biases are ubiquitous in all ML research, *e.g.*, to regularize the model to find sparse solutions [137, Chapter 11.4], but also appear in more concrete scenarios such as, *e.g.*, for introducing fairness constraints to DL models [229].

[137] Murphy (2022), ‘Probabilistic Machine Learning: An introduction.’ 

[229] Zafar, Valera, Gomez-Rodriguez and Gummadi (2017), ‘Fairness Constraints: Mechanisms for Fair Classification.’ 

A less typical example is that of gradient manipulation in MTL, which we cover in detail in **Part I** of this thesis. Taking again the example from **Section 2.1**, where we had two tasks to learn, and where we wanted the model to meet our expectations in terms of the trade-off between tasks,⁷ these algorithms work by taking the vector of tasks gradients, and aggregating them according to the user expectations. That is, the update rule in **Equation 2.3** is re-written to be of the form

7: In this particular case, that both tasks were equally important.

$$\theta^{t+1} = \theta^t - \alpha^t \underset{\text{gradient-manipulation algorithm}}{\overset{\text{vector of task gradients}}{f}}([g_1 \ g_2]), \quad (2.11)$$

where f is a predetermined algorithm. For example, in the case of RotoGrad (introduced in [Chapter 5](#)), f is a function that equalizes the gradients in magnitude by re-scaling them in each iteration, while slowly rotating them during training to match their directions as well. While these algorithms do not ensure that the solution found will meet the given expectations in terms of their trade-off, they make the optimization process move towards solutions that we expect are closer to meeting our trade-off expectations.

Thesis Outline and Contributions

3.

Yugos os quieren poner
gentes de la hierba mala,
yugos que habéis de dejar
rotos sobre sus espaldas.

Miguel Hernández

3.1 Outline 17

3.2 Contributions 17

3.1 Outline

This dissertation is organized in three main parts. The first part is devoted to **multitask learning (MTL)** and the problem of negative transfer. The second, to **probabilistic generative models (PGMs)** and the problem of modality collapse. The third part is instead devoted to the theoretical and practical design of effective **causal generative models (CGMs)**. One last part concludes the dissertation and outlines future research directions that build on the results and insights presented here. Every part starts with an introductory section introducing all the relevant background.

For the curious reader, all the chapters contain plenty of complementary remarks, side notes, figures, and references on the page margins, that supplement the content of the main text. Any other detail omitted in the main text can be found in the appendices of the manuscript and are properly referenced when necessary.

3.2 Main contributions

In this section, we briefly describe the main contributions presented in this dissertation. The most general contributions follow the same ideas and arguments described in the previous two chapters, attempting to formalize and make explicit different ideas in **deep learning (DL)**, as well as expectations that we place on models and assume to universally hold, even if that is not the case in practice. Through this careful formalization, we are then able to develop more specific contributions built on solid foundations for each considered framework. Namely, the main contributions of this dissertation can be summarized as follows:

Part I - MTL. We provide in **Chapter 4** a thorough introduction to MTL, its original motivation as an inductive bias, and the problem of negative transfer, which materializes as gradient conflict issues during training. More importantly, we identify two fundamental problems in the MTL methodology. First, we stress the importance of having a clear scalarization function to optimize *a priori*, *i.e.*, to know the trade-off solution that we expect the model to achieve. This is especially important in the trending interpretation of MTL as a **multi-objective optimization (MOO)** problem. Within this context, we question the utility and reliability of seeking Pareto optimal solutions, given the impossibility to test their

optimality in practice. Second, we unveil the oft-implicit expectation of task comparability, *i.e.*, that the optimization algorithm can reasonably compare quantities from different tasks, and attribute many issues in MTL to this expectation not being met in practice.

In light of these fundamental issues, we introduce in [Chapter 5](#) RotoGrad, a gradient-conflict method that simultaneously homogenizes gradients across tasks both in direction and magnitude, guiding the optimization towards task-impartial solutions, *i.e.*, solutions where all tasks are equally important. RotoGrad is composed of two complementary sub-routines for each type of gradient conflict, and ensures convergence by interpreting the training process as a differentiable Stackelberg game. [Chapter 6](#) focuses instead on methodological problems of MTL and, more specifically, in providing metrics that account for the incomparability of quantities across tasks. To this end, we motivate the use of ranking statistics to compare MTL models, interpreting them as finite-sample approximations of MOO global methods over [cumulative distribution functions \(CDFs\)](#), overcoming all comparability issues. Second, we provide metrics to measure gradient conflict and, using these new tools, show how to probe existing MTL benchmarks to better suit them for testing MTL methods.

Part II - PGMs. The main contribution of this part is the strong connection that we draw between multimodal PGMs and MTL. We start building these connections in [Chapter 7](#), where we present the problem of modality collapse which, just like negative transfer, arises mainly as a result of the incomparability between modalities. Again, we as users place expectations on the models regarding the trade-off between modalities, and in this thesis we introduce the concept of task impartiality as a specific trade-off to pursue. Unlike in MTL, we note that additional expectations are placed on PGMs as a natural consequence of the different tasks they can solve, as we briefly introduced already in [Section 1.2](#).

We start exploiting these connections in [Chapter 8](#), where we propose Lipschitz-standardization, an optimization-inspired preprocessing algorithm that attempts to equalize the optimization landscape of each modality, such that they enjoy similar convergence guarantees. While improving training robustness, preprocessing algorithms can only go thus far. Then, we propose in [Chapter 9](#) to leverage existing MTL algorithms for gradient conflict and, to this end, we introduce the concept of impartiality blocks, which are subsets of the network’s computational graph that are prone to suffer from modality collapse. Crucially, the presented methodology expands on the usual MTL case as it can be applied to non-linear loss functions and to several impartiality blocks simultaneously, as we show that complex multimodal PGMs have several impartiality blocks due to their additional expectations. We empirically demonstrate that the proposed methodology improves the robustness and impartiality of different PGMs in a wide range of scenarios.

Part III - CGMs. We then go one step further, and attempt to learn the underlying causal data-generating process producing our data, instead of their joint distribution. That is, our goal is to have a single DL model that we can reliably use to perform causal inference, *i.e.*, to answer *what-if* queries regarding the modelled generating process. To this end, we introduce all relevant causality concepts to the reader in [Chapter 10](#),

study the problem theoretically in [Chapter 11](#), and apply the resulting knowledge to effectively design CGMs in [Chapter 12](#).

More specifically, the main contribution of [Chapter 11](#) is to find a family of generating processes, [TMI SCMs](#), that it is causally identifiable, meaning that we can recover the one member that generated the data we observe, up to a tolerable error. We do this under quite common assumptions,¹ and simply requiring the causal ordering between variables as an external input. Also crucially, we prove that any other model meeting our assumptions can be reduced to an equivalent member of the TMI SCM family, greatly increasing the applicability of our results. Finally, we provide an alternative implementation of the do-operator for our family of models, enabling their use for causal inference tasks, and extend our results to handle mixed-type data as well as partial causal knowledge.

1: Namely, having bijective generators, no cycles, and no hidden confounders.

We take these theoretical results and put them into practice in [Chapter 12](#). By realizing that [autoregressive normalizing flows \(ANFs\)](#) are members of the TMI SCM family and universal density approximators, we introduce [causal normalizing flows \(Causal NFs\)](#), DL models that can provably learn causal data-generating processes and accurately perform causal inference. In practice, however, it is difficult for Causal NFs to meet their causal expectations solely from data. For this reason, we study different model parametrizations and conclude that, if we have access to the causal graph, we can design Causal NFs that are causally consistent by design, *i.e.*, that they meet our causal expectations. As a result, Causal NFs are the first-of-their-kind models to provide theoretical guarantees in such a broad family of causal models, which we empirically validate by showing that they accurately model the observational, interventional, and counterfactual distributions of a wide variety (*e.g.*, linear, non-linear, synthetic, and semi-synthetic) causal models.

Part IV - Epilogue. We conclude this dissertation by summarizing the main results introduced herein, as well as presenting the initial impact of the works published alongside this thesis. Finally, we provide a short discussion on the high-level topics that this dissertation has gone over, sharing with the reader our prospect on open questions and research directions that we consider important researching in the future.

Part I.

Multitask Learning



Introduction to Multitask Learning

4.

Venceréis, pero no convenceréis.

Miguel de Unamuno, 1936

Multitask learning (MTL) aims at learning several tasks with a single model, amortizing parameters, and using common information about tasks to boost its performance. In the context of this thesis, it serves as the initial grounds to study the interactions between inductive biases and **machine learning (ML)** optimization. In this chapter, we briefly introduce the most relevant aspects of MTL, as well as a historical overview of the field to better contextualize this part of the dissertation.

4.1 Historical overview


To look back at the origins of MTL, we need to go as back as 1993 to the pioneering work of Caruana [16]. Back then, ML research followed a *reductionist perspective*, in which large problems were first reduced into smaller separated pieces that could be solved and then recombined. As a result, the main focus was to train models that could first solve a single task satisfactorily¹ before considering more complex scenarios. Caruana challenged this methodology, and introduced MTL under the following premise:

“If an inductive learner is given several related tasks at the same time, these tasks can be used as valuable sources of inductive bias for each other. This may make learning faster or more accurate, and may allow hard tasks to be learned that are not learnable in isolation.”


The work of Caruana had a lasting impact on the ML community, and nowadays MTL is considered a first-class citizen within ML. Moreover, MTL has experienced notorious success in a number of domains such as computer vision [69, 115], robotics [227], and natural language processing [207, 208], to name a few. To this day, perhaps the best example of such success is GATO [162], a generalist MTL agent that ‘can play Atari, caption images, chat, stack blocks with a real robot arm and much more, deciding based on its context whether to output text, joint torques, button presses, or other tokens.’


Intuitively, what characterizes and differentiates MTL from other approaches in ML is the very notion of *tasks biasing tasks*, which has more recently being popularized under the term *information transfer* [214]. We discussed this idea in Section 2.2, and interpret it as an inductive bias re-defining our objective function where, by adding all task losses together, we introduce additional information about joint dependencies between tasks. However, despite the success of MTL, we should note that this transfer of information does not necessarily translate into improved performance and, depending on whether the effect of this bias improved

4.1 Historical overview	23
4.2 Problem statement	24
4.3 Network architectures . . .	25
4.4 MTL as MOO	26
4.5 Task impartiality	26
4.6 Gradient conflict	28

[16] Caruana (1993), ‘Multitask Learning: A Knowledge-Based Source of Inductive Bias.’ 

1: That is, **single task learning (STL)**.

[162] Reed, Zolna, Parisotto, Colmenarejo, Novikov, Barth-maroon, Giménez, Sulsky, Kay, Springenberg, Eccles, Bruce, Razavi, Edwards, Heess, Chen, Hadsell, Vinyals, Bordbar and Freitas (2022), ‘A Generalist Agent.’ 

[214] Wu, Zhang and Ré (2020), ‘Understanding and Improving Information Transfer in Multi-Task Learning.’ 

or degraded the performance of the model with respect to a baseline, information transfer is further particularized as *positive transfer* or *negative transfer*, respectively.

Therefore, it should come as no surprise that a significant amount of the effort in MTL research has focused on ways of promoting positive transfer and, consequently, on ways of discouraging and alleviating the effects of negative transfer. While we will focus on specific approaches later, here we provide a brief taxonomy of the types of approaches developed over the years to improve information transfer:

- **Task clustering.** These approaches follow the simple idea of grouping those tasks that ‘work well together’ while keeping those that interfere with each other apart,² and where the gist is on providing a good notion of *task similarity* that captures information transfer well. This idea dates back as much as of 1996 with the task clustering algorithm [201], but the field is still active and new approaches are proposed each year [59, 118, 149, 184, 193, 232].
- **Adaptive architectures.** Instead of restricting which tasks are put together, here the idea is to use a single architecture with some selection mechanism, such that it can adaptively adjust which parameters—and thus, which inductive biases—are shared or exclusive across different tasks [115, 131, 195]. This approach is further explained in Section 4.3.
- **Optimization algorithms.** These methods look at information transfer from an optimization standpoint, *i.e.*, they consider it a result of the interaction between task gradients during training. Consequently, these approaches propose variations on the optimization process—usually by explicitly altering each task gradient—such that the training converges towards solutions that exhibit positive transfer. We cover these methods in detail in Section 4.6 and present one in Chapter 5.

Despite the success of MTL, and its consolidation as a field of ML, there exists still a significant gap between the empirical success of MTL and our understanding of information transfer. There has recently been voices among the community questioning how effective MTL approaches really are [99, 106, 167, 220] and, as we will see in Chapter 6, inadequate benchmarking and the lack of metrics can account for quite a fair bit of this criticism. In said chapter, we focus on **task incomparability** as a cause of negative transfer, where information coming from heterogeneous sources cannot be *a priori* meaningfully aggregated.

4.2 Problem statement

In this thesis, we restrict our MTL setting to that where all K tasks share the same input features.³ Specifically, we consider an input dataset X composed of N *i.i.d.* samples from a D -dimensional random vector $\mathbf{x} \in \mathbb{R}^D$ with distribution $P_{\mathbf{x}}$, *i.e.*, $X = \{\mathbf{x}_n\}_{n=1}^N \stackrel{i.i.d.}{\sim} P_{\mathbf{x}}$. For each sample \mathbf{x}_n , we consider a K -dimensional vector of targets that we aim to predict, $\mathbf{y}_n = [\mathbf{y}_1 \ \mathbf{y}_2 \ \dots \ \mathbf{y}_K]$, one per task. More compactly, we have a target dataset $Y = \{\mathbf{y}_n\}_{n=1}^N \subset \mathbb{R}^K$.

2: As we want to study the interaction between tasks, our focus on this thesis will be on other approaches.

[201] Thrun and O’Sullivan (1996), ‘Discovering Structure in Multiple Learning Tasks: The TC Algorithm.’

3: The same assumption appears in Caruana’s thesis [17]. As noted there, this is without loss of generality, as one could always take the concatenation of each task’s input features as the shared input.

Together with the targets, for each task we also have a task-specific loss $L_k : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ which evaluates how close the model prediction, \tilde{y}_k , is to the actual solution, y_k . Without loss of generality, we assume that our objective is to minimize the output of L_k .⁴ With a slight abuse of notation, we will write $L_k(\tilde{y}_k, y_k)$ to refer to the loss of the k -th task on a single element of the dataset, and with

$$L_k(\tilde{Y}_k, Y_k) := \frac{1}{N} \sum_{n=1}^N L_k(\tilde{y}_{nk}, y_{nk}) \quad (4.1)$$

to the sample average loss of the k -th task.

We therefore have a vector of K losses that we want to minimize. However, minimizing a vector is not well-defined and, while we will discuss later in [Section 4.4](#) the implications of this vector-minimization problem, it is a widespread solution to minimize a linear combination of the task losses [182], *i.e.*,

$$L(\tilde{Y}, Y) := \sum_{k=1}^K L_k(\tilde{Y}_k, Y_k). \quad (4.2)$$

4.3 Network architectures

Naturally, the neural-network architecture used to learn multiple tasks plays an important role on how well we can capture these tasks and, inherently, on the information transfer that the model experiences during training. In order to provide some context, here we present a broad taxonomy⁵ of the different classes of MTL architectures that can be found in the literature.

Within the network, we distinguish in general two types of components: **i) backbones**, which are parametrized by a set of common parameters and produce a shared intermediate representation, $z := f_\theta(x)$; and **ii) heads**, which have task-exclusive parameters and transform the intermediate representation into the prediction for their task, $y_k := h_{\phi_k}(z)$. In essence, different architectures differ in the way that they define these shared parameters, θ , and the extent of what sharing means.

We distinguish the following types of architecture:

- **Hard parameter-sharing.** Despite being the original architecture proposed by Caruana in 1993 [16], it is still the *de-facto* MTL architecture in the literature [169] and the one we will focus on. As depicted in [Figure 4.1](#), in hard parameter-sharing all tasks share the same backbone network, and each task has a specific head to make predictions based on z . As a result, information transfer occurs exclusively in the backbone, and z is identical for every task, thus behaving as the *last shared representation* that contains all the necessary information to predict all tasks. That is, z behaves as a *bottleneck* of all the predictive information.
- **Soft parameter-sharing.** Instead of completely sharing the backbone, in soft parameter-sharing we have one backbone per task, f_{θ_k} . To encourage information transfer, the most common approach is to consider all the backbone parameters as a single tensor, $[\theta_1 \ \theta_2 \ \dots \ \theta_K]$, and place some type of structural constraint,

4: Note that $\max f(x) = \min -f(x)$.

5: The taxonomy is an adaption of that from [169] Ruder (2017), ‘An Overview of Multi-Task Learning in Deep Neural Networks.’

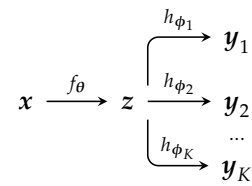


Figure 4.1: Hard parameter-sharing MTL architecture consisting of a shared backbone and K task-specific heads.

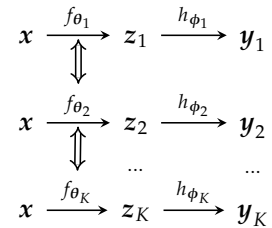


Figure 4.2: Soft parameter-sharing MTL architecture with K backbones and task-specific heads. Double-arrows represent soft constraints that relate backbone parameters across tasks.

6: Namely, linear in the number of tasks.

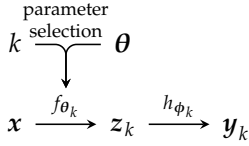


Figure 4.3: Adaptive MTL architecture with a single backbone whose parameters are selected from a shared set of parameters, θ , the task index, and a architecture-specific selection algorithm.

7: This alternative formulation was first proposed in the context of MTL by Sener and Koltun in 2018 [182].

e.g., a particular matrix factorization [98, 165, 225], or to apply some regularization to the tensor via, *e.g.*, the l_2 -norm [49], or the trace norm [226]. While soft parameter-sharing offers more flexibility—parameters across tasks are allowed to differ to some extent—the increase in number of parameters is significant,⁶ which could explain why they are not as widely-used as their counterparts.

- **Adaptive architectures.** Last, we have networks which do not define *a priori* which backbone parameters will be used by each task. Instead, they learn some sort of ‘parameter selection’ algorithm that distills a set of task-specific backbone parameters for each task, θ_k . Many selection algorithms have been proposed in the literature using, *e.g.*, reinforcement learning [195], branching algorithms [69], neural networks [3, 131, 194], or attention-based networks [115]. While we abstractly depict these networks in Figure 4.3, note that there is no set rule on how to define them and, *e.g.*, Long et al. [118] proposed a shared backbone with soft parameter-sharing heads.

4.4 MTL as multi-objective optimization

In Section 4.2, we mentioned that the most widespread training loss in MTL is a weighted sum of the task losses, *i.e.*, $L(\tilde{Y}, Y)$ in Equation 4.2. However, this choice of loss function is rather arbitrary, since there is nothing in the conception of MTL (*i.e.*, solving K tasks simultaneously) suggesting how to combine the tasks losses.

An alternative approach is to strip away this implicit choice for the loss function, and instead formulate MTL as a more general **multi-objective optimization (MOO)** problem.⁷ From this perspective, MTL attempts to solve the following optimization problem:

$$\min \quad L(\tilde{Y}, Y) := [L_1(\tilde{Y}_1, Y_1) \quad L_2(\tilde{Y}_2, Y_2) \quad \dots \quad L_K(\tilde{Y}_K, Y_K)], \quad (4.3)$$

where the outcome $L(\tilde{Y}, Y) \in \mathbb{R}^K$ is now a *vector* of K losses.

Unfortunately, the problem in Equation 4.3 is undefined without further assumptions. While the real numbers with the usual order (\mathbb{R}, \leq) is a totally ordered vector space, its K -dimensional extension with the product order, *i.e.*, (\mathbb{R}^K, \leq^K) where $\mathbf{a} \leq^K \mathbf{b}$ iff $a_i \leq b_i \forall i$, forms just a partially ordered vector space. As a result, two elements in \mathbb{R}^K *may not be comparable*, and thus we cannot choose between a pair of outcomes.

Two important and related notions in MOO are those of dominance and Pareto optimality. We call the set of Pareto-optimal outcomes the Pareto front, which represents the set of best possible outcomes. *The goal of MOO is to achieve Pareto optimality, i.e.*, to reach an element in the Pareto front.

4.5 Task impartiality

While mathematically appealing, the Pareto front, *i.e.*, the (possibly infinite) set of all trade-off solutions of the MOO problem in Equation 4.3, is not that useful from a practitioner perspective: at some point in the

training-and-deployment process, *we need to select a specific model to use*, hopefully within the Pareto front and with the advice of a **decision maker (DM)**, who knows which trade-offs they are willing to make.

MOO has been extensively studied in diverse research areas, and thus a considerable amount of *model selection* approaches (*i.e.*, those that select a model within the Pareto set) have been considered.⁸ All of them, however, come down to defining a way of imposing a total order across outcomes, either by **i)** transforming \mathbb{R}^K to \mathbb{R} with the usual order \leq ; or by **ii)** replacing \leq^K with a total order for \mathbb{R}^K . Some important approaches within the literature are the following:

- **No-preference methods** expect the absence of a DM or preference information, and thus look for a compromise solution *impartial* to any loss function. For example, the global criterion method [231] selects the outcome closest to the ideal solution,

$$\min \|L(\tilde{Y}, Y) - L^{\text{ideal}}\|_p, \quad (4.4)$$

where $p \geq 1$ defines a p-norm, and L^{ideal} is the best possible outcome for each individual objective, *i.e.*,

$$L^{\text{ideal}} := [\min L_1 \quad \min L_2 \quad \dots \quad \min L_K], \quad (4.5)$$

and where we have removed the arguments of each loss function to avoid clutter. However, Equation 4.4 is sensitive to the scaling of the loss functions [130, Chapter 2.1], as we will discuss in Chapter 6.

- **Scalarization methods** assume the existence of a *scalarization function*, $\psi : \mathbb{R}^K \rightarrow \mathbb{R}$, that maps the vector outcome to a scalar value, and select the model that solves $\min \psi(L(\tilde{Y}, Y))$. Of special importance is linear-scalarization, where ψ is a linear combination of the outcome elements, weighted by a set of weights, $\psi(L) = \sum_{k=1}^K \alpha_k L_k$. Linear scalarization is broadly used in practice,⁹ and enjoys important theoretical results. For example, it is known [14] that any (static) linear combination of outcomes can only reach the elements of the Pareto front that intersect with its convex hull. In the case where the set of achievable outcomes is convex, linear scalarization can reach any Pareto optimal outcome.
- **A posteriori methods** take a different approach, and attempt to recover the entire Pareto front, which is afterwards presented to the DM to select the preferred outcome. While we focus on other methods in this thesis, it is worth-noting that there have been efforts within the **deep learning (DL)** community to provide *a posteriori* solutions [107, 108, 121, 168].

The main take-away from the paragraphs above is that, independently of how we choose it, having a clear scalarization function is vital. In other words, while the goal of MOO theory is to reach a Pareto-optimal outcome, *for all practical purposes we need to have a clear image of the objective we are trying to achieve* and, to this end, we must induce a total ordering between the outcomes through a scalarization function.

This is specially important in the interpretation of MTL as a MOO problem (Section 4.4). Under the premise of having foundations in MOO theory, a recent trend in MTL literature is to propose approaches that reach Pareto-stationarity, without specifying any scalarization function

8: For a more in-depth review, refer to [130] Miettinen (1999), ‘Nonlinear multiobjective optimization.’


[130] Miettinen (1999), ‘Nonlinear multiobjective optimization.’

9: Indeed, we introduced the objective of MTL as a linearly scalarized problem in Equation 4.2.

[14] Boyd and Vandenberghe (2004), ‘Convex optimization.’

to follow [27, 110, 111, 114, 227]. As a result, their utility and reliability are questionable, as *any outcome trade-off achieved could be a valid solution, unless we specifically find a dominant outcome*, making it hard to test in practice.

In this thesis, we adopt **task impartiality** as our objective. Tightly related to no-preference methods, task impartiality assumes the absence of any preference towards learning one task over another, and thus looks for a compromise solution that performs well on all tasks without overlooking any of them. While still loosely defined, task impartiality provides a clear objective to achieve, and thus to evaluate and compare MTL approaches. Task impartiality was first mentioned by Liu et al. [112] and, in this thesis, we provide an MTL approach to encourage task impartiality in **Chapter 5**, a way of measuring task-impartiality performance in **Chapter 6**, and we extend the concept of impartial solutions to the context of **probabilistic generative models (PGMs)** in **Part II**.


[112] Liu, Li, Kuang, Xue, Chen, Yang, Liao and Zhang (2021), ‘Towards Impartial Multi-task Learning.’ 

4.6 Gradient conflict

4.6.1 Existing approaches . . . 29

One important concept in this thesis is that of **gradient conflict** which, in short, describes the negative effects that considering several tasks simultaneously can have on the total gradient computation, making it a bad direction to update the network parameters. Again, the seminal work of Caruana [16] provided many relevant insights ahead of its time. Specifically, Caruana said back in 1993:

“Because the MTL net is doing several potentially competing jobs at the same time, the gradients computed from the error signal for each task may interfere (*i.e.*, cause the aggregate gradient to be flatter or less directional), so learning might be slower for some MTL tasks.”

[16] Caruana (1993), ‘Multitask Learning: A Knowledge-Based Source of Inductive Bias.’ 

10: See **Section 4.1** for a quick description of these other MTL solutions.

However, most of the efforts in MTL research did go towards better task clustering and more sophisticated architectures.¹⁰ Starting in 2018 with the works of Chen et al. [26], Kendall et al. [89], Sener and Koltun [182] and Sinha et al. [189], the MTL community started to focus on the interaction between gradient coming from different tasks during training, and the interest on alleviating gradient conflict has increased since then [21, 27, 110–112, 114, 142, 183, 211, 227]. These solutions, which modify the gradient direction to follow, can be interpreted as inductive biases that drive the optimizer towards certain solutions that meet our trade-off expectations, as discussed in **Section 2.2**.

For ease of exposition, let us focus here on the linearly scalarized MTL loss in **Equation 4.2**, and suppose we use a hard parameter-sharing architecture as in **Figure 4.1**, so that all tasks share the parameters θ from the backbone.¹¹ Moreover, since the gradient is a linear function, we can quite easily compute the gradient of the MTL loss *w.r.t.* θ :

11: We cover non-linear scalarization and complex architectures in **Chapter 9**.

$$\nabla_{\theta} L(\mathbf{Y}) = \sum_{k=1}^K \nabla_{\theta} L_k(\tilde{\mathbf{Y}}_k, \mathbf{Y}_k), \quad \text{where} \quad \tilde{\mathbf{Y}}_k := h_{\phi_k}(f_{\theta}(\mathbf{X})). \quad (4.6)$$

From the expression above, it is clear how conflicts between the different task gradients can occur: if there exist big disparities between task

gradients, then the sum can be dominated by a subset of tasks, and these tasks can take over the backbone parameters, θ .

To avoid clutter, let us use \mathbf{g}_k to refer to the gradient *w.r.t.* θ of the k -th task, $\mathbf{g}_k := \nabla_{\theta} L_k(\tilde{\mathbf{Y}}_k, \mathbf{Y}_k)$, and similarly to the total gradient *w.r.t.* θ as $\mathbf{g} := \nabla_{\theta} L(\mathbf{Y})$. Given that \mathbf{g}_k is a $|\theta|$ -dimensional vector, a taxonomy for gradient conflict naturally arises from the vectorial nature of \mathbf{g}_k :

- **Magnitude conflict.** If two gradients differ significantly in their norm, *i.e.*, if $\|\mathbf{g}_i\| \gg \|\mathbf{g}_j\|$ for two different tasks, then the task with bigger magnitude will dominate the total gradient computation and, as a result, the network will prioritize learning the dominant task. While the magnitude of a task gradient in isolation contains useful information about the progress in the optimization of that task, in general tasks are not comparable, and thus their magnitude should not be comparable either, as we will see in [Chapter 6](#).
- **Direction conflict.** Even if two task gradients have equal magnitude, their direction could still be completely different. Therefore, when computing the total gradient in [Equation 4.6](#), their sum could cancel out, and the network explore potentially poor update directions that deteriorate model performance. Intuitively, the direction of \mathbf{g}_k indicates where to explore to immediately improve the performance of the k -th task and, therefore, direction conflict can be understood as the disagreement of two tasks on which parts of the parameter space to explore. However, the direction of \mathbf{g}_k can be highly influenced by the parameters of the task heads, as we will describe and exploit later in [Chapter 5](#).

It is worth noting that other sources of conflict have also been pointed out in the previous literature. For example, Yu et al. [227] considered the curvature of the optimization landscape. However, the two sources above are the most common ones, and to this date little work exists exploring second-order information of gradient conflict in MTL.

4.6.1 Existing approaches

As we have mentioned above, the number of approaches to tackle gradient conflict have significantly increased since 2018 in the context of MTL. In order to get a better understanding of the type of approaches considered in the existing literature, here we provide a non-exhaustive taxonomy of methods to alleviate gradient conflict:

- **Magnitude-aware.** These algorithms seek a set of task weights $\omega := \{\omega_k\}_{k=1}^K \in \mathbb{R}^K$ to re-scale each gradient, $\omega_k \cdot \mathbf{g}_k$. Moreover, these weights are typically constrained to add up to one, $\sum_k \omega_k = 1$. We call them magnitude-aware since scaling \mathbf{g}_k only changes its norm and, therefore, only immediately address magnitude conflict.¹² Examples of these approaches are:
 - GradNorm [26], which treats the weights as parameters to optimize along the network parameters, so that they equalize the gradient magnitudes over training. To reduce overhead, GradNorm considers the gradients *w.r.t.* the last-shared layer.
 - MGDA seeks for a convex combination of weights, such that the norm of the total gradient is minimized. Its dual and primal

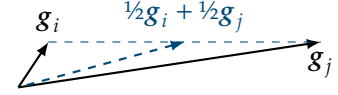


Figure 4.4: Pictorial description of magnitude gradient conflict.

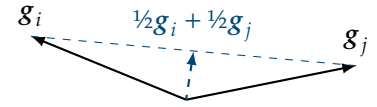


Figure 4.5: Pictorial description of direction gradient conflict.

12: Note, however, that they can also alleviate direction conflict over time.

formulations were first introduced by Fliege and Svaiter [61] in 2000, and by Schäffler et al. [178] in 2002, respectively, and later reintroduced by Désidéri [44] in 2012. In 2018, Sener and Koltun [182] adapted it to MTL, under the name of MGDA-UB, by applying the algorithm to the gradients *w.r.t.* the shared representation, \mathbf{z} , rather than *w.r.t.* the shared parameters, θ . Moreover, generalizations of MGDA such as CAGrad [111] and, more recently, FAMO [110], have been proposed to better deal with the specifics of optimizing MTL architectures.

- IMTL-G [112] finds the task weights under a closed-form solution, such that the projection of the total gradient to each of the task gradient is equal. Similar to MGDA-UB, IMTL-G is applied to the gradients *w.r.t.* \mathbf{z} .
 - Nash-MTL [142] interprets the problem of gradient conflict as a Nash game, and looks for a set of weights such that they achieve a Nash equilibrium. AuxiNash [183] extends Nash-MTL to the case where we seek an asymmetric Nash equilibrium, *i.e.*, where we treat some tasks as auxiliary tasks.
 - RGW [106] proposes to randomly sample the task weights according to a given distribution, and then renormalize them so that they add up to one. The intuition here is that the random weighting can serve as a regularization for the network to not be dominated by a subset of the tasks.
- **Direction-aware.** This type comprehends all other methods in which task gradients are modified in a way other than scaling, such that their directions are explicitly modified. For example:
1. GradDrop [27] drops individual elements of the task gradients randomly, according to how much each task gradient conflicts in direction with the total gradient. Here, the intuition is that GradDrop makes each task gradient to self-correct, and thus to align with the rest of task gradients.
 2. PCGrad [227] is a popular method due to its simplicity. In short, if there exists some direction conflict between two tasks, PCGrad selects one randomly, and removes its conflicting part by computing the projection of this gradient onto the other, *i.e.*, it removes the orthogonal component that would cancel out when computing the sum of both task gradients. More recently, GradVac [208] was proposed as a generalization of PCGrad which, even if there is no direction conflict, it modifies the gradients to encourage positive transfer.

The list above is an incomplete but representative view of the type of approaches proposed to deal with gradient conflict in the MTL literature. Other interesting works have considered the use of, *e.g.*, adversarial training [189], or meta-learning [114]. Moreover, in Chapter 5 we will introduce RotoGrad, an approach to gradient conflict that tackles both magnitude and direction conflict simultaneously.

Gradient Homogenization in Multitask Learning

5.

La dignidad es la pieza clave para poder vivir bien.
Y cuando un pueblo no tiene dignidad,
se pone de rodillas y termina sin comer.
Porque sin dignidad no se come.

Julio Anguita

5.1 Problem statement	31
5.2 Gradient homogenization .	32
5.3 Illustrative examples	36
5.4 Empirical validation	37
5.5 Concluding remarks	41

As discussed previously in [Chapter 4](#), despite [multitask learning \(MTL\)](#) being increasingly adopted in applications domains such as computer vision and reinforcement learning, optimally exploiting its potential remains a major challenge as a consequence of negative transfer. Previous works have tracked down this issue to gradient conflict, *i.e.*, to the disparities between task gradients in their magnitude and direction across tasks, when optimizing the shared network parameters. However, while previous works have acknowledged that gradient conflict is a two-fold problem, existing approaches fall short as they only focus on either [i\)](#) homogenizing the gradient magnitude across tasks; or [ii\)](#) heuristically modifying the gradient directions, overlooking future interactions.

In this chapter, we introduce RotoGrad, an algorithm that tackles gradient conflict as a whole by homogenizing both gradient magnitudes and directions across tasks. Specifically, RotoGrad: [i\)](#) addresses magnitude discrepancies by re-weighting task gradients at each step of the training process, promoting learning those tasks that have converged the least thus far;¹ and [ii\)](#) instead of directly modifying gradient directions, RotoGrad smoothly rotates the shared feature space for each task, seamlessly aligning gradients² in the long run.

Additionally, as shown by our theoretical insights, the cooperation between gradient magnitude- and direction-homogenization algorithms ensures the stability of the overall learning process. Finally, we run extensive experiments to empirically demonstrate that RotoGrad leads to a stable (convergent) learning process, scales up to complex network architectures, and outperforms competing methods in multi-label classification settings on CIFAR10 and CelebA, and in computer vision tasks on the NYUv2 dataset.

5.1 Problem statement

In this section, we introduce the MTL setting assumed in this chapter, and recap some concepts previously introduced in [Chapter 4](#).

First, we assume a common input dataset $X \in \mathbb{R}^{N \times D}$, and our goal is to simultaneously learn K mappings from X to a task-specific set of labels $Y_k \in \mathbb{Y}_k^N$. We consider a hard parameter-sharing architecture, as it is broadly adopted in practice, with: [i\)](#) a *backbone* f parametrized by θ ; [ii\)](#) a shared output $z = f_\theta(x) \in \mathbb{R}^d$, where d is the dimensionality of z ; and [iii\)](#) K head networks with exclusive parameters ϕ_k , that output the task prediction based on z , $h_{\phi_k}(z) = \tilde{y}_k$.



github.com/adrianjav/rotoegrad

This chapter is based on the content of [\[11\]](#): Javaloy and Valera (2022), ‘RotoGrad: Gradient Homogenization in Multitask Learning.’

1: In that way, it attempts to not overlook any task, *i.e.*, to achieve *task impartiality*.

2: And thus, local optima.

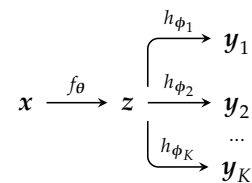


Figure 5.1: Hard parameter-sharing architecture assumed in this chapter.

3: See Section 4.4 in Chapter 4.

4: See Section 4.6 in Chapter 4.

Although MTL is *a priori* a **multi-objective optimization (MOO)** problem,³ in this chapter we adopt the common practice of assuming as loss a linear combination of the task losses, $L = \sum_k L_k$. While this assumption leads to a simpler optimization problem, it may also hurt the overall task performance due to an imbalanced competition among task gradients for the shared parameters, a problem we know as gradient conflict.⁴ Specifically, since θ is updated according to $\nabla_{\theta} L = \sum_k \nabla_{\theta} L_k$, we may face significant differences across: **i)** task magnitudes, leading to a subset of tasks dominating the learning process; and **ii)** task directions, leading to poor update directions that may not benefit any of the tasks.

In this chapter, we tackle negative transfer as a whole by homogenizing tasks gradients both in magnitude and direction. To reduce overhead, we adopt the usual practice of homogenizing gradients with respect to the shared feature z (rather than θ), as all tasks share gradient up to that point, $\nabla_{\theta} L_k = \nabla_{\theta} z \cdot \nabla_z L_k$. Therefore, in this chapter we focus on feature-level task gradients, $\nabla_z L_k$.

5.2 Gradient homogenization

5.2.1 Gradient magnitudes . . .	32
5.2.2 Gradient directions . . .	34
5.2.3 The full picture	35
5.2.4 Practical considerations .	35

In this section, we introduce RotoGrad, an algorithm to homogenize task gradients during training, thus palliating the effect of gradient conflict. Specifically, RotoGrad consists of two building blocks which homogenize task-gradient magnitudes and directions, respectively. These blocks address orthogonal problems, and they complement each other while providing convergence guarantees of the network training.

Next, we detail each of these building blocks and show how they are combined towards an effective MTL learning process. First, we address magnitude homogenization, proposing a parameter-less algorithm that equalizes gradient magnitudes at each iteration. Then, as gradient directions cannot be freely manipulated, we provide an algorithm which smoothly homogenizes gradient directions via task-specific rotations of the shared intermediate feature, z .

5.2.1 Gradient magnitudes

We aim to homogenize gradient magnitudes across tasks, as large magnitude disparities can lead to a subset of tasks dominating the learning process. Thus, the first goal of RotoGrad is to homogenize the magnitude of the gradients across tasks at each step of the training.

First, let us denote the feature-level task-gradient for the k -th task and n -th sample by $g_{nk} := \nabla_z L_k(h_k(x_n), y_{nk})$. Similarly, let us write the batched gradient as $G_k^T := [g_{1k} \ g_{2k} \ \dots \ g_{Bk}]$, where B is the batch size. Then, equalizing gradient magnitudes amounts to finding a set of weights ω_k that normalizes and scales each gradient G_k , *i.e.*,

$$\|\omega_k G_k\| = \|\omega_i G_i\| \quad \forall i \iff \omega_k G_k = \frac{C}{\|G_k\|} G_k = C U_k \quad \forall k, \quad (5.1)$$

where $U_k := G_k / \|G_k\|$ denotes the normalized task gradient, C the common target magnitude, and where we assume $\|G_k\| > 0$. Note that, in the above expression, C is a free parameter that we need to pick.

Dimensions

$$\begin{array}{ll} x \in \mathbb{R}^D & z \in \mathbb{R}^d \\ g_{nk} \in \mathbb{R}^d & G_k, U_k \in \mathbb{R}^{B \times d} \\ C \in \mathbb{R}^+ & \end{array}$$

In RotoGrad, we select C such that all tasks converge at a similar rate, in aims of having an impartial learning process across tasks. We motivate this choice by the fact that, by scaling all gradients, we change their individual step size, interfering with the convergence rates associated with their Lipschitz-smoothness [144]. Therefore, we seek for the value of C providing the best step size for those tasks that have converged the least up to the current iteration, t . To this end, we set C to be a convex combination of the task-wise gradient magnitudes, $C := \sum_k \alpha_k \|G_k\|$, where the weights $\alpha_1, \alpha_2, \dots, \alpha_K$ measure the *relative convergence* of each task and sum up to one, *i.e.*,

$$\alpha_k = \frac{\|G_k\| / \|G_k^0\|}{\sum_i \|G_i\| / \|G_i^0\|} \quad \text{with} \quad \sum_{k=1}^K \alpha_k = 1, \quad (5.2)$$

where G_k^0 is the initial gradient of the k -th task, *i.e.*, its gradient at the training iteration $t = 0$. The resulting algorithm is summarized below:

```

1 function: ComputeWeights
2 input: gradients  $\{G_k\}_{k=1}^K$ 
3 begin
4    $\alpha_k \leftarrow \|G_k\| / \|G_k^0\|$  for  $k = 1, 2, \dots, K$ 
5    $\alpha_k \leftarrow \alpha_k / \sum_i \alpha_i$  for  $i = 1, 2, \dots, K$ 
6    $C \leftarrow \sum_k \alpha_k \|G_k\|$ 
7    $\omega_k \leftarrow C / \|G_k\|$  for  $k = 1, 2, \dots, K$ 
8   return  $\omega$ 
9 end
```

For an introduction to optimization see [144] Nesterov (2004), ‘Introductory Lectures on Convex Optimization - A Basic Course.’

Algorithm 5.1: Function to compute the task weights in RotoGrad. The algorithm takes the batched task-gradients *w.r.t.* \mathbf{z} , G_k , and returns a set of weights, ω , to scale each of these gradients.

Algorithm 5.1 is a parameter-free algorithm that equals gradient magnitudes across tasks to encourage learning the slow-converging tasks. Notably, the proposed approach resembles a multitask version of Normalized Gradient Descent [36], which has been previously proved to quickly escape saddle points during optimization [138]. Consequently, we would expect a similar behaviour for RotoGrad, where slow-converging tasks will force quick-converging tasks to escape from saddle points.

[138] Murray, Swenson and Kar (2019), ‘Revisiting Normalized Gradient Descent: Fast Evasion of Saddle Points.’

However, the simplicity of **Algorithm 5.1** also makes it prone to errors in a number of degenerated settings, *e.g.*, in the presence of: **i)** noisy tasks that do not progress; **ii)** tasks that have fully converged; or **iii)** opposing tasks where, as one task improves, another task deteriorates. In the following proposition⁵ we show that, if task gradients do not excessively conflict in direction, then following the gradient $C \sum_k \mathbf{U}_k$ improves all tasks for the given batch. This result, while simplistic—*e.g.*, we assume a gradient step in feature space—provides insights in favour of having as *desideratum* of an efficient MTL pipeline the absence of gradient conflict.

5: Proved in **Appendix A.1**.

Proposition 5.1 If, at one iteration, the task gradients are such that $\cos(G_i, G_j) > -1/(K-1)$ for every pair of tasks $i, j \in \{1, 2, \dots, K\}$, then for a small-enough $\epsilon > 0$ we have that

$$L_k \left(h_{\phi_k} \left(\mathbf{Z} - \epsilon C \sum_{k=1}^K \mathbf{U}_k \right), \mathbf{Y}_k \right) < L_k (h_{\phi_k}(\mathbf{Z}), \mathbf{Y}_k) \quad (5.3)$$

for every $k \in \{1, 2, \dots, K\}$ and where $C > 0$.

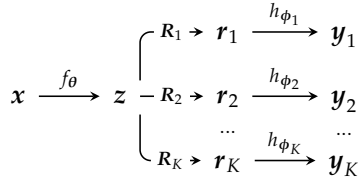


Figure 5.2: Hard parameter-sharing architecture, extended with the rotation matrices R_k from RotoGrad.

6: Closer in the parameter space.

Dimensions

$$R_k \in \mathbb{R}^{d \times d} \quad r_k \in \mathbb{R}^d$$

7: As it is a bijective mapping.

8: The special orthogonal group, $SO(d)$, denotes the set of all (proper) rotation matrices of dimension d .

Remark 5.1 While we keep the expression in Equation 5.4 as in the original publication [11], it is direct to check that this is equivalent to optimizing direction conflict instead, as we later define it in Chapter 6.

9: Which holds that $g_{nk} = R_k^\top \tilde{g}_{nk}$.

Algorithm 5.2: Function to update the rotation matrices introduced by RotoGrad to homogenize the gradient directions across tasks.

5.2.2 Gradient directions

We just motivated the need of reducing direction conflict, so that **Algorithm 5.1** could improve the performance of all tasks. In this section, we introduce the second building block of RotoGrad: an algorithm to homogenize gradient directions, thus complementing the previous scaling algorithm. The idea behind this approach is to smoothly rotate for each task the feature z to reduce the conflict between tasks in the following iterations, by bringing the (local) optima of different tasks closer to each other.⁶ In simple words, RotoGrad uses current local information to seemingly reduce the direction conflict at the following steps.

In order to homogenize gradient directions, for each task k , RotoGrad introduces a matrix R_k at the beginning of each task head so that, instead of optimizing $L_k(z)$ with z being the last shared representation, we optimize an equivalent⁷ loss function $L_k(R_k z)$. Since our sole interest is to change gradient directions, we choose $R_k \in SO(d)$ to be a rotation matrix⁸ leading to task-specific representations $r_k := R_k z$. Therefore, RotoGrad extends the usual hard parameter-sharing architecture by adding task-specific rotations before each head, as depicted in Figure 5.2.

Unlike the network parameters, the matrices R_k do not seek to improve the performance of their task. Instead, these additional parameters are optimized to reduce the direction conflict of the gradients across tasks. To this end, for each task we optimize R_k to maximize the batch-wise cosine similarity or, equivalently, to minimize

$$\mathcal{L}_{\text{rot}}^k := - \sum_n \langle R_k^\top \tilde{g}_{nk}, v_n \rangle, \quad (5.4)$$

where $\tilde{g}_{nk} := \nabla_{r_k} L_k(z_n, y_{nk})$ is the gradient *w.r.t.* r_k ,⁹ and v_n is the target direction that we want all task gradients to follow. In practice, we set the target vector v_n to be the gradient we would have followed if all task gradients weighted the same, *i.e.*, $v_n := \frac{1}{K} \sum_k u_{nk}$, where u_{nk} is the n -th row of the normalized batch gradient matrix U_k , as defined before. The resulting algorithm can be written as follows:

```

1 function: UpdateRotations
2 input: gradients  $\{G_k\}_{k=1}^K$  (with respect to  $z$ )
3 begin
4    $\tilde{G}_k \leftarrow R_k G_k$  for  $k = 1, 2, \dots, K$  # treated as a constant
5    $V \leftarrow \frac{1}{K} \sum_k G_k / \|G_k\|$ 
6   for  $k = 1, 2, \dots, K$  do
7      $\mathcal{L}_{\text{rot}}^k \leftarrow - \sum_n \langle R_k^\top \tilde{g}_{nk}, v_n \rangle$ 
8      $R_k \leftarrow R_k - \eta_{\text{rot}} \nabla_{R_k} \mathcal{L}_{\text{rot}}^k$ 
9   done
10 end
```

As a result of the extra loss function, in each training step with RotoGrad we simultaneously optimize the following two problems:

$$\mathcal{N}_{\text{etwork}}: \underset{\theta, \{\phi\}_k}{\text{minimize}} \sum_k \omega_k L_k, \quad \mathcal{R}_{\text{otation}}: \underset{\{R_k\}_k}{\text{minimize}} \sum_k \mathcal{L}_{\text{rot}}^k. \quad (5.5)$$

A key insight is that the above problem can be interpreted as a Stackelberg game: a two-player game in which a leader and a follower make alternate moves in order to minimize their respective losses, and where the leader knows in advance how the follower will respond to their moves. This

Remark 5.2 In this interpretation, the player learning the rotations introduced by RotoGrad plays the role of the leader since, as a result of optimizing Equation 5.4, it has first-order information of the next ‘move’ of the network parameters via the gradient information.

game-theoretical interpretation enables simple guidelines to guarantee *training convergence*—*i.e.*, guarantees that the network loss will not oscillate indefinitely as a result of optimizing two different objectives.

Specifically, we can leverage the results of Fiez et al. [58], and ensure that the problem in Equation 5.5 converges as long as we learn the rotations (leader) more slowly than the network parameters (follower). That is, if we make the learning rate for the rotations, η_{rot} , decrease faster than that of the network parameters, η , then we know that RotoGrad will converge to a local optimum for both objectives.¹⁰

[58] Fiez, Chasnov and Ratliff (2020), ‘Implicit Learning Dynamics in Stackelberg Games: Equilibria Characterization, Convergence Analysis, and Empirical Study.’



10: A more extensive discussion can be found in Appendix A.2.

5.2.3 The full picture

After describing the two main blocks of RotoGrad in the previous sections, we can now provide a full picture of the proposed algorithm, summarized in Algorithm 5.3. At each step: **i)** RotoGrad homogenizes the gradient magnitudes such that no task dominates the learning dynamics of the shared parameters, θ , and the step size is set according to the slow-converging tasks; in addition, **ii)** RotoGrad smoothly updates the rotation matrices¹¹ to seamlessly align task gradients in the following steps, thus preventing future direction conflicts.

11: Using the local information given by the task gradients.

```

1 input: samples  $X$ , labels  $\{Y_k\}_{k=1}^K$ 
2 begin
3    $Z \leftarrow f(X; \theta)$ 
4   for  $k = 1, 2, \dots, K$  do
5      $Z_k \leftarrow R_k Z$ 
6      $L_k \leftarrow \sum_n L_k(h_{\phi_k}(z_{nk}), y_{nk})$ 
7      $G_k \leftarrow \nabla_z L_k$ 
8   done
9   UpdateRotations( $\{G_k\}$ ) # Algorithm 5.2
10   $\omega \leftarrow \text{ComputeWeights}(\{G_k\})$  # Algorithm 5.1
11   $\theta \leftarrow \theta - \eta \nabla_{\theta} \cdot \sum_k \omega_k G_k$ 
12   $\phi_k \leftarrow \phi_k - \eta \nabla_{\phi_k} L_k$  for  $k = 1, 2, \dots, K$ 
13 end
```

Algorithm 5.3: One training iteration of RotoGrad. In the forward pass, the only additional step is the rotation of the shared features in line 5. In the backward pass, there are two additional steps: **i)** the re-weighting of task gradients in line 9; and **ii)** the update of the rotation matrices in line 10.


5.2.4 Practical considerations

In this section, we discuss the main practical considerations to account for when implementing RotoGrad, and propose how to address them.

Unconstrained optimization. As previously discussed, the parameters R_k are defined as rotation matrices, and thus Rotation in Equation 5.5 is a constrained optimization problem. While this would typically require the use of costly algorithms like Riemannian gradient descent [1], we can leverage recent work on manifold parametrization [18, 19] and, instead, apply unconstrained optimization methods by automatically parametrizing R_k via exponential maps on the Lie algebra of $\text{SO}(d)$.¹²

12: *E.g.*, libraries like Geotorch [18] make this process transparent to the user.

Memory and time complexity. As we need one rotation matrix per task, we have to store $\mathcal{O}(Kd^2)$ additional parameters. In practice, we only need $Kd(d-1)/2$ parameters due to the aforementioned parametrization and, in most cases, this amounts to a small fraction of the total number of network parameters. Moreover, parametrizing R_k enables efficient

[19] Casado and Martínez-Rubio (2019), ‘Cheap Orthogonal Constraints in Neural Networks: A Simple Parametrization of the Orthogonal and Unitary Group.’ 

computations compared with traditional methods [19], with a time complexity of $\mathcal{O}(d^3)$ independently of the batch size. In our case, the time complexity is of $\mathcal{O}(Kd^3)$, which scales better with respect to the number of tasks than existing methods, *e.g.*, the $\mathcal{O}(K^2d)$ time complexity of PCGrad [227]. Moreover, other techniques such as forward-pass caching and GPU parallelization can further reduce training time.

Scaling-up RotoGrad. Despite being able to efficiently compute and optimize the rotation matrix R_k , in domains like computer vision, where the shared representation size, d , is extremely large, the time complexity for updating the rotation matrix may become comparable to that of the network updates. In those cases, we propose to only rotate a subspace of the feature space, *e.g.*, to rotate only the first $m \ll d$ dimensions of z . Then, we can simply apply a transformation of the form $r_k = [R_k z_{1:m}, z_{m+1:d}]$, where $z_{a:b}$ denotes the elements of z with indexes $a, a+1, \dots, b$. While there exist other possible solutions, such as using block-diagonal rotation matrices R_k , we defer them to future work.

5.3 Illustrative examples

In this section, we illustrate the behaviour of RotoGrad in two synthetic scenarios, providing clean qualitative results about its effect on the loss landscape. Appendix A.3.1 provides a detailed description of the experimental setups.

Namely, we consider two multitask regression problems of the form

$$L(x) = L_1(x) + L_2(x) = \varphi(R_1 f_\theta(x), 0) + \varphi(R_2 f_\theta(x), 1), \quad (5.6)$$

where φ is a test function¹³ with a single global optimum whose position is parametrized by the second argument, *i.e.*, both tasks are identical—and thus related—up to a translation. We use a single input $x \in \mathbb{R}^2$ and drop all task-specific parameters. For the backbone, we take a simple network of the form $z = W_2 \max(W_1 x + b_1, 0) + b_2$ with $b_1 \in \mathbb{R}^{10}$, $b_2 \in \mathbb{R}^2$, and $W_1, W_2^\top \in \mathbb{R}^{10 \times 2}$.

For the first experiment we choose a simple convex avocado-shaped objective function and, for the second one, we opt for a non-convex function with several local optima and a single global optimum. Figure 5.3 shows the training trajectories in the presence (and absence) of RotoGrad for both experiments, depicted as level plots in the space of z and r_k , respectively. To provide a clear comparison, we compare with the vanilla case with the same fixed initial rotations as RotoGrad, since the matrices are not initialized to the identity matrix.

For the first experiment, we can observe in Figure 5.3a, that RotoGrad finds both optima by rotating the feature space and matching the (unique) local optima of the tasks. Similarly, for the second experiment in Figure 5.3b—as we have two symmetric tasks and a non-equidistant starting point—the optimization for the vanilla case is dominated by the task closer to an optimum. In contrast, RotoGrad avoids this problem by equalizing gradient magnitudes and, by aligning gradients, it is able to find the optima of both functions.

13: Also known as artificial landscape. See <https://w.wiki/878L>.

Remark 5.3 For the avocado-shaped experiment we use

$$\varphi((x, y), s) = (x - s)^2 + 25y^2,$$

while for the non-convex experiment we instead take

$$\begin{aligned} \varphi((x, y), s) = & -\frac{\sin(3x + 4.5s)}{x + 1.5s} \\ & -\frac{\sin(3y + 4.5s)}{y + 1.5s} \\ & + |x + 1.5s| \\ & + |y + 1.5s|. \end{aligned}$$

Remark 5.4 The duality depicted in Figure 5.3 between rotating the feature z and rotating the loss landscape corresponds in geometry to the concepts of active and passive transformations. These are two perspectives of the same transformation. See <https://w.wiki/878H>.

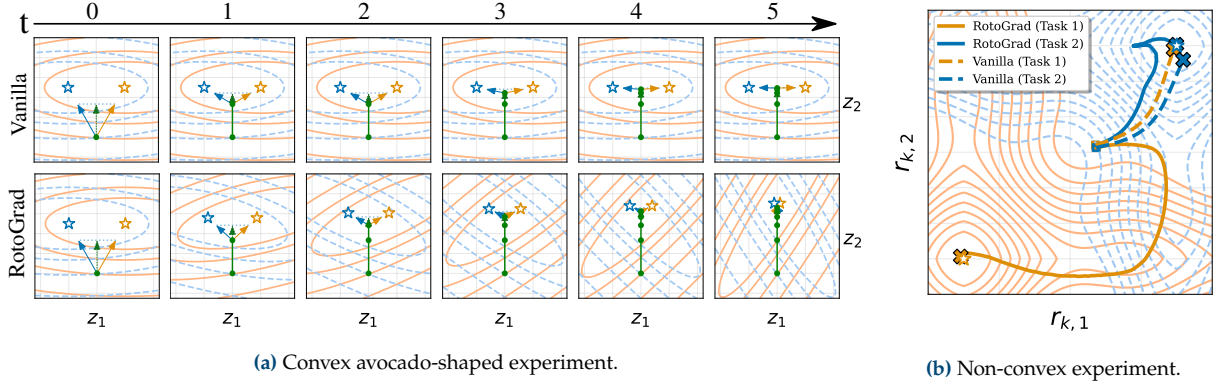


Figure 5.3: Level plots showing the evolution of two regression MTL problems with and without RotoGrad, see Section 5.3. RotoGrad is able to reach the optimum (\star) for both tasks. **(a)** In the space of z , RotoGrad rotates the loss landscapes to align task gradients (blue and orange arrows), finding shared features z (green arrow) closer to the (matched) optima. **(b)** In the space of r_k , RotoGrad rotates the shared feature z , providing task-specific features r_k that better fit each task.

5.4 Empirical validation

In this section, we assess the performance of RotoGrad on a wide range of MTL settings. First, we check the effect that learning rates have on the rotation and network updates for the stability of RotoGrad. Then, with the aim of applying RotoGrad to scenarios with high-dimensional z , we explore the effect of rotating only a subspace of z . Finally, we compare RotoGrad with existing MTL solutions, showing that it consistently outperforms them. Refer to Appendix A.3 for more details on the experiments and additional results.

Relative task improvement. Throughout this section, we resort to the relative task improvement [124] to group task performances. Given the test metrics obtained by a model, M_k , and by a baseline model, B_k , the relative task improvement for the k -th task is

$$\Delta_k := 100 \cdot (-1)^{l_k} \frac{M_k - B_k}{B_k}, \quad (5.7)$$


where $l_k = 1$ if $M_k < B_k$ means that the baseline is worse, and $l_k = 0$ otherwise. Since Δ_k can be quite sensitive, we show different statistics such as the mean ($\text{avg}_k \Delta_k$), maximum ($\max_k \Delta_k$), and median ($\text{med}_k \Delta_k$) across tasks. Refer to Chapter 6 for an in-depth discussion on relative task improvement and more robust alternatives.

Statistical significance. We highlight significant improvements according to a one-sided paired t-test ($\alpha = 0.05$), with respect to MTL with vanilla optimization (marked with † in each table).

5.4.1 Training stability

At the end of Subsection 5.2.2 we discussed that, by casting the problem of Equation 5.5 as a Stackelberg game, we can enjoy convergence guarantees as long as the rotation optimizer is the slow learner. Next, we empirically verify this statement.

5.4.1 Training stability	37
5.4.2 RotoGrad's blocks	38
5.4.3 Subspace rotations	38
5.4.4 Methods comparison	39

[182] Sener and Koltun (2018), ‘Multi-Task Learning as Multi-Objective Optimization.’ 

14: See Subsection 6.1.2 in Chapter 6.

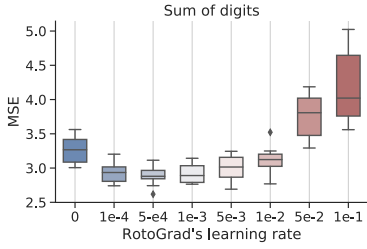


Figure 5.4: Test error on the sum-of-digits task for different values of RotoGrad’s learning rate, η_{rot} , on Multi-MNIST.

Experimental setup. We follow a similar setup to that of Sener and Koltun [182], where we use Multi-MNIST, a multitask version of MNIST [102] composed of a left and right digit, and use as backbone a reduced version of LeNet [101] with light-weight heads. Since the original Multi-MNIST does not exhibit enough conflict,¹⁴ we consider three other tasks besides the original left- and right-digit classification tasks: **i)** sum of digits; **ii)** parity of the digit product; and **iii)** number of active pixels. The idea here is to enforce all digit-related tasks to cooperate, while the (orthogonal) image-related task should not disrupt these learning dynamics.

Results. Figure 5.4 shows the effect, averaged over ten independent runs, that changing the learning rate η_{rot} has on the test error of the sum-of-digits task, while the rest of tasks are shown in Appendix A.3.2. We observe that, the bigger the learning rate is (in comparison to the network learning rate, $\eta = 1e-3$), the larger and noisier the test error becomes. Mean squared error (MSE) keeps decreasing as we lower the learning rate, reaching a sweet spot at half the network learning rate, $\eta_{\text{rot}} = 5e-4$. For smaller values, the rotations are learnt too slowly and results start to resemble those of Vanilla, in which no rotations are applied (leftmost box in Figure 5.4).

5.4.2 RotoGrad’s building blocks

In this section, we evaluate to which extent each of RotoGrad’s components, which we call Scale (Algorithm 5.1) and Rotate (Algorithm 5.2), contribute to the performance gains of RotoGrad.

Experimental setup. We test each component on three different tasks on the NYUv2 dataset [37]: **i)** 13-class semantic segmentation; **ii)** depth estimation; and **iii)** normal surface estimation. Following the setup by Liu et al. [115], we resize all images to a resolution of 288×384 px to speed up training, and apply data augmentation to alleviate overfitting. As MTL architecture, we use SegNet [5] as backbone, where the decoder is split into three convolutional heads.

Results. The three top rows of Table 5.4 show the performance of RotoGrad and its both components in isolation, all of them using the same number of parameters. Compared to Vanilla (4th row), Rotate improves all metrics by homogenizing gradient directions. Scale avoids overlooking the normal estimation task and improves on semantic segmentation by homogenizing gradient magnitudes, at the expense of higher depth estimation error. Remarkably, RotoGrad exploits its scaling and rotation components to obtain the best results in semantic segmentation and depth estimation, while still achieving comparable performance in the normal estimation task.

5.4.3 Subspace rotations

We now evaluate the effect of applying subspace rotations as described at the end of Subsection 5.2.4, assessing the trade-off between avoiding negative transfer and the subspace size rotated by RotoGrad.


[115] Liu, Johns and Davison (2019), ‘End-To-End Multi-Task Learning With Attention.’ 

Table 5.4: Median results, over five runs, on the NYUv2 dataset. RotoGrad obtains great performance in segmentation and depth tasks, and significantly improves the results on normal surfaces. Δ_S , Δ_D , and Δ_N denote the relative task improvement for each task.

Method		Relative improvement \uparrow			Segmentation \uparrow		Depth \downarrow		Normal Surfaces				
		Δ_S	Δ_D	Δ_N	mIoU	Pix Acc	Abs.	Rel.	Angle Dist. \downarrow		Within t° \uparrow		
									Mean	Median	11.25	22.5	30
	Single	0.0	0.0	0.0	39.21	64.59	0.70	0.27	25.09	19.18	30.01	57.33	69.30
With R_k ($m = 1024$)	Rotate	3.3	20.5	-6.6	39.63	66.16	0.53	0.21	26.12	20.93	26.85	53.76	66.50
	Scale	-0.3	20.0	-7.9	38.89	65.94	0.54	0.22	26.47	21.24	26.24	53.04	65.81
	RotoGrad	1.8	24.0	-6.1	39.32	66.07	0.53	0.21	26.01	20.80	27.18	54.02	66.53
	Vanilla	-2.7	20.6	-25.7	38.05	64.39	0.54	0.22	30.02	26.16	20.02	43.47	56.87
	GradDrop	-0.9	14.0	-25.2	38.79	64.36	0.59	0.24	29.80	25.81	19.88	44.08	57.54
	PCGrad	-2.7	20.5	-26.3	37.15	63.44	0.55	0.22	30.06	26.18	19.58	43.51	56.87
	MGDA-UB	-31.2	-0.7	0.6	21.60	51.60	0.77	0.29	24.74	18.90	30.32	57.95	69.88
	GradNorm	-0.6	19.5	-10.5	37.22	63.61	0.54	0.22	26.68	21.67	25.95	52.16	64.95
	IMTL-G	-0.3	17.6	-7.5	38.38	64.66	0.54	0.22	26.38	21.35	26.56	52.84	65.69
	Without R_k	Vanilla [†]	-0.9	16.8	-25.0	37.11	63.98	0.56	0.22	29.93	25.89	20.34	43.92
GradDrop		-0.1	15.7	-27.0	37.51	63.62	0.59	0.23	30.15	26.33	19.32	43.15	56.59
PCGrad		-0.5	20.0	-24.6	38.51	63.95	0.55	0.22	29.79	25.77	20.61	44.22	57.64
MGDA-UB		-32.2	-8.2	1.5	20.75	51.44	0.73	0.28	24.70	18.92	30.57	57.95	69.99
GradNorm		2.2	20.6	-10.2	39.29	64.80	0.53	0.22	26.77	21.88	25.39	51.78	64.76
IMTL-G		1.9	21.4	-6.7	39.94	65.96	0.55	0.21	26.23	21.14	26.77	53.25	66.22

Experimental setup. We test RotoGrad on a 10-task classification problem on CIFAR10 [97], using for all tasks **binary cross-entropy (BCE)** and F1-score as loss and metric, respectively. We use ResNet-18 [74] (without pre-training) as backbone ($d = 512$), and linear layers with sigmoid activation functions at the output as task-specific heads.

Results. The top part of Table 5.5 shows that rotating the entire space provides the best results, and that these worsen as we decrease the size of R_k , *i.e.*, as we decrease m . Rotating only 128 features already outperforms Vanilla with no extra task-specific parameters (1st row); and rotating 256 features already yields comparable results to Vanilla with extra capacity (6th row) despite its larger number of task-specific parameters. These results can be further explained by Figure 5.5, which shows a positive correlation between the size of R_k and cosine similarity.

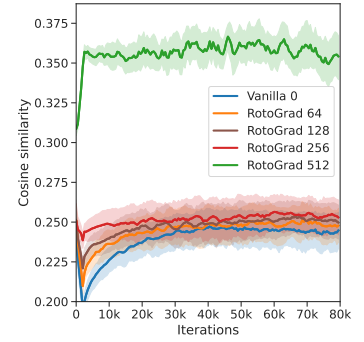


Figure 5.5: Cosine similarity between the task gradients and the update gradient, $\cos(\mathbf{g}_k, \bar{\mathbf{g}})$ on CIFAR10. Results are averaged over tasks and five runs. Shaded areas show 90 % confidence intervals.

5.4.4 Methods comparison

We now compare RotoGrad with the different existing approaches to gradient conflict (for both magnitude and direction) in different real-world datasets, showing that RotoGrad outperforms existing methods while being on par with them in training time.

Experimental setup. In order to provide fair comparisons among methods, all experiments use identical configurations and random initializations. For each experiment, we performed a hyperparameter search and chose the best configuration based on validation error. Unless otherwise specified, all baselines use the same architecture (and thus, number of parameters) as RotoGrad,¹⁵ using each rotation matrix R_k as extra

15: That is, using the extended architecture shown in Figure 5.2.

Table 5.5: Median and standard deviation results over 5 runs. **(Top)** Relative task improvement on CIFAR10 for RotoGrad with different sizes of R_k . **(Bottom)** Comparison with existing methods, extended with rotation matrices as extra task-specific parameters.

Method	d	$\text{avg}_k \Delta_k \uparrow$	$\text{med}_k \Delta_k \uparrow$	$\text{max}_k \Delta_k \uparrow$
Vanilla [†]	0	2.58 ± 0.54	1.90 ± 0.53	11.14 ± 3.35
RotoGrad	64	2.90 ± 0.49	1.79 ± 0.57	13.16 ± 2.40
RotoGrad	128	2.97 ± 1.08	2.25 ± 1.07	12.64 ± 3.56
RotoGrad	256	3.68 ± 0.68	2.16 ± 0.72	14.01 ± 3.22
RotoGrad	512	4.48 ± 0.99	3.67 ± 1.40	15.57 ± 3.99
With R_k ($m = 512$)	Vanilla	3.12 ± 0.79	3.10 ± 1.29	14.23 ± 2.86
	GradDrop	3.54 ± 1.10	3.27 ± 1.61	13.88 ± 2.95
	PCGrad	3.29 ± 0.46	2.67 ± 0.88	13.44 ± 1.86
	MGDA-UB	0.21 ± 0.67	0.57 ± 0.62	4.78 ± 2.15
	GradNorm	3.21 ± 1.04	3.10 ± 1.01	10.88 ± 4.73
	IMTL-G	3.02 ± 0.69	1.81 ± 0.87	12.76 ± 1.77

task-specific parameters. Further experimental details can be found in [Appendix A.3.1](#), as well as extra experiments and complete results in [Appendix A.3.2](#).

NYUv2. [Table 5.4](#) shows the performance of all baselines with and without the extra capacity. RotoGrad significantly improves the performance on all tasks compared with Vanilla, and outperforms all other baselines. Remarkably, we rotate only 1024 dimensions of z (out of a total of 7 millions) and, as a result, RotoGrad stays on par in training time with the baselines.¹⁶ We can also assert the importance of learning the matrices R_k properly by comparing the different baselines with and without the extra task-specific rotations in [Table 5.4](#).

Interestingly, we can observe that the extra parameters do not alleviate negative transfer, but instead *amplify biases*—methods that overlook a subset of tasks, keep overlooking them—and, in the best case, provide trade-off solutions (also shown in [Appendix A.3.2](#)). Moreover, note that RotoGrad (due to the Rotate component) is the only method to tackle conflicting gradient directions that manages to not overlook the normal surfaces task.

CIFAR10. We reuse the setting from [Subsection 5.4.3](#) to compare different MTL baselines in terms of relative improvements ([Table 5.5](#)) and cosine similarity ([Figure 5.6](#)), averaged over five different runs. We can observe in [Table 5.5](#) that, similar to the results in NYUv2, both direction-aware solutions (PCGrad and GradDrop) behave similar to Vanilla, marginally increasing the average improvement. Unlike previous experiments, all magnitude-aware methods substantially worsen (at least) one of the statistics. And, in contrast, RotoGrad improves the relative task improvement across all statistics using the same number of parameters.

[Figure 5.6](#) shows the cosine similarity between task and update gradients, *i.e.*, $\cos(\mathbf{g}_k, \bar{\mathbf{g}})$, averaged over all tasks and runs (shaded areas correspond to 90 % confidence intervals). RotoGrad obtains significantly better cosine similarity than the rest of methods, yet direction-aware approaches effectively align task gradients as well. This result, combined with the low cosine similarity achieved by MGDA-UB, suggests that there exists a positive correlation between cosine similarity and task performance.

16: Around 4 h, see [Appendix A.3.2](#).

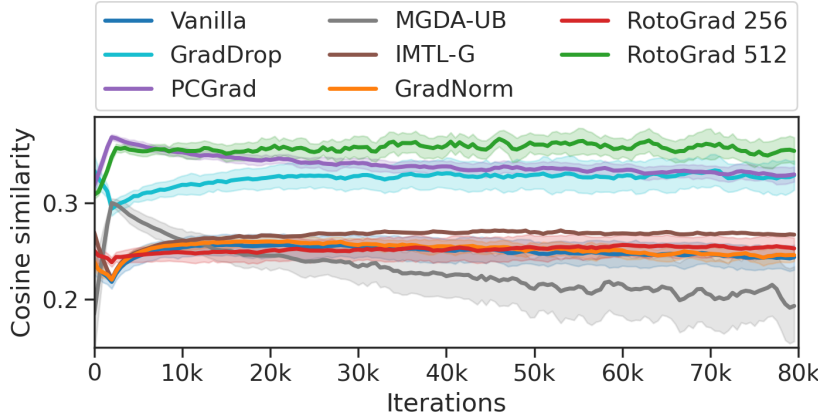


Figure 5.6: Cosine similarity between task and update gradients for different methods on CIFAR10, averaged over all tasks and 5 runs.

Table 5.6: F1-score statistics (median over 5 runs) in CelebA for two network architectures.

Conv. Net. with R_k ($m = 256$)					ResNet-18 with R_k ($m = 1536$)				
Method	Task F1-scores (%) \uparrow				Method	Task F1-scores (%) \uparrow			
	\min_k	med_k	avg_k	$\text{std}_k \downarrow$		\min_k	med_k	avg_k	$\text{std}_k \downarrow$
Vanilla	4.59	50.28	56.03	25.65	Vanilla	19.71	63.56	63.23	21.16
GradDrop	3.18	50.07	54.43	27.21	GradDrop	12.33	62.40	62.74	21.74
PCGrad	1.44	53.05	54.72	27.61	PCGrad	14.71	63.65	62.61	22.22
GradNorm	2.08	52.53	56.71	24.57	GradNorm	9.05	60.20	60.78	22.31
IMTL-G	0.00	37.00	42.24	33.46	IMTL-G	17.11	61.68	60.72	22.80
RotoGrad	4.59	55.02	57.20	24.75	RotoGrad	9.96	63.84	62.81	21.80

CelebA. To conclude, we test all methods in a 40-class multi-classification problem on CelebA [117] with two different architectures: **i)** one using a convolutional network as backbone ($d = 512$); and **ii)** another using a ResNet-18 [74] as backbone ($d = 2048$). Like before, we use BCE and F1-score as loss and metric for all tasks, thus accounting for highly imbalanced attributes. Results in Table 5.6 show that RotoGrad performs great in all F1-score statistics and both architectures, specially with the convolutional backbone, outperforming competing methods.

Moreover, RotoGrad achieves these results rotating 50 % of the shared feature z for the convolutional network, and 75 % for the residual network, which further demonstrates that RotoGrad can scale-up to real-world settings. We believe it is important to remark that, due to the high number of tasks, this setup is specially demanding. Results in Appendix A.3.2 show the performance of all baselines without the rotation matrices, demonstrating the negative effect that the extra capacity can have if not learnt properly, and that RotoGrad stays on par with non-extended baselines in terms of training time.

5.5 Concluding remarks

In this chapter, we have introduced RotoGrad, an algorithm that tackles negative transfer in MTL by homogenizing task gradients in terms of both magnitudes and directions. RotoGrad enforces a similar convergence rate for all tasks, while at the same time smoothly rotates the shared representation differently for each task in order to avoid conflicting gradients. As a result, RotoGrad leads to stable and accurate MTL. Our

empirical results have shown the effectiveness of RotoGrad in many scenarios, staying on top of all competing methods in performance, while being on par in terms of computational complexity with those that better scale to complex networks.

We believe our work opens up interesting venues for future work. For example, it would be interesting to study alternative approaches to further scale up RotoGrad using, *e.g.*, diagonal-block or sparse rotation matrices; to rotate the feature space in application domains with structured features, *e.g.*, channel-wise rotations in images; and to combine different methods, *e.g.*, by scaling gradients using the direction-awareness of IMTL-G and the ‘favor slow-learners’ policy of RotoGrad.

On Task Incomparability and its Effects in Multitask Learning

6.

Tanto amor me confunde.

Isabel Valera, circa 1990

Multitask learning (MTL) poses unique challenges that remain largely ignored to this day. First, as we explained in [Section 4.4](#), the **multi-objective optimization (MOO)** nature of MTL makes model comparison ill-defined in the absence of a scalarization function. Moreover, designing such a scalarization function becomes an arduous task in the presence of heterogeneous tasks due to **task incomparability**, *e.g.*: would it be preferable to have 10 % more accuracy on task A, or 0.5 less regression error on task B? Task incomparability also contributes to the problem of *gradient conflict* explained in [Section 4.6](#) where, during training, task gradients may not be comparable (*e.g.*, having different magnitudes), biasing the model towards learning a subset of all the tasks.

While prior works propose ways of addressing gradient conflict, only a handful of them attempt to actually measure it, usually as the cosine similarity between task gradients and the total gradient. Remarkably, common MTL benchmarks have been inherited and adapted from **deep learning (DL)** without ensuring that gradient conflict is indeed a problem during training. Therefore, it is unclear whether these benchmarks are well-suited to test MTL approaches specifically addressing gradient conflict, despite having been used to cast doubts of the effectiveness of gradient-conflict approaches [[99](#), [220](#)].

In this chapter, we first address the problem of model selection in MTL by showing that simple statistics over task rankings are tightly connected with non-preference scalarization methods over **cumulative distribution functions (CDFs)**, thus explicitly specifying a scalarization function over statistics robust to task incomparability. Then, we provide more fine-grained gradient conflict measures by showing that the usual cosine similarity can be factorized in two quantities, each of them concerning a different type of gradient conflict. To motivate the utility of these metrics, we then take the broadly-used Multi-MNIST benchmark and study how different design choices affect gradient conflict, probing the benchmark, and resulting in a new Conflict benchmark better-suited for MTL. Finally, we use Conflict to compare different methods in the gradient conflict literature, showing that indeed different solutions behave significantly different, in stark contrast with the results shown in previous MTL benchmarks, empirically validating the proposed metrics and methodology.

6.1 Motivation & background .	43
6.2 How to measure	47
6.3 Benchmark probing	52
6.4 Empirical validation	58
6.5 Concluding remarks	62

6.1 Motivation and background

To better contextualize the chapter, in this section we motivate the need for better metrics and benchmarks by summarizing the different findings and results found in previous works. First, we will introduce the different metrics used in the literature to measure both MTL performance and

6.1.1 Previous metrics	44
6.1.2 MTL benchmarks	45
6.1.3 Prior analyses	46

gradient conflict, then we will empirically show the absence of gradient conflict in two popular MTL benchmarks, and finally we will briefly introduce the main conclusions from previous analyses.

6.1.1 Previous metrics

Model performance. As previously mentioned, measuring model performance is a challenging task due to task incomparability and the absence of a clear scalarization function. Some relevant examples of MTL model performance in the literature are the following:

- If the set of tasks are *homogeneous*—*i.e.*, if they are indeed *comparable*—measuring model performance becomes much simpler, as we can take any statistics of the task-metrics vector, *e.g.*, the average.¹ While constraining, sticking to homogeneous tasks is quite a common practice in the existing literature, *e.g.*, using F1-scores [208], BLEU scores [208], perplexities [129], success rates [99, 110, 111, 227, 228], or accuracies [99, 111, 125, 182, 184, 227, 232].
- If we have heterogeneous tasks, there exist variations of common metrics that attempt to normalize task performances. For example, when dealing with classification and regression tasks, it is common to replace the **root mean squared error (RMSE)** in regression tasks by the *normalized* RMSE to match the domain of the classification errors [8, 143, 232]:

$$\text{nRMSE}(\mathbf{y}) = \frac{1}{\sqrt{D_y}} \frac{\|\mathbf{y} - \tilde{\mathbf{y}}\|}{\max_n \mathbf{y}_n - \min_n \mathbf{y}_n}, \quad (6.1)$$

where D_y indicates the target dimensionality.

- Most remarkably, Maninis et al. proposed in 2019 the **relative task improvement** metric [124] in the context of MTL, and it has been widely adopted by the community [21, 110–112, 114, 194, 195]. This metric, which we denote by $\text{avg } \Delta$, is defined as

$$\text{avg } \Delta := \frac{1}{K} \sum_k \Delta_k \quad \text{where} \quad \Delta_k := 100 \cdot (-1)^{l_k} \frac{M_k - B_k}{B_k}, \quad (6.2)$$

with M_k and B_k being, respectively, the k -th task performances of the model M and a baseline model B .² Here, l_k is a binary variable to change the sign depending on the semantics of the metric, *i.e.*, it is set to zero if a higher value of M_k is better, and to one otherwise. Despite its broad adoption, $\text{avg } \Delta$ has two main drawbacks: **i)** it is expensive to compute, as it requires a baseline model for each task; and **ii)** due to task incomparability, it can be quite sensitive to certain metrics, as we show later in Table 6.6.

Gradient conflict. As briefly mentioned before, task incomparability exacerbates gradient conflict as well: if task losses are not comparable, neither are their gradients. This adds to the already difficult task of measuring gradient conflict, as we attempt to summarize in a single number the interactions of task gradients during all training.

Since tasks may not be comparable, and since the range of values that the gradient norms take can change wildly, one sensible approach throughout

1: Note, however, that we are selecting a scalarization function when we choose the statistic to measure performance.

2: This baseline is usually the same architecture solving only the k -th task, *i.e.*, the **single task learning (STL)** model.

the MTL literature is to use cosine similarities between task gradients to measure gradient conflict [47, 104, 208, 227, 228], as it is invariant to the scale of the gradients. In this chapter, instead of using these metrics to measure gradient conflict, we repurpose them to help us have a better understanding on the interaction between tasks. Namely:

- **Parameter sharing.** We estimate the extent to which two different tasks share parameters, by measuring how much their gradients align, $\cos(\mathbf{g}_i, \mathbf{g}_j)$. For example, if two task gradients were orthogonal,³ then the parameters could effectively be divided in two task-exclusive sets, up to a linear mapping.
- **SGD agreement.** To understand how much a task contributes to the total gradient $\boldsymbol{\theta}$, we propose to measure the cosine similarity of its gradient *w.r.t.* the total one, *i.e.*, $\cos(\mathbf{g}_k, \bar{\mathbf{g}})$.⁴ This quantity describes how much the gradient used during training aligns with that of a specific task, and therefore on its share during training.

The alignment between gradients is also important in other contexts beyond MTL, and different metrics have been proposed in the literature. Some relevant examples are:

- **Cosine stiffness** [62], which measures the pairwise alignment between different gradients:

$$\frac{1}{K} \frac{1}{K-1} \sum_{i \neq j} \cos(\mathbf{g}_i, \mathbf{g}_j). \quad (6.3)$$

- **Gradient coherence** [23], that computes the average SGD agreement up to a multiplicative factor:

$$\frac{1}{K} \sum_{k=1}^K \cos(\mathbf{g}_k, \bar{\mathbf{g}}) \|\bar{\mathbf{g}}\|. \quad (6.4)$$

- **Gradient confusion** [175], that measures the *worst* alignment between two task gradients:⁵

$$\min_{i,j} \cos(\mathbf{g}_i, \mathbf{g}_j). \quad (6.5)$$

In [Subsection 6.2.2](#), we will see the connection between these existing metrics and the ones we propose to measure gradient conflict.

6.1.2 MTL benchmarks

As previously mentioned at the beginning of the chapter, all commonly-used MTL benchmarks⁶ have been directly inherited and adapted from DL applications, without making sure that gradient conflict is indeed a problem during training. In this section, we gather some empirical and historical evidence suggesting that common MTL benchmarks may not be well-suited to test MTL approaches.

Multi-MNIST. First proposed by Sabour et al. [171] and brought to MTL by Sener and Koltun [182], the Multi-MNIST benchmark is a widely-adopted experiment to assert the effectiveness of MTL solutions [21, 99, 103, 167, 182, 220, 227]. Specifically, Multi-MNIST is a modified version

Remark 6.1 The cosine similarity between two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ is

$$\cos(\mathbf{a}, \mathbf{b}) := \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|},$$

where \cdot is the usual dot product.

3: See, *e.g.*, the Multi-MNIST benchmark shown later in [Table 6.1](#).

4: In [Subsection 6.2.2](#), we show that the average SGD agreement can be factorized to provide fine-grained measurements of gradient conflict.

5: Or, as we will see in [Subsection 6.2.2](#), the worst direction conflict.

6: Here, with the term benchmark we mean a specific dataset, model, and hyperparameters values.



Figure 6.1: Example samples from the Multi-MNIST dataset.

Table 6.1: Model performance and gradient similarities on the Multi-MNIST benchmark, averaged over 300 runs. Single-task models achieved 96.45 ± 0.22 and 95.45 ± 0.20 on left- and right-digit classification, respectively.

Metric	Value
Left acc. (%)	96.40 ± 0.24
Right acc. (%)	94.94 ± 0.33
$\cos(\mathbf{g}_L, \mathbf{g}_R)$	0.04 ± 0.02
$\mathbb{E}_k[\cos(\mathbf{g}_{k'}, \bar{\mathbf{g}})]$	0.70 ± 0.00

of the MNIST dataset [102] where two digits are randomly placed (with some overlap) on each side of the image, and the goal is to classify both digits, treating them as two different classification tasks. LeNet [101] is used as the backbone, with simple heads consisting of one or two layers, and a shared representation, \mathbf{z} , of dimensionality 50.

In Table 6.1, we show the quantitative results from running the Multi-MNIST benchmark 300 times: 30 for each gradient conflict method considered in Subsection 6.4.2. In both tasks, the model performs on par with single-task models. Averaged over all training, we can also observe that the two task gradients (*w.r.t.* to the backbone parameters, θ) are completely orthogonal, as also indicated by the fact that their angle with respect to the total gradient is of 45° . In short, this evidence suggests that there is no conflict in the Multi-MNIST benchmark, as the backbone parameters are effectively not shared, but partitioned across both tasks. In other words, *the multitask model is effectively two single-task models*.

CelebA. Another popular benchmark in the gradient conflict literature is the CelebA benchmark: directly inherited from Deep Learning, CelebA [117] is a dataset consisting of booking photographs of celebrities, each one accompanied by a set of 40 labels such as gender, or the presence of glasses in the picture. In its MTL setting, the benchmark is treated as a multi-class classification problem, where each of the 40 labels is taken as a different classification task. Regarding the architecture, the common practice [21, 99, 106, 114, 182, 220, 227] is to use ResNet-18 as the backbone, and linear layers as heads. As a result, the backbone has around 11 M parameters, and \mathbf{z} has a dimensionality of 2048.

To illustrate the amount of parameter sharing in the benchmark, we took the official code of the work by Kurin et al. [99], and stored the task gradients during training to measure the overlap in parameter sharing. Figure 6.2 shows the minimum, average, and maximum pairwise parameter sharing⁷ while training the network. As it can be observed, after a few epochs all tasks gradients become almost orthogonal, suggesting that the overlap in network parameters is minimal, and thus each of them has a big portion of task-specific backbone parameters.


Other benchmarks. In their work, Suteu and Guo [197] proposed a method that, instead of aligning task gradients, encourages them to be orthogonal. Interestingly, the authors empirically showed that the two classification tasks in the Multi-MNIST benchmark are orthogonal, despite not discussing it explicitly. Not only that, but they also showed that by making gradients more orthogonal—and therefore sharing less parameters—they could obtain competitive results on the NYUv2 [141] and SUN RGB-D [191] benchmarks, which suggest that they could as well be absent of any relevant gradient conflict, just as we showed above for CelebA and Multi-MNIST.


6.1.3 Prior gradient-conflict analyses

To provide some context, we briefly discuss prior works analysing gradient conflict approaches and trying to measure gradient conflict.

First, Yu et al. [227] introduced what they called ‘the tragic triad’, and

7: *I.e.*, the cosine similarity, $\cos(\mathbf{g}_i, \mathbf{g}_j)$.

[197] Suteu and Guo (2019), ‘Regularizing deep multi-task networks using orthogonal gradients.’ 

[227] Yu, Kumar, Gupta, Levine, Hausman and Finn (2020), ‘Gradient Surgery for Multi-Task Learning.’ 

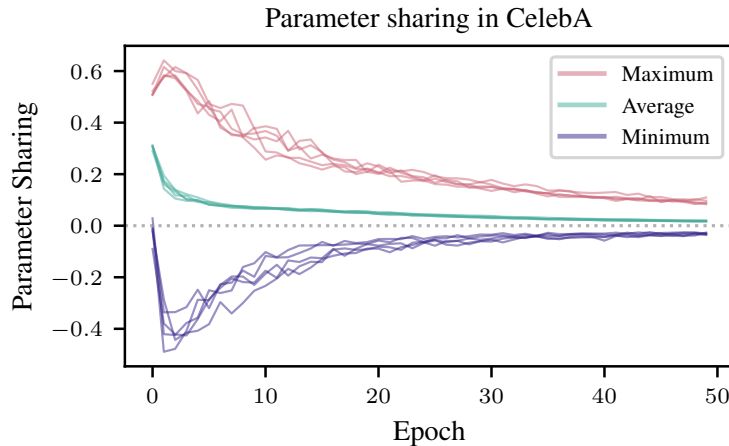


Figure 6.2: Evolution of the pairwise parameter sharing between the 40 tasks of the CelebA benchmark during training, averaged over 5 different runs.

argued that it characterizes negative transfer. This triad is the combination of: **i)** task gradients having negative cosine similarity; **ii)** task gradients significantly differing in magnitude; and **iii)** the optimization landscape having high curvature. Wang et al. [208] later extended this work, and empirically showed that ‘task-gradient (cosine) similarities correlate positively with model quality’. Remarkably, these experiments were conducted in very large models in the context of language translation. Also in the context of multilingual models, Li and Gong [104] similarly found that ‘imbalanced training data poses task interference between high and low resource languages, characterized by nearly orthogonal gradients for major parameters and the optimization trajectory being mostly dominated by high resource ones.’

More works, however, analyse MTL approaches only via their performance. In this direction, recent works have risen some doubts on the effectiveness of gradient-conflict approaches. Specifically, Kurin et al. [99] argued that many existing approaches can be reconsidered as regularization methods, and showed in benchmarks such as Multi-MNIST and CelebA that no method improved the model performance when it was compared to a well-regularized baseline. Similarly, Xin et al. [220] argued that gradient-conflict approaches do not work, as they usually cannot beat their vanilla counterpart on usual MTL benchmarks such as CelebA and NYUv2, when the baseline is properly fine-tuned.

However, we argue that while there is *a priori* no reason to doubt the empirical results reported in these works, the problem may be as well a selection of poorly-suited benchmarks to test MTL approaches, combined with an unclear objective to achieve. Regardless, to make sure that we do not follow this trend, in our experimental setup we perform an exhaustive hyperparameter search where regularization methods (weight decay and early stopping) are taken into account.

6.2 How to measure your MTL model

Previously, we have discussed the difficulties in measuring model performance and gradient conflict, and the ways that prior work have tried to overcome them. In this section, we first motivate the use of simple ranking statistics as an approximation to global MOO methods over CDFs, thus providing a sound metric to evaluate the performance of

[208] Wang, Tsvetkov, Firat and Cao (2021), ‘Gradient Vaccine: Investigating and Improving Multi-task Optimization in Massively Multilingual Models.’ [↗](#)

[104] Li and Gong (2021), ‘Robust Optimization for Multilingual Translation with Imbalanced Data.’ [↗](#)

[99] Kurin, Palma, Kostrikov, Whiteson and Mudigonda (2022), ‘In Defense of the Unitary Scalarization for Deep Multi-Task Learning.’ [↗](#)

[220] Xin, Ghorbani, Gilmer, Garg and Firat (2022), ‘Do Current Multi-Task Optimization Methods in Deep Learning Even Help?’ [↗](#)

6.2.1 Model performance . . . 48

6.2.2 Gradient conflict 50

MTL models. Then, we show that the average SGD agreement can be factorized into two quantities, each of them measuring a different type of gradient conflict, and hence providing more fine-grained metrics to measure gradient conflict.

6.2.1 Model performance

In this section, we demonstrate that simple statistics over the vector of task rankings can be interpreted as finite-sample approximations of MOO global-criterion solutions over CDFs,⁸ and try to convey that these statistics overcome two big problems in MTL model evaluation, namely, task incomparability and the absence of a clear objective to optimize.

For ease of exposition, let us consider a concrete example where we have a 2-task MTL problem, where Figure 6.3 depicts the Pareto front for the two metrics associated with each task. In this example, the tasks are not comparable, as we can observe by noting the different scales of both metrics. Therefore, if we were to, *e.g.*, sum both metrics, those points on the right side of the plot would be highly preferred. We show how to overcome this issue in several steps:

8: We refer to Section 4.4 for a brief introduction to MOO.

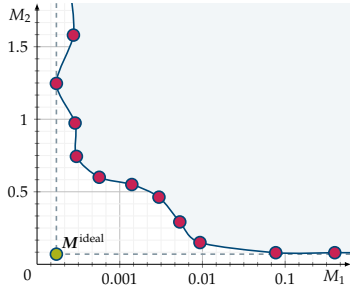


Figure 6.3: Pictorial representation of a metric landscape, the set of Pareto-optimal points, and the ideal point. Note that the x-axis is in log-scale.

CDFs do overcome incomparability. Taking a Bayesian perspective, we can consider each metric as a **random variable (R. V.)**, $M_k \sim P_{M_k}$. Since they are *r.v.s*, they all have an associated CDF, which does not suffer from incomparability issues since:

1. CDFs always exist, and are equally defined for any type of *R. V.*, discrete or continuous, as $F_{M_k}(a) = P_{M_k}(M_k \leq a)$.
2. Their semantics, given by this definition, are easily interpretable: ‘the probability to obtain a value at most as large as a .’
3. They are always defined in the unit interval, are right-continuous, and always reach both ends, *i.e.*, $F_{M_k}(-\infty) = 0$ and $F_{M_k}(\infty) = 1$.

As a consequence, *if we have access to the true CDF*, we have a way of *universally normalizing* our metrics, so that they have the same domain and semantics. That is, we can safely compare the *r.v.s* $U_{M_k} := F_{M_k}(M_k)$ for all k *if we know* F_{M_k} . In the case that all *r.v.s* are continuous, this is known as the probability integral transform, and it can be shown that every U_k then follows a standard uniform distribution [20].

[20] Casella and Berger (2021), ‘Statistical inference.’

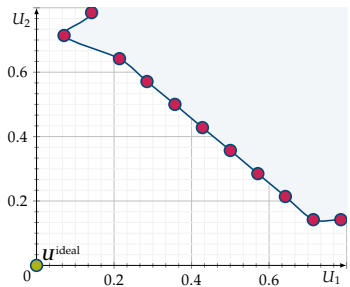


Figure 6.4: Pictorial representation of the CDF landscape after normalizing each point in Figure 6.3.

The ideal point is the origin. From the previous paragraph, it is clear that it is preferable to use U_k instead of M_k . What is not as clear is the location of our ideal point, *i.e.*, the best point that we can reach under utopian circumstances. Recall from Equation 4.5 in Section 4.4 that M^{ideal} is the vector composed of the minimum metric value of each task when individually optimized, *i.e.*, the *smallest value plausible* for each metric:

$$M_k^{\text{ideal}} = \inf \{ M_k \in \mathbb{R} \mid F_{M_k}(M_k) > 0 \}, \quad (6.6)$$

where if we, just as before, transform the ideal (continuous) metrics using their CDFs, $U_k^{\text{ideal}} = F_{M_k}(M_k^{\text{ideal}})$, the equation above becomes

$$U_k^{\text{ideal}} = \inf \{ U_k \in [0, 1] \mid U_k > 0 \} = 0. \quad (6.7)$$

$\uparrow F_{M_k}$ is right-continuous

We depict in [Figure 6.4](#) the result of normalizing the example from [Figure 6.3](#) where now the ideal point is the origin.

Rankings approximate CDFs. Unfortunately, we usually have no access to the true CDF, but only to the metric evaluations for different models as a sequence of *i.i.d.* samples, *i.e.*, $\{M_{k1}, M_{k2}, \dots, M_{kn}\} \stackrel{i.i.d.}{\sim} P_{M_k}$. We are going to show now that their rankings can asymptotically approximate the value of their CDF. To this end, we can sort these samples and define their order statistics, *i.e.*, $M_{kR_k(1)} \leq M_{kR_k(2)} \leq \dots \leq M_{kR_k(n)}$. Here, $R_k(i)$ denotes the ranking of the i -th R.V.,⁹ *i.e.*, its index after sorting the variables, which we can define as $R_k(i) := \sum_{j=1}^n \mathbf{1}_{\{M_{kj} \leq M_{ki}\}}$. Similarly, the *empirical CDF* can be defined by simply counting the number of elements smaller than the input, *i.e.*, $\hat{F}_{M_k}(t) := \frac{1}{n} \sum_{j=1}^n \mathbf{1}_{\{M_{kj} \leq t\}}$.

Therefore, if the input to \hat{F}_{M_k} is one of the samples, M_{ki} , it is direct to check that $\hat{F}_{M_k}(M_{ki}) = R_k(i)/n$, which we can show that it is an unbiased estimator of the true CDF evaluated at that point, $F_{M_k}(M_{ki})$. Specifically, since $\mathbf{1}_{\{M_{kj} \leq M_{ki}\}}$ is a Bernoulli R.V. with $p = F_{M_k}(M_{ki})$, then $R_k(i)$ is a Binomial R.V. with mean $nF_{M_k}(M_{ki})$ and variance $nF_{M_k}(M_{ki})(1 - F_{M_k}(M_{ki}))$. It is important to remark that the relative order between elements is always preserved, *i.e.*, if $F_{M_k}(M_{ki}) < F_{M_k}(M_{kj})$, then $\hat{F}_{M_k}(M_{ki}) < \hat{F}_{M_k}(M_{kj})$ for any sequence containing both samples.

Connection to global methods. We can now put all the results together. Assume that we were to solve a MOO global-criterion problem, *i.e.*,

$$\underset{M}{\text{minimize}} \quad \|M - M^{\text{ideal}}\|_p, \quad (6.8)$$

then, after replacing M_k by U_{M_k} , [Equation 6.8](#) simply becomes

$$\underset{U}{\text{minimize}} \quad \|U\|_p, \quad (6.9)$$

where p is the value of the p -norm chosen by the **decision maker (DM)**.¹⁰ Moreover, we have seen above that we can approximate each element of the random vector using the ranking between samples, *i.e.*,

$$U_{ki} = F_{M_k}(M_{ki}) \approx \frac{1}{n} R_k(i), \quad (6.10)$$

reducing the initial problem in [Equation 6.8](#) to

$$\underset{i \in \{1, 2, \dots, n\}}{\text{minimize}} \quad \|R(i)\|_p = \|[R_1(i) \ R_2(i) \ \dots \ R_K(i)]\|_p, \quad (6.11)$$

i.e., to finding the model whose ranking vector has the smallest statistic.

Statistics and p -norms. We are left with how to choose a value for p . While this is a task for the DM, we can interpret [Equation 6.11](#) similar to L^p -regularization [67]. That is, the optima are the first points intersecting with a ball centred at the origin and with growing radius, where the shape of the ball depends on the choice of p (see [Figure 6.5](#)). In other words, our choice of p indicates which solutions we prefer.

Conveniently, certain values of p are connected with easy-to-interpret statistics. For example, with $p = 1$ we have $\|R(i)\|_1 = \sum_{k=1}^K R_k(i)$, which

9: We use the *uprank* and thus, if there is a tie, both elements get the maximum ranking.

Remark 6.2 Since $R_k(i) \sim \text{Bin}(n, p)$, we have that $\hat{F}_{M_k}(M_{ki})$ has mean

$$\mathbb{E}[\hat{F}_{M_k}(M_{ki})] = \frac{1}{n} \mathbb{E}[R_k(i)] = p,$$

and variance

$$\begin{aligned} \mathbb{V}[\hat{F}_{M_k}(M_{ki})] &= \frac{1}{n^2} \mathbb{V}[R_k(i)] \\ &= \frac{p(1-p)}{n}, \end{aligned}$$

and thus $\mathbb{V}[\hat{F}_{M_k}(M_{ki})] \xrightarrow{n \rightarrow \infty} 0$.

10: *I.e.*, the person making decisions on the model behaviour, see [Section 4.4](#).

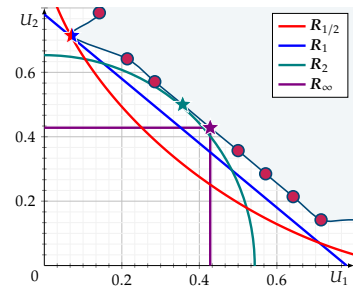


Figure 6.5: CDF landscape, and the p -balls that intersect with the set of plausible metrics for $p \in \{1/2, 1, 2, \infty\}$. The points selected for any of the scalarization functions are marked with a star.

Remark 6.3 The solution selected by our choice of p may not be unique. For example, if we use mean rank ($p = 1$) and the Pareto front is a diagonal line (almost as in Figure 6.5), then all the front intersects with the p -ball, *i.e.*, all points have equal mean rank.

Remark 6.4 Note that the weights in Equation 6.12 can be seen as a variation of the Leximin weights [222], making the problem equivalent to a lexicographic MOO problem where our first priority is to optimize R_∞ , and then R_1 .

equals up to a K factor the *mean rank* statistic previously used in the MTL literature [142, 183]. Similarly, a value of $p = \infty$ results in the maximum rank, *i.e.*, in $\|R(i)\|_\infty = \max_{k \in \{1, 2, \dots, K\}} R_k(i)$, turning Equation 6.11 into a minimax problem to find the most robust model across all tasks.

Robust ranking. Because of the finite number of samples, if we were interested in selecting robust models using R_∞ , the likelihood of obtaining ties can be quite high, specially as the number of tasks increases. To untie elements, we propose what we call a **robust ranking**, R_R , which we define as a combination of two ranking statistics:

$$\|R(i)\|_R := \|R(i)\|_\infty + \frac{1}{n+1} \|R(i)\|_1. \quad (6.12)$$

In the expression above, the second term is always smaller than the smallest increment in the first term. As a result, we have an estimator that still converges to $\|U\|_\infty$ as $n \rightarrow \infty$, but that it prefers the model that performs better on average in case of ties.

6.2.2 Gradient conflict

The following quantities help us understand how much a task shares parameters and contributes to learning them during training. To measure both types of conflict individually, we propose the following:

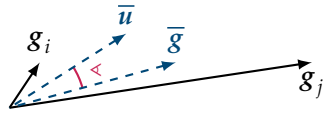


Figure 6.6: Pictorial description of the proposed metric for magnitude conflict.

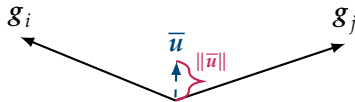


Figure 6.7: Pictorial description of the proposed metric for direction conflict.

Magnitude conflict. To measure the gradient drift caused by magnitude differences between task gradients, we propose to use the cosine similarity of the average task gradient and the average unitary task gradient, *i.e.*, $\cos(\bar{g}, \bar{u})$. Specifically, to make the metric lie in the unit interval, and such that zero and one represent the absence and total presence of conflict, we use instead $(1 - \cos(\bar{g}, \bar{u}))/2$.

Direction conflict. To discard the effect of the gradient magnitudes, we propose to measure the norm of the average unitary gradient, *i.e.*, $\|\bar{u}\|$. This quantity also lies in the unit interval, and we propose to measure $1 - \|\bar{u}\|$ so that intuitively measures the strength of the average gradient in the absence of magnitude conflict. Like the previous metric, in the absence of direction conflict it has a value of one, and zero in the case where gradients are opposite to each other.

[175] Sankararaman, De, Xu, Huang and Goldstein (2020), ‘The Impact of Neural Network Overparameterization on Gradient Confusion and Stochastic Gradient Descent.’

Gradient confusion [175]. The previous metrics summarize the interactions between all tasks. Sometimes, we find useful to use gradient confusion as a finer metric, since it measures the worst direction conflict between all pairs of tasks gradients, $\min_{ij} \cos(g_i, g_j)$.

Training dynamics. With metrics as the ones presented above, we are interested in summarizing their values *during all training*, and not a single iteration. To this end, we propose to measure their signed area under the curve, normalized by the number of training iterations, to take early stopping into account. In practice, we approximate the integral using the trapezoidal rule, and compute the metrics *w.r.t.* θ every 50 iterations

to reduce computational overhead. Every training metric shown in this chapter is thus an average over training.

Relationship to other metrics. The metrics above can look unusual at first. However, we show that they are actually not that exotic, and that they can be related with other metrics from the literature, thus providing them with a richer context and different interpretations. Specifically:

- The **average SGD agreement**, which measures how much on average the total gradient differs from the direction of each task gradient, can be factorized in the product of magnitude and direction conflict,

$$\underbrace{\frac{1}{K} \sum_k \cos(\mathbf{g}_k, \bar{\mathbf{g}})}_{\text{Avg. SGD Agreement}} = \frac{1}{K} \sum_k \frac{\mathbf{g}_k \cdot \bar{\mathbf{g}}}{\|\mathbf{g}_k\| \|\bar{\mathbf{g}}\|} = \left(\frac{1}{K} \sum_k \frac{\mathbf{g}_k}{\|\mathbf{g}_k\|} \right) \cdot \frac{\bar{\mathbf{g}}}{\|\bar{\mathbf{g}}\|}$$

$$= \frac{\bar{\mathbf{u}} \cdot \bar{\mathbf{g}}}{\|\bar{\mathbf{g}}\|} = \underbrace{\frac{\bar{\mathbf{u}} \cdot \bar{\mathbf{g}}}{\|\bar{\mathbf{u}}\| \|\bar{\mathbf{g}}\|}}_{\text{Magnitude Conflict}} \cdot \underbrace{\frac{\|\bar{\mathbf{u}}\|}{\|\bar{\mathbf{g}}\|}}_{\text{Direction Conflict}}. \quad (6.13)$$

- Similarly, we can show that the **average unitary SGD agreement** is exactly the same as direction conflict, thus providing another interpretation for this metric,

$$\frac{1}{K} \sum_k \cos(\mathbf{u}_k, \bar{\mathbf{u}}) = \frac{1}{K} \sum_k \frac{\mathbf{u}_k \cdot \bar{\mathbf{u}}}{\|\mathbf{u}_k\| \|\bar{\mathbf{u}}\|} = \frac{(\frac{1}{K} \sum_k \mathbf{u}_k) \cdot \bar{\mathbf{u}}}{\|\bar{\mathbf{u}}\|}$$

$$= \frac{\|\bar{\mathbf{u}}\|^2}{\|\bar{\mathbf{u}}\|} = \|\bar{\mathbf{u}}\|. \quad (6.14)$$

- Maybe more surprisingly, we can also show that the **average pairwise cosine similarity** between all tasks gradients equals the square direction conflict,

$$\frac{1}{K^2} \sum_{ij} \frac{\mathbf{g}_i \cdot \mathbf{g}_j}{\|\mathbf{g}_i\| \|\mathbf{g}_j\|} = \frac{1}{K} \sum_i \frac{\mathbf{g}_i \cdot \left(\frac{1}{K} \sum_j \frac{\mathbf{g}_j}{\|\mathbf{g}_j\|} \right)}{\|\mathbf{g}_i\|} = \frac{1}{K} \sum_i \frac{\mathbf{g}_i \cdot \bar{\mathbf{u}}}{\|\mathbf{g}_i\|}$$

$$= \|\bar{\mathbf{u}}\| \cdot \frac{1}{K} \sum_i \frac{\mathbf{g}_i \cdot \bar{\mathbf{u}}}{\|\mathbf{g}_i\| \|\bar{\mathbf{u}}\|}$$


$$= \|\bar{\mathbf{u}}\| \cdot \frac{1}{K} \sum_i \frac{\mathbf{u}_i \cdot \bar{\mathbf{u}}}{\|\mathbf{u}_i\| \|\bar{\mathbf{u}}\|} = \|\bar{\mathbf{u}}\|^2, \quad (6.15)$$

where we have used the previous result for the last step.

- Lastly, we also demonstrate that the **cosine stiffness** introduced by Fort et al. [62] is also fully described by the direction conflict,

$$\frac{1}{K} \frac{1}{K-1} \sum_{i \neq j} \frac{\mathbf{g}_i \cdot \mathbf{g}_j}{\|\mathbf{g}_i\| \|\mathbf{g}_j\|} = \frac{1}{K} \frac{1}{K-1} \left(\sum_{i \neq j} \frac{\mathbf{g}_i \cdot \mathbf{g}_j}{\|\mathbf{g}_i\| \|\mathbf{g}_j\|} + K - K \right)$$

$$= \frac{1}{K} \frac{1}{K-1} \left(\sum_{ij} \frac{\mathbf{g}_i \cdot \mathbf{g}_j}{\|\mathbf{g}_i\| \|\mathbf{g}_j\|} - K \right)$$

[62] Fort, Nowak, Jastrzebski and Narayanan (2019), ‘Stiffness: A new perspective on generalization in neural networks.’ 

$$\begin{aligned}
&= \frac{1}{K} \frac{1}{K-1} \sum_{ij} \frac{\mathbf{g}_i \cdot \mathbf{g}_j}{\|\mathbf{g}_i\| \|\mathbf{g}_j\|} - \frac{1}{K-1} \\
&= \frac{K}{K-1} \frac{1}{K^2} \sum_{ij} \frac{\mathbf{g}_i \cdot \mathbf{g}_j}{\|\mathbf{g}_i\| \|\mathbf{g}_j\|} - \frac{1}{K-1} \\
&= \frac{K}{K-1} \|\bar{\mathbf{u}}\|^2 - \frac{1}{K-1}, \tag{6.16}
\end{aligned}$$

where, again, we have used the prior result for the last equality.

6.3 Benchmark probing

6.3.1 Experimental setup . . .	52
6.3.2 Adding additional tasks .	53
6.3.3 Changing the dataset . .	54
6.3.4 Resizing z	55
6.3.5 Increasing the backbone	56
6.3.6 The Conflict benchmark	57

In this section, we demonstrate how the metrics from [Section 6.2](#) can be used to probe the existence of gradient-conflict in an existing benchmark, and refine the benchmark to increase the amount of conflict encountered. To this end, we take the Multi-MNIST benchmark introduced in [Subsection 6.1.2](#) as an example, and refer to its refined version as Conflict. Additional details can be found in [Appendix B.1](#).

6.3.1 Experimental setup and methodology

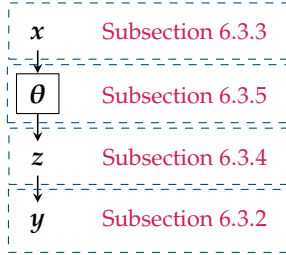


Figure 6.8: Schematic of the components present in the considered MTL pipeline (as in [Section 4.3](#)), and the subsections where we probe each of them.

Table 6.2: Loss and performance metric considered for each type of task.

Type	Loss	Metric
Class.	NLL	Acc.
2-Class.	BCE	F1-score
Regress.	MSE	MSE

Methodology. To refine the existing benchmark, we consider a number of design choices, see [Figure 6.8](#), and study how each of them affect gradient conflict during training. Specifically, since the number of design options grows exponentially, we keep computations tractable by proceeding as follows: **i)** choose the key design elements we wish to evaluate and an order to follow; and **ii)** iteratively ablate each design choice, keeping at each step the one that increases gradient conflict the most.

Experimental setup. Following prior works, we use LeNet [\[101\]](#) as the network backbone, which takes the common input x and produces the last-shared representation z , and simple ReLU networks as task-specific heads, which take z and predict the target y_k . We use Adam [\[92\]](#) as optimizer, train every model for 300 epochs to ensure convergence, and repeat each experiments 30 times. We consider the following types of tasks: **i)** classification, with [negative log-likelihood \(NLL\)](#) as loss and accuracy as task metric; **ii)** binary classification, with [binary cross-entropy \(BCE\)](#) as loss and F1-score as metric; and **iii)** regression, with [mean squared error \(MSE\)](#) as both loss and task metric.

Hyperparameter tuning. We use BOHB [\[55\]](#), a state-of-the-art gray-box algorithm, to tune the hyperparameters of each model in our experiments.¹¹ Specifically, during tuning we run each configuration three times to avoid poor generalization across runs, and tune the learning rate, weight decay, decay rate, and the learning-rate scheduler. Every experiment has a budget of 6 h, and we take the configuration of hyperparameters with the best validation loss.

¹¹: Meaning, a specific network with a gradient-conflict algorithm.

6.3.2 Adding additional tasks

It is broadly-known that different task combinations can have a large impact on the individual task performance in MTL [193, 201] *e.g.*, in task clustering, whose sole purpose is to find the optimal combination of tasks, was covered in Section 4.1. Therefore, it is sensible to think that different task combinations also have different effects on gradient conflict and, in fact, the cosine similarity between task gradients has been previously used as a task clustering criterion [59].

To explore the effect of having different tasks combinations, we consider new tasks for the Multi-MNIST benchmark. Specifically, we consider the following: **i)** the original classification tasks, **L** and **R**; **ii)** classifying the parity of the product of digits, **O**; **iii)** predicting the number of active pixels in the image, **D**; and **iv)** predicting the two-digit number in the image as a regression or classification task, **N** and **B**, respectively. Additionally, we normalize the two regression tasks (**N** and **D**) so that the predictions always lay in the unit interval. As a result, the losses of all classification tasks (**L**, **R**, **O**, and **B**) are of the same order of magnitude on average, while **N** and **D** are, respectively, one and two order of magnitudes smaller. See Appendix B.1.3 for further details.


[59] Fifty, Amid, Zhao, Yu, Anil and Finn (2021), ‘Efficiently Identifying Task Groupings for Multi-Task Learning.’ 

Table 6.3: Type and mathematical description of the tasks considered for the Multi-MNIST dataset.

Name	Type	Expression
Left	Class.	y_L
Right	Class.	y_R
Odd	2-Class.	$y_L y_R \bmod 2$
Density	Regress.	$\sum_{ij} \mathbf{1}_{\{x_{ij} > 0.5\}}$
Number	Regress.	$10y_L + y_R$
Both	Class.	$10y_L + y_R$

Effect of a third task. First, we study the effect that adding another task has on the gradient conflict of the original Multi-MNIST benchmark. We observe in Figure 6.9 that, despite all experiments achieving similar accuracy on the two original tasks (numbers in parentheses), each extra task has a different effect on gradient conflict. Specifically, Figure 6.9 shows a clear difference between adding a classification or regression task. In other words, *having a heterogeneous set of tasks increases gradient conflict*. This effect is also reflected in the task performance of the additional tasks, as we show in the Table 6.4 where, *despite solving the exact same task*, the performance on **B** improves over single-task learning, and that of **N** worsens instead. This result strengthens the importance that the loss landscape has on task conflict.

Table 6.4: Performance on the **Both** and **Number** tasks when included in the Multi-MNIST benchmark, compared with the STL baseline.

	Both \uparrow	Number \downarrow
STL	89.38 ± 0.33	7.08 ± 0.43
LRB	91.06 ± 0.25	-
LRN	-	10.38 ± 0.36

Effect of adding more tasks. Now, we investigate the effect of adding several tasks to the original Multi-MNIST dataset. Interestingly, Figure 6.9 shows that adding **B** as a third task reduces both types of conflict, a trend

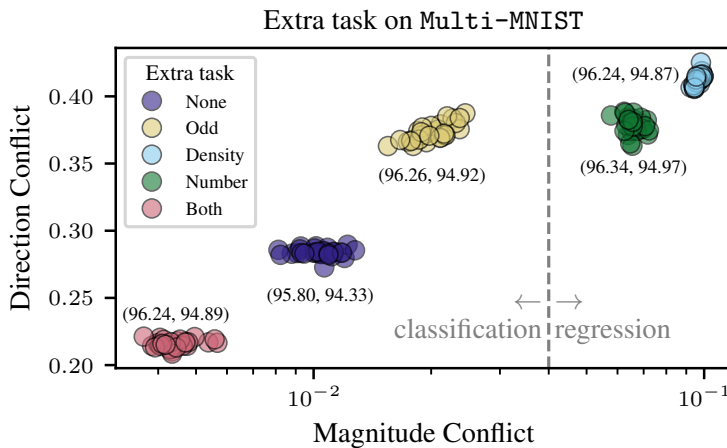
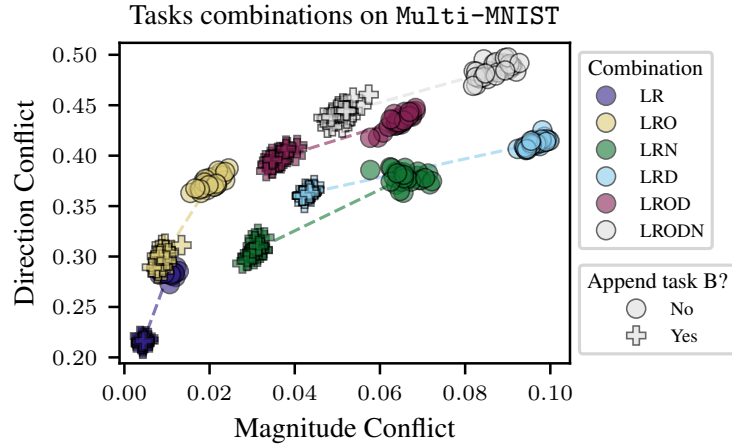


Figure 6.9: Gradient conflict (higher is more conflict) on the Multi-MNIST dataset when an extra task is added. Numbers in parentheses represent performance on the **Left** and **Right** tasks, averaged over 30 runs.

Figure 6.10: Gradient conflict of different extra tasks in the Multi-MNIST benchmark, and the effect of adding the **Both** task as well. Dashed lines connect both cluster pairs to help the reader.



which is consistent with every task combination shown in [Figure 6.10](#). However, the effect of adding additional tasks seems to be in general context-dependent, *e.g.*, adding **O** to the combination **LROD** reduces the magnitude conflict but increases the direction conflict, while adding **N** on top of that increases both types of gradient conflict.

In summary, these results suggest that: **i)** the dynamics of different task combinations during training are rather complex, **ii)** that task performance may not provide enough information to study these effects, and **iii)** that the properties of the loss landscape—determined by the task type and loss function, among others—are an important factor to take into account. For the following experiments, we keep the task combination **LRODN** (white circles in [Figure 6.10](#)) as the default choice.

6.3.3 Changing the dataset

Now, we focus on the effect that different data distributions have on gradient conflict. To this end, we take Multi-MNIST with the tasks **LRODN**, and swap the MNIST dataset with other drop-in replacements. Namely, we consider: **i)** the FMNIST [219] and KMNIST [31] datasets, composed of $C = 10$ classes representing clothes and kanji characters, respectively; and **ii)** the Letter and Balanced splits from the EMNIST dataset [33], both composed of letters and, in the case of Balanced, also digits. Since the input size remains the same, we simply replace MNIST by one of the aforementioned datasets, changing the last layer of the **L** and **R** tasks to match the number of classes when necessary.

It is important to remark that, since the EMNIST splits have a slightly different format, we need to take these differences into account when running the experiments. Firstly, these datasets contain more training samples, and thus we make sure that all experiments have converged by running a sufficiently large number of epochs. Similarly, we normalize all training metrics by the number of total iterations to make them comparable (see [Subsection 6.2.2](#)). Secondly, since the Letter dataset has $C = 37$ and Balanced $C = 47$ classes, we normalize the regression task **N** by the total number of classes. See [Appendix B.1.4](#) for details.

Relationship between problem complexity and gradient conflict. We summarize the results in [Figure 6.11](#). Namely, [Figure 6.11a](#) shows that

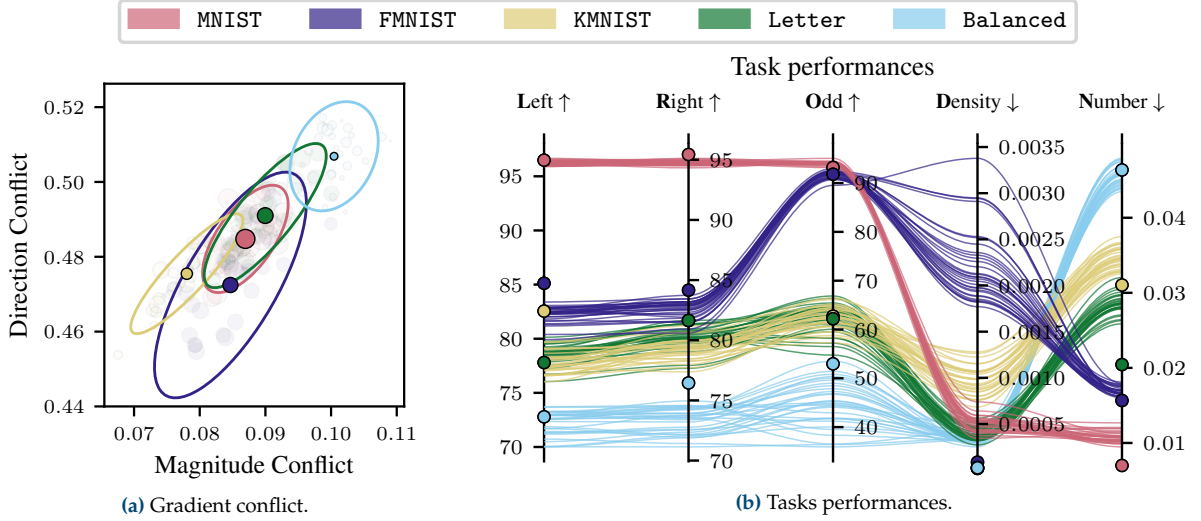


Figure 6.11: Effect on the Multi-MNIST benchmark when changing the dataset on: (a) gradient conflict during training; and (b) task performances. Marker size on (a) are proportional to R_R values across datasets (bigger means better performance), and markers on (b) represent the performance of STL in each dataset. Confidence ellipses on (a) cover points two standard-deviations away from the average.

KMNIST and FMNIST exhibit less conflict than MNIST, while Balanced shows significantly more gradient conflict than any other considered dataset. Moreover, the marker sizes in Figure 6.11a represent the performance of each experiment with respect to the rest,¹² and we can observe that there is no apparent relationship between problem difficulty and the amount of gradient conflict: best and worst-performing experiments are scattered all across the gradient-conflict spectrum. More in detail, if we look at the gap between STL and MTL for each task and dataset in Figure 6.11b, we see that each dataset is particularly difficult for particular tasks, *e.g.*, FMNIST on **D**, Letter on **N**, or Balanced on **O**.

12: Measured via R_R over the task relative performance, more details in Appendix B.1.5.

Other determining factors. We hypothesize that, just as shown in Subsection 6.3.2, two factors determine the amount of conflict in these experiments: **i)** the loss landscape, which is affected by the data distribution; and **ii)** the number of tasks, which is affected by the number of classes in the dataset. To expand more on the latter, Figure 6.11a shows a correlation between number of classes and gradient conflict and, similar to the CelebA setup, we can interpret each C -classification task (**L** and **R**) as C binary-classification tasks. Therefore, Letter and Balanced would increase conflict by introducing additional tasks.

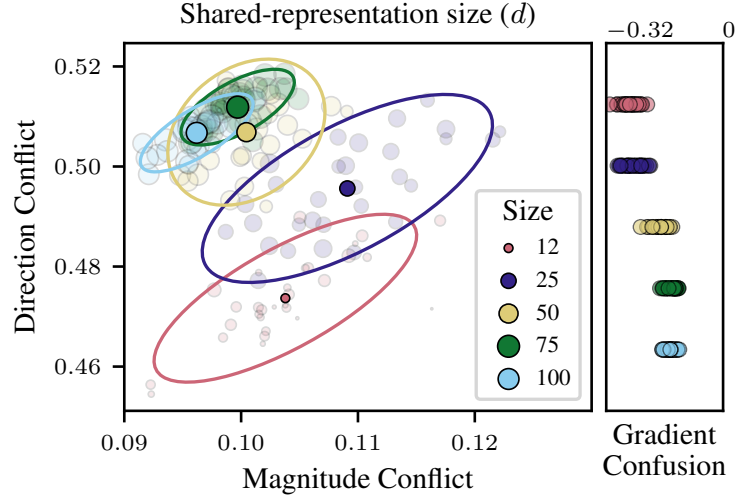
In short, this experiment reinforces again the idea that the loss landscape is a key factor in the amount of conflict present during training, and hints at architectural changes¹³ as another important component to better understand gradient conflict. From here onward, we consider Balanced as the default dataset for every experiment.

13: As a result from increasing the number of classes, C .

6.3.4 Resizing the last shared representation

We explore now explicit architectural changes on the network and, specifically, in this section we study how changing the dimensionality of the last shared representation affects gradient conflict. To this end, we take the most conflictive experiment from the previous section, and

Figure 6.12: Gradient conflict and confusion as we change the dimension of \mathbf{z} , the last shared representation, averaged over 30 runs. Marker size represents R_R performance (bigger markers perform better). Confidence ellipses cover two standard-deviations from the average.



consider a shared representation, \mathbf{z} , of size $d \in \{12, 25, 50, 75, 100\}$, where $d = 50$ is the value used in previous experiments.

Effect of d on gradient conflict. Figure 6.12 shows the gradient conflict and confusion as we increase d , and where the marker size indicates its R_R performance *w.r.t.* the rest of experiments. First, we observe that there is a clear positive correlation between the size of d and model performance, and a negative one on the variance of all measurements. As for gradient conflict, we observe a consistent decrease in gradient confusion as we increase d , and an interesting non-linear relationship with respect to both magnitude and direction conflict: as we increase d , both measures increase to then decrease.

These observations suggest the existence of a ‘sweet spot’ where gradient conflict is maximized. However, reducing d could render the network unable to learn the tasks at all. In the extreme case where $d = 1$, we would have a direction conflict of either 1 or -1 , but all the shared information would be reduced to a single number. If, instead, d is sufficiently large, the network could distribute the information for each task in different components of \mathbf{z} , effectively having independent subnetworks, as we saw in Subsection 6.1.2. Therefore, we continue the following experiments with $d = 25$ as a good compromise between gradient conflict and model performance.

6.3.5 Increasing the backbone size

Lastly, we study how the number of shared parameters affect gradient conflict. To this end, we follow the same parametrization for the backbone as Ruchte and Grabocka [167], where a multiplicative factor c determines the number of backbone parameters. We provide a detailed explanation of the architecture in Appendix B.1.2. Following up the setup from the last section, we consider now backbones with capacity $c \in \{0.1, 0.25, 0.5, 0.75, 1.0, 1.25, 1.50\}$, being $c = 1.0$ the value in previous experiments.

[167] Ruchte and Grabocka (2021), ‘Multi-task problems are not multi-objective.’

Results. Figure 6.13 shows the gradient conflict and confusion as we change the backbone capacity. We observe a clear difference in

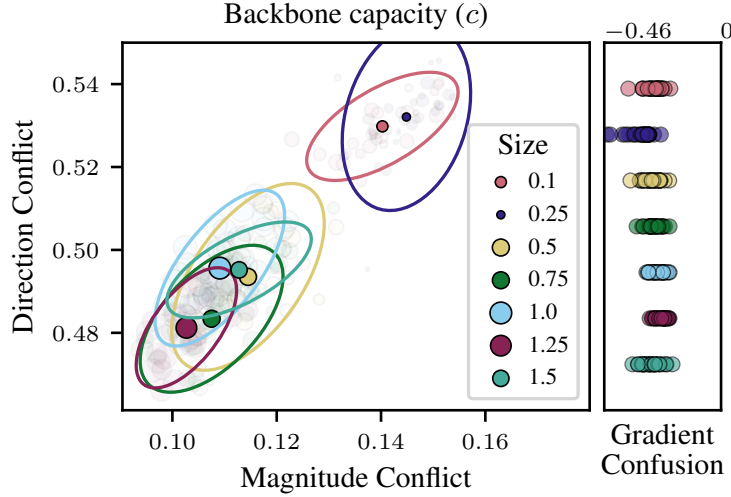


Figure 6.13: Gradient conflict and confusion as we change the backbone-capacity parameter, c , averaged over 30 runs. Marker size represents R_R performance (bigger means better performance). Confidence ellipses cover two standard-deviations from the average.

gradient conflict and performance (marker sizes) between the two smallest backbones, $c \in \{0.1, 0.25\}$, and the rest. However, once that the backbone has enough capacity, we do not observe any clear differences, *i.e.*, the larger cluster seems to not follow any particular pattern with respect to c . Interestingly, we observe that the biggest backbone, $c = 1.5$, performs poorly in terms of model performance, and exhibits relatively high gradient conflict among well-performing models. As little difference as there is, we choose to ensure that the model has enough capacity to learn all tasks appropriately, and use $c = 1.5$ for the rest of experiments.

6.3.6 The Conflict benchmark

We collect all the changes carried out in the Conflict benchmark, a refinement of the Multi-MNIST benchmark, result of probing the effect of different design choices on the metrics from Section 6.2. Therefore, Conflict serves as a practical example on how to systematically increase the gradient conflict on an existing benchmark, making it more suitable for testing MTL approaches, as we corroborate later in Section 6.4. Moreover, we have also obtained insights on gradient conflict that, although not necessarily generalizable to every scenario, help us better understand the role of gradient conflict. As a summary of Conflict:

- **Tasks: LRODN.** We observed in Subsection 6.3.2 that the loss landscape can be as important as the relationship between tasks, since highly correlated tasks can still increase conflict (*e.g.*, tasks **O** and **N**). Moreover, having heterogeneous tasks seems crucial to increase gradient conflict.
- **Dataset: Balanced.** In Subsection 6.3.3 we found no correlation between gradient conflict and the complexity of the problem to solve. Again, the loss landscape¹⁴ seemed to be the main factor behind gradient conflict.
- **Size of \mathbf{z} : $d = 25$.** Working as a bottleneck, we found in Subsection 6.3.4 that the size of \mathbf{z} is an important factor for gradient conflict, and that there seems to follow a non-linear relationship. We therefore chose a compromise option with significant conflict and good performance.

Table 6.5: Summary of the differences between Multi-MNIST and Conflict.

Parameter	Multi-MNIST	Conflict
Tasks	LR	LRODN
Dataset	MNIST	Balanced
d	50	25
c	1	1.5

14: Via the input distribution and the number of output classes.

- **Backbone capacity:** $c = 1.5$. We found in Subsection 6.3.5 that the number of shared parameters does not have a clear effect on gradient conflict, once that the backbone is large enough. To ensure that this is the case, we took the largest backbone in our experiments.

6.4 Empirical validation

6.4.1 Benchmark comparison .	58
6.4.2 GC approaches	59
6.4.3 Homogenizing θ v.s. z .	61

In Section 6.3, we took the Multi-MNIST benchmark, and showed how to systematically modify it to make it more suitable for MTL benchmarking, which we summarized in the Conflict benchmark. In this section, we empirically validate Conflict, and show the advantages of evaluating gradient conflict approaches with Conflict instead of Multi-MNIST.

We follow the same experimental setup as in Section 6.3. However, it is important to remark that we tune all the hyperparameters for each gradient-conflict approach independently, to make sure that tuning bias is not an issue [99].

[99] Kurin, Palma, Kostrikov, Whiteson and Mudigonda (2022), ‘In Defense of the Unitary Scalarization for Deep Multi-Task Learning.’ [📄](#)

6.4.1 Comparing Multi-MNIST and Conflict

First, we show the extent to which Conflict is different from the original Multi-MNIST benchmark. That is, we want to verify that Conflict provides richer interactions between tasks during training, and therefore that it is better suited for testing MTL approaches.

In Figure 6.14, we provide a summary of the quantitative differences between both benchmarks, showing that the improvement margin in comparison with STL is bigger in Conflict and, moreover, Conflict exhibits more magnitude and direction conflict, despite having more parameter sharing than Multi-MNIST. It is important to recall that Conflict have three additional tasks, which are also used to compute the three last metrics on the figure.

We further explore the interactions between tasks in terms of parameter sharing¹⁵ in Figure 6.15, showing the pairwise interactions at the parameter level (θ , upper diagonal), at the shared-representation level (z , lower diagonal), as well as the individual SGD agreement (θ , diagonal), as we defined them in Subsection 6.1.1. We show these metrics

15: Or, equivalently, in terms of gradient cosine similarities.

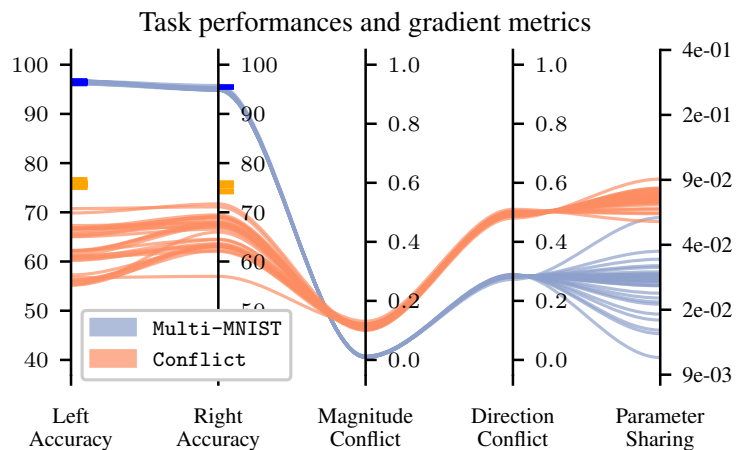


Figure 6.14: Summary of task performance and conflict measures for the Multi-MNIST and Conflict benchmarks. Each coloured tick represents task performance of a single-task model.

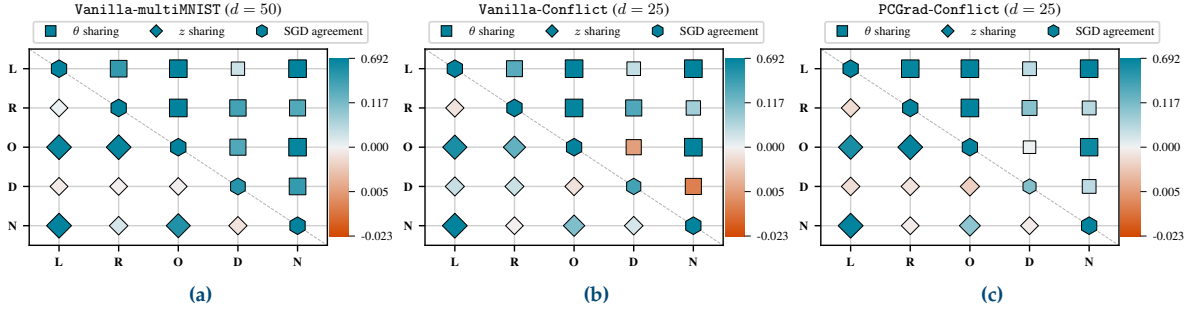


Figure 6.15: Interaction of different tasks during training in the Multi-MNIST (a) and Conflict benchmarks, (b) and (c). Both size and colour represent the magnitude of the metric. Specifically, diagonal elements represent SGD agreement, *i.e.*, the contribution of each task in the gradient followed; upper diagonal elements represent parameter sharing *w.r.t.* the shared parameters θ ; and lower diagonal elements *w.r.t.* the last shared representation, z . We observe that Multi-MNIST shows no conflict at all, and PCGrad reduces the conflict compared with the vanilla approach in Conflict. Best viewed in colour.

for three different experiments: (a) the original Multi-MNIST benchmark with the LRODN tasks,¹⁶ (b) the proposed Conflict benchmark; and (c) Conflict using PCGrad [227], a gradient-conflict approach.

16: As in Subsection 6.3.2.

As we observed in Subsection 6.1.2, we see in Figure 6.15a that Multi-MNIST exhibits no conflict anywhere, *i.e.*, either tasks share parameters, or they do not interact at all. In contrast, we see in Figure 6.15b that in Conflict there are more interactions between tasks, *e.g.*: i) D now conflicts at the parameter level with the O and N tasks; and ii) the positive interactions between gradients have lost strength with respect to Multi-MNIST. Additionally, we observe in Figure 6.15c that PCGrad works as intended, *i.e.*, we observe that PCGrad has: i) removed any conflict between D and the other tasks; and ii) increased the cooperation of, *e.g.*, tasks L and R.

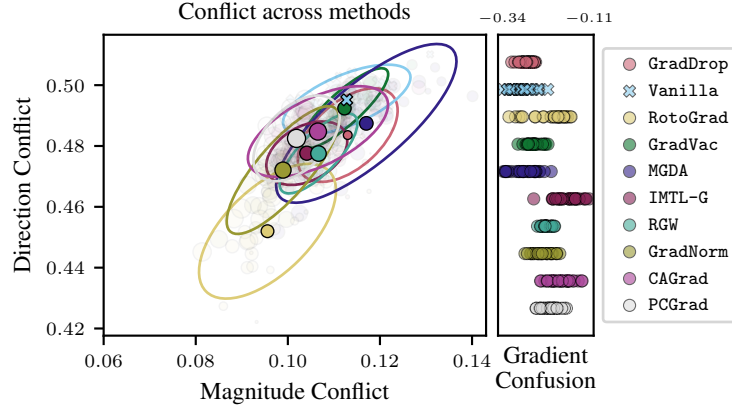
Therefore, Figure 6.15 shows that the gradient dynamics in Conflict are richer than those in the original Multi-MNIST benchmark. Additionally, it also empirically shows that these dynamics do not need to match when we compute them *w.r.t.* θ or z , *e.g.*, the negative interactions in Figure 6.15b only appear in one side of the diagonal. This suggests that working with the gradients *w.r.t.* z to reduce computational overhead [26, 27, 182] may not always produce the expected results.

6.4.2 Gradient-conflict approaches in Conflict

We now turn our attention to gradient-conflict approaches, which we briefly reviewed in Subsection 4.6.1. Specifically, we explore whether existing approaches are significantly different from the baseline (that we call the Vanilla approach), when applied to the Conflict benchmark.

Considered methods and metrics. Here, we focus on gradient-conflict methods that dynamically modify the gradients during training, as it requires minimal changes to Conflict, and consider the following approaches: GradNorm [26]; GradDrop [27]; PCGrad [227] and its extension, GradVac [208]; IMTL-G [112]; MGDA [45] and its extension, CAGrad [111]; RGW [106]; and RotoGrad from Chapter 5. For each specific method, we accordingly apply them to modify the gradient *w.r.t.* the shared parameters, θ , or *w.r.t.* the last-shared representation, z , depending

Figure 6.16: Gradient conflict of different approaches on the Conflict benchmark, averaged over 30 runs. Marker size represents R_R performance (larger means better performance).



on their original implementation. Regarding MTL performance metrics, we consider those discussed in [Subsection 6.1.1](#) and [Section 6.2](#): robust ranking, R_R ; average ranking, R_1 ; relative task improvement, $\text{avg } \Delta$; and we additionally consider $\text{med } \Delta$, where we take the median across tasks to make it more robust.

Results. [Figure 6.16](#) shows the gradient conflict and confusion for each method, and where the marker size indicates R_R performance. [Table 6.6](#), instead, presents the quantitative performance of each method per task. First, note that Vanilla performs significantly worse than other approaches (e.g., PCGrad) which is consistent with their reduction in gradient conflict, as shown in [Figure 6.16](#). However, RotoGrad reduces the most conflict, yet it obtains average results; after inspecting [Table 6.6](#), we see that it performs great on the two dominated regression tasks at the expense of the digit-classification tasks, which we can attribute to the aggressive behaviour of its scaling algorithm.¹⁷ Overall, we observe that with Conflict different methods result in distinguishable gradient-conflict and MTL performance values with respect to Vanilla.

17: This ill-behaviour was previously mentioned in [Subsection 5.2.2](#).

With Conflict, we can now compare also different approaches and their follow-ups. We observe that CAGrad consistently beats MGDA in conflict and MTL performance, by not focusing solely on the **O** task. Interestingly, PCGrad instead significantly outperforms GradVac, and that this is also reflected in the gradient conflict from [Figure 6.16](#). Moreover, in both cases, the best method also has the least gradient conflict.

Table 6.6: Task and MTL performance of existing gradient-conflict approaches when applied to the Conflict benchmark. Highlighted numbers indicate that their average \pm one standard deviation do not overlap with the average \pm one standard deviation of Vanilla. The MSE values of **Density** and **Number** are scaled by 10^4 and 10^3 , respectively. Average and standard deviation over 30 runs.

Method	Task Metrics					MTL Performance			
	Left \uparrow	Right \uparrow	Odd \uparrow	Density \downarrow	Number \downarrow	$R_R \downarrow$	$R_1 \downarrow$	med $\Delta \uparrow$	avg $\Delta \uparrow$
STL	75.68 \pm 0.44	75.02 \pm 0.80	53.83 \pm 2.62	0.22 \pm 0.00	46.67 \pm 2.73	-	-	-	-
PCGrad [227]	64.22 \pm 2.62	65.34 \pm 2.72	44.42 \pm 2.71	0.61 \pm 0.10	51.59 \pm 2.11	5.13 \pm 1.31	3.45 \pm 1.10	-15.47 \pm 3.39	-46 \pm 9
CAGrad [111]	63.90 \pm 4.13	64.46 \pm 5.12	41.39 \pm 3.32	0.28 \pm 0.03	52.46 \pm 3.37	6.01 \pm 1.84	3.47 \pm 1.79	-15.79 \pm 5.81	-18 \pm 6
GradNorm [26]	60.78 \pm 2.99	62.11 \pm 2.03	40.04 \pm 3.89	5.85 \pm 1.18	55.34 \pm 2.36	7.27 \pm 1.02	5.85 \pm 1.09	-20.98 \pm 5.35	-524 \pm 107
RGW [106]	65.44 \pm 1.49	66.51 \pm 1.36	38.18 \pm 3.26	6.54 \pm 1.07	54.43 \pm 1.89	7.65 \pm 0.76	4.65 \pm 0.69	-18.23 \pm 4.99	-585 \pm 98
IMTL-G [112]	55.00 \pm 7.15	58.25 \pm 6.07	51.68 \pm 3.23	0.30 \pm 0.05	48.78 \pm 4.09	7.76 \pm 2.30	3.90 \pm 1.42	-19.39 \pm 8.08	-18 \pm 8
MGDA [45]	56.41 \pm 6.94	64.38 \pm 3.40	52.38 \pm 3.73	7.88 \pm 1.54	55.23 \pm 4.87	8.27 \pm 1.38	5.17 \pm 1.74	-18.80 \pm 6.61	-703 \pm 141
RotoGrad	55.52 \pm 8.51	51.32 \pm 1146	48.74 \pm 4.28	0.49 \pm 0.39	48.88 \pm 4.86	8.31 \pm 2.09	4.45 \pm 1.45	-23.61 \pm 7.85	-38 \pm 36
GradVac [208]	61.99 \pm 2.26	63.18 \pm 2.26	39.24 \pm 3.87	2.56 \pm 5.19	62.55 \pm 2.30	8.82 \pm 0.71	5.89 \pm 0.93	-25.80 \pm 6.99	-228 \pm 463
Vanilla	62.13 \pm 4.61	65.89 \pm 3.24	37.95 \pm 6.87	8.79 \pm 0.83	51.80 \pm 3.21	8.98 \pm 0.71	5.28 \pm 1.93	-18.08 \pm 6.50	-788 \pm 76
GradDrop [27]	57.96 \pm 3.51	60.50 \pm 2.61	29.09 \pm 4.60	7.93 \pm 1.58	64.35 \pm 3.45	9.57 \pm 0.42	8.00 \pm 0.64	-36.60 \pm 7.49	-721 \pm 138

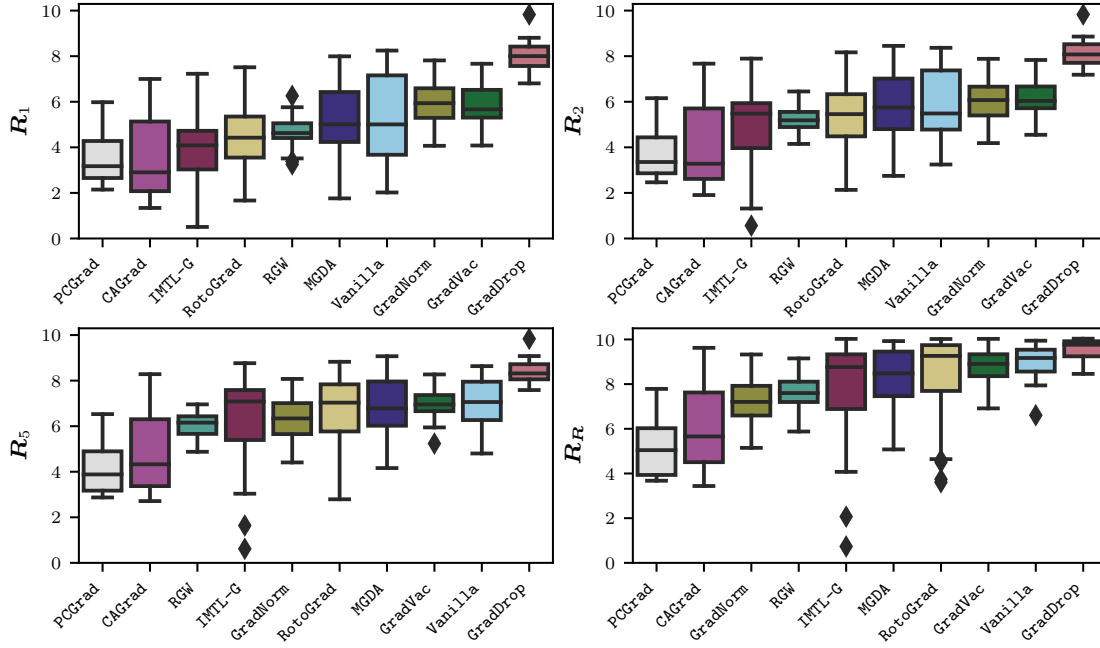


Figure 6.17: Box plots of different gradient-conflict approaches applied to the Conflict benchmark, averaged over 30 runs, according to four different rankings from [Subsection 6.2.1](#).

MTL performance metrics. These experiments also validate the use of R_R over the widely adopted $\text{avg } \Delta$. Besides the theoretical motivation introduced in [Subsection 6.2.1](#), we can observe in [Table 6.6](#) and [Figure 6.16](#) that it also positively correlates with performance improvement and reduction in gradient conflict, respectively.¹⁸ [Table 6.6](#) also shows important shortcomings of Δ , *e.g.*: $\text{avg } \Delta$ clearly focuses on **D** task performance (since it has comparatively small values), whereas $\text{med } \Delta$ shows no significant differences between most methods.

18: Beyond exceptions like RotoGrad, which can be understood by looking at the gradient confusion.

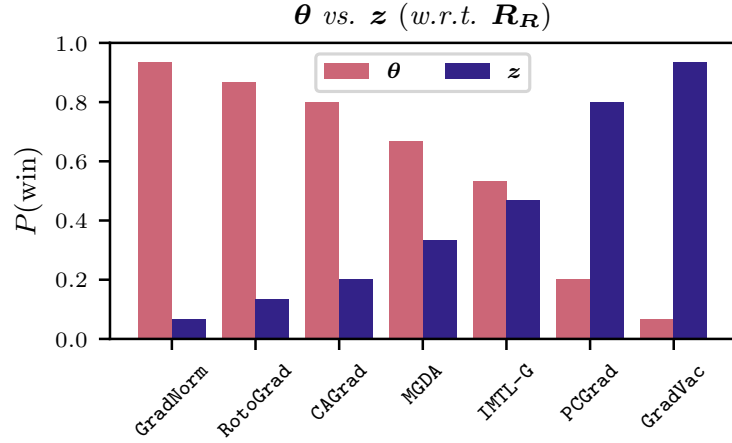
As discussed in [Subsection 6.2.1](#), it is important to have a clear objective to achieve. To emphasize this point, we show in [Figure 6.17](#) the performance of all methods in [Table 6.6](#) according to four rankings: R_1 , R_2 , R_5 , and R_R . As we discussed, the differences between rankings (and thus, objectives) can be dramatic, *e.g.*, GradNorm is among the worst methods based on mean rank, R_1 , but among the best according to R_R and, in contrast, the high variance of RotoGrad in this benchmark makes it a solid method on average, but a weak one if we care about best worst performance.

Combining the evidence gathered in the experiments above, we can clearly see that, when tested on an appropriately designed benchmark, different well-tuned gradient-conflict approaches do have distinguishable characteristics on both gradient conflict and MTL performance, which is in stark contrast to those analyses previously discussed in [Subsection 6.1.3](#).

6.4.3 Where to solve the gradient conflict

To complete the previous results, we now show how a benchmark like Conflict can be used to shed light on unanswered questions. Specifically, we are interested in understanding whether, by solving the conflict of the gradients *w.r.t.* \mathbf{z} instead of $\boldsymbol{\theta}$, we can obtain similar improvements while reducing the overhead, as we discussed in [Subsection 4.6.1](#). To the best

Figure 6.18: Probability of different approaches to obtain better MTL performance when applied to the gradients *w.r.t.* the shared parameters, θ , or the shared feature, z , measured using R_R and computed over 30 different runs.



of our knowledge, there is still no experiment in the existing literature trying to answer which of the two approaches is preferable, if any.

19: Except for GradDrop and RGW.

To this end, we consider most methods from previous section¹⁹ and benchmark them once more, this time applying them on the opposite gradients from those they were initially conceived to work with. Then, we compare the original method and its counterpart according to R_R performance, and average the number of times each method wins over 30 runs. In the case of RotoGrad, we keep the rotation matrices on the heads, and only change the scaling algorithm (see Chapter 5).

Results. Figure 6.18 summarizes the results for each approach. First, we can observe that there are methods on all the spectrum of probabilities, so we can say that no approach is preferred, and that there is *a priori* no way of knowing whether a new method will work better in θ or z . However, there seems to exist some trend, as the two projection-based methods (PCGrad and GradVac) are consistently better when applied on z . Similarly, all scaling methods work better when directly applied to the gradients *w.r.t.* θ . Of holding this trend on other benchmarks, this could be a strong indicator that solving direction conflict at the shared-feature level is more effective, and that solving magnitude direction is more beneficial at the parameter level, which would certainly expand our understanding on the nature of gradient conflict.

6.5 Concluding remarks

In this chapter, we have drawn our attention to the problem of measuring and benchmarking in MTL, heavily exacerbated by task incomparability, *i.e.*, to the difficulty of making meaningful comparisons across tasks, both at evaluation and training time. To overcome these issues, we have provided better metrics to compare the performance of MTL models at evaluation time, and to measure the conflict among tasks gradients at training time. We have connected rank metrics to MOO global methods over CDFs, hence addressing task incomparability while making explicit the goal we attempt to achieve.

Moreover, we have shown that this inability to perform meaningful comparisons can have a significant impact on our conclusions, as benchmarks

and methods are developed without accounting for gradient conflict. With the proposed metrics at hand, we have shown through a practical example how to probe an existing benchmark, systematically increasing the amount of gradient conflict found during training by studying how different design choices affect gradient conflict. Finally, to validate the resulting Conflict benchmark, we have empirically shown that it presents richer relationships between tasks, and that gradient-conflict approaches are quantitatively different when tested on it, defying recent analyses on existing MTL benchmarks.

We consider the construction of Conflict an illustrative example, and we are aware that the insights obtained here may not generalize to all settings. Therefore, to tackle fundamental MTL problems such as gradient conflict, we consider it important to revisit existing benchmarks and experiments, gaining a deeper understanding on the design factors to address, and thus enabling more effective MTL benchmarking.

While we have proven the proposed gradient conflict metrics to be useful tools towards these goals, one should be aware that we still summarize all training interactions to a pair of values, inevitably losing information in the process, and thus finer future analyses will ultimately require more accurate metrics. One way to alleviate this issue is to complement the proposed metrics with others, such as the recently proposed rAU [223] which measures the conflict between tasks with respect to individual model parameters, greatly enriching the MTL toolbox.

[223] Yang, Pan, Wang, Yu, Shen, Chen, Xiao, Jiang and Guo (2023), 'AdaTask: A Task-Aware Adaptive Learning Rate Approach to Multi-Task Learning.' [🔗](#)

Part II.

Probabilistic Generative Models



Deep Learning and Probabilistic Modelling

7.

Hay, empero, otra noción de patria.
No la tierra de los padres, como decía
Nietzsche, sino la tierra de los hijos.

José Ortega y Gasset

7.1 Problem statement	67
7.2 Exponential family	68
7.3 Latent-variable models . .	69
7.4 Modality collapse & MTL .	73

In this part, we consider the problem of modelling the joint distribution of the observed dataset. This is a significant change with respect to **Part I**, as we have moved from a supervised to an unsupervised, probabilistic setting. However, we will see in these next chapters that the two settings are not only related, but that we can draw strong connections between both formulations, and that we can indeed reuse from **multitask learning (MTL)** the terminology, knowledge, and approaches, to significant improve the performance of **probabilistic generative models (PGMs)**.

In this chapter, we first describe the problem statement, then lay the foundations necessary to understand the following chapters and, lastly, provide a first view on the connections between MTL and PGMs. Upon understanding this connection, we will see that they are indeed quite similar, and that they share many inductive biases.

7.1 Problem statement

Similar to the problem statement from **Part I**, we assume that there is a D -dimensional **random variable (R. V.)** $\mathbf{x} := [x_1 \ x_2 \ \dots \ x_D] \sim P_{\mathbf{x}}$. Again, we assume access to an input dataset $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N \stackrel{i.i.d.}{\sim} P_{\mathbf{x}}$ composed of N *i.i.d.* samples from \mathbf{x} . Note that we do not make any assumptions on the modalities, allowing for \mathbf{x}_d of different shapes (e.g., images and their labels) and types (e.g., continuous *v.s.* discrete variables).

The goal of this part of the thesis is to have a **deep learning (DL)** model (generally a neural network) with parameters θ that models the joint distribution of the R. V. \mathbf{x} , *i.e.*,

$$p_{\mathbf{x}}(x_1, x_2, \dots, x_D) \approx p_{\theta}(x_1, x_2, \dots, x_D), \quad (7.1)$$

where $p_{\mathbf{x}}$ is ground-truth density followed by \mathbf{x} , and p_{θ} the approximation obtained by our probabilistic model with parameters θ .

Multimodal data. In this part, we focus on multimodal data, *i.e.*, on data that contains information coming from different sources. That is, we consider each \mathbf{x}_d to represent the data coming from the d -th modality. Furthermore, we refer to \mathbf{x} as a **heterogeneous R. V.** when each modality \mathbf{x}_d in \mathbf{x} is unidimensional. Thus, we consider heterogeneous data as a special case of multimodal data.¹ In a slight abuse of notation, we use D for both the number of modalities and the input size, as it will be clear from the context and it does not change the results presented here.

In probabilistic **machine learning (ML)**, the usual practice when modelling the joint distribution $p_{\mathbf{x}}$ is to assume a fixed family of distributions for

Dimensions

$$\mathbf{x} \in \mathbb{R}^D \quad \mathbf{X} \in \mathbb{R}^{N \times D}$$

Remark 7.1 The notation $p_{\theta}(\mathbf{x})$ and $p(\mathbf{x}; \theta)$ are equivalent and we use them interchangeably, depending on whether we want to emphasize the role of the parameters or not.

1: In the literature, however, heterogeneous and multimodal problems are often studied independently [143, 185].

p_{θ} —e.g., Gaussian distributions—while a neural network outputs the values for the distributional parameters, η . Moreover, as we make no assumptions on the modality types, the usual practice is to assume that the joint likelihood fully factorizes across modalities, *i.e.*,

$$p(\mathbf{x}; \boldsymbol{\theta}) = \prod_{d=1}^D p_d(\mathbf{x}_d; \boldsymbol{\eta}_d(\boldsymbol{\theta})), \quad (7.2)$$

where $\boldsymbol{\eta}_d(\boldsymbol{\theta})$ is the output of the network parametrized by $\boldsymbol{\theta}$, and p_d describes the likelihood of the R. V. \mathbf{x}_d .

Selecting the proper likelihood for a modality is, in general, a difficult task that requires specific domain knowledge for each use case. For example, in the multimodal applications in [Chapter 9](#), we choose multivariate Laplace distributions for the image modalities. In an attempt to automatize likelihood selection, when working with heterogeneous data in [Chapters 8 and 9](#), we select the likelihood by looking at easy-to-check properties of the modality domain. Namely:

Property	Distribution of \mathbf{x}_d
Real-valued:	$\mathcal{N}(\mu, \sigma)$
Positive real-valued:	$\text{Log-normal}(\mu, \sigma)$
Count data:	$\text{Poiss}(\lambda)$
Binary:	$\text{Bern}(p)$
Categorical:	$\text{Cat}(\pi_1, \pi_2, \dots, \pi_D)$

7.2 Exponential family

In this section, we provide a quick introduction to the exponential family, as it appears ubiquitously in the rest of this thesis. Indeed, every distribution we consider will be a member of the exponential family.

The exponential family is a parametric set of distribution functions whose densities have the following form:²

$$p(\mathbf{x}; \boldsymbol{\eta}) = h(\mathbf{x}) \exp [\boldsymbol{\eta}^\top \mathbf{T}(\mathbf{x}) - A(\boldsymbol{\eta})], \quad (7.3)$$

2: For simplicity, we introduce the uni-dimensional case.

3: Note that additional constraints on the values that $\boldsymbol{\eta}$ can take are possible.

where: **i)** $\boldsymbol{\eta} \in \mathbb{R}^I$ are the natural parameters of size I ;³ **ii)** $\mathbf{T}(\mathbf{x}) \in \mathbb{R}^I$ are the sufficient statistics, which contain all the information of $\boldsymbol{\eta}$ in the data; **iii)** $h(\mathbf{x}) \in \mathbb{R}$ is the base measure, which restricts the space where \mathbf{x} can take values; and **iv)** $A(\boldsymbol{\eta}) \in \mathbb{R}$ is the log-partition function, which ensures that the distribution integrates to one.

One parametric member of the exponential family is fully determined by the selection of \mathbf{T} and h , whereas $\boldsymbol{\eta}$ determines one specific distribution of the family. For example, the Normal distribution forms an exponential family with $\mathbf{T}(\mathbf{x}) = [\mathbf{x} \ \mathbf{x}^2]$ and $h(\mathbf{x}) = 1/\sqrt{2\pi}$ and, similarly, the truncated Normal distribution within the interval $[a, b]$ also forms an exponential family with same \mathbf{T} and $h(\mathbf{x}) = \mathbf{1}_{\{a \leq \mathbf{x} \leq b\}}/\sqrt{2\pi}$.

The exponential family is the most-studied family of distributions in statistics, as it enjoys uncountable nice properties such as: **i)** forming the ‘probability manifold’ that lies the foundations of information geometry [\[4\]](#); **ii)** having conjugate priors, vital for the development of

[\[4\]](#) Amari and Nagaoka (2000), ‘Methods of information geometry.’

Bayesian statistics; or **iii)** being the only family in the real line with sufficient statistics of finite dimension, given infinite amount of data (Pitman-Koopman-Darmois theorem). For our purposes, however, it suffices to say that it is a family analytically nice to work with, as well as flexible enough to model most real-world observed data.

In statistics, it is a common practice to not work with η directly, but rather use an alternative parametrization θ , *e.g.*, the mean μ and standard deviation σ in the Normal distribution, such that $\eta = \eta(\theta)$. Therefore, it is also common in ML to work with these alternative parametrizations directly, *e.g.*, having a neural network that outputs the mean. It is worth noting, however, that sometimes it is more convenient to work directly with η , and indeed we will use neural networks to parametrize the natural parameters in **Chapter 8** and part of **Chapter 9**.

7.3 Latent-variable models

In this part, we focus on the broad class of latent-variable models, which assume that each of the *i.i.d.* observed variables \mathbf{x}_n are generated as a stochastic function after observing: **i)** a local latent variable, \mathbf{z}_n ; and **ii)** a global latent variable, β . Namely, the joint distribution takes the form:

$$p_{\theta}(\mathbf{X}, \mathbf{Z}, \beta) = p(\beta) \prod_{n=1}^N p(\mathbf{z}_n) p_{\theta}(\mathbf{x}_n | \mathbf{z}_n, \beta), \quad (7.4)$$

where $\mathbf{Z} = \{\mathbf{z}_n\}_{n=1}^N$ is the random matrix composed of all the local latent variables. As the name indicates, latent variables are unobserved, and their prior distributions in **Equation 7.4** needs then to be fixed beforehand. Moreover, different choices for the latent variables and the observed likelihood define different latent-variable models.

Inference. To perform model inference, *i.e.*, to learn the parameters of the model that explain the data, in this part we focus on **black-box variational inference (BBVI)** [160]. In BBVI we solve two problems at once: besides learning the parameters that explain the likelihood of our data in **Equation 7.4**, we also learn an approximation to the **posterior distribution** of our latent variables, \mathbf{Z} and β . In other words, we have an additional model q_{ϕ} with parameters ϕ such that

$$q_{\phi}(\mathbf{Z}, \beta | \mathbf{X}) \approx p_{\theta}(\mathbf{Z}, \beta | \mathbf{X}) = \frac{p_{\theta}(\mathbf{X}, \mathbf{Z}, \beta)}{p_{\theta}(\mathbf{X})}, \quad (7.5)$$

where $p(\mathbf{X})$ is the marginal distribution of the joint in **Equation 7.4**. BBVI learns the parameters θ and ϕ by performing stochastic gradient ascent⁴ to maximize the **evidence lower bound (ELBO)**, *i.e.*,

$$\text{ELBO}(\theta, \phi) = \mathbb{E}_{q_{\phi}(\mathbf{Z}, \beta)}[\log p_{\theta}(\mathbf{X} | \mathbf{Z}, \beta)] - \text{KL}(q_{\phi}(\mathbf{Z}, \beta) \| p(\mathbf{Z}, \beta)), \quad (7.6)$$

where KL denotes the **Kullback-Leibler divergence (KL)**. The intuition here is that the first term of **Equation 7.6** serves as a ‘reconstruction loss’, whereas the second term works as a regularization term to keep the approximation to the posterior close to the prior distribution.

7.3.1 Mixture model	70
7.3.2 Matrix factorization	70
7.3.3 Variational autoencoders	71

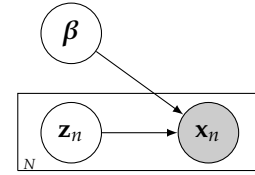


Figure 7.1: Graphical model of a general latent-variable model.

[160] Ranganath, Gerrish and Blei (2014), ‘Black Box Variational Inference.’

⁴ In practice, using any DL optimizer over stochastic gradients.

The ELBO is far from the only loss to use within the BBVI framework, and we will explore more complex loss functions in [Chapter 9](#). Now, we briefly review the models that we will use in [Chapters 8](#) and [9](#).

7.3.1 Mixture model

[54] Everitt (1996), ‘An introduction to finite mixture distributions.’

Remark 7.2 From a Bayesian standpoint, $p(\beta)$ is usually the conjugate prior of the likelihood. However, as we will select the likelihood of each x_{nd} based on the properties of the d -th modality, we keep the prior as a standard normal.

This is not a big setback, as we only use the priors to numerically compute the KL in [Equation 7.6](#).

First, we introduce mixture models [54], which build arbitrarily complex distributions by combining M simple distributions. In other words, given a datapoint \mathbf{x}_n , a mixture model assumes that it was generated from one of the M simple distributions, chosen at random.

A mixture model defines the global latent $R.V. \beta$ as the parameters of all the distributions, and given an element of the dataset \mathbf{x}_n , it chooses the specific cluster via the cluster-assignment local latent $R.V. \mathbf{z}_n$. Therefore, a mixture model defines the priors of β and \mathbf{z}_n as

$$p(\beta) := \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \text{and} \quad p(\pi_n) := \mathcal{U}(1, M), \quad (7.7)$$

and the likelihood function as

$$p_{\theta}(\mathbf{x}_n | \mathbf{z}_n, \beta) := p(\mathbf{x}_n; \mathbf{z}_n^{\top} \beta), \quad (7.8)$$

where \mathbf{z}_n is a one-hot vector encoding the selected cluster. For the posterior distributions, a mixture model assumes the following factorization

$$q_{\phi}(\mathbf{Z}, \beta | \mathbf{X}) = q_{\phi}(\beta | \mathbf{X}) \prod_{n=1}^N q_{\phi}(\mathbf{z}_n | \mathbf{x}_n), \quad (7.9)$$

where

$$q_{\phi}(\beta | \mathbf{X}) := \mathcal{N}(\mu, \Sigma), \quad \text{and} \quad q_{\phi}(\mathbf{z}_n | \mathbf{x}_n) := \text{Cat}(\pi_n), \quad (7.10)$$

and where $\phi := [\mu \ \Sigma \ \pi_1 \ \pi_2 \ \dots \ \pi_N]$ are the parameters we need to learn using the ELBO. Moreover, note that in this model $\theta := \emptyset$, i.e., we do not have likelihood parameters to learn.

Optimizing discrete variables. In order to learn the parameters π_n with automatic differentiation, we need to approximate the sampling from a categorical distribution, since it is a non-differentiable function. To this end, we replace the categorical distribution during training with a Gumbel-Softmax distribution [81] to learn the parameters π_n , and use a regular categorical distribution during evaluation.

[81] Jang, Gu and Poole (2017), ‘Categorical Reparameterization with Gumbel-Softmax.’

7.3.2 Matrix factorization

[172] Salakhutdinov and Mnih (2007), ‘Probabilistic Matrix Factorization.’

Following the previous section, maybe the best way of understanding the matrix factorization model [172] is as an extension of the clustering model, in which we replace the discrete cluster assignments for continuous variables. Specifically, we now define the priors of β and \mathbf{z}_n as

$$p(\beta) := \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \text{and} \quad p(\pi_n) := \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad (7.11)$$

and with the same form for the likelihood as before,

$$p_{\theta}(\mathbf{x}_n | \mathbf{z}_n, \beta) := p(\mathbf{x}_n; \mathbf{z}_n^{\top} \beta), \quad (7.12)$$

where, as mentioned before, \mathbf{z}_n is now a continuous vector of the same size as $\boldsymbol{\beta}$. We assume the same posterior factorization as in the mixture model, *i.e.*,

$$q_\phi(\mathbf{Z}, \boldsymbol{\beta} \mid \mathbf{X}) = q_\phi(\boldsymbol{\beta} \mid \mathbf{X}) \prod_{n=1}^N q_\phi(\mathbf{z}_n \mid \mathbf{x}_n), \quad (7.13)$$

but this time with

$$q_\phi(\boldsymbol{\beta} \mid \mathbf{X}) := \mathcal{N}(\mu_\beta, \Sigma), \quad \text{and} \quad q_\phi(\mathbf{z}_n \mid \mathbf{x}_n) := \mathcal{N}(\mu_n, \sigma), \quad (7.14)$$

where we note that the standard deviation σ is shared across samples. Again, we have that the likelihood has no parameters to optimize, $\boldsymbol{\theta} := \emptyset$, and therefore we only need to learn the parameters for the posterior approximation, $\boldsymbol{\phi} := [\mu_\beta \ \Sigma \ \sigma \ \mu_1 \ \mu_2 \ \dots \ \mu_N]$.

7.3.3 Variational autoencoders

Variational autoencoders (VAEs), popularized by Kingma and Welling [93] in 2014, are a well-established class of DL probabilistic models. In short, they are a class of probabilistic models that heavily rely on neural networks, and that are able to easily scale up to a million of samples.

Compared with the previous models, VAEs show two main differences. First, VAEs lack global latent variables, so they only model the local variables, \mathbf{z}_n . Second, unlike the previous models, VAEs have a more flexible *linking function* describing how the latent variables influence the likelihood function. Namely, it is usually assumed to have a standard Gaussian as prior, *i.e.*,

$$p(\mathbf{z}_n) := \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad (7.15)$$

and the likelihood is of the form

$$p_\theta(\mathbf{x}_n \mid \mathbf{z}_n) := p(\mathbf{x}_n; \boldsymbol{\eta}_\theta(\mathbf{z}_n)), \quad (7.16)$$

where $\boldsymbol{\eta}_\theta$ is a neural network with parameters $\boldsymbol{\theta}$ that takes \mathbf{z}_n as input, and outputs the parameters of the likelihood function. Once again, the approximation to the posterior is assumed to factorize per sample, *i.e.*,


$$q_\phi(\mathbf{Z} \mid \mathbf{X}) := \prod_{n=1}^N q_\phi(\mathbf{z}_n \mid \mathbf{x}_n), \quad (7.17)$$

where, similar to the likelihood function, we assume the posterior for each local variable to be a Gaussian distribution parametrized by a neural network, *i.e.*,

$$q_\phi(\mathbf{z}_n \mid \mathbf{x}_n) := \mathcal{N}(\mu_\phi(\mathbf{x}_n), \Sigma_\phi(\mathbf{x}_n)), \quad (7.18)$$

where $\eta_\phi = [\mu_\phi \ \Sigma_\phi]$ is a neural network with parameters $\boldsymbol{\phi}$ that takes \mathbf{x}_n as input, and outputs the mean and covariance of the Gaussian posterior. In practice, the specific way that the posterior is parametrized can slightly change, *e.g.*, by keeping the covariance matrix fixed, shared across samples, or by modelling only the diagonal.

In the following, we present different extensions of the vanilla VAE architecture introduced above, that will become relevant in **Chapter 9**.

[93] Kingma and Welling (2014), ‘Auto-Encoding Variational Bayes.’ 

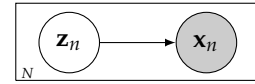


Figure 7.2: Graphical model of a VAE.

Remark 7.3 VAEs share many similarities with classical autoencoders [65] and, hence, it is common in the literature to refer to q_ϕ and p_θ as encoder and decoder, respectively.

[143] Nazabal, Olmos, Ghahramani and Valera (2020), ‘Handling incomplete heterogeneous data using VAEs.’

5: Remember, in the heterogeneous case we assume each \mathbf{x}_d to be unimodal.

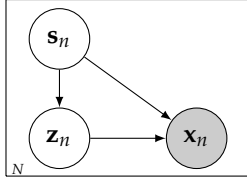


Figure 7.3: Graphical model of a HI-VAE.

Heterogeneous-Incomplete VAE

The **heterogeneous-incomplete VAE (HI-VAE)** [143] is a model specialized on handling heterogeneous data.⁵ While HI-VAEs differ from a standard VAE in several technical aspects, *e.g.*, including a data normalization layer for each type of likelihood, here we present the main differences in terms of the probabilistic model it defines. Specifically, the main change in HI-VAE is the introduction of an additional local latent variable, \mathbf{s}_n , that renders a hierarchical structure in the probabilistic model. Similar to the mixture model introduced in Subsection 7.3.1, the pair $(\mathbf{s}_n, \mathbf{z}_n)$ forms a uniform Gaussian mixture model prior, *i.e.*,

$$p(\mathbf{s}_n) := \mathcal{U}(1, M), \quad \text{and} \quad p(\mathbf{z}_n | \mathbf{s}_n) := \mathcal{N}(\mathbf{s}_n^\top \mu_{\mathbf{z}}, \mathbf{I}), \quad (7.19)$$

where \mathbf{s}_n is a one-hot encoding of the same size as the vector of mixture means, $\mu_{\mathbf{z}}$. Now, the complete joint distribution is of the form

$$p_{\theta}(\mathbf{X}, \mathbf{Z}, \mathbf{S}) = \prod_{n=1}^N p_{\theta}(\mathbf{x}_n | \mathbf{z}_n, \mathbf{s}_n) p(\mathbf{z}_n | \mathbf{s}_n) p(\mathbf{s}_n), \quad (7.20)$$

where $\mathbf{S} = \{\mathbf{s}_n\}_{n=1}^N$, and where the likelihood is parametrized by a neural network with parameters θ , *i.e.*,

$$p_{\theta}(\mathbf{x}_n | \mathbf{z}_n, \mathbf{s}_n) := p_{\theta}(\mathbf{x}_n; \eta_{\theta}(\mathbf{z}_n, \mathbf{s}_n)). \quad (7.21)$$

For the posterior approximation, HI-VAEs factorize it akin to the prior,

$$q_{\phi}(\mathbf{Z}, \mathbf{S} | \mathbf{X}) = \prod_{n=1}^N q_{\phi}(\mathbf{z}_n | \mathbf{s}_n, \mathbf{x}_n) q_{\phi}(\mathbf{s}_n | \mathbf{x}_n), \quad (7.22)$$

where the posteriors for \mathbf{z}_n and \mathbf{s}_n are, respectively, Gaussian and categorical distributions whose parameters are given by neural networks,

$$q_{\phi}(\mathbf{z}_n | \mathbf{s}_n, \mathbf{x}_n) := \mathcal{N}(\mu_{\phi}(\mathbf{s}_n, \mathbf{x}_n), \Sigma_{\phi}(\mathbf{s}_n, \mathbf{x}_n)), \quad \text{and} \quad (7.23)$$

$$q_{\phi}(\mathbf{s}_n | \mathbf{x}_n) := \text{Cat}(\pi_{\phi}(\mathbf{x}_n)), \quad (7.24)$$

[81] Jang, Gu and Poole (2017), ‘Categorical Reparameterization with Gumbel-Softmax.’

where we use the Gumbel-Softmax distribution [81] during training.

Mixture-based multimodal VAEs

6: Where the variables \mathbf{x}_d are, in general, multidimensional.

7: For example, sample the caption for a given image, or *vice versa*.

In the context of multimodal data,⁶ one recurrent application researchers seek for is *conditional generation*, *i.e.*, the ability of sampling a modality having observed a different one, while representing the same underlying concept.⁷ However, if we have a single encoder that takes all modalities as input, handling a missing modality is not straight-forward.

Mixture-based multimodal VAEs solve this problem by fully factorizing the network architecture: instead of using one VAE, these type of models create D VAEs—one per modality, as we described at the beginning of Subsection 7.3.3—and connect them through the local latent variables by considering the posterior approximation a mixture model of the form:

$$q_{\phi}(\mathbf{z}_n | \mathbf{x}_n) = \frac{1}{M} \sum_{A \in \mathcal{A}} q_A(\mathbf{z}_n | \mathbf{x}_{nA}), \quad (7.25)$$

where $\mathcal{A} \subset \mathcal{P}(D)$ is a subset of all the possible subsets of modalities, and $q_{\mathcal{A}}(\mathbf{Z} \mid \mathbf{X}_{\mathcal{A}})$ is known as an *expert*, and it is defined as the product of all the posteriors of the modalities in $\mathcal{A} \subset \{1, 2, \dots, D\}$, i.e.,

$$q_{\mathcal{A}}(\mathbf{z}_n \mid \mathbf{x}_{n\mathcal{A}}) \propto \prod_{d \in \mathcal{A}} q_{\phi_d}(\mathbf{z}_n \mid \mathbf{x}_{nd}). \quad (7.26)$$

Recently, a number of works have proposed multimodal VAEs following this structure, and we can specify them by choosing a specific value for the ‘mixture of experts’ set \mathcal{A} . For example, we employ the following multimodal VAEs in [Chapter 9](#):

$$\text{MVAE [213]: } \mathcal{A} = \{\{1, 2, \dots, D\}\},$$

$$\text{MMVAE [185]: } \mathcal{A} = \{\{1\}, \{2\}, \dots, \{D\}\},$$

$$\text{MoPoE [199]: } \mathcal{A} = \mathcal{P}(D),$$

which consider, respectively, a single joint expert, all unimodal experts, and every possible expert we can define by combining modalities.

Remark 7.4 The notation \propto means ‘proportional to’ and, in general, it is not guaranteed that $q_{\mathcal{A}}$ exists except, e.g., in a product of Gaussian distributions.

7.4 Modality collapse and MTL

In this section, we introduce the concept of modality collapse, and draw the initial connections between probabilistic modelling in DL and MTL.

Inductive biases. Similar to MTL, PGMs change their objective through an inductive bias. However, in this case, the probabilistic setting makes it clear: as we discussed in [Chapter 2](#), PGMs exploit the synergetic information across modalities by optimizing the joint density, rather than the D marginals individually. In this way, they can also model the dependencies between modalities.

Modalities as tasks. For the purposes of this discussion, let us start with a concrete example to get a feeling on the commonalities between modalities and tasks. If we recall from [Part I](#), we had a set of K tasks who were commonly optimized by adding up their losses, i.e., $\sum_k L_k$. Now, instead, we have a set of D modalities, and we learn to model the joint likelihood in [Equation 7.2](#) by optimizing the ELBO in [Equation 7.6](#) and, putting both equations together,

$$\begin{aligned} \text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}) &= \sum_{d=1}^D \mathbb{E}_{q_{\phi}(\mathbf{Z}, \boldsymbol{\beta})} [\log p_d(\mathbf{X}_d; \boldsymbol{\eta}_d(\mathbf{Z}, \boldsymbol{\beta}; \boldsymbol{\theta}))] \\ &\quad - \text{KL}(q_{\phi}(\mathbf{Z}, \boldsymbol{\beta}) \parallel p(\mathbf{Z}, \boldsymbol{\beta})). \end{aligned} \quad (7.27)$$

We can now see in the first term a ‘sum of losses’ with shared parameters $\boldsymbol{\theta}$ similar to that from MTL. From [Equation 7.27](#), the initial connection between probabilistic modelling and MTL should be clear. Therefore, it is sensible to expect more similarities in terms of similar issues, concepts, and solutions, between both frameworks and, indeed, we will see in the following chapters that to be the case.

8: E.g., classification *v.s.* regression.

Comparability. In Chapter 6, we introduced the concept of task incomparability within the context of MTL. In short, we *a priori* cannot directly compare quantities coming from different sources, as they have disparate ranges and semantics, especially if the set of tasks is heterogeneous.⁸ This very same comparability issue exists in probabilistic modelling, where we cannot *a priori* compare the (log-)likelihoods of two different modalities. Far from obscure, this is a well-known concept, and it is particularly clear if one modality is continuous and the other discrete; in that case, we would be comparing two completely different objects, respectively, a probability density and a probability mass function.

Impartiality. In Section 4.5 from Part I, we discussed that, without any knowledge on the preferences of the **decision maker (DM)**, one sensible approach in **multi-objective optimization (MOO)** was to consider a no-preference method which, in layman terms, means that we remain impartial towards the importance of the individual objectives. Moreover, we also introduced task impartiality, an analogue objective in MTL which refers to the desire of learning all tasks together, without overlooking any of them. Therefore, it should be of no surprise to find a similar concept in probabilistic modelling.

9: I.e., sampling \mathbf{x} by first sampling \mathbf{z} and β from the prior distribution.

If we consider the analogies drawn above, the natural approach would be to think of ‘modality impartiality’, in which we want to remain impartial toward which modality we learn. While intuitively true, it is an overly vague definition, as PGMs are usually built with a purpose or application in mind. For example, in the basic use-case in which we want to perform joint generation,⁹ it is vital that we achieve a good likelihood fit for all modalities during training. If, instead, we want to perform more complex tasks such as imputing missing modalities given the observed ones, our impartiality goals would not only concern being impartial towards the likelihood terms, but also towards their posteriors and the modalities we use for conditioning. We will expand this idea later in Chapter 9.

We encompass all these cases under the common adjective *impartial*. Therefore, we want in general to achieve modality impartiality and, to this end, the PGMs should go through an impartial learning process via impartial optimization. Whether our goal is to achieve likelihood impartiality (Chapters 8 and 9), or more involved variants of these concepts such as expert impartiality (Chapter 9), should be explicitly stated and will be clear throughout the thesis.

Modality collapse. The paragraphs above should already make us raise an eyebrow: ‘If, as in MTL, we have a sum of incomparable quantities, should we not observe some form of negative transfer?’ We indeed should and, only recently, a number of works have explicitly talked about the difficulties of training PGMs in this setting [120, 143, 185]. For example, quoting Nazabal et al. [143]:

“Preventing a few dimensions of the data dominating the training is a crucial aspect to effectively deploy deep generative models suitable for heterogeneous data.”

In this thesis, we refer to this limitation of PGMs as *modality collapse* where, similar to negative transfer, the model focuses on a subset of

the modalities, overlooking the rest of them (*e.g.*, by fitting the image while neglecting the caption). To better exemplify the effects of modality collapse, we show in Figure 7.4 a real example on a VAE extracted from the experiments in Chapter 9.

Gradient conflict and probabilistic constraints. Equation 7.27 hints us that gradient conflict will appear in the first term of the loss, and this is indeed the case. However, it should also be apparent that there exist additional challenges and constraints that should be taken into account. First, Equation 7.27 has an additional regularization term which does not directly involve any modality. Moreover, it is a common practice to train PGMs with more sophisticated losses than the ELBO which, *e.g.*, involve the use of non-linear scalarization functions. Second, the probabilistic setting imposes additional constraints. For example, the joint likelihood of \mathbf{x} should always integrate to one and, thus, naively multiplying each log-likelihood in Equation 7.27, *e.g.*, employing a gradient-conflict approach like GradNorm [26], would break this constraint.

We deepen into these connections to MTL and how to overcome the aforementioned challenges in the following chapters. In Chapter 8, we attempt to achieve likelihood impartiality by pre-processing the data, and thus preserving probability constraints. In Chapter 9, instead, we identify the sub-computational graphs of the training process where gradient conflict is located, and leverage existing MTL solutions to palliate the effects of gradient conflict.

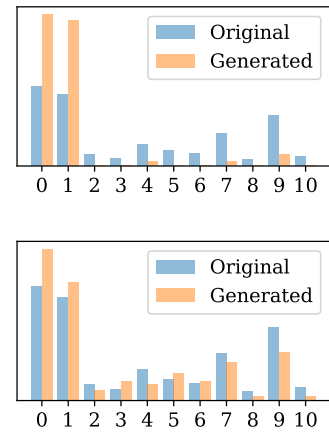


Figure 7.4: Histogram of a categorical $R.V.$ showing the effects of modality collapse on a tabular dataset (top), and the same histogram of a model without modality collapse (bottom).

On Modality Collapse and Data Preprocessing

8.

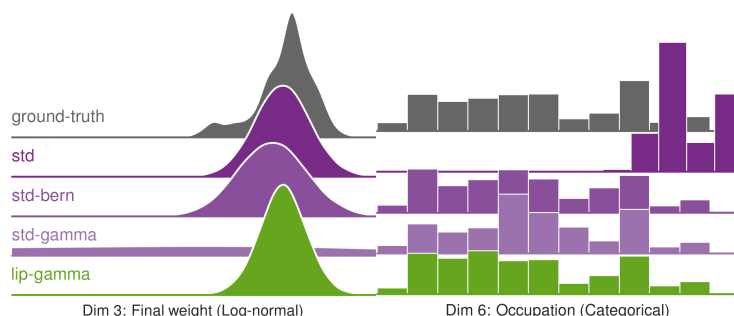
Des einen Einsamkeit ist
die Flucht des Kranken;
des anderen Einsamkeit
die Flucht vor den Kranken.

Friedrich Wilhelm Nietzsche

In the previous chapter, we discussed how approaches like **black-box variational inference (BBVI)** have enabled **probabilistic generative models (PGMs)** to scale up, and are widely-used to learn the joint likelihood of our data, being **variational autoencoders (VAEs)** a remarkable example. However, we also argued that *modality collapse* is a common problem, where the model fits only a subset of the observed variables, leading to poor results in real-world applications. In this chapter, we investigate how data-preprocessing affects modality collapse.

To motivate the chapter, let us go through an illustrative example in **Figure 8.1**, where we want to model the `Adult` dataset [48] using a simple VAE. First, we follow the textbook and standardize the continuous variables as a preprocessing step. The final model (`std`) does a reasonable job fitting the variable `Final weight`, but a poor one with the discrete variable `Occupation`. We relax the constraint across the categorical vectors, and model each as Bernoulli distributions (`std-bern`),¹ which works much better. As discrete data cannot be standardized, we relax them even further, and treat them as continuous variables (`std-gamma`) that can be standardized.² Unexpectedly, when we reconstruct data, the log-normal variable is really poorly modelled. If we were to use Lipschitz standardization instead (`lip-gamma`), introduced in **Section 8.4**, then we would be able to preprocess all variables and capture them significantly better.

In this chapter, we first show that data standardization often helps impartial learning with common continuous distributions, such as the Gaussian distribution (**Section 8.3**). Unfortunately, often is not always. Hence, we propose Lipschitz standardization to scale the data such that the Lipschitz smoothness of the log-likelihood is comparable across variables (**Section 8.4**). As illustrated in **Figure 8.1**, Lipschitz standardization facilitates a more accurate fitting across all variables. Finally, our results in **Section 8.5** show the effectiveness of the proposed method as a step forward towards impartial learning in PGMs.



8.1 Problem Statement	78
8.2 Impartial learning	79
8.3 Scaling & smoothness . . .	79
8.4 Lipschitz standardization .	82
8.5 Empirical evaluation	84
8.6 Concluding remarks	86



github.com/adrianjav/lipstd

This chapter is based on the content of:
[1]: Javaloy and Valera (2020), ‘Lipschitz standardization for multivariate learning.’

1: Introduced in **Subsection 8.4.1** as the Bernoulli trick.

2: Again, introduced in **Subsection 8.4.1** as the Gamma trick.

Figure 8.1: Marginals of continuous (left) and discrete (right) variables from the `Adult` dataset reconstructed from trained VAEs with different preprocessing. All plots are drawn in the original data space. See the accompanying text for a description of the methods.

8.1 Problem Statement

8.1.1 Data preprocessing . . . 78

In this chapter, we assume the same setup as in [Chapter 7](#). That is, we assume that there is a multimodal D -dimensional **random variable (R. V.)** $\mathbf{x} := [x_1 \ x_2 \ \dots \ x_D] \sim P_{\mathbf{x}}$ whose distribution we want to model using a PGM, as well as an input dataset $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N \stackrel{i.i.d.}{\sim} P_{\mathbf{x}}$ of N *i.i.d.* samples. Moreover, we take a heterogeneous modelling perspective, in which we make no assumptions on the distribution of each unidimensional modality, x_d , and thus we factorize the likelihood model, *i.e.*,

$$p(\mathbf{x}; \boldsymbol{\theta}) = \prod_{d=1}^D p_d(x_d; \boldsymbol{\eta}_d(\boldsymbol{\theta})), \quad (8.1)$$

3: See [Section 7.2](#) in [Chapter 7](#).

where p_d is a member of the exponential family,³ and $\boldsymbol{\eta}_d$ denotes their natural parameters as a function of $\boldsymbol{\theta}$. Our goal then is to learn values for the parameters $\boldsymbol{\theta}$ that properly fit each modality likelihood.

4: However, we are positive that the results should translate to others frameworks such as **maximum likelihood estimation (MLE)**.

We consider **latent variable models (LVMs)** as our PGMs, and BBVI [[160](#)] as our training framework.⁴ In short, LVMs assume that each observed variable is generated after a local latent variable, \mathbf{z}_n , and a global one, $\boldsymbol{\beta}$, whose joint distribution is of the form

$$p_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{Z}, \boldsymbol{\beta}) = p(\boldsymbol{\beta}) \prod_{n=1}^N p(\mathbf{z}_n) p_{\boldsymbol{\theta}}(\mathbf{x}_n | \mathbf{z}_n, \boldsymbol{\beta}). \quad (8.2)$$

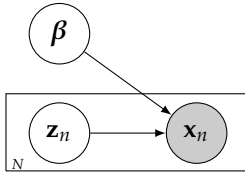


Figure 8.2: Graphical model of a general latent-variable model.

BBVI is then used to jointly learn the likelihood $p_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{Z}, \boldsymbol{\beta})$ and the posterior distribution over the latent variables $q_{\boldsymbol{\phi}}(\mathbf{Z}, \boldsymbol{\beta} | \mathbf{X})$, where $\boldsymbol{\phi}$ are the set of parameters for the posterior. Namely, BBVI finds the values for $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ by maximizing the **evidence lower bound (ELBO)**,

$$\text{ELBO}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{Z}, \boldsymbol{\beta})} [\log p_{\boldsymbol{\theta}}(\mathbf{X} | \mathbf{Z}, \boldsymbol{\beta})] - \text{KL}(q_{\boldsymbol{\phi}}(\mathbf{Z}, \boldsymbol{\beta}) \parallel p(\mathbf{Z}, \boldsymbol{\beta})), \quad (8.3)$$

using stochastic gradient optimization over the parameters.

As discussed in [Section 7.4](#), each modality contributes additively to the gradient computation in [Equation 8.3](#), whereas the data do not directly affect the second term. Therefore, LVMs can suffer from gradient conflict, and thus from modality collapse. Similarly, to study the effect of data preprocessing, we only need to study its effect on the first term of [Equation 8.3](#), *i.e.*, on the (scaled) log-likelihoods.

8.1.1 Data preprocessing

In this chapter, we focus on data-scaling preprocessing algorithms of the form $\tilde{x} = \omega x$ since they: **i)** preserve important properties of the data distribution, such as tails; and **ii)** are broadly used in practice [[73](#)]. Besides, we assume that no method shifts data,⁵ as it can violate data constraints (*e.g.*, non-negativity). Our main focus is on three broadly-used data scaling methods:

- **Standardization:** $\tilde{x} = x / \text{std}_d$ (**std**)
- **Normalization:** $\tilde{x} = x / \max_d$ (**max**)
- **Interquartile range:** $\tilde{x} = x / \text{iqr}_d$ (**iqr**)

[[73](#)] Han, Pei and Kamber (2011), ‘Data mining: concepts and techniques.’

5: However, as a first step we centre any (log-)normal R. V. in the experiments.

where we divide, in order, by the empirical standard deviation, absolute maximum, and interquartile range of the d -th modality.

Notation. To avoid clutter, in this chapter we use the shorthand $\ell_d(\theta)$ for the log-likelihood of the d -th modality, $\ell_d(\theta) := \log p_d(x_d; \eta_d(\theta))$. Similarly, we use a tilde as a shorthand to refer to the scaled counterpart of, *e.g.*, a variable (\tilde{x}), a parameter ($\tilde{\eta}$), or a log-likelihood ($\tilde{\ell}$).

8.2 Multivariate impartial learning

As discussed above, our goal is to prevent modality collapse in LVMs by properly scaling the data as a preprocessing step, *i.e.*, we want to incentivize an impartial learning process. We could express this objective by saying that we want, at each step t , that the normalized log-likelihood improvement is the same across modalities, *i.e.*,

$$\frac{\ell_d(\theta^{t+1}) - \ell_d(\theta^t)}{\ell_d(\theta^0)} = C^t \quad \text{for every } d \in \{1, 2, \dots, D\}, \quad (8.4)$$

where θ^0 denotes the parameters at initialization, and C^t the shared constant improvement at step t . At first, however, Equation 8.4 seems like a hard objective to work with. By looking instead at the learning process through the gradients, we can make our goal more amenable.

Lipschitz-smoothness. To this end, let us first introduce the class of L -smooth functions, which are a class of functions extensively studied in machine learning (ML) optimization as they enjoy convergence guarantees in gradient descent [144]. Specifically, a function $\ell(\theta)$ is called L -smooth on Q w.r.t. θ if it is twice-differentiable and, for any $a, b \in Q$, it holds that

$$\|\nabla_{\theta} \ell(a) - \nabla_{\theta} \ell(b)\| \leq L \|a - b\|. \quad (8.5)$$


Then, instead of looking at the log-likelihood differences as in Equation 8.4, we can focus on the gradient differences, *i.e.*,

$$\frac{\|\nabla_{\theta} \ell_d(\theta_d^{t+1}) - \nabla_{\theta} \ell_d(\theta_d^t)\|}{\|\nabla_{\theta} \ell_d(\theta_d^0)\|} = C^t \quad \text{for every } d \in \{1, 2, \dots, D\}, \quad (8.6)$$

and, if we were to force equal (tightest) L -smoothness across modalities, then we would have that

$$\|\nabla_{\theta} \ell_d(\theta^{t+1}) - \nabla_{\theta} \ell_d(\theta^t)\| \leq \underbrace{L \|\nabla_{\theta} \ell_d(\theta_d^0)\|}_{C^t} \|\theta^{t+1} - \theta^t\|. \quad (8.7)$$

In practice, we presume a good initialization for θ such that the initial gradient magnitudes are comparable across modalities, and thus we can blend the term $\|\nabla_{\theta} \ell_d(\theta_d^0)\|$ within C^t .⁶

[144] Nesterov (2004), ‘Introductory Lectures on Convex Optimization - A Basic Course.’ 

6: Alternatively, we could always make C^t depend on the initial value of each modality, *i.e.*, $C_d^t := C^t \cdot \|\nabla_{\theta} \ell_d \theta_d^0\|$.

8.3 Data scaling and smoothness

In this section, we first analytically study the impact that data scaling has

8.3.1 Exponential family	80
8.3.2 Smoothness computation	81
8.3.3 Standardization	82

Table 8.1: Multiplicative and additive factors for some common continuous distributions, see [Proposition 8.1](#). When f_i or g_i are omitted, they are the constant functions 1 and 0, respectively. Together with the distributions, we print their traditional parametrization to help the reader.

Distribution	[Params]	$T_1(x)$	$T_2(x)$
(Log-)Normal	$[\mu \ \sigma]$	$f_1 = \omega$	$f_2 = \omega^2$
Gamma	$[\alpha \ \beta]$	$g_1 = \log \omega$	$f_2 = \omega$
Inverse Gaussian	$[\mu \ \lambda]$	$f_1 = \omega$	$f_2 = 1/\omega$
Inverse Gamma	$[\alpha \ \beta]$	$g_1 = \log \omega$	$f_2 = 1/\omega$
Exponential	$[\lambda]$	$f_1 = \omega$	
Rayleigh	$[\sigma]$	$f_1 = \omega^2$	

on the log-likelihood L -smoothness. Then, we compute the effect of data standardization in specific, and find that, while it often helps to even the L -smoothness across modalities, in some cases it can also be harmful.

Similar to the way Santurkar et al. showed that batch normalization [80] smooths out the optimization landscape [177], we show that data standardization often smooths out the log-likelihood optimization landscape, and it does so similarly across a wide range of log-likelihood choices.

[177] Santurkar, Tsipras, Ilyas and Madry (2018), ‘How Does Batch Normalization Help Optimization?’ [🔗](#)

Remark 8.1 This is a similar argument to the one of homogenizing gradients *w.r.t.* θ or z that we discussed in [Part I](#). However, as we do preprocessing, this is the only option.

Given that we work on preprocessing data and thus have no access to the parameters, in the following we focus on the data-dependent part of the gradient computation, *i.e.*, on $\nabla_{\eta} \ell(\eta)$. Assuming that the model-dependent part $\nabla_{\theta} \eta$ is similar across modalities, we can see equal L -smoothness *w.r.t.* them implies equal L -smoothness *w.r.t.* the full gradient, since $\nabla_{\theta} \ell(\eta(\theta)) = \nabla_{\eta} \ell(\eta) \cdot \nabla_{\theta} \eta$.

8.3.1 Scaling the exponential family

We consider each observed modality x_d to follow some distribution of the exponential family,⁷ *i.e.*,

$$p(x; \eta) = h(x) \exp [\eta^T T(x) - A(\eta)], \quad (8.8)$$

where η and $T(x)$ are vectors of the same size I . Most importantly, working with the exponential family let us analytically relate the gradients of the scaled and original log-likelihoods. Namely:

Proposition 8.1 Let $p(x; \eta)$ be a member of the exponential family with $x \in \mathbb{R}$ and $\eta \in \mathbb{R}^I$, and let $\omega > 0$ be the weight to scale the data, *i.e.*, $\tilde{x} := \omega x$. Then, if every sufficient statistic can be factorized as $T_i(\tilde{x}) = f_i(\omega)T_i(x) + g_i(\omega)$, it holds that:

$$\partial_{\tilde{\eta}_i}^j \tilde{\ell}(\tilde{\eta}) = f_i(\omega)^j \partial_{\eta_i}^j \ell(\eta), \quad (8.9)$$

where $\partial_{\eta_i}^j$ and $\partial_{\tilde{\eta}_i}^j$ denote the j -th partial derivatives with respect to η_i and $\tilde{\eta}_i := \eta_i / f_i(\omega)$, respectively.

Although the requirements of [Proposition 8.1](#) look restrictive at first, many common distributions fulfil them, as we report in [Table 8.1](#). Note that, in the case of the log-normal distribution, we can apply the proposition using the scaling function $x \mapsto x^\omega$, instead of $x \mapsto \omega x$. A complete version of the proposition and its proof can be found in [Appendix C.2](#).

7: See [Section 7.2](#) for an introduction.

8.3.2 Computing the local smoothness

As discussed at the beginning of the chapter, during preprocessing we assume no knowledge on the model parameters, and therefore the weights $\omega := [\omega_1 \ \omega_2 \ \dots \ \omega_D]$ cannot depend on them. Therefore, in this chapter we consider that our transformations are performed around the MLE estimator of the marginal natural parameters, which we denote by $\hat{\eta}$. As an example, if we had a Gaussian R.V. with empirical mean $\hat{\mu}$ and standard deviation $\hat{\sigma}$, we would consider $\hat{\eta}_1 = \hat{\mu} / \hat{\sigma}^2$ and $\hat{\eta}_2 = -1/2 \hat{\sigma}^2$.

We justify this unorthodox take on data preprocessing methods by noting that standardization (**std**) scales the data such that it has unit variance *evaluated on the empirical parameters*. That is, if we let $\text{std}_{\eta}(\eta) = \tilde{\eta}$ be the function scaling the natural parameters with the standardization weights, then $\mathbb{V}_p(\tilde{x}; \text{std}_{\eta}(\hat{\eta}))[\tilde{x}] = 1$, but this does not hold as we evaluate std_{η} far from $\hat{\eta}$. To understand how data scaling affects the log-likelihood L -smoothness, we first need to introduce a similar local concept.

Remark 8.2 We use $\mathbb{V}_p[f(x)]$ to denote the variance of $f(x)$ when $x \sim p$, i.e., $\mathbb{E}_p[(x - \mathbb{E}_p[x])^2]$.

Infinitesimal L -smoothness. We introduce a suitable definition of L -smoothness for an infinitesimally-small region centred around a point. Namely, we say that ℓ is (infinitesimally) L -smooth around η with respect to η_i if, for any $\epsilon > 0$, we can find a neighbourhood of η such that ℓ is locally L -smooth with respect to η_i , up to an error of ϵ . In other words, if we can find a neighbourhood Q of η such that, for any $a, b \in Q$,

$$\|\nabla_{\eta} \ell(a) - \nabla_{\eta} \ell(b)\| \leq L \|a - b\| + \epsilon. \quad (8.10)$$

Note that this is a relaxation of the (local) L -smoothness in **Equation 8.5**. Moreover, we have that ℓ is always $\|\nabla_{\eta} \partial_{\eta_i} \ell(\eta)\|$ -smooth around η w.r.t. η_i . Indeed, if we consider the first-order Taylor expansion of ℓ around η (i.e., if we *linearize* ℓ at η), we have that

$$\partial_{\eta_i} \ell(\eta + h) - \partial_{\eta_i} \ell(\eta) = \nabla_{\eta} \partial_{\eta_i} \ell(\eta) \cdot h + o(\|h\|), \quad (8.11)$$

where $h \in \mathbb{R}^I$, and $\|h\| = \epsilon$ for an arbitrarily small $\epsilon > 0$. Then, by taking norms and using the Cauchy-Schwarz inequality, we obtain that

$$\|\partial_{\eta_i} \ell(\eta + h) - \partial_{\eta_i} \ell(\eta)\| \leq \|\nabla_{\eta} \partial_{\eta_i} \ell(\eta)\| \cdot \|h\| + o(\epsilon), \quad (8.12)$$

that is, we have that ℓ is $\|\nabla_{\eta} \partial_{\eta_i} \ell(\eta)\|$ -smooth around η w.r.t. η_i .

Using this result and **Proposition 8.1**, we can then write the L -smoothness of $\tilde{\ell}$ around its scaled empirical parameters, $\hat{\eta}$,⁸ in terms of $\nabla_{\eta} \partial_{\eta_i} \ell(\hat{\eta})$:

$$\begin{aligned} \tilde{L}_i(\omega) &= \|\nabla_{\tilde{\eta}} \partial_{\tilde{\eta}_i} \tilde{\ell}(\hat{\eta})\| = |f_i(\omega)| \|\nabla_{\tilde{\eta}} \partial_{\tilde{\eta}_i} \ell(\hat{\eta})\| \\ &= |f_i(\omega)| \|f(\omega) \odot \nabla_{\eta} \partial_{\eta_i} \ell(\hat{\eta})\|, \end{aligned} \quad (8.13)$$

where $f(\omega) := [f_1(\omega) \ f_2(\omega) \ \dots \ f_I(\omega)]$, and \odot is the element-wise (or Hadamard) multiplication. It is important to remark that, if ℓ is L_i -smooth for each η_i , then it is $\sum_i L_i$ -smooth with respect to η . Similarly, if a function ℓ_1 is L_1 -smooth w.r.t. η , and another function ℓ_2 is L_2 -smooth w.r.t. η , then their sum $\ell_1 + \ell_2$ is $(L_1 + L_2)$ -smooth w.r.t. η .⁹ These properties are proved in **Appendix C.1**.

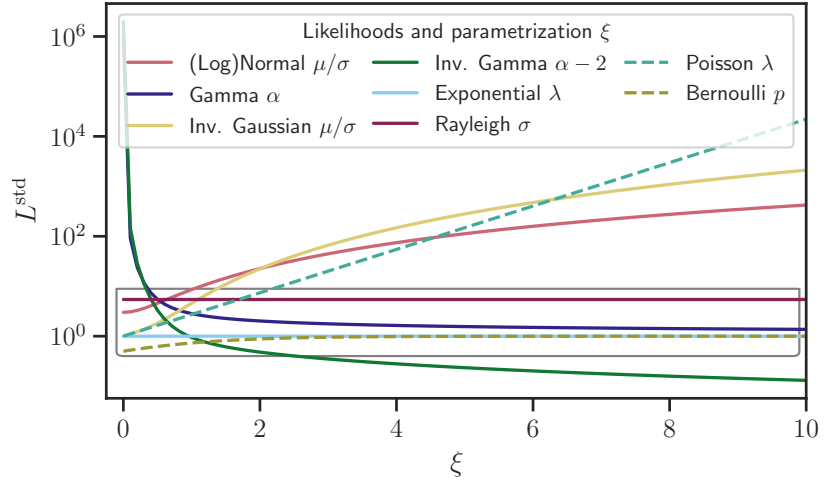
Remark 8.3 Indeed, any L -smooth function on Q is (infinitesimally) L -smooth around any point in Q .

8: We abuse notation here and call $\hat{\eta}$ the scaled empirical parameters.

Remark 8.4 For the exponential family, the sum $\sum_i L_i = \sum_i \|\nabla_{\eta} \partial_{\eta_i} \ell(\eta)\|$ corresponds to the $L_{2,1}$ -norm of the covariance of sufficient statistics, i.e., $\|\text{Cov}(T(x), T(x))\|_{2,1}$.

9: Note that, in that case, there could still exist a tighter bound.

Figure 8.3: Log-scale plot of the L -smoothness for different continuous log-likelihoods around their true original parameters, after standardizing. Dashed lines correspond to discrete distributions. To unify the distributions, we plot each one as a function ξ of their traditional parameters.



8.3.3 Smoothness and standardization

For the particular case of data standardization (**std**), we consider the distributions shown in [Table 8.1](#), and compute their L -smoothness around $\hat{\eta}$ after standardizing the data.

[Figure 8.3](#) shows their L -smoothness as a function of their traditional parameters.¹⁰ The plot sheds some light on the reason why standardizing works well in many cases, as it makes the L -smoothness around $\hat{\eta}$ comparable across modalities for many common log-likelihoods. For example: **i)** the exponential and Rayleigh distributions have constant values; **ii)** a centred (log-)normal distribution ($\mu = 0$) is 3-smooth; and **iii)** the Gamma distribution is (approximately) 2-smooth as long as its shape parameter α (which is scale-invariant) is larger than one.

However, [Figure 8.3](#) shows other cases where the Lipschitz constants are not comparable. First, we see that some discrete distributions (*e.g.*, Poisson) can have arbitrarily large L -constants. Besides, standardization does not work well for all continuous variables. For example, suppose that we have two variables following a standard normal and an inverse Gamma with $\hat{\alpha} = 10$, respectively. After standardizing, the normal R. V. is one order of magnitude smoother.¹¹ However, it is possible to show using [Equation 8.13](#) that, by scaling the inverse Gamma variable by $\omega_2 = 2$ instead of $\omega_2 = 1/\text{std}$, both log-likelihoods would have been 3-smooth. This is the key motivation for the proposed Lipschitz standardization.

8.4 Lipschitz standardization

8.4.1 Discrete data	83
8.4.2 In practice	84

Leveraging the insights from the previous section, we propose *Lipschitz standardization*, a novel data-scaling algorithm that homogenizes the L -smoothness of each modality around their empirical parameters, thus encouraging impartial learning. Intuitively, our algorithm puts the data into a region of the parameter space where each dimension has similar convergence rate guarantees.

Lipschitz standardization finds, for the d -th modality, the optimal weight

10: The full derivations are presented in [Appendix C.4](#).


11: Concretely, the normal is 3-smooth and the inverse Gamma 0.16-smooth.

ω_d^* that makes the L -smoothness match a target smoothness L^* , *i.e.*,

$$\omega_d^* := \arg \min_{\omega_d} \left(\sum_{i=1}^{I_d} \tilde{L}_{di}(\omega_d) - L^* \right)^2, \quad (8.14)$$

where $\tilde{L}_{di}(\omega_d)$ are the scaled Lipschitz constants in Equation 8.13. We set $L^* = 1/(D\alpha)$, where α is the initial learning rate set by the practitioner.

We motivate this choice following classical results on convex optimization [144] which states that, given an L -smooth function $\ell(\theta)$, the optimal step-size for gradient descent is $\alpha^* = 1/L$. As we fit multiple functions, each one being L_d -smooth—each with different optimal learning rates, we set the target smoothness such that the joint log-likelihood is both balanced across modalities, and close to the one optimal for a given learning rate,¹² *i.e.*, $L^* = 1/(D\alpha)$ and thus $\tilde{L} = \sum_d \tilde{L}_d \approx \sum_d 1/(D\alpha) = 1/\alpha$.

[144] Nesterov (2004), ‘Introductory Lectures on Convex Optimization - A Basic Course.’ 

12: Set by the practitioner.

8.4.1 Discrete data

Up to this point, our algorithm only suits continuous data, as we cannot scale discrete variables. However, real-world data often present mixed continuous and discrete data types. Here, we extend Lipschitz standardization to Bernoulli, Poisson, and categorical distributions.

Gamma trick. Our approach can be summarized in four steps: **i)** dequantize the discrete data x with additive noise, *i.e.*, $\tilde{x} = x + (1 + \epsilon)$; **ii)** model \tilde{x} as a Gamma R. V., and scale it with Lipschitz standardization to ease impartial learning, $\tilde{x} = \omega \tilde{x}$; **iii)** train the model with \tilde{x} to learn the model parameters $\tilde{\theta}$; and **iv)** obtain the original distribution parameters, η , from the unscaled continuous ones, $\bar{\eta}$.¹³

13: A full description can be found in Appendix C.3.1.

Additive noise. In the dequantization step above, we add a constant 1 to promote that the shape parameter α of the Gamma likelihood is far from zero, so that L_1 does not become arbitrarily large.¹⁴ Moreover, we select the noise ϵ such that its domain is a non-zero measure subset of the unit interval, so that the original value is identifiable, and its distribution preserves the original data shape as much as possible.

14: See Figure 8.3 and Appendix C.4 for more details about the Gamma case.

Recovering the original parameters. Bernoulli and Poisson distributions are characterized by their expected value. Hence, to recover their distributional parameters on evaluation, we perform mean matching between the original distribution and its unscaled Gamma counterpart. Note that the mean of the discrete variable x is given by $\mu = \bar{\mu} - \mathbb{E}[\epsilon] - 1$, where $\bar{\mu}$ is the mean of \tilde{x} , *i.e.*, $\bar{\alpha} / \bar{\beta}$ under the unscaled Gamma distribution with parameters $\bar{\alpha}$ and $\bar{\beta}$. Therefore, we estimate the mean of the Bernoulli distribution as $p = \max(0, \min(1, \mu))$, and the rate of the Poisson as $\lambda = \max(\delta, \mu)$, where $0 < \delta \ll 1$ to ensure that λ is positive.

As the categorical distribution has several parameters, a **Bernoulli trick** is applied first. Namely, we adopt a one-hot representation of the M -class categorical distribution, and treat each class as independent Bernoulli distributions, which are suitable for the Gamma trick. To recover the categorical parameters, $\pi = [\pi_1 \ \pi_2 \ \dots \ \pi_M]$, we take the mean of

each Bernoulli variable, μ_m , and normalize them to sum up to one, *i.e.*, $\pi_m = \mu_m / \sum_i \mu_i$. Note that, when we apply Lipschitz standardization to a categorical variable, we account for the fact that it has been divided into M Gamma distributions, by setting their smoothness target to L^* / M so the total target remains unchanged, *i.e.*, $\sum_m L_m^* = L^*$.

8.4.2 Lipschitz standardization in practice

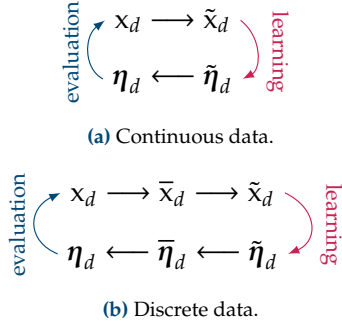


Figure 8.4: Pipeline in this chapter. For training, data is transformed and their natural parameters inferred. To evaluate, the original parameters are recovered from the transformed ones.

Figure 8.4 summarizes the pipeline employed in this chapter, where we distinguish between the **learning** and **evaluation** steps.

First, in the **learning** step we use the scaled data \tilde{x} , where we dequantize the discrete modalities beforehand with the Gamma trick so that we can scale them as well. During training, we optimize θ by using \tilde{x} as input data and infer the scaled likelihood parameters $\tilde{\eta}$.

During the **evaluation** step, we need to *undo* the entire process. Specifically, we recover the original likelihood parameters by transforming $\tilde{\eta}$ back into η , or in $\bar{\eta}$ if the original variable were discrete. For the considered likelihood functions, the mapping is given by $\eta = f(\omega) \odot \tilde{\eta}$, as described in **Proposition 8.1**. For discrete variables, we then recover the original parameters with a mapping $\bar{\eta} \mapsto \eta$ as described in **Subsection 8.4.1**. Finally, we can use η together with the original distributions.

Remark. We use **Equation 8.13** and automatic differentiation to efficiently compute the L -smoothness around $\hat{\eta}$, as well as root-finding methods to find the optimal scaling factors ω_d^* .¹⁵ However, gradient descent and backpropagation can be used out-of-the-box to solve **Equation 8.14** using $\tilde{L}_{di}(\omega_d) = \|\nabla_{\tilde{\eta}} \partial_{\tilde{\eta}_i} \tilde{\ell}(\tilde{\eta})\|$ instead of **Equation 8.13**. As a result, Lipschitz standardization is easily applicable to other distributions. We defer other models and frameworks to future work.

8.5 Empirical evaluation

In this section, we ablate the individual components that form Lipschitz standardization. Since preprocessing methods entirely rely on data, we test them on missing data imputation problems, and show that Lipschitz standardization reduces the overall imputation error without overlooking any variable, which is especially clear in the more demanding settings.

Experimental setup. We consider six datasets from the UCI repository [48] and apply BBVI to fit three PGMs: **i**) a mixture model, MM; **ii**) a matrix factorization model, MF; and **iii**) a variational autoencoder, VAE. We pick each modality distribution based on their observable properties¹⁶ and, to provide a fair initialization across all methods and datasets, continuous variables are always standardized beforehand. **Appendix C.5** contains further details and the full tabular results.

15: Details in **Appendix C.3**.

[48] Dua and Graff (2017), ‘UCI Machine Learning Repository.’

16: *E.g.*, real positive or categorical data, see **Section 7.1** for a description.

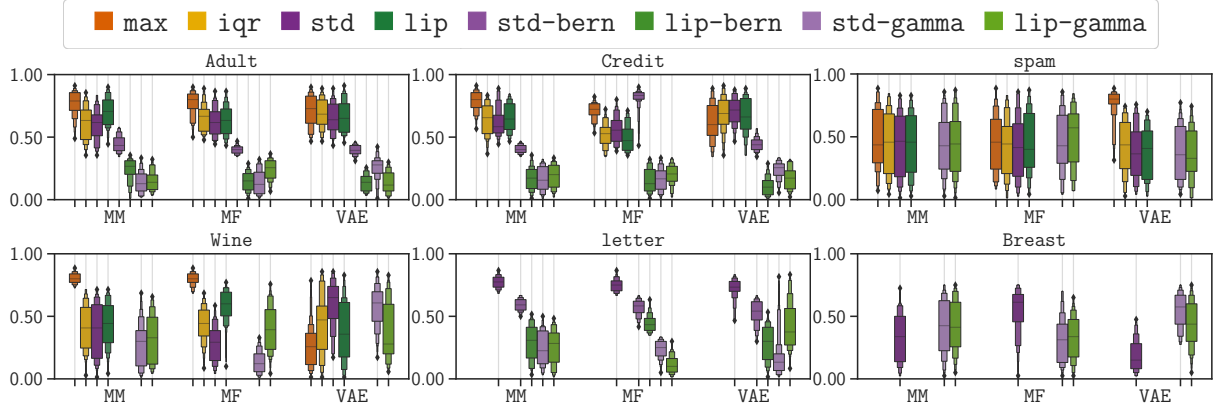


Figure 8.5: Ranking across methods based on mean ranking (R_1 , see Subsection 6.2.1) for different datasets and models (lower is better). Each method appears only when applicable to the dataset, and it is shown in the same order as in the legend.

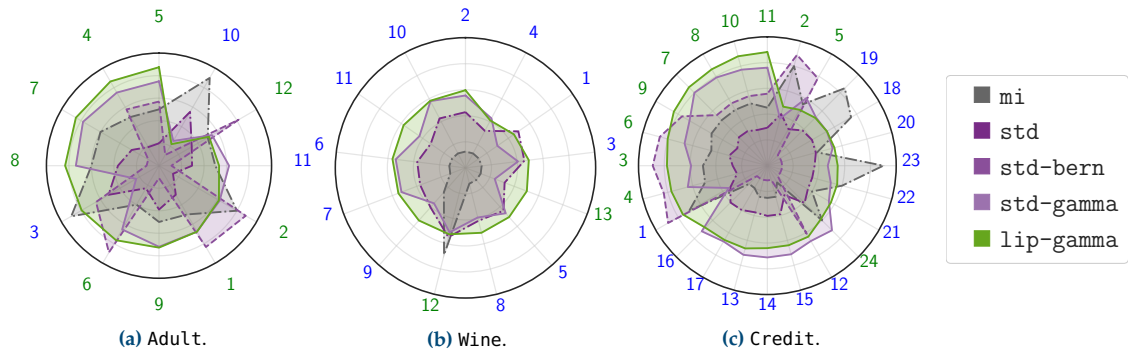


Figure 8.6: Per-modality rankings (farther from the centre is better) for different methods on the Adult, Wine, and Credit datasets. Solid lines apply the Gamma trick, dashed lines the Bernoulli trick, and dash-dotted lines apply none. Green and blue numbers represent discrete and continuous modalities.

Methods. We consider different combinations of preprocessing methods, which we clearly reflect in our naming nomenclature. Namely, we scale continuous variables with: **i)** `std`, standardization; **ii)** `max`, normalization; **iii)** `iqr`, divides by the interquartile range; and **iv)** `lip`, Lipschitz standardization. If we treat discrete variables, we add the following suffixes: **i)** `bern`, for the Bernoulli trick; and **ii)** `gamma`, if we add the Gamma trick on top of that. For example, the proposed Lipschitz standardization applies the Bernoulli and Gamma tricks, and then Lipschitz standardization to all the data, so we denote it as `lip-gamma`.

Metrics. Similar to Nazabal et al. [143], we evaluate the performance of missing-data imputation tasks using the normalized MSE for numerical variables, and the error rate for nominal ones. In Figure 8.5 we compute their mean ranking, R_1 , as described in Subsection 6.2.1, and show their per-modality rankings in Figure 8.6.

Results. Figure 8.5 summarizes the results, averaged over three missing rates of 10, 20, and 50 %, over 10 independent runs each.¹⁷ We distinguish two main groups: **i)** the methods that keep discrete data unaltered, which perform either equal or worse than the rest, with `max` performing significantly worse in some cases, and `lip` providing comparable results to the best counterpart; and **ii)** the methods which treat the categorical distributions in some way, performing equal or better than the former, and with

[143] Nazabal, Olmos, Ghahramani and Valera (2020), ‘Handling incomplete heterogeneous data using VAEs.’

¹⁷: More detailed results can be found in Appendix C.6.

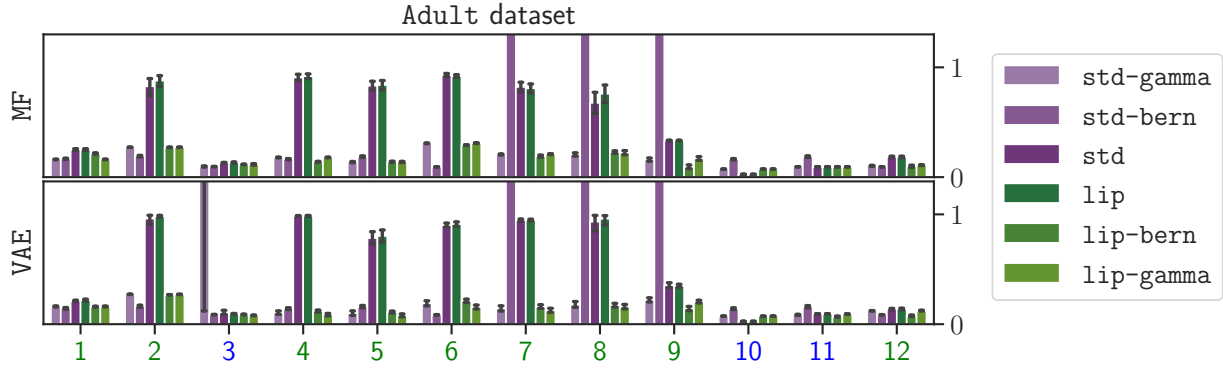


Figure 8.7: Per-modality error (normalized MSE for numerical variables, and error rate for nominal ones) on the Adult dataset. **(Top)** Matrix factorization. **(Bottom)** VAE. Note that all methods but `lip-gamma` overlook a subset of the variables. Green and blue numbers represent discrete and continuous modalities, respectively.

those methods applying the Gamma trick being more robust in general. These trends become particularly clear on the most demanding datasets (Credit and Adult), where it clearly shows that applying Lipschitz standardization with the Bernoulli trick (`lip-bern`) beats standardization (`std-bern`), even if we scale only continuous variables.

We study these results more closely in Figure 8.6, where we show the average ranking per modality for a subset of the datasets and methods, with the addition of mean imputation (`mi`) as a baseline. First, we observe that `mi` tends to rank really well in one of the modalities, while performing significantly bad in all the others. Interestingly, we find a lesser but similar trend in both `std` and `std-bern`, where there are really long spikes in their rankings, namely, `std` in Credit, and `std-bern` in Adult. Finally, we find that both `std-gamma` and `lip-gamma` rank similarly well on most of the modalities, and that `lip-gamma` covers the most area and more uniformly than any other considered method, showing that indeed Lipschitz-standardization helps towards an impartial learning.

To study robustness, we go more in detail, and focus on the Adult dataset for the MF and VAE models, with Figure 8.7 showing the modality errors for every `std` and `lip` variant. Remarkably, we find that `lip-gamma` improves the overall imputation error across dimensions *without overlooking any variable*. On the other hand, all `std` variants overlook some modalities, including `std-gamma`, where we see that the 3rd modality of Adult is completely ignored by the VAE model. More importantly, we note that this behaviour is not exclusive of this particular case, as we find similar trends in all experiments as numerically shown in Figures C.3 and C.4 and Tables C.4 to C.6 in Appendix C.5. These additional results support the effectiveness of Lipschitz standardization, which improves the performance across modalities on most settings, and *does not completely overlook a modality in any of the 540 independent runs*.

8.6 Concluding remarks

In this chapter, we have studied how to tackle modality collapse from the perspective of preprocessing methods, specifically using data-scaling approaches. We have shed new insights on the behaviour of data scaling and, in particular, data standardization, analytically showing that it

makes the log-likelihood comparably smooth around their empirical parameters, for a wide range of common distributions.

Finally, we have proposed Lipschitz standardization, a data-scaling algorithm that eases an impartial learning process by making the infinitesimal L -smoothness equal across all (discrete and continuous) data modalities around their empirical parameters. The application of Lipschitz standardization can therefore be seen as a soft-constraint inductive bias towards achieving modality-impartial solutions. Our experiments show that Lipschitz standardization outperforms existing methods, especially when the data is highly heterogeneous.

Interesting research directions include considering other probabilistic learning settings different from BBVI (*e.g.*, Hamiltonian Monte-Carlo methods [10]), as well as to investigate how to adapt our implementation into existing probabilistic programming pipelines such as Pyro [11]. Moreover, we would also like to extend the ideas of Lipschitz standardization to the in-processing setting, in which we can adapt the modality weights during optimization.

[10] Betancourt (2017), ‘A conceptual introduction to Hamiltonian Monte Carlo.’


Mitigating Modality Collapse via Impartial Optimization

9.

Si se calla el cantor, calla la vida.
Porque la vida, la vida misma, es todo un canto.

Mercedes Sosa; Si se calla el cantor

In previous chapters, we have seen how **latent variable models (LVMs)** can occasionally fit solely a subset of all the modalities, a problem we refer to as *modality collapse*, and that we have tackled with data preprocessing. While **variational autoencoders (VAEs)** [93] enjoy great success in domains such as images, text, and temporal data [127, 203, 221], their application to multimodal data remains being a challenge. In recent years, a number of tailored models for tabular [120, 143] and multimodal data [185, 199] have emerged to model this type of data, yet they still seem to suffer from the same issues to different extents.

In this chapter, we go beyond data preprocessing and study modality collapse as a result of experiencing gradient conflict in specific parts of the computational graph, which we refer to as *impartiality blocks*, and introduce in **Section 9.2**. Then, we propose a training pipeline that leverages existing gradient-conflict solutions from **multitask learning (MTL)** to encourage an impartial optimization process that does not favour a subset of modalities over the rest.

We show the flexibility of our approach by applying this training pipeline to several VAE models previously proposed in the literature to fit multimodal and tabular data (**Section 9.3**). Our empirical results on different datasets, models, and training losses (**Section 9.4**) show that impartial optimization results in a more accurate fit of the marginal, joint, and conditional distributions over all modalities.

Notation. We extensively use the set indexing notation x_A , where A is a set of indexes, *e.g.*, $\mathbf{z}_{1:K}$ denotes a sequence with indices ranging from 1 to K . Moreover, we denote by $\mathbf{1}$ the vector full of ones, and by $[\cdot]$ the concatenation operator.

9.1 Preliminaries

In this section, we briefly introduce the concepts needed for the chapter. Refer to **Chapter 7** for a more detailed introduction.

Multimodal data. We consider the input data to be multimodal, *i.e.*, to be composed of variables coming from different sources. Namely, we consider as input *i.i.d.* samples from a multimodal **random variable (R.V.)** $\mathbf{x} = [x_1 \ x_2 \ \dots \ x_D]$, where the d -th modality is fully described by x_d . Note that we do not make any assumptions on the modalities, allowing for x_d of different shapes (*e.g.*, images and their labels) and types (*e.g.*, continuous *v.s.* discrete). We refer to \mathbf{x} as heterogeneous when each modality x_d in \mathbf{x} is unidimensional. Thus, we consider heterogeneous data as a special case of multimodal data.

9.1 Preliminaries	89
9.2 Impartial optimization	90
9.3 Extending our framework	93
9.4 Experiments	97
9.5 Concluding remarks	101



github.com/adrianjav/impartial-vaes

This chapter is based on the content of: [III]: Javaloy, Meghdadi and Valera (2022), ‘Mitigating Modality Collapse in Multimodal VAEs via Impartial Optimization.’

9.1.1 State-of-the-art	90
----------------------------------	----

[93] Kingma and Welling (2014), ‘Auto-Encoding Variational Bayes.’ 

VAEs [93]. These **probabilistic generative models (PGMs)** that model the data by assuming the existence of some latent variable \mathbf{z} . Specifically, they learn the likelihood function that best approximates the input (*aka.* decoder), $p_{\theta}(\mathbf{x} | \mathbf{z})$, and an approximation to the posterior distribution of \mathbf{z} (*aka.* encoder), $q_{\phi}(\mathbf{z} | \mathbf{x})$. During training, VAEs maximize a function of the following form:

$$L(\theta, \phi) = \mathbb{E}_{\mathbf{x}} \left[\mathbb{E}_{\mathbf{z}_{1:K} \sim q_{\phi}} \left[\log \frac{1}{K} \sum_{k=1}^K \frac{p_{\theta}(\mathbf{x}, \mathbf{z}_k)}{q_{\phi}(\mathbf{z}_k | \mathbf{x})} \right] \right], \quad (9.1)$$

where $p_{\theta}(\mathbf{x}, \mathbf{z}) = p_{\theta}(\mathbf{x} | \mathbf{z})p(\mathbf{z})$, and $\mathbf{z}_{1:K}$ is an *i.i.d.* sequence of length K . This formulation includes the original **evidence lower bound (ELBO)** [93], and others like the **importance weighted autoencoder (IWAE)** loss [15].

One important detail here is that the functional form of p_{θ} (and q_{ϕ}) is usually fixed beforehand—*e.g.*, as a normal distribution—and a neural network determines its parameters, η . Also remarkably, when dealing with multimodal data, the usual practice is to assume that the likelihood fully factorizes across modalities, *i.e.*,

$$p_{\theta}(\mathbf{x} | \mathbf{z}) = \prod_{d=1}^D p_d(x_d; \eta_d(\mathbf{z}; \theta)), \quad (9.2)$$

where p_d accounts for the statistical properties of x_d .

9.1.1 State-of-the-art

Heterogeneous data. To this date, the most prominent VAEs found in the literature are HI-VAE [143],¹ originally designed for missing-data imputation tasks, and VAEM [120], designed instead for active data-acquisition tasks. More recently, SHIVAE [8] has been proposed as an extension of HI-VAE that also deals with temporal data.

1: We introduced it in Subsection 7.3.3.

Multimodal data. In this chapter, we focus on mixture-based VAEs,² which are an active area of research. While MVAE [213], MMVAE [185], and MoPoE [199] are the reference models, different extensions compatible with our framework keep being proposed, *e.g.*, using alternative training functions [186, 198]. For a survey on other multimodal methods, refer to the works of, *e.g.*, Baltrušaitis et al. [7] and Guo et al. [70].

2: Also introduced in Subsection 7.3.3.

9.2 Impartial optimization in multimodal VAEs

9.2.1 Our approach 92

In this section, we investigate the standard assumptions and goals made when designing multimodal VAEs, as well as discuss the optimization challenges that cause modality collapse. Then, we propose a flexible learning pipeline to mitigate modality collapse.

First, let us bring multimodal modelling into context. When we think of multimodal applications (*e.g.*, missing-data imputation, or joint data-generation), these are tasks that not only involve explaining the different data modalities, but jointly capturing the interactions and dependencies between each pair of modalities. In other words, the main—and often

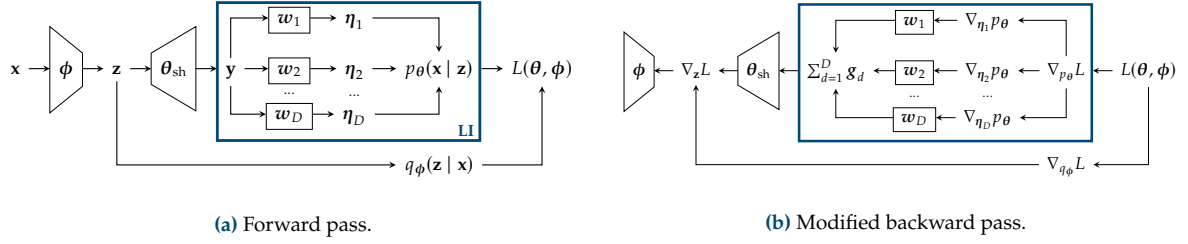


Figure 9.1: Schematic computational graph of a basic multimodal VAE: **(a)** forward pass, taking \mathbf{x} as input and producing the training objective, $L(\theta, \phi)$; **(b)** backward pass, taking $L(\theta, \phi)$, and applying the chain rule backwards along the network to generate the gradients. The impartiality block, which encloses gradient conflict, is highlighted in blue.

implicit—goal of multimodal learning is to accurately approximate the marginal, joint, and conditional distributions over *all* modalities.

Goal LI - Likelihood Impartiality To accurately approximate the marginal, joint, and conditional distributions, it is essential to accurately fit the likelihood of *all* modalities without neglecting any of them.

We thus aim for a training process that it is *impartial* to learning the likelihood of the different modalities.

In this chapter, we argue that the reason of **LI** being often unsatisfied when training multimodal VAEs lies on the computational graph resulting from the likelihood factorization in Equation 9.2. We illustrate this idea in Figure 9.1, where we highlight in blue the problematic sub-graph of the computational graph, which we refer to as **impartiality block**.

As an example, let us suppose that the last layer of the decoder is a linear layer with parameter \mathbf{W} , and let us denote by θ_{sh} the remaining decoder parameters, which are *shared* across all modalities. Then, we can write the likelihood parameters as $\boldsymbol{\eta} = \sigma(\mathbf{y}\mathbf{W})$, where \mathbf{y} is the output of the decoder up to the last layer, and σ is an element-wise transformation to ensure that each parameter satisfies its distributional constraints.³ By making the modalities operations explicit, $[\eta_1 \ \eta_2 \ \dots \ \eta_D] = \sigma(\mathbf{y}[\mathbf{w}_1 \ \mathbf{w}_2 \ \dots \ \mathbf{w}_D])$, it is clear that all modalities share \mathbf{y} , while the parameters \mathbf{w}_d are exclusive of the d -th modality.

3: E.g., having a positive variance.

An impartiality block (blue square in Figure 9.1a) encloses a sub-graph in which a split-and-merge pattern across modalities appears, which we will recurrently observe later in Section 9.3. In the forward pass, the impartiality block takes a shared \mathbf{y} as input, which is *independently* fed to each modality-specific head to compute η_d . Then, these computations are collected to produce a common output, the total likelihood $p_{\theta}(\mathbf{x} | \mathbf{z})$. Note that, outside this block, all computations are shared across modalities.

Impartiality blocks play an essential role in explaining modality collapse in multimodal VAEs. First, we need to understand the effect that the split-and-merge pattern has on the update rule of the shared parameters during optimization. That is, we need to compute the gradient of $L(\theta, \phi)$ w.r.t. θ_{sh} , passing through the computational block:⁴

4: Similar computations follow in the case of the encoder parameters, ϕ .

$$\begin{aligned}
\nabla_{\theta_{\text{sh}}} L(\theta, \phi) &= \nabla_{\theta_{\text{sh}}} \mathbf{y} \nabla_{\mathbf{y}} \eta \nabla_{\eta} p_{\theta} \nabla_{p_{\theta}} L \\
&= \nabla_{\theta_{\text{sh}}} \mathbf{y} \left(\sum_{d=1}^D \nabla_{\mathbf{y}} \eta_d \nabla_{\eta_d} p_{\theta} \right) \nabla_{p_{\theta}} L \\
&= \nabla_{\theta_{\text{sh}}} \mathbf{y} \sum_d \mathbf{g}_d,
\end{aligned} \tag{9.3}$$

where $\mathbf{g}_d := \nabla_{\mathbf{y}} \eta_d \nabla_{\eta_d} p_{\theta} \nabla_{p_{\theta}} L$ is the gradient of the loss *w.r.t.* \mathbf{y} through the d -th modality, as it is computed during back-propagation [170].

Equation 9.3 reveals why modality collapse may occur during training. Intuitively, each gradient \mathbf{g}_d represents the update direction that the model should follow to better explain the d -th modality. However, if there exist large discrepancies between different gradients, *i.e.*, in the presence of *gradient conflict*, the total gradient computation⁵ can benefit some modalities more than others, leading to an update of the shared parameters that prioritizes a subset of the modalities.

5: Namely, the sum of gradients $\sum_d \mathbf{g}_d$.

Therefore, our goal is to ensure impartiality across modalities in the computations that output the impartiality block, such that no modality is neglected, hence the name. Gradient conflict is however not an exclusive problem of multimodal VAEs. We recall that gradient conflict played a central role in Part I and we extensively discussed it in the context of MTL, and refer the reader to Chapter 4 for an overview.

9.2.1 Our approach

In this section, we propose to prevent modality collapse by modifying the backward pass of the impartiality block during training—since all outer computations are shared across modalities. To achieve this, we leverage existing MTL solutions to enforce impartial optimization.

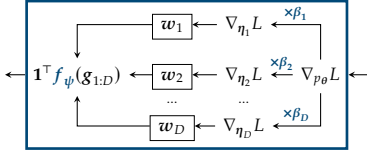


Figure 9.2: Modified backward pass within the impartiality block from Figure 9.1b to tackle gradient conflict and thus mitigate modality collapse.

We illustrate the proposed approach in Figure 9.2 and Algorithm 9.1, highlighting in blue those parts that differ from usual back-propagation. We propose two modifications within the impartiality block to bring impartiality with respect to the modalities:

- **Local step.** While back-propagating, we re-weight the gradients *w.r.t.* the likelihood parameters η_d by a factor of $\beta_d \in \mathbb{R}^+$ to keep them at a comparable scale. In this chapter, we let β_d simply be one over the dimensionality of \mathbf{x}_d , as it works well in practice, similar to other works in the literature [185]. However, more complex approaches could also be adapted to our framework [28, 89, 112]. Note, however, that in prior work re-weighting was an *ad-hoc* fix in the forward pass (rather than in the backward pass), despite breaking probabilistic assumptions.⁶ This step draws similarities to loss balancing in MTL [89].
- **Global step.** Instead of propagating to the shared parameters, θ_{sh} and ϕ , the gradient *w.r.t.* the common representation \mathbf{y} , we leverage existing MTL solutions to alleviate gradient conflict. These approaches can be commonly represented as a (parametrized) function f_ψ that takes a sequence of gradients $\mathbf{g}_{1:D}$, and returns another of equal length $\tilde{\mathbf{g}}_{1:D} := f_\psi(\mathbf{g}_{1:D})$, where the function f_ψ is selected to mitigate conflicts in the impartiality block. Thus, we apply f_ψ to the gradients *w.r.t.* \mathbf{y} , and back-propagate $\sum_d \tilde{\mathbf{g}}_d$

6: Specifically, that the likelihood needs to integrate to one.

instead of $\sum_d \mathbf{g}_d$. Note that the specific form of \mathbf{f}_ψ is determined by the MTL method that is being applied.

To sum up, we address modality collapse within each impartiality block by: **i)** scaling by β_d the local gradients *w.r.t.* η_d to make them more comparable; and **ii)** leveraging existing MTL solutions to modify the gradients *w.r.t.* \mathbf{y} such that they do not conflict, propagating these impartial gradients further into the network towards the shared parameters.

We need to make two important remarks. First, the local character of impartiality blocks is in stark contrast with traditional MTL; we do not make any assumption on the outer computational graph, nor the number of blocks in the graph. Second, the optimal choice of \mathbf{f}_ψ depends on the problem setting, with no clear winner among existing MTL solutions.⁷ Therefore, we treat the selection of algorithm \mathbf{f}_ψ as a hyperparameter, which we need to cross-validate.

```

1 input: Gradient  $\nabla_{p_\theta} L$ 
2 begin
3   for  $d = 1, 2, \dots, D$  do
4      $\mathbf{h}_d \leftarrow \beta_d \nabla_{\eta_d} p_\theta \nabla_{p_\theta} L$ 
5      $\mathbf{g}_d \leftarrow \nabla_{\mathbf{y}} \eta_d \cdot \mathbf{h}_d$ 
6   done
7    $\tilde{\mathbf{g}}_{1:D} \leftarrow \mathbf{f}_\psi(\mathbf{g}_{1:D})$ 
8   return  $\sum_d \tilde{\mathbf{g}}_d$ 
9 end

```

Finally, while we explain our approach for the impartiality block shown in [Figure 9.1](#), we provide in [Appendix D.1](#) a general formulation that allows for the more complex cases we will see in [Section 9.3](#).

MTL gradient-conflict solutions. For the choice of \mathbf{f}_ψ we consider the same solutions as those considered in [Part I](#), and refer to [Subsection 4.6.1](#) for a brief introduction on this type of approaches. Note that the contribution of this chapter is identifying *where* to modify gradients, rather than *how* to do it. As a result, our approach is orthogonal to the choice of \mathbf{f}_ψ , and new algorithms can be easily included.

9.3 Extending our framework

Next, we revisit different VAEs from [Subsection 7.3.3](#) tailored to handle multimodal data, and show how to apply the ideas from [Section 9.2](#) to them in order to prevent modality collapse.

9.3.1 Heterogeneous VAE models

The HI-VAE [\[143\]](#) is a model specialized on handling heterogeneous data. Two differences with the vanilla VAE model are particularly relevant for this chapter. First, HI-VAE introduces a Gaussian mixture prior which, according to Nazabal et al., helps to ‘overcome the limitations of having assumed a generative model that fully factorizes for every dimension’.⁸ Second, the authors also pointed out to gradient conflict as a potential issue, and introduced a normalization layer to palliate its effects. Quoting Nazabal et al.:

7: Similarly, in [Subsection 6.4.3](#) we saw that there is *a priori* no winner between modifying the gradients *w.r.t.* the common features or parameters.

Algorithm 9.1: Algorithm for the modified backward pass within an impartiality block. The proposed additions are highlighted in blue.

9.3.1 Heterogeneous VAEs . . . 93

9.3.2 Multimodal VAEs 94

[\[143\]](#) Nazabal, Olmos, Ghahramani and Valera (2020), ‘Handling incomplete heterogeneous data using VAEs.’

8: Recall [Equation 9.2](#).

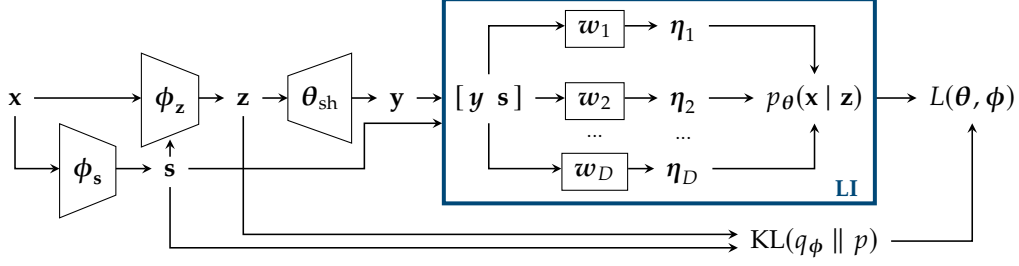


Figure 9.3: Computational graph of the forward pass of a HI-VAE, with the impartiality block highlighted in blue.

“The heterogenous nature of the data [...] results in broadly different likelihood parameters, leading in practice to heterogenous (and potentially unstable) gradient evaluations. To avoid that the gradient evaluations are dominated by a subset of attributes, we apply a batch (de-)normalization layer at the input (resp. output) of the model.”

HI-VAE introduces an additional latent variable \mathbf{s} that makes the computational graph of the model a bit more complex. However, as we show in Figure 9.3, we can still identify an impartiality block and, similar to the example in Section 9.2, the last layer of the model is linear, such that the parameters are obtained as $\boldsymbol{\eta} = \sigma([\mathbf{y} \ \mathbf{s}]\mathbf{W})$. Therefore, the derivations we made in Equation 9.3 remain being valid for \mathbf{y} . Moreover, \mathbf{s} also suffers from gradient conflict:

$$\nabla_{\phi_s} p_{\theta} \nabla_{p_{\theta}} L(\boldsymbol{\theta}, \boldsymbol{\phi}) = \nabla_{\phi_s} \mathbf{s} \left(\sum_{d=1}^D \nabla_{\mathbf{s}} \boldsymbol{\eta}_d \nabla_{\boldsymbol{\eta}_d} p_{\theta} \right) \nabla_{p_{\theta}} L. \quad (9.4)$$

Therefore, HI-VAE contains an impartiality block with two different inputs, \mathbf{y} and \mathbf{s} . We propose to tackle modality collapse by applying our approach (Algorithm 9.1) to both inputs, which implies using MTL methods twice, one for \mathbf{f}_{ψ_y} and another for \mathbf{f}_{ψ_s} .

9.3.2 Multimodal VAE models

As we mentioned in Subsection 7.3.3, one desirable application for multimodal VAEs is performing *conditional generation*, *i.e.*, sampling a modality having observed a different one, while representing the same underlying concept. For example, sample the caption for a given image, or *vice versa*. Mixture-based multimodal VAEs ease this task by introducing D modality-exclusive encoders (and decoders), using as posterior approximation a mixture model of the form:

$$q_{\phi}(\mathbf{z} | \mathbf{x}) = \frac{1}{M} \sum_{A \in \mathcal{A}} q_A(\mathbf{z} | \mathbf{x}_A), \quad (9.5)$$

where $q_A(\mathbf{z} | \mathbf{x}_A)$ is an *expert* composed of the modalities in A ,

$$q_A(\mathbf{z} | \mathbf{x}_A) \propto \prod_{d \in A} q_{\phi_d}(\mathbf{z} | \mathbf{x}_d), \quad (9.6)$$

and where $\mathcal{A} \subset \mathcal{P}(D)$ is a subset of all the possible combinations of modalities. Moreover, this generic formulation allow us to recover existing

models from the literature by selecting different values for \mathcal{A} :

$$\text{MVAE [213]: } \mathcal{A} = \{\{1, 2, \dots, D\}\},$$

$$\text{MMVAE [185]: } \mathcal{A} = \{\{1\}, \dots, \{D\}\},$$

$$\text{MoPoE [199]: } \mathcal{A} = \mathcal{P}(D).$$

One setback of considering q_ϕ a mixture model is that sampling from it is no longer differentiable. However, we can overcome this issue by using stratified sampling on the ELBO [134, 185] (Equation 7.6):

$$L(\theta, \phi) = \sum_{A \in \mathcal{A}} \mathbb{E}_{\mathbf{x}, \mathbf{z}_{1:K}^A} \left[\log \sum_{k=1}^K \frac{p_\theta(\mathbf{x}, \mathbf{z}_k^A)}{q_\phi(\mathbf{z}_k^A | \mathbf{x})} \right]. \quad (9.7)$$

We refer to Equation 9.7 as a loose loss since a tighter objective, SIWAE, can be derived using Jensen's inequality after stratifying [134, 185]:

$$\tilde{L}(\theta, \phi) = \mathbb{E}_{\mathbf{x}, \{\mathbf{z}_{1:K}^A\}_{A \in \mathcal{A}}} \left[\log \sum_{A \in \mathcal{A}} \sum_{k=1}^K \frac{p_\theta(\mathbf{x}, \mathbf{z}_k^A)}{q_\phi(\mathbf{z}_k^A | \mathbf{x})} \right]. \quad (9.8)$$

Despite being tighter, this objective is notoriously known for suffering from modality collapse. For example, Shi et al. [185] discarded using it, showing empirical evidence of modality collapse and arguing that 'it leads to situations where the joint variational posterior collapses to one of the experts in the mixture.'

Impartial optimization

Recall that our main goal is to accurately approximate the marginal, joint, and conditional distributions over all modalities (LI). To achieve this objective, we now identify in mixture-based multimodal VAEs different impartiality blocks that may stray us from our goal.

Looking at their computational graph in Figure 9.4, we find an upper impartiality block which corresponds once again to the evaluation of the factorized likelihood (Equation 9.2). For each expert A , we find such an impartiality block, having each decoder as a head and its latent variable \mathbf{z}_A as the common input. Hence, we can improve LI by applying Algorithm 9.1 to each of these blocks.

Next, we focus on specific problems of these models that may also contribute to modality collapse. Just as in Section 9.2, we first describe the goals to pursue towards enabling conditional generation.

Goal EEI - Encoder Expert-Impartiality In order to enable conditional generation, we need interchangeable encoders, so that we can replace them when modalities are missing. In other words, we need the ability to generate encoder samples that are *impartial* to the expert used.

Given a latent sample from an expert, \mathbf{z}_A , we can compute how likely is of coming from an expert A' by computing $q_{A'}(\mathbf{z}_A | \mathbf{x}_{A'})$. Similar to the way \mathbf{y} could receive gradients from $p_\theta(\mathbf{x} | \mathbf{z})$ benefiting a subset of modalities (see Section 9.2), \mathbf{z}_A can receive gradients from the mixture $q_\phi(\mathbf{z} | \mathbf{x})$ that favour a subset of modalities. This impartiality block can be

[213] Wu and Goodman (2018), 'Multimodal Generative Models for Scalable Weakly-Supervised Learning.'

[185] Shi, Narayanaswamy, Paige and Torr (2019), 'Variational Mixture-of-Experts Autoencoders for Multi-Modal Deep Generative Models.'

[199] Sutter, Daunhawer and Vogt (2021), 'Generalized Multimodal ELBO.'

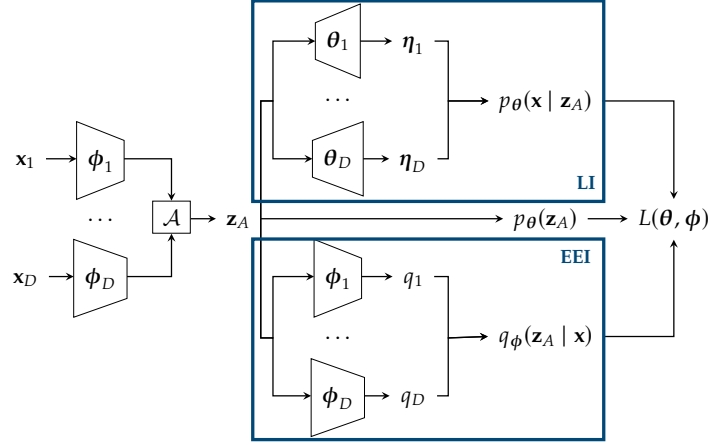


Figure 9.4: Forward pass of a mixture-based multimodal VAE. Here we only represent one \mathbf{z}_d from \mathcal{A} , i.e., only one term of the sum in Equation 9.5.

observed in the bottom part of Figure 9.4, or by computing the gradients of $L(\theta, \phi)$ w.r.t. \mathbf{z}_A passing through $q_\phi(\mathbf{z}_A | \mathbf{x})$, i.e.:

$$\nabla_{\phi_d} \mathbf{z}_A \nabla_{\mathbf{z}_A} q_\phi \nabla_{q_\phi} L(\theta, \phi) = \nabla_{\phi_d} \mathbf{z}_A \left(\sum_{A' \in \mathcal{A}} \nabla_{\mathbf{z}_A} q_{A'} \right) \nabla_{q_\phi} L. \quad (9.9)$$

Modality collapse can thus appear as a consequence of gradient conflict in Equation 9.9, having experts whose samples can only substitute a subset of other experts. We can again prevent it by applying Algorithm 9.1 to these impartiality blocks.

9: As the computational graphs become convoluted, we defer it to Figure D.2 in Appendix D.1.

Finally, we identify a third source of gradient conflict:⁹

Goal DEI - Decoder Expert-Impartiality To have proper conditional generation, similar to EEI, we need interchangeable decoders that can generate their modality using any latent sample. That is, we need decoders that are *impartial* to the expert generating the sample.

DEI relates to the passive role of the latent samples, where the decoder parameters are optimized taking these samples as input.¹⁰ In particular, each decoder $p_{\theta_d}(x_d | \mathbf{z})$ is optimized to explain the R.V. x_d given the samples from each expert, $\mathbf{z}_A \sim q_A(\mathbf{z} | \mathbf{x}_A)$, which is explicitly shown via stratification in Equations 9.7 and 9.8.

In this case, modality collapse would lead to decoders that can only generate their modality based on a subset of experts. Taking derivatives again, we can find for each decoder $p_{\theta_d}(x_d | \mathbf{z})$ an impartiality block:

$$\nabla_{\theta_d} L(\theta, \phi) = \left(\sum_{A \in \mathcal{A}} \nabla_{\theta_d} p_{\theta_d}(x_d | \mathbf{z}_A) \nabla_{p_{\theta_d}^A} L \right). \quad (9.10)$$

Note that the impartiality block in Equation 9.10 has as shared input the decoder parameters, θ_d , and each sample \mathbf{z}_A as modality-specific head. However, due to the flexibility offered by the impartiality blocks, we can reason and tackle modality collapse just as we did in the other cases: applying Algorithm 9.1 to each impartiality block.

In total, there are $2M + D$ impartiality blocks in a mixture-based VAE, for which we can use Algorithm 9.1 to palliate modality collapse. Extra details on their application can be found in Appendix D.1.

10: We do not consider the encoder parameters, since in practice we use the sticking-the-landing estimator [164].

Table 9.2: Median test reconstruction errors over five seeds for different datasets and VAE models. Statistically different values according to a corrected paired t-test ($\alpha = 0.1$) are highlighted. Models trained with our approach outperforms the baseline in most cases.

			Heterogeneous								Homogeneous			
			Adult	Credit	Wine	Diam.	Bank	IMDB	HI	rwm5yr	labour	ElNino	Magic	BooNE
Standard VAE	ELBO	Vanilla	0.213	0.128	0.086	0.187	0.203	0.082	0.170	0.105	0.109	0.109	0.064	0.042
		Ours	0.104	0.041	0.071	0.139	0.043	0.032	0.041	0.026	0.063	0.068	0.058	0.039
	IWAE	Vanilla	0.226	0.134	0.075	0.185	0.199	0.090	0.155	0.094	0.098	0.086	0.053	0.037
		Ours	0.129	0.051	0.066	0.125	0.076	0.035	0.042	0.032	0.066	0.061	0.048	0.035
	DReG	Vanilla	0.234	0.132	0.077	0.176	0.191	0.088	0.153	0.094	0.096	0.085	0.050	0.037
		Ours	0.168	0.075	0.065	0.139	0.103	0.055	0.042	0.026	0.076	0.069	0.046	0.036
	HI-VAE	Vanilla	0.127	0.107	0.126	0.114	0.141	0.079	0.105	0.044	0.100	0.098	0.062	0.039
		Ours	0.081	0.060	0.117	0.011	0.095	0.049	0.109	0.024	0.069	0.015	0.033	0.038

9.4 Experiments

In this section, we assess the approaches discussed in Sections 9.2 and 9.3 for heterogeneous and multimodal settings. All results shown here are averaged over 5 different seeds and bold numbers represent statistically significant values according to a one-sided Student’s t-test with $\alpha = 0.1$, unless stated otherwise. Additional details and results can be found in Appendices D.3 and D.4.

9.4.1 Heterogeneous data . . . 97

9.4.2 Multimodal data 98

9.4.1 Heterogeneous data

First, we focus on heterogeneous data modelling. While the setting may look simple at first, we need to deal with plenty of modalities, each one with unique properties. Moreover, models are comparatively simple, forming a breeding ground for modality collapse.

For these experiments, we use VAEs as the one introduced in Section 9.2, using as objectives the ELBO [93], IWAE [15], and DReG [202]. Additionally, we include HI-VAE [143] as an example of tailored heterogeneous model.¹¹ We consider 12 datasets collected from the UCI [48] and R [157] repositories, covering a wide range of dataset sizes and likelihoods. We assign 4 likelihood types (normal, log-normal, Poisson, and categorical) depending on the modality domain.¹² Since likelihoods are not comparable, we use the normalized **mean squared error (MSE)** as metric for numerical data, and the error rate for categorical data, following the work of Nazabal et al. [143].

11: See Subsection 7.3.3 and 9.3.1.

12: Explained in Section 7.1.

[143] Nazabal, Olmos, Ghahramani and Valera (2020), ‘Handling incomplete heterogeneous data using VAEs.’

Do we reconstruct better? Explaining the observed data explicitly appears in the objective function in Equation 9.1. Hence, if our approach works, reconstruction error should be reduced as a result of impartially learning to explain all modalities. Table 9.2 (left) shows the reconstruction error for 9 heterogeneous datasets, for which the models trained with our approach improve over those without it, Vanilla, in a statistically significant manner in **30 out of 36 cases**. Interestingly, our approach specially benefits the standard VAE model, outperforming HI-VAE¹³ in several datasets. Remarkably, for the majority of datasets, the performance of HI-VAE is also significantly improved via impartial optimization, outperforming the rest of VAE models, *e.g.*, in Adult and Diam.

13: Trained with both Vanilla and Ours.

Table 9.3: Error on the heterogeneous experiments for the baseline and our framework, aggregated by likelihood type.

	Poiss	Cat	$\log \mathcal{N}$	\mathcal{N}
Vanilla	0.058	0.158	0.064	0.041
Ours	0.083	0.065	0.057	0.039

Remark 9.1 The observation in the text corroborates a similar observation shown in Figure 8.3 of Chapter 8, where the L -constant of the Poisson *r.v.s* were significantly higher than for the rest of variables.

Remark 9.2 These results on the homogeneous setting defy the common believe that equally-evaluated tasks are less susceptible of suffering from negative transfer.

14: The same pair plot for *all* modalities is shown in Figure D.3.

15: The results for other losses are shown in Appendix D.4.3.

Remark 9.3 Specifically, SVHN has a dimensionality of 1024, MNIST of 784, and Text of only 10.

Where does the improvement come from? Table 9.3 shows again reconstruction error, but aggregated instead by data type. Here, we observe that our approach improves across all data types (and especially in categorical variables) by slightly worsening reconstruction on Poisson variables. In Appendix D.2, we show that, in our data pipeline, the gradient norm of the Poisson variables are one order of magnitude larger than for the rest of variable types, and thus dominate the learning process under standard optimization. Essentially, the trade-off we observe in Table 9.3 is the result of preventing this dominance.

Does impartial optimization help in homogeneous settings? One could reasonably suspect that modality collapse only appears when each modality uses a different likelihood type. Assigning now exclusively normal likelihoods to all variables, we show in Table 9.2 (right) that modality collapse also occurs in homogeneous settings, and that our approach may significantly improve model training even if all modalities share the same data type.

Can we generate faithful data? An important use case of heterogeneous modelling is data generation. As a qualitative example, we train on the HI dataset a VAE-ELBO using vanilla and impartial optimization. In Figure 9.5, we show three modalities generated by the two models, compared against ground-truth test data.¹⁴ While both VAEs generate well the two continuous marginals, only our approach can properly generate the categorical variable (middle), which concurs with the previous analysis on data types. More importantly, the VAE trained with our framework is able to faithfully recreate the dependencies between modalities, as it can be observed in the off-diagonal figures.

9.4.2 Multimodal data

We focus now on mixture-based multimodal VAE models. Besides the obvious architectural differences, these experiments are significantly more demanding, involving millions of parameters and high-dimensional modalities. We use SIWAE (Equation 9.8) for most of the results in the main text, as it is specially prone to modality collapse.¹⁵

We reproduce the setup of Sutter et al. [199], using the same architectures, and taking MNIST-SVHN-Text as dataset, which randomly matches positive pairs from MNIST [102] and SVHN [145], and a one-hot-encoded text generated from the common label. This is a well-suited dataset for our purposes, since the high disparity in dimensionality between modalities should exacerbate modality collapse during training. Note that in all experiments we divide the log-likelihood by the number of dimensions (local step, see Subsection 9.2.1), to offer fair comparisons, as it is a common practice in the field.

We consider MVAE, MMVAE, and MoPoE as models, which differ in the choice of experts $\mathcal{A} \subset \mathcal{P}(D)$ for the posterior approximation, $\frac{1}{M} \sum_{A \in \mathcal{A}} q_A$, as explained in Subsection 9.3.2.

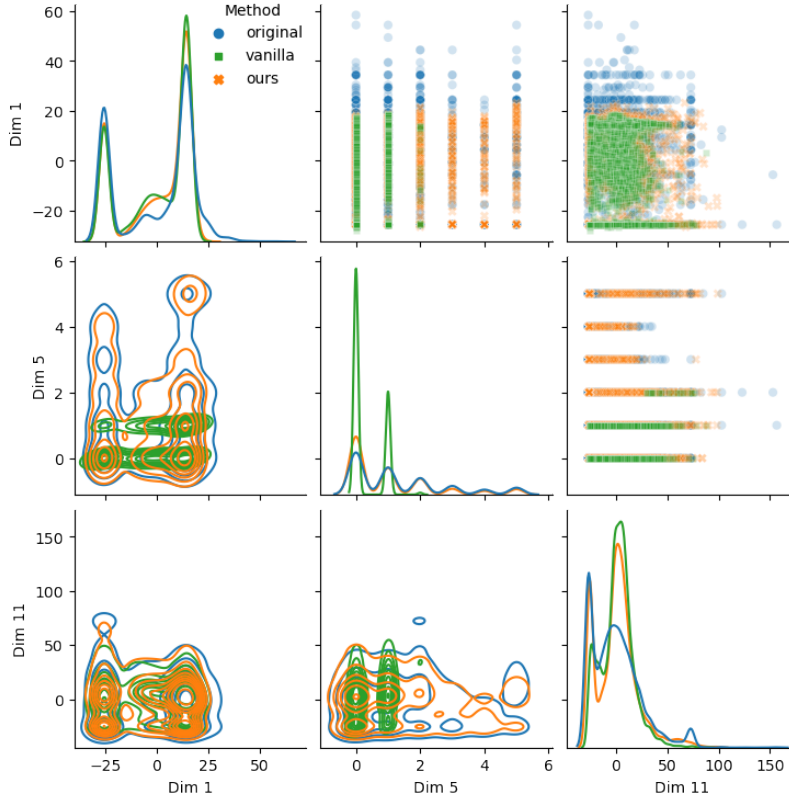


Figure 9.5: Pair plot of three dimensions of HI, generated from different VAE models. The diagonal shows marginals, upper-diagonals are scatter plots, and lower-diagonals are kernel density estimates. The VAE trained with our approach is able to generate faithful samples.

Do we reconstruct better? As a sanity check, we check again how well we are able to reconstruct each modality. Following the existing literature, we measure reconstruction capabilities in terms of *generative coherence*. Specifically, we generate latent samples using *all* the modalities as input, and reconstruct each modality x_d . Then, we feed the reconstructions to modality-specific digit classifiers, and compute the accuracy *w.r.t.* the ground-truth digit. Table 9.4 shows that our framework improves reconstruction coherence for all cases and models, sometimes by a statistically significant margin. It is also worth-noting that, in the case of MoPoE, the statistical test is inconclusive as the vanilla case has large variances.

Table 9.4: Reconstruction coherence for each modality $A \in \{M, S, T\}$ and model, trained with SIWAE as loss function.

	x_d	M	S	T
MVAE	Van.	97.37	87.47	98.83
	Ours	97.42	87.63	99.20
MMVAE	Van.	58.95	61.27	63.27
	Ours	74.16	68.93	78.17
MoPoE	Van.	75.10	67.16	76.61
	Ours	96.91	89.01	99.28

Do we improve conditional generation? One desirable ability of a multimodal VAE model is that of generating coherent samples based on other modalities. In our case, this translates to generating samples of the same digit as the input. We use again generative coherence as metric. This time, given an expert $A \subset \mathcal{P}(D)$, and an output modality x_d , we impute $x_d \sim p_{\theta_d}(x_d | \mathbf{z}_A)$ and check if the imputed value matches the original digit. Besides, for each modality we distinguish between self coherence, where we compute the average accuracy of samples conditioned on that same modality ($d = \{d\}$); and cross coherence, where samples are instead conditioned on experts not containing that modality (*i.e.*, on every $A \in \mathcal{P}(D)$ such that $d \notin A$).

Table 9.5: Self and cross generation coherence results (%) for different models on MNIST-SVHN-Text, trained using SIWAE and averaged over five different seeds. Models trained with our framework are able to sample more coherent modalities.

		Self coherence			Cross coherence									
		x_d	M	S	T	M			S			T		
		A	M	S	T	S	T	ST	M	T	MT	M	S	MS
MVAE	Vanilla		82.06	12.08	36.67	10.34	17.12	19.19	49.99	19.31	31.19	62.50	10.82	64.25
	Ours		87.63	12.47	78.88	10.75	25.99	27.85	50.02	33.13	29.62	61.17	11.67	63.63
MMVAE	Vanilla		95.90	48.30	53.02	28.43	52.52	40.45	84.44	51.08	67.77	96.80	39.96	68.38
	Ours		95.90	58.20	88.70	49.33	79.32	64.30	87.29	76.17	81.71	96.70	57.86	77.28
MoPoE	Vanilla		92.32	11.60	69.05	10.13	51.02	34.67	41.93	46.39	51.58	85.19	10.57	67.54
	Ours		90.99	12.00	83.82	10.63	62.75	52.08	28.19	46.91	43.34	79.64	10.81	90.33

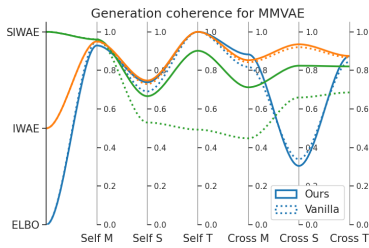


Figure 9.6: Generation coherence of MMVAE with ELBO, IWAE, and SIWAE. We improve most metrics *w. r. t.* the baseline.

Table 9.5 shows the self and cross coherence results for all models and both approaches. While there are trade-offs, we can observe that our framework in general improves both self and cross coherence across all models. For example, Text (T) and SVHN (S) were overlooked in MVAE and MMVAE, respectively, and the impartial VAE model increased self coherence for those modalities, as well as cross coherence when they appear in the expert A. Also, while SIWAE is prone to modality collapse, we note that all objectives benefit from our framework. Figure 9.6 shows a parallel coordinate plot with the generative coherence results for MMVAE, evaluated on all objectives. While SIWAE significantly improves with impartial optimization (as expected), we also improve all the different metrics for all losses.

Table 9.6: Self and cross latent classification accuracy (%) for different models and losses on MNIST-SVHN-Text.

Self latent classification				
		ELBO	IWAE	SIWAE
MVAE	Van.	69.68	69.14	68.58
	Ours	69.95	69.06	69.75
MMVAE	Van.	71.81	87.55	71.30
	Ours	87.83	90.78	85.55
MoPoE	Van.	89.85	87.23	67.58
	Ours	91.47	90.74	69.26
Cross latent classification				
		ELBO	IWAE	SIWAE
MVAE	Van.	33.60	39.15	38.36
	Ours	35.25	49.73	46.23
MMVAE	Van.	44.25	76.81	40.60
	Ours	71.42	84.80	60.50
MoPoE	Van.	66.14	83.71	40.36
	Ours	84.52	90.48	53.24

Do we generate more informative latent spaces? One key aspect of latent-space generative models is that the latent space should be rich and informative. Following the existing literature, we evaluate the quality of the latent space by training a linear classifier to predict the ground-truth label, taking samples of z as input.

Another key aspect, this time of mixture-based multimodal VAE models, is that the encoder outputs should be as similar as possible (EEI), and thus their latent spaces. Just as before, here we distinguish between self and cross latent classification accuracy. For each expert A, self latent classification refers to classifying test samples from the same expert the classifier was trained with, while cross latent classification refers to classifying test samples coming from an expert different from the one the classifier was trained with.

We show in Table 9.6 the classification accuracies, averaged over experts. We can observe that MMVAE and MoPoE significantly improve self latent classification accuracy when they are trained with our framework. More importantly, all models significantly improve the cross latent classification accuracy, independently of the loss they were trained with, indicating that the latent spaces between experts are more similar between them (*i.e.*, that the models satisfy EEI better).

Does impartial optimization add a lot of overhead? Table 9.7 shows the training times for MMVAE as we change the number of blocks for which

we apply [Algorithm 9.1](#). As expected, the training time increases as we apply more MTL algorithms during training. In the case of MMVAE, we have 9 different impartiality blocks, and yet the training time increases only by 18%, going from 10 h of training to 11.89 h. Each additional step increased in 25 min the training time, which makes us believe that the extra overhead in the second row is due to our implementation to manipulate the backward pass for [Algorithm 9.1](#).

Table 9.7: Training times of MMVAE when addressing different goals, and the number of times [Algorithm 9.1](#) is applied.

LI	EEI	DEI	hours	#
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	10.06	0
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	11.42	D
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	11.64	$2D$
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	11.89	$3D$

9.5 Concluding remarks

In this chapter, we have studied the problem of modality collapse in multimodal VAEs, showing that it can be understood as a consequence of the conflict between gradients of different modalities during training, similar to negative transfer in [Part I](#) of the thesis. Therefore, we argued in this chapter that similar inductive biases as those from MTL could be leveraged to overcome modality collapse.

We confined this conflict to a sub-graph of the computational graph, the impartiality block, and proposed a general pipeline to enforce impartial optimization across modalities. We have analysed different tailored models that contained several impartiality blocks, proving the flexibility of our modular approach. Finally, we have empirically shown that our approach can significantly improve the performance of these models on a range of datasets, losses, and metrics.

We believe the results of this chapter open interesting venues for future research. First, as our method relies on off-the-shelf solutions from MTL, it would be interesting to develop gradient-conflict solutions tailored to multimodal VAEs. Second, we believe that exploring extensions of the introduced impartiality blocks for specific applications, *e.g.*, non-modular designs that reduce the current overhead, or impartiality blocks that take into account missing patterns in real-world data, could lead to exciting future works.

Part III.

Causal Generative Models



Introduction to Causal Inference in Deep Learning

10.

A ti Señor te encomiendo unas preguntas,
por aquí abajo dudan sobre tu existencia,
por no ayudar cuando el mundo está en ayunas.

Titó; Mis palabras al cielo

10.1 Correlation \rightarrow causation	105
10.2 SCMs	106
10.3 Causal inference	107
10.4 Problem statement	109
10.5 Existing works	109

In the previous parts, we have discussed how to introduce inductive biases that steer the model towards impartial solutions. In **Part I**, we focused on jointly predicting different tasks whose measurements may not be comparable. In **Part II**, we extrapolated this knowledge to **probabilistic generative models (PGMs)** which try to capture the joint likelihood of heterogeneous **random variables (*r.v.s*)**, *i.e.*, to model $p(\mathbf{x})$. In this part, we go one step further and consider **causal generative models (CGMs)** which, in sort, try to capture the causal mechanisms defining the true *data-generating process*, instead of their joint distribution.

This chapter serves as an introduction to key concepts of causality, where we state the problem we aim to solve, as well as the relation of this part with extant literature. **Chapter 11** is a purely theoretical chapter, in which we prove that causal identifiability is possible under a sensible set of assumptions, if we provide just a causal ordering between the variables. Then, **Chapter 12** shows how to use these results to design effective hard-constraint inductive biases for CGMs, such that we can guarantee that they will capture arbitrarily well the underlying causal data-generating process, and therefore that we can perform causal reasoning with them.

10.1 Correlation does not imply causation

‘Correlation does not imply causation’ is a remarkable quote that despite (or maybe, due to) its technical wording, has been adopted by the public and that we can hear on TV series, debates, social media, or even depicted on Internet memes (see **Figure 10.1**). This phrase perfectly summarizes the fact that, just because we have observed a certain relationship between two different events, it is not necessarily the case that there is a *cause-and-effect* relationship between them.

Conceptually, **causal analysis** rests on these very same foundations, *i.e.*, on going one step beyond studying the statistical relationship between variables, and consider instead their cause-and-effect relationships. There are notorious examples that help us introduce the field to the public, *e.g.*, the positive correlation between chocolate consumption and the amount of Nobel laureates per capita [128], or the amount of stork breeding pairs and human birth rates across Europe [126]. Far from exceptional cases, there are thousands of these **spurious correlations** showing, *e.g.*, that the per-capita consumption of margarine in the US almost-perfectly correlates with the divorce rate in the state of Maine.¹

While it is easy to explain *why* causal analysis is important, formalizing a mathematical framework to model causal systems is a more complex



Figure 10.1: ‘Correlation does not imply causation’ meme. Despite what the image suggests, the gull (most likely) did not bend the fence. Found on Twitter: <https://twtr.to/F-WQG>.

[128] Messerli (2012), ‘Chocolate consumption, cognitive function, and Nobel laureates.’


[126] Matthews (2000), ‘Storks deliver babies ($p=0.008$).’

1: You can find more examples, along with artificial intelligence made-up explanations, at <https://tylervigen.com/spurious-correlations>.

[212] Woodward (2023), ‘Causation and Manipulability’

matter. In this thesis, we will take a pragmatic approach when it comes to causality, and adopt the framework that works best for us. In particular, we consider the framework of **structural causal models (SCMs)** by Pearl, which is built on a notion of causality based on manipulability [212]. However, it is worth-noting that there are other causal frameworks such as the Neyman–Rubin model based on *potential outcomes* [181].

10.2 Structural causal models

[152] Pearl (2009), ‘Causality.’ 

In this part, we model causal systems using SCMs [152]. In stark contrast with the PGMs that we saw in **Part II**, which model the (symmetric) statistical relationship between two observed variables through the joint distribution: ‘If I observe that x_i is high, x_j is likely to be high as well,’ with SCMs we model the (asymmetric) **data-generating process** that generated the observed data: ‘If x_i is set to a high value, it will make x_j have a high value as well.’

Mathematically, we can define a SCM as a tuple $\mathcal{M} = (\mathbf{f}, P_{\mathbf{u}}) \in \mathcal{F} \times \mathcal{P}_{\mathbf{u}}$ that takes a D -dimensional exogenous *R.V.*, $\mathbf{u} \sim P_{\mathbf{u}}$, and generates another D -dimensional (observed) endogenous *R.V.*, \mathbf{x} , according to \mathbf{f} . Specifically, the endogenous variables are computed as follows:

$$\mathbf{u} := [u_1 \ u_2 \ \dots \ u_D] \sim P_{\mathbf{u}}, \quad \text{and} \quad x_d = f_d(\mathbf{x}_{\text{pa}_d}, u_d), \quad (10.1)$$

for every $d \in \{1, 2, \dots, D\}$, and where each x_d is generated after all its causal dependencies, \mathbf{x}_{pa_d} , have been instantiated as well. In other words, each d -th component of the vector function \mathbf{f} maps the d -th exogenous variable to the d -th endogenous variable, given the subset of the endogenous variables that directly cause it.² Intuitively, the exogenous variables \mathbf{u} represent the stochastic component of each observed variable that cannot be explained away by the causal effects with other variables.

2: Also known as the parents of x_d .

Family of data-generating processes. In the definition above, the sets \mathcal{F} and $\mathcal{P}_{\mathbf{u}}$ are crucial, as they describe the set of possible causal generators and exogenous distributions, respectively, that we consider for the family of data-generating processes described by the SCMs. Depending on our choices for these two families, our SCMs will be more or less expressive, as well as easier or more difficult to recover from data.

Induced causal graph. Generally, we can define a causal graph as a directed graph $\mathcal{G} = (V, E)$ with one node per variable, $V = \{x_d\}_{d=1}^D$, and where two nodes are connected, $x_i \rightarrow x_j$, if there exists a causal dependency from x_i to x_j . In particular, a SCM $\mathcal{M} = (\mathbf{f}, P_{\mathbf{u}})$ induces a causal graph where an edge $x_i \rightarrow x_j$ exists if x_i is in the set of parents of x_j , *i.e.*, $x_i \in \mathbf{x}_{\text{pa}_j}$. Note, however, that this definition incurs circular dependencies, as it implies an *a priori* knowledge on the causal graph through the set of parent nodes.

Instead, we can also define an edge $x_i \rightarrow x_j$ if there exists any *functional* dependency of x_i on x_j through the j -th causal generator, *i.e.*, if

$$\partial_{x_i} f_j(\mathbf{x}, \mathbf{u}) \neq 0 \quad \text{for any } (\mathbf{x}, \mathbf{u}) \text{ such that} \quad \mathbf{x} = \mathbf{f}(\mathbf{x}, \mathbf{u}). \quad (10.2)$$

Note that, while this definition relies on checking all the set of valid (\mathbf{x}, \mathbf{u}) pairs, it only assumes that \mathbf{f} is a valid causal generator to induce the set of parents.³ More compactly, we can define the **adjacency matrix** of the induced causal graph as $A := \nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x}, \mathbf{u}) \neq \mathbf{0}$, where $\mathbf{0}$ is the constant zero function, and the comparisons are made elementwise.

Following the graph analogy, the direct causes of the d -th variable are called the **parents** of x_d , as it corresponds to the parent nodes of the d -th node in \mathcal{G} . Similarly, all the causes of a variable (direct or indirect) are referred to as its **ancestors**, and we denote them as an_d . As an example, **Figure 10.2** shows the causal graph of a simple 3-chain with adjacency matrix A , where $x_1, x_2 \in \text{an}_3$ and $x_2 \in \text{pa}_3$.

Moreover, if \mathcal{G} is acyclic, a topological ordering of the graph is called a **causal ordering**, and describes which variables do *not* cause others, and which ones *may* cause them. Specifically, a D -permutation π is said to be a causal ordering of \mathcal{M} if, for every x_i that directly causes x_j , we have that $\pi(i) < \pi(j)$. Without loss of generality, in this part we assume that the variables are ordered according to a causal ordering.

3: *I.e.*, that *there* is a correct way of applying \mathbf{f} as in **Equation 10.1**, yet we do not know *how* to do it exactly.

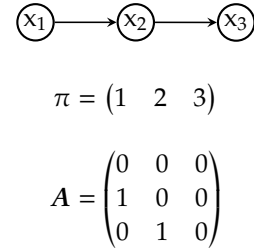


Figure 10.2: The causal graph, causal ordering π , and adjacency matrix A , of a 3-node SCM. Note that the exogenous variables were omitted.

10.3 Causal inference

In the previous sections, we have made clear the differences between correlation and causation. However, it still remains unclear what would a causal system offer that a probabilistic one would fail to provide.

PGMs are powerful tools to answer statistical questions about the observed variables that describe a system but, quoting Schölkopf and Kügelgen [179], they ‘exploit that some of the variables allow the prediction of others as long as the experimental conditions do not change.’ In other words, if the data-generating process remains the same, then a probabilistic model can make good predictions.

If the data-generating process experiences some type of change, however, PGMs fall sort. By modelling the causal mechanism \mathbf{f} that generated the data, SCMs enable **causal inference**, *i.e.*, they allow us to answer **interventional queries**.⁴ This is a significant change with respect to PGMs, as we now can simulate and predict hypothetical scenarios where an external agent performed an intervention on the system.

There are many ways in which we could modify the data-generating process. However, in this thesis we focus on (perfect) **hard interventions**, in which we assume that we can fix the value of an endogenous variable, nullifying any causal effect that other variables could have on it, while keeping the rest of the data-generating process intact. These types of interventions are plausible, *e.g.*, in clinical-trial scenarios, where we get to set the drug that test subjects take.

10.3.1 Do-operator

We simulate the effect of interventions by means of the *do-operator* [154]. Specifically, if we want to intervene on the variable x_i by fixing it to the value α , the do-operation⁵ $do(x_i := \alpha)$ simulates a physical intervention on the SCM \mathcal{M} , inducing another model $\mathcal{M}^{\mathcal{I}}$ that fixes the endogenous variable $x_i = \alpha$, and removes any causal dependency on x_i .

10.3.1 Do-operator 107

10.3.2 Ladder of causation . 108

[179] Schölkopf and Kügelgen (2022), ‘From statistical to causal learning.’

4: Which we can also call hypothetical or *what-if* questions.

[154] Pearl (2012), ‘The Do-Calculus Revisited.’

5: The symbol $:=$ stresses that x_i is the variable being modified.

Traditionally, the do-operator is implemented by yielding a SCM of the form $\mathcal{M}^{\mathcal{I}} = (\mathbf{f}^{\mathcal{I}}, P_{\mathbf{u}})$, result of replacing the i -th component of \mathbf{f} by a constant function, $f_i^{\mathcal{I}} := \alpha$, reflecting the data-generating process after an external intervention.


It is important to note that, as a result of modifying the causal mechanism that generated the endogenous variable x_i , we have in general that the induced distribution by $\mathcal{M}^{\mathcal{I}}$ do *not* correspond to the conditional distribution given that we observed $x_i = \alpha$, *i.e.*,

$$P_{\mathcal{M}}(\mathbf{x} \mid x_i = \alpha) \neq P_{\mathcal{M}}(\mathbf{x} \mid do(x_i := \alpha)) =: P_{\mathcal{M}^{\mathcal{I}}}(\mathbf{x}). \quad (10.3)$$

Remark 10.1 The equality holds if and only if the variable x_i happens to be a root node, *i.e.*, if it does not have any parent nodes.

We will revisit the do-operator later in [Chapters 11 and 12](#).

10.3.2 Pearl's ladder of causation

[152] Pearl (2009), 'Causality.' 

Equipped with the do-operator, Pearl's ladder of causation [152] is a hierarchical classification of the types of queries that a SCM can respond. These three levels (or *rungs*) correspond to semantically different questions that require different levels of knowledge on the causal mechanisms that generated the data to be answered. Specifically, these levels are:

1. **Observational.** They correspond to the passive action of *seeing*, and answer questions that only involve raw observational data. These are the queries answered by PGMs of the form $P_{\mathcal{M}}(\mathbf{x})$, *e.g.*, 'How likely am I of being sick given that I have pimples?'
2. **Interventional.** They relate with the active action of *doing*, *i.e.*, these are questions involving the effect that a certain intervention has in a population, $P_{\mathcal{M}}(\mathbf{x} \mid do(x_i := \alpha))$. For example, 'How likely am I of being sick if I decide to pop my pimples?'. Intuitively, removing your pimples this way will not have an effect on the sickness that produced them in the first place.
3. **Counterfactual.** These are the most complex queries, associated with the action of *imagining*, and answer retrospective questions with respect to an existing situation, *e.g.*, 'Would have been sick now, had I take this medicine a week ago?'. Mathematically, this is represented as $P_{\mathcal{M}}(\mathbf{x}^{\text{cf}} \mid do(x_i := \alpha), \mathbf{x}^{\text{f}})$, where \mathbf{x}^{f} is the observed factual and \mathbf{x}^{cf} is the counterfactual variable. Counterfactuals in SCMs are a bit more involved, and they are computed following a three-step procedure. Namely:
 - a) **Abduction.** Replace the exogenous distribution $P_{\mathcal{M}}(\mathbf{u})$ by the posterior of \mathbf{u} given \mathbf{x}^{f} , *i.e.*, by $P_{\mathcal{M}}(\mathbf{u} \mid \mathbf{x}^{\text{f}})$.
 - b) **Action.** Perform an external intervention $do(x_i := \alpha)$ on \mathcal{M} to obtain the intervened SCM $\mathcal{M}^{\mathcal{I}}$.
 - c) **Prediction.** Generate \mathbf{x}^{cf} using the modified SCM and the posterior distribution from the abduction step.

Both, interventional and counterfactual queries, leverage knowledge about the causal model, and thus enable causal reasoning.

10.4 Problem statement

We briefly formalize the problem to tackle in the next two chapters. In short, our goal is to approximate arbitrarily well the SCM generating our observations, such that we can then manipulate it to answer queries on all levels of the ladder of causation.

Specifically, we assume the existence of an underlying causal model \mathcal{M} that describes the data-generating process of a D -dimensional $R.V.$ $\mathbf{x} := [x_1 \ x_2 \ \dots \ x_D] \sim P_{\mathcal{M}}$. Similar to the problem statements of [Parts I](#) and [II](#), we only have access to a dataset of N *i.i.d.* samples from \mathcal{M} , *i.e.*, $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N \stackrel{i.i.d.}{\sim} P_{\mathcal{M}}$. However, in this case we also assume that some knowledge on the induced causal graph is provided. Namely, we always know at least the causal ordering, π , and at most we have knowledge about the entire adjacency matrix, A .

Our objective in this part is to design and learn a CGM that is guaranteed to approximate the underlying \mathcal{M} in *all three rungs* of Pearl's ladder of causation given enough resources,⁶ so that we can use it later to perform causal inference. Specifically, for the most complex case of counterfactual queries, this implies that the model provides a way of faithfully performing the abduction-action-prediction steps *w.r.t.* the underlying ground-truth causal model \mathcal{M} .

Remark 10.2 As we will prove later in [Chapter 11](#), this assumption is necessary as otherwise we cannot learn \mathcal{M} using only observational data.

6: *I.e.*, data, time, and parameters.

10.5 Existing works


In this section, we briefly review the existing body of work on learning the underlying SCM that generated the data using [deep learning \(DL\)](#).


The most common approach is to *individually* estimate the conditional distribution of each endogenous variable given its causal parents, *i.e.*, to learn the mapping $x_d := f_d(\mathbf{x}_{\text{pa}_d}, \mathbf{u}_d)$, thus using an independent model per observed variable. Previous works have relied on different [deep neural networks \(DNNs\)](#) to learn this mapping, *e.g.*: [normalizing flows \(NFs\)](#) [[140](#), [148](#), [151](#)], generative adversarial networks [[96](#), [218](#)], [variational autoencoders \(VAEs\)](#) [[88](#), [224](#)], Gaussian processes [[88](#)], or denoising diffusion probabilistic models [[22](#)]. These approaches follow the recursive formulation of SCMs, so they are guaranteed to be causally consistent and easily intervened upon. However, they may also suffer from error propagation⁷ and a high parameter count, which is addressed in practice with ad-hoc parameter amortization techniques [[148](#), [151](#)]. Moreover, several approaches also rely on implicit distributions [[22](#), [96](#), [151](#), [173](#)], and thus do not allow evaluating the learnt distribution.


In contrast, and similar to previous works, in this thesis we aim at learning the full causal data-generating process using a single DNN.⁸ To the best of our knowledge, the closest works to ours are those from Khemakhem et al. [[91](#)], Sánchez-Martín et al. [[174](#)] and Zečević et al. [[230](#)], as they all capture the whole causal data-generating process using a single DNN. In particular, Khemakhem et al. [[91](#)] connected affine [autoregressive normalizing flows \(ANFs\)](#) with additive noise models [[77](#)], and proposed the use of affine ANFs to learn this type of SCMs, providing identifiability results for Gaussian exogenous variables. As such, the work carried out in [Chapters 11](#) and [12](#) can be seen as a strict generalization of that of

7: Which worsens the longer the causal-graph diameter becomes.

8: Namely, using a [causal normalizing flow \(Causal NF\)](#) in [Chapter 12](#).

[[91](#)] Khemakhem, Monti, Leech and Hyvärinen (2021), 'Causal Autoregressive Flows.' 

[[174](#)] Sánchez-Martín, Rateike and Valera (2022), 'VACA: Designing Variational Graph Autoencoders for Causal Queries.' 

[[230](#)] Zečević, Dhimi, Velivcković and Kersting (2021), 'Relating Graph Neural Networks to Structural Causal Models.' 

Khemakhem et al. [91], as it provides significantly more general causal identifiability results and connects them with general ANFs.

Another relevant set of works is that of Sánchez-Martín et al. [174] and Zečević et al. [230], which connect SCMs with **graph neural networks (GNNs)** and, while making little assumptions on the underlying SCM, they also lack any theoretical guarantees, *e.g.*, interventions on the GNN are performed by severing the graph, which we show in **Appendix E.1** do not work in the general case. Nevertheless, it is worth noting that the way we introduce inductive biases in the architecture of ANFs in **Chapter 12** is inspired by these previous works.

Our work significantly extends this line of research. More specifically, in the following chapters we study how to learn the underlying causal data-generating process using a single DNN, both theoretically and in practice. In **Chapter 11**, we show how to transform a broad family of SCMs into a reduced family that is causally identifiable from observational data and a causal ordering. Moreover, we extend these results to handle discrete data and partial-knowledge about the causal ordering, and equip this family of SCMs with an implementation of the do-operator well-suited for them. Then, we show in **Chapter 12** how to put these results into practice, proving that the family of ANFs can be naturally understood as parametric members of the aforementioned reduced family of SCMs, and that we can effectively learn the underlying SCMs with them if we introduce inductive biases that exploit our knowledge about the causal graph. As a result, Causal NFs are the first-of-their-kind DL models to provably approximate such a broad family of SCMs.

Causal Identifiability Given a Causal Ordering

11.

Meto el dedo en la llaga y me escuece por dentro
porque también es mi llaga y no soy el mejor ejemplo.
Y no me des la razón si no la tengo,
pero no permitiré que vengan a fisgar mi templo.

Santiuве; Oración

In the previous chapter, we have discussed that our final objective is to train a **causal generative model (CGM)** that can approximate an unknown **structural causal model (SCM)** using only observational data, and information about the causal graph induced by the true SCM. In this chapter, we focus entirely on the theory of causal identifiability, and study how to constraint the family of possible CGMs, such that the causal models become causally identifiable from the known information,¹ while keeping the type of data-generating processes that we can model as broad as possible.

To this end, we will first make a number of common sensible assumptions on the causal generators and the distribution of the exogenous variables, which will allow us to specify our final goal (see [Section 10.4](#)) as a set of clearly defined objectives to achieve. Then, we exploit the fact that SCMs can be algebraically modified to reduce the set of possible SCMs, thus restricting the family of causal generators. Under this family of SCMs, we then formally prove causal identifiability from observational data and a causal ordering by leveraging existing results from the non-linear **independent component analysis (ICA)** literature.

Finally, we make these results better suited for real-world scenarios in two ways. First, we extend our identifiability results to more realistic settings where the data could be discrete, and where we might have only partial knowledge about the causal ordering. Second, we provide a re-implementation of the do-operator tailored to our assumptions that can be deployed on any SCM without modifying the causal generators, enabling general CGMs to solve causal inference tasks.

Additional notation. To ease the notation and focus on reasoning about the causal dependencies between variables, we introduce the notion of structural equivalence. Namely, we say that two matrices S and R are **structurally equivalent**, denoted $S \equiv R$, if both matrices share the same zero entries, *i.e.*, if they have zeroes exactly in the same positions. Similarly, we say that S is **structurally sparser** than R , denoted as $S \leq R$, if whenever an element of R is zero, the same element of S is zero.² Moreover, we use similar definitions for linear functions (*e.g.*, the Jacobian map) and talk instead of constant zero functions.

11.1 Solution characterization	111
11.2 SCM representations	113
11.3 Causal identifiability	116
11.4 Real-world extensions	119
11.5 Do-operator	122
11.6 Concluding remarks	124

This chapter is based on the content of:
[JV]: Javaloy, Sánchez-Martín and Valera (2023), ‘Causal normalizing flows: from theory to practice.’

1: *I.e.*, that we can recover them from data and a causal ordering.

2: This is reminiscent of the notion of absolute continuity in measure theory.

11.1 Solution characterization

Our final goal is, under the assumption that there is an underlying causal model \mathcal{M} that generated our input dataset X , and of which we know

11.1.1 Assumptions	112
11.1.2 Characterization	112

its causal ordering, to find a parametric model that approximates \mathcal{M} as closely as possible. In this chapter, however, we relax this goal and assume directly that our approximation of \mathcal{M} is another member of the same family of SCMs, $\mathcal{M} \in \mathcal{F} \times \mathcal{P}_{\mathbf{u}}$. Our focus is then on **causal identifiability**, *i.e.*, on whether we can identify the member of $\mathcal{F} \times \mathcal{P}_{\mathbf{u}}$ that generated the observational distribution, and which admits a causal ordering equal to the observed one.

11.1.1 Assumptions

First, we restrict the family of SCMs we consider by imposing some regularity conditions that ensure a well-behaved data-generating process, ruling out unobserved variables in the system. Specifically, we make the following assumptions:

1. **Real-valued noise.** We consider the exogenous variables to be in the real domain, *i.e.*, $\mathbf{u} \in \mathbb{R}^D$, such that the causal generators can be continuous and differentiable.
2. **Diffeomorphic generators.** We therefore assume the generator \mathbf{f} to be invertible, and that both \mathbf{f} and its inverse are differentiable. As a consequence, we can differentiate the data-generating process and go back and forth between exogenous and endogenous variables.
3. **Acyclic graphs.** We assume that there are no feedback loops in the data-generating process. This is equivalent to saying that the induced causal graph is a **directed acyclic graph (DAG)**, or that there exists at least one way of sampling endogenous variables from the SCM following [Equation 10.1](#).
4. **Causal sufficiency.** We also consider that there are no hidden confounders, *i.e.*, that there are no unobserved endogenous variables that may be having a causal effect in two or more of the considered endogenous variables. This is equivalent to saying that the endogenous distribution fully factorizes, *i.e.*, that $P_{\mathbf{u}}(\mathbf{u}) = \prod_d P_{u_d}(u_d)$.

Far from exotic, this set of assumptions turn out to be quite common in the literature: the first two assumptions in the identifiability literature [90, 140, 216], as it allows to unambiguously invert the generative process and to use real analysis tools; and the last two assumptions, are commonly made in the causality literature to have a ‘well-defined, unique observational distribution’ [64, 179].

11.1.2 Characterization

While we have a clear goal in terms of the *functionality* that our approximation to \mathcal{M} should provide,³ we do not have a clear *characterization* of how such an approximation should be. It turns out that, in order to successfully answer each type of query in the ladder, we need to impose extra constraints on the set of valid solutions. Namely:

1. **Expressive.** In order to match the observational queries, we need to make sure that our space of solutions is expressive enough to approximate arbitrarily well the joint observational distribution of \mathcal{M} , *i.e.*, $P_{\mathcal{M}}(\mathbf{x})$. However, this is not a concern for this chapter, as we directly consider the space of SCMs, $\mathcal{F} \times \mathcal{P}_{\mathbf{u}}$.

3: *I.e.*, make the same predictions as the original model in all three levels of the ladder of causation.

2. **Causally consistent.** While the observational distribution can be matched using spurious correlations,⁴ in order to match any interventional distribution, the model approximation $\tilde{\mathcal{M}}$ should exhibit the same causal dependencies as the original model between variables. Intuitively, since interventions modify the causal graph, if the two models induce different graphs, then their intervened distributions cannot match: **i)** either information would leak between variables; or **ii)** necessary information would be cut off.
3. **Identifiable.** Due to the abduction step (see [Subsection 10.3.2](#)), the approximation $\tilde{\mathcal{M}}$ from \mathbf{x} should be able to isolate the exogenous variables in order to match counterfactual distributions, *i.e.*, the **random variable (R.V.)** \tilde{u}_d recovered by $\tilde{\mathcal{M}}$ should contain only information about the original u_d .⁵ Similar to the previous case, we would otherwise find that information between variables would be intertwined, even if we were successfully removing their causal connections during the intervention.

4: As probabilistic models do in general.

5: Recovering the exact exogenous variables is not necessary, as we care about the information the variable contains, and not its representation.

Therefore, in order for a causal model to faithfully approximate the underlying SCM in *all three levels* of causal inference queries, the model should be identifiable, causally consistent and, in the case of the parametric models considered in [Chapter 12](#), expressive enough to match the observational distribution.

11.2 The multiple representations of SCMs

In this section, we present one of the core ideas of this part of the thesis. Specifically, we exploit the fact that SCMs are a special case of a system of equations, and thus we can perform algebraic manipulations to the right side of the equations to come up with alternative representations of the same causal system that, as we will shortly see, bring us new perspectives and enable the theory presented below.

[11.2.1 Illustrative example](#) . . . [113](#)

[11.2.2 Non-linear case](#) . . . [115](#)

11.2.1 Illustrative linear example

Let us start with an illustrative linear example, which will help us build an intuition on the different representations that any SCM can have.

Recursive SCM. First, assume that we are given a linear SCM described in the same way as we introduce them in [Equation 10.1](#), where the d -th equation explicitly depends on u_d , and the rest of dependencies with the exogenous variables are implicitly stated through the direct causes of x_d , *i.e.*, through \mathbf{x}_{pa_d} . In our particular example,

$$\begin{cases} x_1 = u_1 \\ x_2 = 2x_1 + u_2 \\ x_3 = 3x_2 + u_3 \end{cases} \quad (11.1)$$

which we can compactly write as $\mathbf{x} = \mathbf{G}\mathbf{x} + \mathbf{I}\mathbf{u}$. While this recursive formulation is intuitive for understanding the causal dependencies between endogenous variables (see [Figure 11.1a](#)), it is also computationally inconvenient, as it entails iteratively solving the system according to the causal dependencies for tasks such as sampling new data.

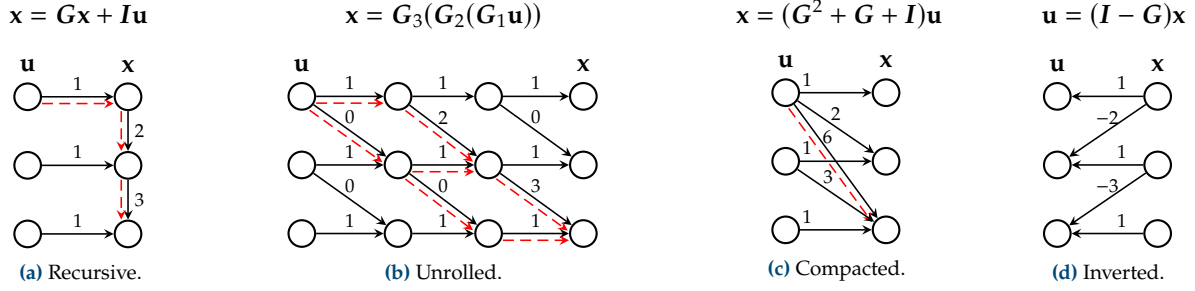


Figure 11.1: Example of the linear SCM $\{x_1 := u_1; x_2 := 2x_1 + u_2; x_3 := 3x_2 + u_3\}$ written **(a)** in its usual recursive formulation; **(b)** without recursions, with each step made explicit; **(c)** without recursions, as a single step; and **(d)** writing u as a function of x . Red dashed arrows show the influence of u_1 on x_3 for all $u \mapsto x$ systems. Note that in the linear case we have $A := G \neq 0$, and that $G_1, G_2, G_3 \leq G + I$ are any three matrices such that their product equals $G^2 + G + I$.

Unrolled SCM. Instead, we can write the equations as a function from u to x by removing the recursive dependencies iteratively, substituting each x_d on the right-hand side by their current expression. That is, we can unroll the equations:

$$\begin{cases} x_1 = u_1 \\ x_2 = 2x_1 + u_2 \\ x_3 = 3x_2 + u_3 \end{cases} \Rightarrow \begin{cases} x_1 = u_1 \\ x_2 = 2u_1 + u_2 \\ x_3 = 3(2x_1 + u_2) + u_3 \end{cases} \Rightarrow \begin{cases} x_1 = u_1 \\ x_2 = 2u_1 + u_2 \\ x_3 = 6u_1 + 3u_2 + u_3 \end{cases} \quad (11.2)$$

We can define this process as a multi-step function

$$\begin{cases} z_1^1 = u_1 \\ z_2^1 = u_2 \\ z_3^1 = u_3 \end{cases} \Rightarrow \begin{cases} z_1^2 = z_1^1 \\ z_2^2 = 2z_1^1 + z_2^1 \\ z_3^2 = z_3^1 \end{cases} \Rightarrow \begin{cases} x_1 = z_1^2 \\ x_2 = z_2^2 \\ x_3 = 3z_2^2 + z_3^2 \end{cases} \quad (11.3)$$

which we can compactly write as three linear operations $x = G_3(G_2(G_1u))$. Note that the matrices G_1, G_2, G_3 are not unique, and that they are valid as long as $G_i \leq G$ and produce the same final output. Despite removing the recursions, we have now as many functions as the recursion depth of the original system of equations (see [Figure 11.1b](#)).

Compacted SCM. Therefore, a natural step here is to compress this sequence of operations into a single linear function. That is, we can directly use the linear operation described at the end of [Equation 11.2](#):

$$\begin{cases} x_1 = u_1 \\ x_2 = 2u_1 + u_2 \\ x_3 = 6u_1 + 3u_2 + u_3 \end{cases} \quad (11.4)$$

Moreover, we can derive the same compacted linear form by directly unrolling the equations in matrix form:

$$\begin{aligned} x &= Gx + Iu \\ x &= G(Gx + Iu) + Iu \\ x &= G(G(Gx + Iu) + Iu) + Iu \\ x &= \cancel{G^3}^0 x + G^2u + Gu + Iu = (G^2 + G + I)u, \end{aligned}$$

Remark 11.1 This derivation can help us get an intuition of what happens in the non-linear case, where the Jacobian plays the role of G .

where G^3 is a zero matrix as the induced causal graph has a diameter of two. Now, we have a single function that directly maps exogenous to endogenous variables, $\mathbf{u} \mapsto \mathbf{x}$, with the caveat that we cannot longer distinguish direct and indirect paths: all paths have collapsed into one.

Inverted SCM. Last, we can take any of these different representations and solve the system of equations to get the inverse map, $\mathbf{x} \mapsto \mathbf{u}$. In this case, the simplest approach is to take the recursive formulation:

$$\begin{cases} x_1 = u_1 \\ x_2 = 2x_1 + u_2 \\ x_3 = 3x_2 + u_3 \end{cases} \Rightarrow \begin{cases} u_1 = x_1 \\ u_2 = x_2 - 2x_1 \\ u_3 = x_3 - 3x_2 \end{cases} \quad (11.5)$$

and, in vectorial form,

$$\mathbf{x} = G\mathbf{x} + I\mathbf{u} \Rightarrow \mathbf{u} = (I - G)\mathbf{x}. \quad (11.6)$$

This turns out to be a convenient SCM representation to work with, as we can easily perform the abduction step needed for counterfactuals.

11.2.2 Non-linear SCM representations

In this section, we discuss how we can manipulate non-linear SCMs similarly to their linear counterparts, and generalize the reasoning about the causal relationships of a SCM carried out in the previous section to the general case.

To this end, let us suppose that we have a non-linear SCM \mathcal{M} of the form $\mathbf{x} = \mathbf{f}(\mathbf{x}, \mathbf{u})$, which induces the same causal graph as the linear example in Figure 11.1, so that the reader can use Figure 11.1 as a reference again, *i.e.*, let us assume that

$$\nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x}, \mathbf{u}) \equiv A := \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad \nabla_{\mathbf{u}} \mathbf{f}(\mathbf{x}, \mathbf{u}) \equiv I := \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (11.7)$$

We omit the recursive representation in the following, as we defined SCMs using this formulation in Equation 10.1.

Unrolled SCM. As in the linear case, we can unroll the equations by having multiple functions $\mathbf{z}^l = \mathbf{f}_l(\mathbf{z}^{l-1})$, and at each step unroll the equations for which we already know the non-recursive formulation of its parents. Again, we can write these operations in multiple ways, as long as they produce the same final function,⁶ and that they respect the causal dependencies provided by $I + A$.

6: After composing the functions, $\mathbf{f}_1 \circ \mathbf{f}_2 \circ \dots \circ \mathbf{f}_L = \mathbf{f}$.

Collapsed SCM. Just as before, we can collapse all operations into a single function. Since all functions are structurally equivalent to $I + A$, it is straightforward to show that their composition has, in general, a Jacobian matrix structurally equivalent to $I + \sum_l^{\text{diam}(A)} A^l$.

Remark 11.2 For functions, as structural equivalence needs to hold at every point, we have for most cases that, if $A_1 \equiv A_2$ and $B_1 \equiv B_2$, then $A_1 B_1 \equiv A_2 B_2$. Hence, we have that $\prod_l \nabla_{\mathbf{z}^{l-1}} \mathbf{f}_l(\mathbf{z}^{l-1}) \equiv \prod_l (I + A)$ which equals $I + \sum_l^{\text{diam}(A)} A^l$.

Reverse SCM. By assumption, each $f_d(\mathbf{x}_{\text{pa}_d}, u_d)$ is bijective with respect to u_d , and we can thus always compute its inverse to obtain u_d as a function of the observed values, *i.e.*, $u_d = f_d^{-1}(\mathbf{x}_{\text{pa}_d}, x_d)$. Then, since $\nabla_{\mathbf{x}} \mathbf{f} \equiv \mathbf{A}$, we have in general that $\nabla_{\mathbf{u}} \mathbf{f}^{-1} \equiv \mathbf{I} - \mathbf{A}$.

In summary, for any SCM we can always reason as we did with the illustrative linear example, but using the Jacobian matrices to reason about causal dependencies between variables through their structural equivalence, independently of the complexity of the causal model.

11.3 Causal identifiability

11.3.1 SCM quotient space . 116

11.3.2 TMI SCM identifiability 118

In this section, we prove causal identifiability for our family of SCMs. To this end, we first show that we can reduce the space of all the considered SCMs by removing all redundant models, *i.e.*, those that generate the same causal system. Then, we prove that this reduced set is causally identifiable by showing that they are identifiable and causally consistent, as we discussed before in [Subsection 11.1.2](#).

11.3.1 SCM quotient space

First, we precisely formalize the meaning of two SCMs being equal, and being causally equivalent. Specifically, we say that two SCMs $\mathcal{M}, \tilde{\mathcal{M}} \in \mathcal{F} \times \mathcal{P}_{\mathbf{u}}$ are **equal** if the tuple that defines them is the same, *i.e.*,

$$\mathcal{M} = \tilde{\mathcal{M}} \iff \{\mathbf{f} \stackrel{\text{a.e.}}{=} \tilde{\mathbf{f}} \text{ and } P_{\mathbf{u}} = P_{\tilde{\mathbf{u}}}\}, \quad (11.8)$$

where $\stackrel{\text{a.e.}}{=}$ means ‘equal almost everywhere’ in the measure-theoretical sense.⁷ In contrast, we say that the two SCMs are **causally equivalent** if, with the same $P_{\mathbf{u}}$, they induce the same observational, interventional, and counterfactual distributions. In mathematical terms,

$$\mathcal{M} \sim \tilde{\mathcal{M}} \iff \{P_{\mathcal{M}} = P_{\tilde{\mathcal{M}}}\} \text{ and } \{P_{\mathbf{u}} = P_{\tilde{\mathbf{u}}}\} \text{ and} \quad (11.9)$$

$$\{P_{\mathcal{M}}(\cdot \mid do(\mathbf{x}_d := \alpha)) = P_{\tilde{\mathcal{M}}}(\cdot \mid do(\mathbf{x}_d := \alpha))\} \\ \forall d \in \{1, 2, \dots, D\}, \forall \alpha \in \mathbb{R} \} \text{ and} \quad (11.10)$$

$$\{P_{\mathcal{M}}(\cdot \mid do(\mathbf{x}_d := \alpha), \mathbf{x}^f) = P_{\tilde{\mathcal{M}}}(\cdot \mid do(\mathbf{x}_d := \alpha), \mathbf{x}^f)\} \\ \forall d \in \{1, 2, \dots, D\}, \forall \alpha \in \mathbb{R}, \forall \mathbf{x}^f \in \mathbb{R}^D\}. \quad (11.11)$$

Remark 11.3 In layman’s terms, causal equivalence groups all the different ways of modifying the structural equations without changing the causal model, *e.g.*, as in [Section 11.2](#).

Despite the similar nature of both definitions, note that they are *not* equivalent, *i.e.*, different SCMs can generate the same causal data-generating process. For example, if $P_{\mathbf{u}}$ were an isotropic multivariate Gaussian distribution, then $(\mathbf{f}, P_{\mathbf{u}})$ and $(\mathbf{f} \circ \mathbf{R}, P_{\mathbf{u}})$, where \mathbf{R} is a rotation operation, would generate the exact same causal data-generating process.

It is straightforward to show that causal equivalence is indeed an equivalence relation⁸ and, therefore, we can define the quotient set of $\mathcal{F} \times \mathcal{P}_{\mathbf{u}}$ according to \sim , *i.e.*,

$$\mathcal{F} \times \mathcal{P}_{\mathbf{u}} / \sim := \{[\mathcal{M}] : \mathcal{M} \in \mathcal{F} \times \mathcal{P}_{\mathbf{u}}\}, \quad (11.12)$$

where $[\mathcal{M}]$ is the equivalence class of \mathcal{M} , *i.e.*, the set of all SCMs that are causally equivalent to \mathcal{M} ,

$$[\mathcal{M}] := \{\tilde{\mathcal{M}} \in \mathcal{F} \times \mathcal{P}_{\mathbf{u}} : \mathcal{M} \sim \tilde{\mathcal{M}}\}. \quad (11.13)$$

7: That is, that the two functions only differ in a set of measure zero.

8: That is, that \sim is a reflexive, symmetric, and transitive relation.

In other words, $\mathcal{F} \times \mathcal{P}_{\mathbf{u}}/\sim$ is the reduction of all considered SCMs to those that are not causally equivalent. What we have left to prove is that the quotient set is isomorphic to a more practical set of SCMs.

Triangular monotonically increasing (TMI) maps. We consider the set of TMI SCMs, *i.e.*, causal data-generating processes where the structural equations are TMI maps, $\text{TMI} \times \mathcal{P}_{\mathbf{u}}$. In particular, we call \mathbf{f} a TMI map if the function is triangular, *i.e.*, the d -th output depends only on the first d inputs, and monotonically increasing, *i.e.*, the d -th output is monotonically increasing *w.r.t.* the d -th input. In mathematical terms,

$$f_d(\mathbf{x}) = f_d(x_1, x_2, \dots, x_d) \quad \text{and} \quad \partial_{x_d} f_d(\mathbf{x}) > 0. \quad (11.14)$$

Knöthe-Rosenblatt (KR) transport [95, 166]. If we have the densities of two distributions, say $P_{\mathcal{M}}$ and $P_{\mathbf{u}}$ from a SCM $\mathcal{M} = (\mathbf{f}, P_{\mathbf{u}})$, the KR transport from $P_{\mathbf{u}}$ to $P_{\mathcal{M}}$ is a function that maps the density of one distribution to the other, *i.e.*, $\text{KR}_{\#} p_{\mathbf{u}} = p_{\mathcal{M}}$. Moreover, it can be shown that the KR function is a TMI map and that, given an ordering for the variables, this TMI map is unique almost everywhere [176].

[176] Santambrogio (2015), ‘Optimal transport for applied mathematicians.’

In a slight abuse of notation, we define the KR transport of a SCM $\mathcal{M} = (\mathbf{f}, P_{\mathbf{u}})$ as the TMI SCM where the KR transport that follows the same *causal ordering* as \mathcal{M} substitutes the structural equations, *i.e.*,

$$\text{KR}(\mathcal{M}) := (\text{KR}, P_{\mathbf{u}}) \quad \text{where} \quad \text{KR}_{\#} p_{\mathbf{u}} = p_{\mathcal{M}}. \quad (11.15)$$

Furthermore, the set of KR-transformed SCMs is the set of TMI SCMs,

$$\{\text{KR}(\mathcal{M}) : \mathcal{M} \in \mathcal{F} \times \mathcal{P}_{\mathbf{u}}\} = \text{TMI} \times \mathcal{P}_{\mathbf{u}}, \quad (11.16)$$

and it is also useful to note that $\text{KR}(\text{KR}(\mathcal{M})) = \text{KR}(\mathcal{M})$.

Remark 11.4 To prove Equation 11.16, the \subseteq part is directly given by the definition of KR transport, and the \supseteq is a consequence of the uniqueness of the KR to map $P_{\mathbf{u}}$ into $P_{\mathcal{M}}$.

SCMs as TMI maps. Finally, we demonstrate that any SCM following the assumptions in Subsection 11.1.1, $\mathcal{M} \in \mathcal{F} \times \mathcal{P}_{\mathbf{u}}$, is causally equivalent to a *single* TMI SCM. To this end, we show that the quotient space of causally equivalent SCMs is isomorphic to the set of TMI SCMs, *i.e.*,

$$\mathcal{F} \times \mathcal{P}_{\mathbf{u}}/\sim = \text{TMI} \times \mathcal{P}_{\mathbf{u}}. \quad (11.17)$$

First, given a SCM $\mathcal{M} = (\mathbf{f}, P_{\mathbf{u}})$ with $\mathbf{f} : \mathbb{X} \times \mathbb{U} \rightarrow \mathbb{X}$ as in Equation 10.1, we can always unroll \mathbf{f} by recursively replacing each x_i in the causal equation by its function f_i , obtaining an equivalent non-recursive function $\mathbf{f} : \mathbb{U} \rightarrow \mathbb{X}$.⁹ Second, we always have that

9: See Figure 11.1b for an example.

$$\mathcal{M} \sim \text{KR}(\mathcal{M}) \in \text{TMI} \times \mathcal{P}_{\mathbf{u}}, \quad (11.18)$$

which can be proved by noting that $\text{KR}(\mathcal{M})$ produces the same observational distribution, and shares both the exogenous distribution and the induced causal graph.¹⁰ In other words, for every \mathcal{M} we always have one TMI candidate for representative of the equivalence class $[\mathcal{M}]$. Last, we can show that this candidate is *unique* by recalling the uniqueness of the KR transports in the TMI space, *i.e.*, we have that

10: This is direct to prove using the definition of the KR map. Hence, $\text{KR}(\mathcal{M})$ fulfils the conditions from Subsection 11.1.2.

$$\mathcal{M} \sim \tilde{\mathcal{M}} \in \text{TMI} \times \mathcal{P}_{\mathbf{u}} \Rightarrow \text{KR}(\mathcal{M}) \sim \tilde{\mathcal{M}} \Rightarrow \text{KR}(\mathcal{M}) = \tilde{\mathcal{M}}. \quad (11.19)$$

\uparrow
by Equation 11.18
 \uparrow
uniqueness in TMI space [176]

11.3.2 TMI causal identifiability

We have shown that, using the KR transport, we can reduce the space of SCMs $\mathcal{F} \times \mathcal{P}_{\mathbf{u}}$ to only those causally non-equivalent SCMs, and that this set is exactly the set of TMI SCMs, $\text{TMI} \times \mathcal{P}_{\mathbf{u}}$.

We can now leverage existing identifiability results from the context of non-linear ICA to show that the set of TMI SCMs is causally identifiable. Specifically, Xi and Bloem-Reddy [216] proved the following result, which we have re-stated to match our setting:

[216] Xi and Bloem-Reddy (2023), ‘Indeterminacy in Generative Models: Characterization and Strong Identifiability.’

Theorem 11.1 (Identifiability) If two elements of the family $\text{TMI} \times \mathcal{P}_{\mathbf{u}}$ produce the same observational distribution, then the exogenous distribution of the two data-generating processes differ by an invertible, component-wise transformation.

Proof. By Theorem 2.2 of Xi and Bloem-Reddy [216], $\text{TMI} \times \mathcal{P}_{\mathbf{u}}$ is identifiable up to TMI maps that transport exogenous distributions. The KR map is the unique TMI map that does this and, since all $P_{\mathbf{u}}$ are fully-factorized, each component of the KR map is a composition of marginal cumulative and quantile functions, *i.e.*, it is a component-wise transformation. \square

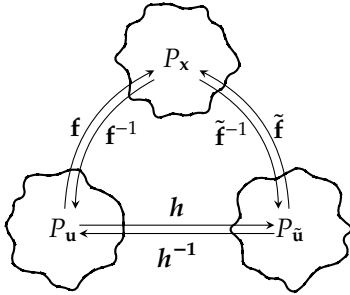


Figure 11.2: Theorem 11.1 shown as a commutative diagram, $\mathcal{M} = (\mathbf{f}, P_{\mathbf{u}})$ and $\tilde{\mathcal{M}} = (\tilde{\mathbf{f}}, P_{\tilde{\mathbf{u}}})$ are two TMI SCMs.

Theorem 11.1 tell us that, if our approximation to the underlying SCM, $\tilde{\mathcal{M}} = (\tilde{\mathbf{f}}, P_{\tilde{\mathbf{u}}}) \in \text{TMI} \times \mathcal{P}_{\mathbf{u}}$, matches the observational distribution generated by $\mathcal{M} = (\mathbf{f}, P_{\mathbf{u}}) \in \mathcal{F} \times \mathcal{P}_{\mathbf{u}}$, then we know that the exogenous distribution recovered by $\tilde{\mathcal{M}}$ differ from the real one by a function of each component independently, *i.e.*, $\tilde{\mathbf{f}}^{-1}(\mathbf{f}(\mathbf{u})) = \mathbf{h}(\mathbf{u}) = (h_1(u_1), h_2(u_2), \dots, h_D(u_D))$, where $\mathbf{u} \sim P_{\mathbf{u}}$ and each h_i is an invertible function. One direct consequence of Theorem 11.1 is that the functional dependencies of $\tilde{\mathcal{M}}$ must match those of \mathcal{M} , *i.e.*, $\tilde{\mathcal{M}}$ must be causally consistent *w.r.t.* \mathcal{M} . We formally present this result in the following corollary:

Corollary 11.2 (Causal consistency) If two elements of the family $\text{TMI} \times \mathcal{P}_{\mathbf{u}}$ produce the same observational distribution, then they are causally consistent.

Proof. Let $\mathcal{M}, \tilde{\mathcal{M}} \in \text{TMI} \times \mathcal{P}_{\mathbf{u}}$. By Theorem 11.1, we have that $\tilde{\mathbf{f}}^{-1} \circ \mathbf{f} = \mathbf{h}$ and $\mathbf{h}_{\#}p_{\mathbf{u}} = p_{\tilde{\mathbf{u}}}$, where \mathbf{h} is a component-wise transformation. Then,

$$\nabla_{\mathbf{u}} \mathbf{f}(\mathbf{u}) = \nabla_{\mathbf{u}} (\tilde{\mathbf{f}} \circ \mathbf{h})(\mathbf{u}) = \nabla_{\tilde{\mathbf{u}}} \tilde{\mathbf{f}}(\tilde{\mathbf{u}}) \cdot \nabla_{\mathbf{u}} \mathbf{h}(\mathbf{u}) \equiv \nabla_{\tilde{\mathbf{u}}} \tilde{\mathbf{f}}(\tilde{\mathbf{u}}), \quad (11.20)$$

where the last expression is a result of \mathbf{h} being a component-wise transformation. Similarly, we have that

$$\nabla_{\mathbf{x}} \tilde{\mathbf{f}}^{-1}(\mathbf{x}) = \nabla_{\mathbf{x}} (\mathbf{h} \circ \mathbf{f}^{-1})(\mathbf{x}) = \nabla_{\mathbf{u}} \mathbf{h}(\mathbf{u}) \cdot \nabla_{\mathbf{x}} \mathbf{f}^{-1}(\mathbf{x}) \equiv \nabla_{\mathbf{x}} \mathbf{f}^{-1}(\mathbf{x}). \quad (11.21)$$

That is, the SCMs are causally consistent. \square

To summarize, we have shown that if we find a SCM in the set $\text{TMI} \times \mathcal{P}_{\mathbf{u}}$ that matches the observational distribution of \mathcal{M} , then the causal model: **i)** is causally consistent; and **ii)** can identify the exogenous distribution up to component-wise transformations. In other words, we have proven that *the family of TMI SCMs is causally identifiable*.

11.4 Extension to real-world settings

To bring theory closer to practice, we extend our theoretical results from the previous section to two scenarios. Specifically, we consider the case when we have mixed discrete-continuous observational data, and the case when we possess only *partial* knowledge about the underlying causal graph, which are both common properties of real-world problems.

11.4.1 Discrete data

To extend the results presented in [Section 11.3](#) to the case where an observed variable is discrete, we restate the more general data-generating process assumed by Xi and Bloem-Reddy [216], which we adopted in previous sections.

Let us assume that the causal mechanism has been already unrolled, *i.e.*, that we have a function \mathbf{f} mapping \mathbf{u} to \mathbf{x} . Let us assume also, without loss of generality, that only the d -th observed variable is discrete, and let us focus on the way this particular *R.V.* is generated. Now, Xi and Bloem-Reddy consider the additional existence of a (fixed) noise distribution P_ϵ and mechanism g , such that the x_d is generated as

$$\mathbf{u} \sim P_{\mathbf{u}}, \quad \epsilon \sim P_\epsilon, \quad \mathbf{x} = g(\mathbf{f}(\mathbf{u}), \epsilon), \quad (11.22)$$

with $\epsilon \perp\!\!\!\perp \mathbf{u}$, and where Xi and Bloem-Reddy focus on the noiseless case under the following assumption: if we have two noise variables such that $\epsilon \stackrel{d}{=} \tilde{\epsilon}$, then $g(\mathbf{f}(\mathbf{u}), \epsilon) \stackrel{d}{=} g(\mathbf{f}(\tilde{\mathbf{u}}), \tilde{\epsilon})$ if and only if $\mathbf{f}(\mathbf{u}) \stackrel{d}{=} \mathbf{f}(\tilde{\mathbf{u}})$.

To accommodate the *discrete* observed variable \tilde{x}_d to our setting, we make the assumption that it is the transformation of a continuous exogenous *R.V.*, u_d , with a function \tilde{f}_d that fulfils our assumptions¹¹ and has undergone a quantization process, *e.g.*, $x_d = f_d(\mathbf{u}_{\text{an}_d}, u_d) := \lfloor \tilde{f}_d(\mathbf{u}_{\text{an}_d}, u_d) \rfloor$. Therefore, it is clear that f_d is no longer bijective, as we are clamping real numbers into integers, and that the observational distribution of x_d is discrete.

We take advantage of the noise assumption, and dequantize the observed variable x_d by assuming an additive noise mechanism of the form $x_d := f_d(\mathbf{u}_{\text{an}_d}, u_d) + \epsilon$, with ϵ distributed within $[0, 1]$ according to any continuous distribution.¹² As a result: **i)** \tilde{x}_d is again a continuous *R.V.*, as the sum of independent discrete and continuous *r.v.s* is a continuous *R.V.*; and **ii)** the discrete distribution of the noiseless case is always recoverable via $P(\tilde{x}_d = c) = P(c \leq x_d \leq c + 1)$.


More importantly, all the theoretical insights from the work of Xi and Bloem-Reddy [216] are preserved by working with the noisy case rather than the noiseless one. Indeed, they assume in their analysis a single \mathbf{u} in the domain of the generator, for which we can merge the generator and noise mechanism $g \circ f_d : \mathbb{R}^D \rightarrow \mathbb{R}$,¹³ by mapping the non-injective part of \mathbf{u} to ϵ itself, *i.e.*, by using the function

$$(g \circ f_d)(\mathbf{u}) = f_d(\mathbf{u}) + F_\epsilon^{-1} \left(\tilde{f}_d(\mathbf{u}) - f_d(\mathbf{u}) \right), \quad (11.23)$$

where F_ϵ^{-1} is the quantile function of P_ϵ . This new function is a diffeomorphism almost everywhere, as it is a composition of almost-everywhere

11.4.1 Discrete data 119

11.4.2 Partial knowledge . . . 120

[216] Xi and Bloem-Reddy (2023), ‘Indeterminacy in Generative Models: Characterization and Strong Identifiability.’ 

Remark 11.5 $\mathbf{x} \stackrel{d}{=} \mathbf{y}$ means that \mathbf{x} and \mathbf{y} are equal *r.v.s in distribution*, meaning that, for every value \mathbf{a} , $P(\mathbf{x} \leq \mathbf{a}) = P(\mathbf{y} \leq \mathbf{a})$.

11: *I.e.*, that \tilde{f}_d is a diffeomorphism.

12: For the experiments of [Chapter 12](#), we consider $P_\epsilon = \mathcal{U}(0, 1)$.

13: Instead of $g \circ f_d : \mathbb{R}^D \times [0, 1] \rightarrow \mathbb{R}$.

Remark 11.6 The intuition is that [Equation 11.23](#) embeds in the noise ϵ the distribution of the (unrecoverable) error of f_d .

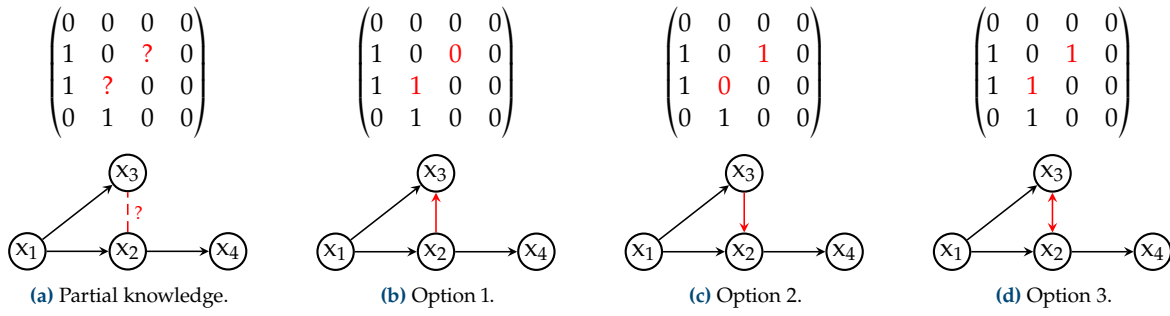


Figure 11.3: Example of a SCM with partial knowledge about the causal graph (a) and three possible realities: (b) only the edge $x_2 \rightarrow x_3$ exists; (c) only the edge $x_3 \rightarrow x_2$ exists; and (d) both edges exist. We omit the case where no edge exists.

diffeomorphisms, and we have effectively replaced the noise variable by the error of $f_d(\mathbf{u})$ after quantization. In particular, note that if the noise is uniformly distributed, $P_\epsilon = \mathcal{U}(0, 1)$, we have that $g \circ f_d = \tilde{f}_d$, if x_d were discretized by taking its integer part.

Intuitively, by adding noise to discrete variables while keeping them recoverable, we learn a mapping between continuous variables that represents ‘the generator function before the observed variables were somehow discretized.’ Importantly, the observed discrete distribution is always recoverable, independently of whether we learn the (unknown and unrecoverable) underlying continuous distribution.

11.4.2 Partial knowledge

We now expand our results to the case in which we have partial knowledge about the underlying causal graph, *i.e.*, when we are certain about some causal dependencies, but not all of them. To this end, we first introduce the way we deal with partial knowledge, and then clarify the theoretical implications that it has with respect to the theory of Section 11.3.

Illustrative example. Similar to Section 11.2, we motivate the method with the illustrative example shown in Figure 11.3. Suppose that we are given a SCM such as the one in Figure 11.3a, where we know all relationships but the one between x_2 and x_3 . Note that, in this case, we lack even information about the causal ordering. Then, there are four possibilities: **i)** there is no edge; **ii)** the edge $x_2 \rightarrow x_3$ exists; **iii)** the edge $x_3 \rightarrow x_2$ exists; or **iv)** both edges exist.¹⁴

Let us switch now to Figure 11.4. To solve the original problem in Figure 11.4a, one natural approach is to assume that all unknown edges may exist and group the nodes with unknown relationships. This, effectively, is equivalent to modelling the joint distribution of block variables with a general function, and assuming block TMI generators. However, if we want to keep the triangular structure for *all* variables (Figure 11.4c), we can model the joint distribution within the blocks with a TMI map using an arbitrary fixed ordering.¹⁵

One subtle detail is, however, that we need to increase the granularity of all inter-block edges from node- to block-wise relationships, defining an edge between blocks if there exists one for *at least* one pair of variables

14: That is, there are cycles in the graph or there exists a confounder.

15: Which we can always do by applying the KR map with that ordering.

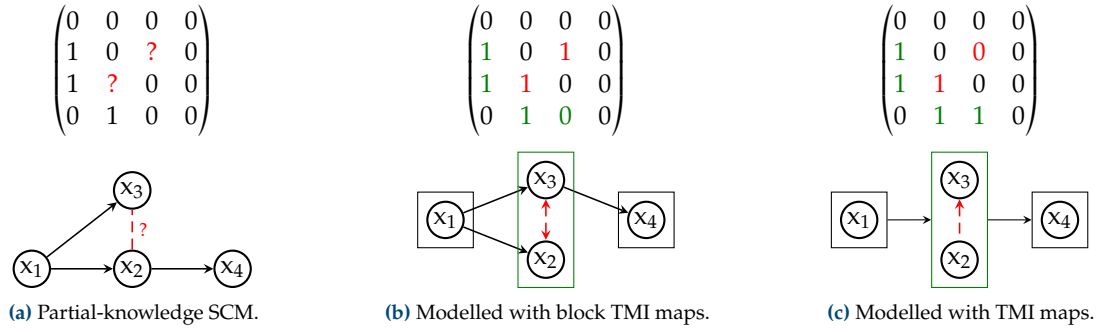


Figure 11.4: Illustrative example (same as in Figure 11.3) describing our method for partial information. First, we apply Tarjan’s algorithm [200] to find the SCCs of the graph (rectangles) and build a new DAG where each node is a subset of the original nodes. (b) If, for the SCCs, we consider block TMI maps, we keep the individual node edges unaltered. (c) If we instead use proper TMI maps, we pick an arbitrary order within each SCC, and extend the node edges to SCC edges. Red represents intra-SCC edges, and green inter-SCC edges. See Subsection 11.4.2 for more details.

between blocks. To see why, assume that the real graph of the example is that of Figure 11.3c, yet we assume the ordering $\pi = [1 \ 2 \ 3 \ 4]$ and the graph A without inter-block modifications.¹⁶ In that case, we would have that x_4 depends on x_3 through x_2 . However, a causally consistent model would not be able to model this dependency, and thus x_4 would depend on u_1 , u_2 , and u_4 but not on u_3 .

16: I.e., we assume the adjacency matrix

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}. \quad (11.24)$$

Constructing a DAG. Therefore, to extend the results from Section 11.3 using TMI maps, the block adjacency matrix needs to be the one from Figure 11.4c, which can be built as follows:

1. Run Tarjan’s algorithm [200] to group all nodes into their **strongly connected components (SCCs)**, forming a DAG.¹⁷
2. Transform the node edges into SCC edges. That is, introduce edges between every pair of nodes of two SCCs, if there exists at least one edge between their nodes.
3. Choose an ordering consistent with the known inter-SCC edges.

17: Note that, unlike in the illustrative example, there could be more than one cluster of unknown relationships.

Theoretical results. Extending the results of Section 11.3 to the partial-knowledge case is rather simple. The reason is that these results require a fixed ordering, not necessarily the causal one.¹⁸ Therefore, the proves of both Theorem 11.1 and Corollary 11.2 still apply to this fixed ordering, and we only need to re-state their implications with respect to the true causal data-generating process \mathcal{M} .

18: Since the KR transports described in Section 11.3 are unique given *any* variable ordering.

Then, we only need to note that all the possible graphs, after reducing them to a DAG using the SCCs-partition, are the same graph. In other words, every possible graph shares *the same causal dependencies between SCCs*. Hence, if we treat each SCC as a (multivariate) variable, calling $S_d \subset \{1, 2, \dots, D\}$ the SCC containing the d -th node, and defining pa_{S_d} and an_{S_d} as the union of parents and ancestors of every node in the SCC, then it is clear that we can write, for every plausible graph \tilde{A} , the observed variables as $x_d = \tilde{f}_d(x_{\text{pa}_{S_d}}, u_{S_d})$. Moreover, we can write the ‘exogenous’ variables as a function of the true ones, i.e., $\tilde{u}_{S_d} = \text{KR}(u_{S_d})$, where KR is the KR transport such that $\text{KR}_{\#}p_u = p_{\tilde{u}}$.

Theorem 11.3 (Identifiability – Partial knowledge) If we have two SCMs $\mathcal{M} \in \mathcal{F} \times \mathcal{P}_{\mathbf{u}}$, and $\tilde{\mathcal{M}} \in \tilde{\mathcal{F}} \times \mathcal{P}_{\mathbf{u}}$, where \mathcal{F} and $\tilde{\mathcal{F}}$ are TMI maps with equal inter-SCC orderings, and they generate the same observational distribution, then the two SCMs are identifiable up to an invertible SCC-wise transformation.

Proof. W.l.o.g. pick \mathcal{M} , and apply a KR transport to write it down as a member of $\tilde{\mathcal{F}} \times \mathcal{P}_{\mathbf{u}}$, call it $\hat{\mathcal{M}}$, with identical observational distribution as both \mathcal{M} and $\tilde{\mathcal{M}}$. Using [Theorem 11.1](#), we know that the $\tilde{\mathcal{M}}$ and $\hat{\mathcal{M}}$ differ by an invertible component-wise transformation \mathbf{h} . Moreover, we can write $\hat{\mathbf{u}}_{S_d}$ as a function of \mathbf{u}_{S_d} , as argued above. Putting it all together, we have that $\tilde{\mathbf{u}}_d = h_d(\hat{\mathbf{u}}_d) = h_d(\text{KR}_d(\mathbf{u}_{S_d}))$ and, for each SCC S_d ,

$$\tilde{\mathbf{u}}_{S_d} = \mathbf{h}_{S_d}(\text{KR}_{S_d}(\mathbf{u}_{S_d})) = (\mathbf{h}_{S_d} \circ \text{KR}_{S_d})(\mathbf{u}_{S_d}), \quad (11.25)$$

which only depends on the elements of \mathbf{u} within the SCC. \square

Corollary 11.4 (Causal consistency – Partial knowledge) If, as in the previous theorem, we have two SCMs with equal inter-SCC ordering producing the same observational distribution, then they are causally consistent at the SCC level.

Proof. The proof is identical to the one for [Corollary 11.4](#), but using the SCC-wise transformation \mathbf{h} given by [Theorem 11.3](#). \square


[Theorem 11.3](#) and [Corollary 11.4](#) extend the results from [Subsection 11.3.2](#). It is important to note, however, that causal consistency here refers to the causal relationships between SCCs. The reason is simple: as we fix an arbitrary ordering within each SCC, it might not correspond to a causal ordering. In other words, when we reason about SCCs instead of individual nodes,¹⁹ we can safely talk about SCC-wise causal identifiability and consistency, and we can perform interventions and compute counterfactuals on SCCs treated as a whole.

19: Note that if the whole causal graph is known, then every SCC contains just a single node, and we recover the results from [Subsection 11.3.2](#).

11.5 Reimplementing the do-operator

While we have reduced the space of SCMs to a subset that is causally identifiable, the implementation of the do-operator described in [Subsection 10.3.1](#) is not well-suited for other than the recursive SCM formulation from [Equation 10.1](#). In this section, we propose an implementation of the do-operator that works for every considered SCM, independently of the specific form of the structural equations, such that we can use them to answer causal-inference queries.

Semantics. Recall from [Subsection 10.3.1](#) that the do-operator [\[154\]](#), denoted as $do(x_i := \alpha)$, is a mathematical operator that simulates a physical intervention on a SCM \mathcal{M} , inducing an alternative model $\mathcal{M}^{\mathcal{I}}$ that fixes the observational value $x_i = \alpha$, and thus removes any causal dependency influencing x_i .

[\[154\]](#) Pearl (2012), ‘The Do-Calculus Revisited.’ 

Usual implementation. Usually, the do-operator is implemented by yielding a SCM $\mathcal{M}^{\mathcal{I}} = (\mathbf{f}^{\mathcal{I}}, P_{\mathbf{u}})$ result of replacing the i -th component of \mathbf{f} with a constant function, $f_i^{\mathcal{I}} := \alpha$. Unfortunately, this implementation only works for the recursive representation of the SCM, as otherwise the sampling process is not iterative along the exogenous variables, and the effect of replacing $f_i^{\mathcal{I}}$ is not reflected in the rest of variables.

Proposed implementation. Instead, we propose to manipulate the SCM by modifying the exogenous distribution $P_{\mathbf{u}}$, while keeping the structural equations \mathbf{f} unaltered. Specifically, an intervention $do(x_i := \alpha)$ constraints $P_{\mathbf{u}}$ to place density only on those values that, when transformed to endogenous variables, the intervened variable yields the expected value, *i.e.*, $x_i = \alpha$, while keeping the distribution of the rest of variables unaltered. In other words, we define the intervened SCM as $\mathcal{M}^{\mathcal{I}} = (\mathbf{f}, P_{\mathbf{u}}^{\mathcal{I}})$, where the density of the updated distribution $P_{\mathbf{u}}^{\mathcal{I}}$ is of the form

$$p^{\mathcal{I}}(\mathbf{u}) \propto p(\mathbf{u}) \cdot \delta_{\{f_i(\mathbf{u})=\alpha\}}(\mathbf{u}), \quad (11.26)$$

and where the distributions of the rest of variables remain the same.

Using the acyclic assumption, we know that the only way of altering the distribution of x_i without altering those of its parents is through u_i and, using the causal sufficiency assumption, we can squeeze the Dirac delta directly in the distribution of the i -th exogenous variable, such that:

$$p^{\mathcal{I}}(\mathbf{u}) = p_i^{\mathcal{I}}(u_i | \mathbf{u}_{j \neq i}) \cdot \prod_{j \neq i} p_j(u_j), \quad \text{with} \quad (11.27)$$

$$p_i^{\mathcal{I}}(u_i | \mathbf{u}_{j \neq i}) \propto p_i(u_i) \cdot \delta_{\{f_i(u_i, \mathbf{u}_{j \neq i})=\alpha\}}(\mathbf{u}). \quad (11.28)$$


By assumption, the structural equations are bijective, and thus the set $\{f_i(u_i, \mathbf{u}_{j \neq i}) = \alpha\}$ contains a single value given $\mathbf{u}_{j \neq i}$. Note that, to be well-defined, the density at this point should be positive, *i.e.*, the element that yields α (and therefore α) should be a plausible value. Interestingly, this implementation resembles the implementations for soft interventions [51] and backtracking counterfactuals [204]. Since this implementation does not make any assumption in the functional form of the structural equations, it can be used on any of the considered SCMs.

Next, we prove that the proposed implementation removes every dependency of the ancestors on the descendants that go through the intervened variable. For simplicity, we consider a 3-chain case with unidimensional variables, but the proof works for the general case replacing derivatives by differentials and taking \mathbf{u}_{an_2} instead of u_1 .

Proposition 11.5 Assume that $\mathcal{M} \in \mathcal{F} \times \mathcal{P}_{\mathbf{u}}$ is a 3-chain model where x_1 indirectly causes x_3 through x_2 . Then, after applying $do(x_2 := \alpha)$ as described above, there is no causal dependency of x_1 on x_3 .

Proof. To check that there is no causal dependency is equivalent to show no functional dependency in the structural equations. Since we are fixing the value of u_2 to produce $x_2 = \alpha$, we can use implicit differentiation to compute the influence of u_1 (and therefore x_1) on x_2 via u_2 :

$$\alpha = x_2(u_1, u_2) \xRightarrow{du_1} 0 = \frac{\partial x_2}{\partial u_1} + \frac{\partial x_2}{\partial u_2} \frac{du_2}{du_1}, \quad (11.29)$$

[51] Eberhardt and Scheines (2007), ‘Interventions and Causal Inference.’ 

[204] Von K  gelgen, Mohamed and Beckers (2023), ‘Backtracking counterfactuals.’

and any *indirect* influence of the ancestor, u_1 , on the descendant, x_3 , through this intermediate variable, x_2 , cancels out:

$$\frac{\partial x_3}{\partial x_2} \frac{dx_2}{du_1} = \frac{\partial x_3}{\partial x_2} \left(\frac{\partial x_2}{\partial u_1} + \frac{\partial x_2}{\partial u_2} \frac{du_2}{du_1} \right) = \frac{\partial x_3}{\partial x_2} \cdot 0 = 0. \quad (11.30)$$

□

Along with [Proposition 11.5](#), we also provide empirical evidence of the validity of the implementation in [Chapter 12](#) and [Appendix E.2](#). This is especially clear in the pair plots shown in [Figures E.2](#) and [E.3](#).

11.6 Concluding remarks

In this chapter, we have investigated the question of how to restrict the class of considered SCMs to gain causal identifiability. We have shown that, under a fairly general set of assumptions, this family of SCMs admits a reduction to a subset of causally equivalent models, the set of TMI SCMs, which we proved to be causally identifiable solely from observational data and a causal ordering, up to component-wise transformations of the original exogenous distribution.

As a consequence, we can provide causal identifiability guarantees in data-driven approaches, *e.g.*, [deep learning \(DL\)](#) models, by building a sufficiently rich parametric family that lives in the TMI SCMs family. That is, by reducing the hypothesis space of the models to those that follow a given causal ordering. Moreover, the extensions provided in this chapter for mixed-type data and impartial causal-graph knowledge further improves the applicability of these results to real-world scenarios. This is precisely what we do next in [Chapter 12](#) yet, as we will see soon, using the causal ordering as an inductive bias is not enough to successfully model the underlying SCMs in practice.

The results in this chapter are inherently limited by the assumptions made herein, and application domains that exhibit, *e.g.*, hidden confounders, or cycles in the induced causal graph, are not suitable to the framework presented in this chapter. Therefore, future work relaxing any of these assumptions are of special interest. Finally, this chapter implicitly relies on the validity of the given causal ordering, and it is not clear to which extent a mismatch between the real causal ordering and the one provided deteriorates the presented results.

Effective Deep Causal Inference with Causal Normalizing Flows

12.

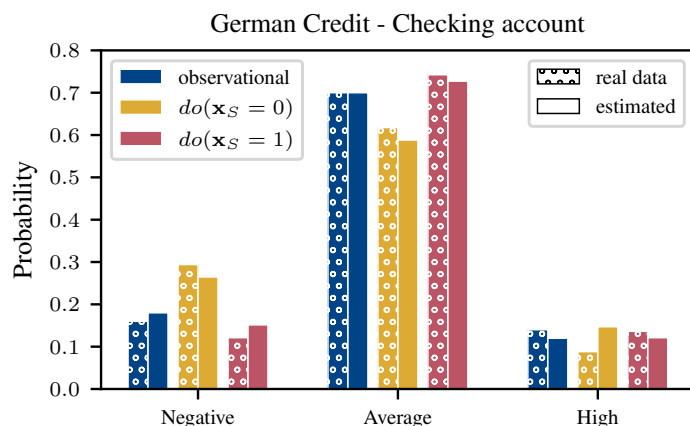
El que está en lucha que sueñe,
el que sufra que se indigne,
el que no hace nada no merece nada,
así de simple.

Charly Efe; La Tuerka

In the previous chapter, we have explored the theory of causal identifiability, and found that the family of **triangular monotonically increasing (TMI) structural causal models (SCMs)** provides us with causal-identifiability guarantees at no cost in expressivity, since we can always reduce a SCM to a causally-equivalent TMI SCM. In this chapter, we put the theory to practice, and use these results to design **causal generative models (CGMs)** that can learn the underlying causal data-generating process.

Specifically, we find **autoregressive normalizing flows (ANFs)** to be ideal candidates for this task, as they are **probabilistic generative models (PGMs)** and universal TMI approximators. We capitalize these findings to introduce in this chapter **causal normalizing flows (Causal NFs)**, a variation of ANFs that use the knowledge about the underlying causal ordering as an inductive bias to design the flow architecture. As a result, Causal NFs are the first **deep learning (DL)** model in the existing literature to provably approximate the underlying SCM from observational data under general assumptions. This is clearly illustrated in **Figure 12.1**, where the Causal NF is able to estimate the (unobserved) causal effect of externally intervening on the sensitive attribute (red and yellow distributions), using solely the observed data (blue distribution) and (partial) information about the causal graph.

However, we often find that the model is not able to effectively approximate the underlying SCM, as it gets stuck in local optima with spurious causal dependencies. To address this issue, we analyse different design and learning choices for Causal NFs, and using the causal graph as another inductive bias, show how to build Causal NFs that are causally consistent *by design*. As a result, the optimization process is significantly eased, and we can efficiently train Causal NFs.



12.1 Causal NFs	126
12.2 Causal inference queries	129
12.3 Ablation study	130
12.4 Model comparison	133
12.5 Fairness use-case	133
12.6 Concluding remarks	134



github.com/psanch21/causal-flows
github.com/adrianjav/causal-flows

This chapter is based on the content of:
[IV]: Javaloy, Sánchez-Martín and Valera
(2023), ‘Causal normalizing flows: from
theory to practice.’

Figure 12.1: Observational and interventional distributions of the categorical variable Checking account of the Credit dataset [48], and their estimated values according to a causal normalizing flow. x_S is a binary variable representing the applicant’s sex.

12.1 Causal normalizing flows

12.1.1 Intro to NFs 126

12.1.2 Connection with SCMs 127

12.1.3 Network design . . . 127

In this section, we introduce Causal NFs and explain their properties and connections with the theory built in [Chapter 11](#). Then, by exploiting explicit causal knowledge, we study different network designs that incorporate hard-constraint inductive biases to ease the optimization process and effectively learn SCMs in practice.

12.1.1 Introduction to normalizing flows

[147] Papamakarios, Nalisnick, Rezende, Mohamed and Lakshminarayanan (2021), ‘Normalizing Flows for Probabilistic Modeling and Inference.’

Normalizing flows (NFs) [147] are a family of PGMs that express the joint density of a set of observations by taking advantage of the change-of-variables rule. Given an observed random vector \mathbf{x} of size D , an NF is an invertible neural network with parameters θ that takes \mathbf{x} as input, and outputs $\mathbf{u} := T_\theta(\mathbf{x}) \in \mathbb{R}^D$, which we assume to be distributed as $\mathbf{u} \sim P_{\mathbf{u}}$. Therefore, the log-likelihood of \mathbf{x} according to T_θ is

$$\log p(\mathbf{x}) = \log p_{\mathbf{u}}(T_\theta(\mathbf{x})) + \log |\det \nabla_{\mathbf{x}} T_\theta(\mathbf{x})|, \quad (12.1)$$

where the second term is finite because T_θ is invertible, and where the distribution of \mathbf{u} , $P_{\mathbf{u}}$, is usually such that evaluating the log-likelihood and sampling are cheap to compute. Since we can evaluate the log-likelihood of the observed data, the usual practice is to perform **maximum likelihood estimation (MLE)** [12] and learn the network parameters, θ , by maximizing [Equation 12.1](#) with gradient-based approaches.

[12] Bishop (2006), ‘Pattern Recognition and Machine Learning (Information Science and Statistics).’

One remarkable property of NFs is their compositionality. Similar to the way that layers in a DL model can be stacked to construct highly complex architectures, *flow layers* can also be composed to build multi-layered flows that iteratively apply the change-of-variable rule. As a result, it is common to have NFs with several flow layers.

In this chapter, we focus on a specific family of NFs called **autoregressive normalizing flows (ANFs)** [94, 146]. Namely, ANFs factorize the joint distribution according to a given ordering π such that

[94] Kingma, Salimans, Jozefowicz, Chen, Sutskever and Welling (2016), ‘Improved Variational Inference with Inverse Autoregressive Flow.’

[146] Papamakarios, Murray and Pavlakou (2017), ‘Masked Autoregressive Flow for Density Estimation.’

$$P_\theta(\mathbf{x}) = \prod_{d=1}^D P_\theta(x_{\pi_d} \mid x_{\pi_1}, x_{\pi_2}, \dots, x_{\pi_{d-1}}), \quad (12.2)$$

which is achieved by designing a flow layer of the following form:

$$\mathbf{u}_d := \tau_d(\mathbf{x}_d; \mathbf{h}_d), \quad \text{with} \quad \mathbf{h}_d := c_d(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{d-1}), \quad (12.3)$$

where τ_d and c_d are known in the literature as the **transformer** and the **conditioner**, respectively. The transformer is a strictly monotonic function of \mathbf{x}_d , while the conditioner can be arbitrarily complex and only takes the variables preceding \mathbf{x}_d as input.

As a result, ANFs have triangular Jacobian matrices with positive diagonals, whose log-determinant can be computed in $\mathcal{O}(D)$ instead of the usual $\mathcal{O}(D^3)$. Moreover, when we stack them and shuffle the ordering π each layer, they become universal density approximators [147]. These two properties make them a really appealing option for designing NFs.

12.1.2 Connection with structural causal models

Another interesting way of looking at ANFs is as data-generating processes. Similar to SCMs, we can see an ANF as a tuple $(T_{\theta}^{-1}, P_{\mathbf{u}})$ that generates data by first sampling $\mathbf{u} \sim P_{\mathbf{u}}$ and then transforming them using the flow, $\mathbf{x} = T_{\theta}^{-1}(\mathbf{u})$. From this perspective, the connections of ANFs with the results and SCMs seen in Chapter 11 are easy to draw.

First, we note that an ANF layer and its inverse are parametric TMI maps, which is clear given that their Jacobian matrices are triangular with positive diagonal. Moreover, if we fix the same ordering π for all layers, then the combination of ANF layers is still a TMI map defined over the given ordering.

If, instead of any ordering, we choose π to be the known causal ordering, then it is clear that an ANF¹ is a TMI SCM as the ones we discussed in Chapter 11. We call such a flow a **causal normalizing flow**. Not only are Causal NFs a subset of this family of causal models, but it is direct to prove that they are universal TMI SCM approximators as well.² In other words, the family of Causal NFs is **expressive** and, as a consequence, **causally identifiable**, as defined in Subsection 11.1.2.

To summarize, if we manage to match the observational distribution of the underlying SCM \mathcal{M} generating the input dataset, which we directly incentivize by learning θ via MLE in Equation 12.1, then the flow will be causally equivalent to \mathcal{M} , *i.e.*, it will generate the same observational, interventional, and counterfactual distributions.

1: Seen as a data-generating process.

2: By fixing the ordering of the universal density approximator proof in [147].

12.1.3 Effective network design

We just showed that Causal NFs are a natural choice to learn the underlying SCM generating the data. In practice, however, we find that reaching the optimal parameters may be tricky as: **i)** we only have access to a finite amount of training data; and **ii)** the optimization process for Causal NFs (like for any neural network) can converge to local optima. In this section, we propose different design choices for Causal NFs to guide the optimization towards solutions that do not only provide an accurate fit of the observational distribution, but allow us to also accurately answer to interventional and counterfactual queries.

Let us start by recalling the illustrative example from Subsection 11.2.1, where we are given the linear SCM shown in Figure 12.2. On the one hand, as we discussed in Section 11.3, we can algebraically modify the structural equations to obtain causally equivalent SCMs which, crucially, preserve the causal dependencies between variables. On the other hand, a usually effective **inductive bias** when trying to approximate an object that obeys a certain structure is to embed the same structure in our approximation to the problem, significantly reducing the space of possible solutions. Therefore, we follow this intuition to design Causal NFs architectures that mimic the structure of specific SCM representations.

$$\mathbf{x} = \mathbf{G}\mathbf{x} + \mathbf{I}\mathbf{u}$$

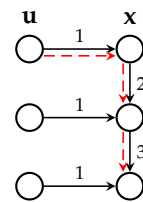


Figure 12.2: Illustrative example of a linear SCM in its recursive formulation.

Generative model. In the first architecture, we define the Causal NF as a function from \mathbf{u} to \mathbf{x} , hence the name, and it is designed to imitate the unrolled equations (see Figure 12.3). To this end, using the knowledge about the causal graph A , we replicate the structural sparsity per layer

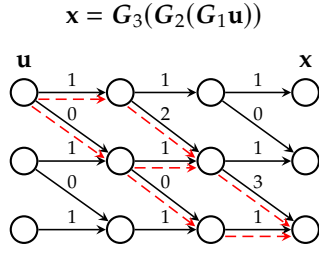


Figure 12.3: Illustrative example of a linear SCM in its unrolled formulation.

[174] Sánchez-Martín, Rateike and Valera (2022), ‘VACA: Designing Variational Graph Autoencoders for Causal Queries.’

3: That is, if $I + A$ were a dense lower-triangular matrix.

$$\mathbf{u} = (\mathbf{I} - \mathbf{G})\mathbf{x}$$

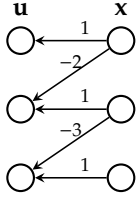


Figure 12.4: Illustrative example of a linear SCM in its inverted formulation.

by adequately masking each flow layer with $\mathbf{I} + \mathbf{A}$. In this way, the information from \mathbf{u} to \mathbf{x} is restricted to flow as if we were unrolling the causal model [174]. As a result, each layer of a Causal NF is of the form:

$$\mathbf{z}_d^{l-1} = \tau_d(\mathbf{z}_d^l; \mathbf{h}_d^{l-1}), \quad \text{where} \quad \mathbf{h}_d^{l-1} = c_d(\mathbf{z}_{\text{pa}_d}^l), \quad (12.4)$$

with $l = 1, 2, \dots, L$, and where \mathbf{z}^l is the output of the l -th layer, with $\mathbf{z}^0 := \mathbf{x}$ and $\mathbf{z}^L := \mathbf{u}$ (note the reversed order). By restricting each layer such that $\nabla_{\mathbf{z}^l} \tau(\mathbf{z}^l) \equiv \mathbf{I} + \mathbf{A}$, we discard all possible shortcuts. For example, the (indirect) information of u_1 to generate x_3 by first generating x_2 needs to go through the middle nodes in Figure 12.3. However, as shown by [174], we need at least $L = \text{diam}(\mathbf{A})$ layers to ensure that the information flows between the two furthest nodes.

In contrast, if we only know the causal ordering π ,³ then during training the Causal NF would need to rule out the spurious correlations by learning the necessary zeroes to fulfil causal consistency (see Corollary 11.2), *i.e.*, such that $\nabla_{\mathbf{x}} T_{\theta}(\mathbf{x}) \equiv \mathbf{I} + \mathbf{A}$ and $\nabla_{\mathbf{u}} T_{\theta}^{-1}(\mathbf{u}) \equiv \mathbf{I} + \sum_{i=1}^{\text{diam}(\mathbf{A})} \mathbf{A}^i$.

Abductive model. Reminiscent to the abduction step, another natural choice is to model the inverted equations of the SCM as in Figure 12.4, hence defining a Causal NF from \mathbf{x} to \mathbf{u} . Using again the information about the causal graph \mathbf{A} , we can use extra masking to force each layer to be structurally equivalent to $\mathbf{I} - \mathbf{A}$, *i.e.*,

$$\mathbf{z}_d^l = \tau_d(\mathbf{z}_d^{l-1}; \mathbf{h}_d^l), \quad \text{where} \quad \mathbf{h}_d^l = c_d(\mathbf{z}_{\text{pa}_d}^{l-1}). \quad (12.5)$$

Remarkably, this architecture is capable of capturing all indirect dependencies of \mathbf{u} on \mathbf{x} , even with a single layer. This is a result of the autoregressive nature of the ANFs used to build Causal NFs, as their inverse has the structure of the recursive formulation. In the example in Figure 12.4, the indirect influence of u_1 on x_3 via x_2 has to necessarily generate x_2 first, as seen in Figure 12.2. Similar to the previous case, when we know only the causal ordering, the Causal NF would need to rely on optimization to discard all spurious correlations.

Causal consistency by design. As stated in Corollary 11.2, the Causal NF needs to share the causal dependencies of the underlying SCM at the optima, *i.e.*, needs to be causally consistent. This is equivalent to saying that their Jacobian matrices need to be structurally equivalent, *i.e.*,

$$\nabla_{\mathbf{x}} T_{\theta}(\mathbf{x}) \equiv \mathbf{I} - \mathbf{A}, \quad \text{and} \quad \nabla_{\mathbf{u}} T_{\theta}^{-1}(\mathbf{u}) \equiv \mathbf{I} + \sum_{l=1}^{\text{diam}(\mathbf{A})} \mathbf{A}^l. \quad (12.6)$$

Therefore, we can study their causal consistency by directly computing their structural equivalence. On the one hand, given the (partial) causal adjacency matrix \mathbf{A} , the generative model in Equation 12.4 holds the latter equivalence by design for any sufficient number of layers since

$$\nabla_{\mathbf{z}^{l-1}} \tau(\mathbf{z}^{l-1}) \equiv \mathbf{I} + \mathbf{A} \quad \text{and} \quad \prod_{l=1}^L (\mathbf{I} + \mathbf{A}) \equiv \mathbf{I} + \sum_{l=1}^{\text{diam}(\mathbf{A})} \mathbf{A}^l \quad (12.7)$$

Table 12.1: Summary of the considered design choices, their induced properties, and their time complexity for density evaluation and sampling. See [Subsection 12.1.3](#) for an in-depth discussion.

Design Choices		Model Properties		Time Complexity	
Network Type	Causal Assumption	Causal Consistency		Sampling	Evaluation
		$\mathbf{u} \rightarrow \mathbf{x}$	$\mathbf{x} \rightarrow \mathbf{u}$		
$\mathbf{u} \mapsto \mathbf{x}$	Generative	Ordering π	\times	$\mathcal{O}(L)$	$\mathcal{O}(DL)$
	Generative	Graph A	\checkmark	$\mathcal{O}(L)$	$\mathcal{O}(DL)$
$\mathbf{x} \mapsto \mathbf{u}$	Abductive	Ordering π	\times	$\mathcal{O}(DL)$	$\mathcal{O}(L)$
	Abductive ($L > 1$)	Graph A	\times	$\mathcal{O}(DL)$	$\mathcal{O}(L)$
	Abductive ($L = 1$)	Graph A	\checkmark	$\mathcal{O}(DL)$	$\mathcal{O}(L)$

for $L \geq \text{diam } A$. Unfortunately, since in a SCM it is always the case that $I - A \leq I + \sum_{l=1}^{\text{diam}(A)} A^l$, we have no guarantees that the first equivalence holds in general: the model should remove these spurious dependencies during training. On the other hand, we run into a similar problem for the abductive model in [Equation 12.5](#) for the general case, *i.e.*,

$$\nabla_{\mathbf{z}^l} \tau(\mathbf{z}^l) \equiv I - A \quad \text{and} \quad \prod_{l=1}^L (I - A) \not\equiv I - A, \quad (12.8)$$

and spurious dependencies need to be removed during training. However, we note that the abductive model is *causally consistent by design* for the case $L = 1$. In those cases where the Causal NF does not ensure causal consistency by design, but we still have access to the adjacency matrix, we can use this extra information to regularize the MLE problem as

$$\underset{\theta}{\text{minimize}} \quad \mathbb{E}_{\mathbf{x}} \left[-\log p_{\theta}(\mathbf{x}) + \|\nabla_{\mathbf{x}} T_{\theta}(\mathbf{x}) \odot (\mathbf{1} - A)\|_2 \right], \quad (12.9)$$

where $\mathbf{1}$ is a matrix of ones, and thus penalizes spurious correlations from \mathbf{x} to $T_{\theta}(\mathbf{x})$. [Table 12.1](#) summarises the properties of the design choices considered in this chapter. Remarkably, the abductive model with a single layer ($L = 1$) is expressive, as ANF layers are universal TMI approximators, and causally consistent by design *w.r.t.* the provided adjacency matrix A , greatly simplifying the optimization process.

Remark 12.1 Intuitively, the mapping $\mathbf{x} \rightarrow \mathbf{u}$ is structurally sparser than the mapping $\mathbf{u} \rightarrow \mathbf{x}$ since in the former u_i depends on pa_i , and the latter x_i depends on an_i .

12.2 Causal inference queries

In this section, we describe how to efficiently answer causal inference queries with Causal NFs. To this end, we revisit the alternative implementation of the do-operator proposed in [Section 11.5](#), as it works for any SCM representation, and therefore for any of the architectures considered in [Subsection 12.1.3](#). See [Subsection 10.3.1](#) and [Section 11.5](#) for an introduction to the do-operator and its re-implementation, respectively.

As a summary, we can implement the do-operation $\text{do}(x_i := \alpha)$ on a Causal NF T_{θ} by modifying the distribution $P_{\mathbf{u}}$ while keeping the network unaltered. Namely, we restrict $P_{\mathbf{u}}$ to the (unique) value of u_i , given its parents, that yields the intervened value α when transformed with T_{θ}^{-1} , *i.e.*, the new density is of the form

$$p^{\mathcal{I}}(\mathbf{u}) = \delta_{\{T_{\theta}^{-1}(u_i, \mathbf{u}_{j \neq i}) = \alpha\}}(\mathbf{u}) \cdot \prod_{j \neq i} p_j(u_j). \quad (12.10)$$

Remark 12.2 Both algorithms use the bijectivity of the NF to sample from the delta. Importantly, note that only u_i is changed, as otherwise spurious correlations would be introduced to the descendants of x_i .

Algorithm 12.1: Algorithm to sample with a Causal NF from the interventional distribution, $P_\theta(\mathbf{x} \mid \text{do}(x_i := \alpha))$.

```

1 function: SampleIntervenedDist
2 input: index  $i$ , value  $\alpha$ 
3 begin
4    $\mathbf{u} \sim P_{\mathbf{u}}$ 
5    $\mathbf{x} \leftarrow T_\theta^{-1}(\mathbf{u})$                                 # Sample from the flow
6    $x_i \leftarrow \alpha$                                 # Set  $x_i$  to the intervened value  $\alpha$ 
7    $\mathbf{u}_i \leftarrow T_\theta(\mathbf{x})_i$                           # Change the  $i$ -th value of  $\mathbf{u}$ 
8    $\mathbf{x} \leftarrow T_\theta^{-1}(\mathbf{u})$ 
9   return  $\mathbf{x}$                                 # Return the intervened sample
10 end

```

Algorithm 12.2: Algorithm to sample with a Causal NF from the counterfactual distribution $P_\theta(\mathbf{x}^{\text{cf}} \mid \text{do}(x_i := \alpha), \mathbf{x}^f)$.

```

1 function: SampleCounterfactual
2 input: factual  $\mathbf{x}^f$ , index  $i$ , value  $\alpha$ 
3 begin
4    $\mathbf{u} \leftarrow T_\theta(\mathbf{x}^f)$                                 # Perform the abduction step
5    $x_i^f \leftarrow \alpha$                                 # Set  $x_i$  to the intervened value  $\alpha$ 
6    $\mathbf{u}_i \leftarrow T_\theta(\mathbf{x}^f)_i$                           # Change the  $i$ -th value of  $\mathbf{u}$ 
7    $\mathbf{x}^{\text{cf}} \leftarrow T_\theta^{-1}(\mathbf{u})$ 
8   return  $\mathbf{x}^{\text{cf}}$                                 # Return the counterfactual sample
9 end

```

12.3 Ablation study

12.3.1 Network design . . .	130
12.3.2 Time complexity . . .	131
12.3.3 Exogenous distribution	132
12.3.4 Flow architecture . .	132

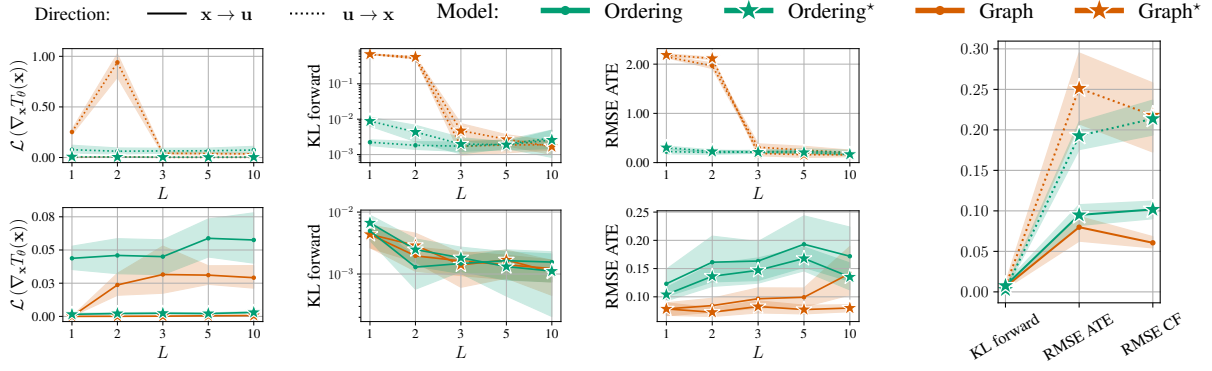
In this section, we extensively validate the design of Causal NFs from [Subsection 12.1.3](#) and the different insights discussed in [Subsection 12.1.3](#). Experiment details can be found in [Appendix E.2](#).

12.3.1 Network design

Experimental setup. We evaluate every network combination described in [Table 12.1](#) on a 4-chain SCM⁴ and assess the extent to which these models: **i)** capture the observational distribution, measured by $\text{KL}(p_{\mathcal{M}}(\mathbf{x}) \parallel p_\theta(\mathbf{x}))$; **ii)** remain causally consistent *w.r.t.* the original SCM, measured with $\mathcal{L}(\nabla_{\mathbf{x}} T_\theta(\mathbf{x})) := \|\nabla_{\mathbf{x}} T_\theta(\mathbf{x}) \cdot (\mathbf{1} - \mathbf{A})\|_2$ from [Equation 12.9](#); and **iii)** perform at causal-inference tasks, such as estimating the **average treatment effect (ATE)** [153] or predicting counterfactuals, both of which we measure with the **root mean squared error (RMSE)** *w.r.t.* the original causal model. Every experiment is repeated 5 times and every Causal NF uses **masked autoregressive flows (MAFs)** [146] as layers.

Results. [Figure 12.5](#) shows the results for different Causal NF designs. Specifically, we ablate: **i)** network design, generative $\mathbf{u} \rightarrow \mathbf{x}$ *v.s.* abductive

4: Which has a diameter of 3 and a very sparse Jacobian.



(a) Results as we vary the number of layers L . (Top) Generative architectures ($u \mapsto x$). (Bottom) Abductive architectures ($x \mapsto u$).

(b) Comparison of the two best models of each row in (a).

Figure 12.5: Ablation of different Causal NFs designs on their causally consistency, and causal inference capabilities. The use of regularization on the Jacobian (Equation 12.9) is indicated with the \star superscript. The abductive Causal NF with information on A and $L = 1$ outperforms the rest of models across all metrics, demonstrating its efficacy despite its simplicity.

$x \rightarrow u$; ii) causal knowledge, ordering *v.s.* graph; iii) number of layers L ; and iv) whether to add regularization, Equation 12.1 *v.s.* 12.9.

First, we see in Figure 12.5a (top) that, as expected, the generative models ($u \mapsto x$) using the causal graph cannot capture the SCM with $L < \text{diam } A = 3$. Furthermore, we observe that abductive models $x \mapsto u$ in Figure 12.5a (bottom) accurately fit the observational distribution, and that embedding the causal graph in the architecture significantly improves ATE estimation. Second, we compare the two network designs in Figure 12.5b, and observe that in general abductive models result in more accurate estimates of the observational distribution, as well as of interventional and counterfactual queries. Finally, we observe that regularization works well in all cases, yet it renders useless for the abductive model with $L = 1$ and knowledge on the graph, since it is causally consistent by design. In summary, this experiment confirms that, despite its simplicity, the causal abductive model with $L = 1$ outperforms the rest of design choices. As a consequence, in the following sections we will stick to this particular design choice.

12.3.2 Time complexity

Next, we evaluate the time complexity of the design choices introduced in Subsection 12.1.3. Figure 12.6 summarizes the results, where the x-axis represents the number of nodes in the dataset, D , and the y-axis indicates the time⁵ required for a single forward pass of the NF during training.

5: In microseconds, μs .

Results. First, abductive models ($x \rightarrow u$, solid lines) train significantly faster than generative models ($u \rightarrow x$, dotted lines), which emphasizes the computational cost associated with inverting ANFs during training. Moreover, the time complexity of abductive models remains constant regardless of input size D , whereas that of generative models increases linearly with D , as indicated in the right-most column of Table 12.1. Note that we should see the exact opposite behaviour when sampling. Second, the inclusion of Jacobian regularization (\star marker) introduces significant overhead, as clearly depicted in Figure 12.6. Finally, leveraging causal graph information or relying solely on ordering (represented in orange and green, respectively) has minimal impact on computational time.

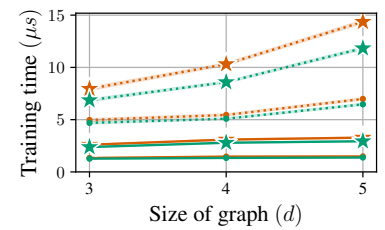


Figure 12.6: Time complexity comparison between the different design choices discussed in Subsection 12.1.3. Same legend as that of Figure 12.5.

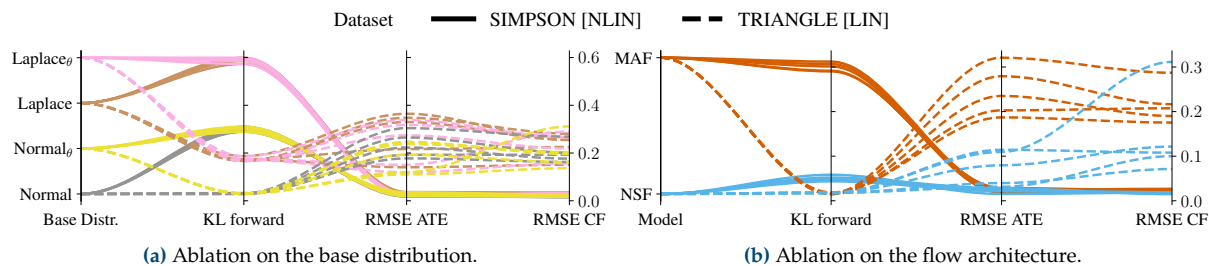


Figure 12.7: Performance on the $\text{Simpson}_{\text{NLIN}}$ and $\text{Triangle}_{\text{LIN}}$ datasets of Causal NFs with (a) different base distributions, Normal and Laplace, where θ indicates that we learn their parameters; and (b) flow architectures, MAF and NSF. Differences in base distribution only affects KL divergence, while the choice of flow architecture influences the overall performance.

12.3.3 Exogenous distribution

We now assess to which extent a mismatch of the distribution $P_{\mathbf{u}}$ between the underlying SCM and the Causal NF affects performance. To this end, we consider two more complex SCMs, Simpson [63] and Triangle [174], and two exogenous distributions for the Causal NF, Normal and Laplace distributions, for which we either fix or learn their parameters during training. Both synthetic SCMs use standard normal distributions.

Results. Figure 12.7a reveals a notable distinction between base distributions in terms of the observational-distribution fitting. However, this discrepancy appears to have minimal implications on the two causal-inference tasks. We hypothesize that this disparity originates from dissimilarities in the tails of the distributions, as it can be inferred by the slight edge of Normal over Laplace on the last column of Figure 12.7a, which measures *per sample* differences, where larger errors occur at the outliers which are, by definition, scarce. Interestingly, with this particular architecture of Causal NF, every model had a difficult time modelling the linear Triangle SCM.

12.3.4 Flow architecture

Considering the observed challenges faced by the MAF [146] flow to accurately model the Triangle SCM in the previous experiment, we further investigate the potential impact that flow architecture has on performance. We consider a Causal NF with one MAF layer, and another flow with a NSF [50] layer. Note that NSFs were built on top of MAFs and differ in that they are no longer affine ANFs, which made MAFs *not* universal TMI approximators. Therefore, we expect NSFs to outperform MAFs in general.

Results. Figure 12.7b summarises our results, with MAFs and NSFs depicted in orange and blue, respectively. Our empirical analysis reveals that indeed NSFs consistently outperform MAFs on all three metrics, *i.e.*, in observational, interventional, and counterfactual queries. Whilst expected, these findings highlight the practical implications of selecting an appropriate flow architecture, which should be taken into consideration when designing Causal NFs.

[50] Durkan, Bekasov, Murray and Papamakarios (2019), ‘Neural Spline Flows.’
[↗](#)

Table 12.4: Comparison, on three non-linear SCMs, of the proposed CausalNF, VACA [174], and CAREFL [91] with the do-operator proposed in Section 11.5. The results are averaged over 5 different runs.

Dataset	Model	Performance			Time Evaluation (μ s)		
		KL	ATE _{RMSE}	CF _{RMSE}	Training	Evaluation	Sampling
Triangle NLIN [174]	CausalNF	0.00 _{0.00}	0.12 _{0.03}	0.13 _{0.02}	0.52 _{0.07}	0.58 _{0.07}	1.07 _{0.12}
	CAREFL [†]	0.00 _{0.00}	0.12 _{0.03}	0.17 _{0.03}	0.57 _{0.18}	0.83 _{0.26}	1.68 _{0.62}
	VACA	7.71 _{0.60}	4.78 _{0.01}	4.19 _{0.04}	28.82 _{1.21}	23.00 _{0.55}	70.65 _{3.70}
LargeBD NLIN [63]	CausalNF	1.51 _{0.04}	0.02 _{0.00}	0.01 _{0.00}	0.52 _{0.10}	0.60 _{0.17}	3.05 _{0.66}
	CAREFL [†]	1.51 _{0.05}	0.05 _{0.01}	0.08 _{0.01}	0.84 _{0.47}	1.18 _{0.17}	8.25 _{1.29}
	VACA	53.66 _{2.07}	0.39 _{0.00}	0.82 _{0.02}	164.92 _{11.10}	137.88 _{15.72}	167.94 _{25.75}
Simpson SYMPROD [63]	CausalNF	0.00 _{0.00}	0.07 _{0.01}	0.12 _{0.02}	0.59 _{0.17}	0.60 _{0.11}	1.51 _{0.30}
	CAREFL [†]	0.00 _{0.00}	0.10 _{0.02}	0.17 _{0.04}	0.49 _{0.15}	0.81 _{0.19}	1.91 _{0.33}
	VACA	13.85 _{0.64}	0.89 _{0.00}	1.50 _{0.04}	49.26 _{4.09}	37.78 _{3.41}	79.20 _{14.60}

12.4 Model comparison

In this section, we compare the proposed model, CausalNF,⁶ with other two relevant works: **i)** CAREFL [91], an abductive NF with knowledge on the causal ordering and exclusively using affine flow layers; and **ii)** VACA [174], a variational auto-encoding graph neural network (GNN) with knowledge on the causal graph. To provide fair comparison, every model uses the same budget for hyperparameter tuning, we restrict CausalNF to affine flow layers, and CAREFL has been modified to use the proposed do-operator from Section 11.5.⁷ We increase the complexity of the synthetic SCMs and consider: **i)** Triangle, a 3-node SCM with a dense causal graph; **ii)** LargeBD [63], a 9-node SCM with non-Gaussian P_u and made out of two chains with common initial and final nodes; and **iii)** Simpson [63], a 4-node SCM simulating a Simpson’s paradox [188], where the relation between two variables changes if the SCM is not properly approximated.

6: Using A , abductive, and with $L = 1$, *i.e.*, bottom row of Table 12.1.

7: As the original implementation only works in root nodes. See Appendix E.1.

Results. In a nutshell, Table 12.4 shows that the proposed CausalNF outperforms both CAREFL and VACA in terms of performance in all metrics as well as in computational efficiency. In general, VACA shows poor performance, and it is considerably slower due to the inherent complexity of GNNs. Our CausalNF outperforms CAREFL in interventional and counterfactual estimation tasks with identical observational fitting, stressing once more the importance of being causally consistent. Moreover, our CausalNF is significantly faster than CAREFL, as usually the best-performing CAREFL architectures have more than one layer.

In Appendix E.2.3 we extend these experiments to a total of 12 SCMs with a variety of settings, showing that these results hold in general.

12.5 Fairness auditing and classification

To show the practical impact of Causal NFs, we reproduce the fairness use-case of Sánchez-Martín et al. [174] on the Credit dataset [48], a dataset from the UCI repository where the likelihood of individuals repaying a loan is predicted based on a small set of features, including *sensitive* attributes such as their sex. Extra details can be found in Appendix E.3.


[174] Sánchez-Martín, Rateike and Valera (2022), ‘VACA: Designing Variational Graph Autoencoders for Causal Queries.’ 

Table 12.5: Accuracy, F1-score, and counterfactual unfairness of the audited classifiers. Causal NFs enable both fair classifiers and accurate unfairness metrics. Results are averaged over 5 different runs.

	Logistic classifier				SVM classifier			
	full	unaware	fair \mathbf{x}	fair \mathbf{u}	full	unaware	fair \mathbf{x}	fair \mathbf{u}
F1-score	72.28 _{6.16}	72.37 _{4.90}	59.66 _{8.57}	73.08 _{4.38}	76.04 _{2.86}	76.80 _{5.82}	68.28 _{5.74}	77.39 _{1.52}
Accuracy	67.00 _{3.83}	66.75 _{2.63}	54.75 _{5.91}	66.50 _{3.70}	69.50 _{3.11}	71.00 _{3.83}	59.25 _{2.99}	69.75 _{1.26}
Unfairness	5.84 _{2.93}	2.81 _{0.72}	0.00 _{0.00}	0.00 _{0.00}	6.65 _{2.45}	2.78 _{0.40}	0.00 _{0.00}	0.00 _{0.00}

Experimental setup. As proposed by Chiappa [29], we use a partial causal graph that groups the 7 discrete features of the dataset in 4 different blocks with known causal relationships, putting in practice the techniques described in Section 11.4. For the Causal NF, we employ the abductive model with a single non-affine NSF layer [50]. Our ultimate goal is to train a Causal NF that captures well the underlying SCM producing the data, and use it to train and audit classifiers that predict the (additional) binary feature `Credit risk`, while remaining counterfactually fair *w.r.t.* the binary variable `Sex`, x_S .

In this context, we call a binary classifier $\kappa : \mathbb{X} \rightarrow \{0, 1\}$ counterfactually fair [100] if, for every possible factual value $\mathbf{x}^f \in \mathbb{X}$, its counterfactual unfairness remains zero. That is, if we have that

$$\mathbb{E}_{\mathbf{x}^f}[P(\kappa(\mathbf{x}^{cf}) = 1 \mid do(x_S := 1), \mathbf{x}^f) - P(\kappa(\mathbf{x}^{cf}) = 1 \mid do(x_S := 0), \mathbf{x}^f)] = 0,$$


where $\mathbf{x}^{cf} \sim P(\mathbf{x}^{cf} \mid do(x_S := s), \mathbf{x}^f)$, are counterfactual variables for the intervention values $s \in \{0, 1\}$.

We audit the following types of classifier: **i)** `full`, a model that takes all observed variables; **ii)** `unaware`, a model that leaves the sensitive attribute x_S out; **iii)** `fair \mathbf{x}` , a fair model that only considers non-descendant variables of x_S ; and, to demonstrate the ability to learn counterfactually-fair classifiers with Causal NFs, we include `fair \mathbf{u}` , a classifier that takes $\mathbf{u} = T_\theta(\mathbf{x})$ as input, but leaves the sensitive exogenous variable u_S out.

Results. Table 12.5 summarizes the performance and unfairness of the different type of classifiers, using logistic regression [38] and SVMs [35] as classification methods. Here, we observe that by taking the non-sensitive exogenous variables from the Causal NF, the resulting classifiers achieve comparable or better accuracy than the rest of approaches, while at the same time being counterfactually fair. Moreover, the estimations of counterfactual unfairness obtained with the Causal NF match our expectations [100], with `full` being the most unfair classifier, followed by `unaware`, and the two other classifiers being counterfactually fair. With this use-case, we demonstrate that Causal NFs may indeed be a valuable asset for real-world causal inference problems.

12.6 Concluding remarks

In this chapter, we have introduced Causal NFs, ANFs that follow a given causal ordering, and showed that they are universal approximators of TMI SCMs, making them a natural choice to provably learn a causal system by leveraging the theoretical results from Chapter 11. Unfortunately, as local

[100] Kusner, Loftus, Russell and Silva (2017), ‘Counterfactual Fairness.’ 

optima may hamper reaching these solutions in practice, we have explored different network designs to introduce further structural inductive biases by exploiting the available knowledge on the causal graph, hence making Causal NFs causally consistent by design. Furthermore, we have revisited the do-operator proposed in [Chapter 11](#) and designed algorithms to efficiently solve causal inference tasks with Causal NFs. Finally, we have empirically validated our findings, and demonstrated that Causal NFs consistently outperforms existing approaches and can deal with mixed-typed data and partial knowledge on the causal graph, which is key for real-world use-cases.

We firmly believe that this chapter opens a number of interesting directions to explore. For example, it remains an open question how to extend Causal NFs to deal with hidden confounders, or whether other loss functions than MLE require looser inductive biases. Moreover, it would be exciting to see Causal NFs applied to other problems such as causal discovery [\[63\]](#), fair decision-making [\[100\]](#), or neuroimaging [\[68\]](#), among others. However, we would like to stress that, in the above contexts, it would be essential to validate the suitability of our framework (*e.g.*, using experimental data) to prevent potential harms.

Part IV.

Epilogue



Si te encuentro gritaré a viva voz,
que prefiero verte que ganar la guerra.
Levántate, mi corazón,
te escondiste a la sombra de la sierra.

La raíz; A la sombra de la sierra

13.1 Summary and impact . 139

13.2 Prospects 141

We conclude this dissertation by summarizing the main results covered herein, the early impact that the research presented has had on subsequent works, as well as by giving our honest suggestions on future research directions that we hope this dissertation has helped moving towards.

13.1 Summary and impact

A common theme in this dissertation has been the specification of our expectations on the model behaviour, and to consider them as actual constraints in the optimization process, for which we use inductive biases as a way of guiding the model parameters towards solutions satisfying these constraints. Of meeting our expectations, we will more accurately predict the functionality of the **deep learning (DL)** models we train and, therefore, feel more in control over these models.

We first studied the case of **multitask learning (MTL)** where we showed that, besides meeting our expectation of learning to solve many tasks simultaneously, it is necessary to *a priori* set our expectations on the trade-off (or Pareto) solutions that we aim to reach. Moreover, we made explicit that, in order to aggregate information from different tasks in an informed manner, the tasks should first be comparable. We demonstrated that we can meet this comparability expectation if we had access to the true **cumulative distribution function (CDF)** for each task, provided metrics to approximate them out of samples, as well as different metrics to measure the effect of incomparability (in the form of gradient conflict) during optimization. For those cases where we seek to achieve task impartiality as our trade-off solution,¹ we introduced RotoGrad [II], which has had significant impact since its publication and, *e.g.*, has been successfully applied in the context of cancer-grading to help doctors diagnose cancer under missing-modalities situations [205].

We then showed that similar expectations are also posed to multimodal **probabilistic generative models (PGMs)** and connected both fields, where modalities play now the role of tasks. Because of the probabilistic constraints of PGMs, we first proposed Lipschitz-standardization, an optimization-inspired preprocessing algorithm that attempts to equalize the optimization landscape of each modality. Then, by reducing the study of gradient conflict to the computational graphs of the models, we showed that complex multimodal PGMs can (and do) exhibit multiple sub-computational graphs prone to comparability problems. This way, we systemized the study of gradient conflict and leveraged existing MTL algorithms in the context of multimodal PGMs. Since the publication of

1: That is, where all tasks are equally important to learn.

[205] Wang, Zhang, Zhu, Jiang, Qin and Yuan (2024), ‘MGIML: Cancer Grading With Incomplete Radiology-Pathology Data via Memory Learning and Gradient Homogenization.’

our work [III], we have brought awareness to the problem of modality collapse and to the interaction between gradients of different modalities during training [25, 52, 53, 116, 180, 209, 215, 233].

Finally, we have made significant contributions to the field of **causal generative models (CGMs)**. Theoretically, we demonstrated the existence of a family of **structural causal models (SCMs)**, which are not only component-wise identifiable, but that are causally equivalent to a much broader family of SCMs which may have generated our data. We provided tools to perform causal inference in this new model family, and extended our results to more realistic scenarios with discrete data and missing causal information. More crucially, we identified a family of DL models which are natural members of this new family of SCMs, and provided effective network architectures to meet our causal expectations with the model by design. As a result, the proposed **causal normalizing flows (Causal NFs)** became the first-of-its-kind DL model to provide theoretical guarantees on its approximation capabilities of causal models, and we empirically demonstrated that it can satisfactorily be used for causal inference tasks. Despite the recency of our work [IV], Causal NFs have already had significant impact and its effectiveness extensively tested:

- Empirically, concurrent works have explored the idea of adding causal structure to **autoregressive normalizing flows (ANFs)**, and corroborated our findings. Namely:

1. Parafita and Vitrià [148] embedded the causal structure in their model leveraging graphical normalizing flows [210], and successfully used it to solve causal-inference tasks in a different set of benchmarks that our work did.
2. Balgi et al. [6] used graphical normalizing flows in the same way, and instead applied the resulting model for policy learning, afterwards answering counterfactual queries.
3. Fan et al. [56] proposed an architecture similar to ours, and used the resulting flow to model the latent space of a **variational autoencoder (VAE)** to successfully achieve disentangled representations.
4. Chen et al. [24] also proposed causally-masked ANFs, but instead focused on finding effective ways of masking the network to embed the causal structure, and used it to solve causal-inference tasks in a different set of problems.

[148] Parafita and Vitrià (2022), ‘Estimand-Agnostic Causal Query Estimation With Deep Causal Graphs.’

[6] Balgi, Peña and Daoud (2022), ‘Personalized Public Policy Analysis in Social Sciences Using Causal-Graphical Normalizing Flows.’

[56] Fan, Hou and Gao (2023), ‘Cf-vae: Causal disentangled representation learning with vae and causal flows.’

[24] Chen, Shi, Gao, Baptista and Krishnan (2023), ‘Structured Neural Networks for Density Estimation and Causal Inference.’

[41] Dance and Bloem-Reddy (2024), ‘Causal Inference with Cocycles.’

[217] Xi, Gonzalez and Bloem-Reddy (2023), ‘Triangular Monotonic Generative Models Can Perform Causal Discovery.’

[123] Majumdar and Valera (2024), ‘CARMA: A practical framework to generate recommendations for causal algorithmic recourse at scale.’

- Recently, Dance and Bloem-Reddy [41] reproduced some of the results reported in our work, and studied the robustness of Causal NFs under different misspecifications.
- Xi et al. [217] extended our theory and showed that Causal NFs ‘can also be used to perform conditional independence based causal discovery by finding the maximally sparse permutation.’
- Finally, Majumdar and Valera [123] leveraged Causal NFs to learn the underlying causal system, and provide recommendations for algorithmic recourse by working directly in the exogenous space.

We hope that the insights presented in this thesis, and the works that composed it, help further advance the DL community towards developing more trustworthy models that behave according to our expectations.

13.2 Prospects

In this section, we discuss open questions and research directions that we would love to see develop in the future. We hope that the insights shared in this dissertation help develop future research in these venues.

13.2.1 Expectations	141
13.2.2 Grounded MTL	141
13.2.3 Comparability	142
13.2.4 Structured DL	143

13.2.1 Expectations as optimization constraints

We have explored during all the dissertation the idea, in different settings, that the expectations we place on the models during deployment should be included during training as optimization constraints. We firmly believe that this idea should indeed become a de-facto standard, rather than an exception, where we clearly state exactly what are we expecting from the DL models we train when they are deployed. Once that the optimization constraints (in this case, our expectations) are clearly stated, we can study to which extent we can introduce them in the training process.

This is a pressing problem to address if we want DL to be more robust and trustworthy. Fortunately, the research community is slowly becoming more aware of this issue. Recently, a number of researchers discussed the tightly related with the problem of *underspecification* [40], where ‘predictors returned by underspecified pipelines are often treated as equivalent based on their training domain performance, but such predictors can behave very differently in deployment domains.’ Similar to the way that design-by-contract programming [132] argues that software designers should define a formal, precise, and verifiable interface specifications for software, DL engineers should also establish a clear behavioural contract (in the form of the optimization problem to solve) when training DL models, where all expectations are explicitly stated.

[40] D’Amour, Heller, Moldovan, Adlam, Alipanahi, Beutel, Chen, Deaton, Eisenstein, Hoffman et al. (2022), ‘Underspecification presents challenges for credibility in modern machine learning.’

[132] Mitchell, McKim and Meyer (2001), ‘Design by contract, by example.’

13.2.2 Towards grounded MTL research

Another topic that we flew over during **Part I** of this thesis is the additional care that should be put when performing research in MTL methods. Next, we discuss two particular topics within this context.

The potential dangers of MOO

In **Section 4.4** we discussed how MTL can be interpreted as a **multi-objective optimization (MOO)** problem, and that this interpretation was increasingly popular in recent years. While useful and appealing (for once, it presents the optimization problem disentangled from any loss function that combines the different task losses), we reiteratedly stressed the necessity of *a priori* establishing a scalarization function to solve within this framework, rather than looking for *any* Pareto-optimal solution.

We motivate this argument in two ways. First, reaching any Pareto front without the user having control of the specific trade-off solution that we reach is of little to no use, similar to the inconsistent trade-off problem that we discussed in **Section 2.1**. Second, comparing different MTL models under the premise that they reach Pareto-optimal solutions quickly becomes unfeasible: in order to verify that the model reach a Pareto-optimal point, we need to run enough methods and random initializations

to explore the entire Pareto front, which grows exponentially with the number of tasks. As a result, systematically testing and corroborating findings in MTL research under the MOO interpretation becomes close to impossible in any practical terms.

A MOO perspective can be extremely useful, but just as we discussed in the previous section, our expectations on the Pareto solution to reach should be clearly specified from the beginning.

The need for MTL-tailored benchmarks

Another important topic is the need for well-suited benchmarks in MTL research, understanding benchmark as the combination of datasets, network architecture, and the set of specific hyperparameters. Similar to how we should clearly state our expectations on the models that we train, we should also clearly specify the problem that we attempt to address with our research, and ensure that the benchmarks we employ help us show that this problem has indeed been palliated. This is clearly illustrated in the case of gradient-conflict methods we discussed in [Chapter 6](#): if our interest is on appropriately combining the information from different tasks, measuring task performance and performing a hyperparameter search in the network architecture may not be the best choice, as bigger models do exhibit less gradient conflict in general.

13.2.3 Comparability: the elephant in the room

The main expectation we discussed in [Parts I and II](#) was the comparability of quantities coming from different, respectively, tasks and modalities. Similar to how we connected these two fields in [Part II](#), we expect similar patterns to emerge everywhere in DL where data from heterogeneous sources are combined. Ultimately, comparability and the proper aggregation of information in DL deserve to be studied as its own abstract framework, so that it can effectively impact different areas of DL.

One key aspect towards this unification is to identify which areas of DL may suffer from comparability problems, for which we suggest to look for similar solutions as those we discussed in this dissertation. For example, in the context of gradient-conflict methods, which emerged in MTL as a response to the incomparability of task gradients, we can find similar approaches in areas such as domain generalization [[159](#), [187](#)], continual learning [[57](#), [72](#)], federated learning [[42](#), [206](#)], auxiliary learning [[47](#), [109](#)], meta learning [[34](#), [60](#)], fairness [[122](#)], imbalanced learning [[78](#)], neural architecture search [[135](#)], diffusion sampling [[46](#)], dynamic-depth neural-network training [[196](#)], among others.

Of unifying all these different areas under a single framework, solutions and ideas could easily transfer from one area to another. For example, of standardizing the connections between negative transfer and modality collapse, we could design more manageable datasets by using a probabilistic perspective, where we could generate our own synthetic data to meet the requirements we need to test our methods. Moreover, as a result of better understanding the root cause of our problems, we can borrow ideas from other fields in science, *e.g.*, for the proof of ranking statistics as finite-sample approximations in [Subsection 6.2.1](#), we were initially

inspired by the fact that, in the social welfare literature, comparisons need to be done by means of relative rankings due to the interpersonal incomparability of utility [136, 156].

13.2.4 Structured Deep Learning

We showed in [Part III](#) the importance of adding structure to the architecture of our DL models in order to provably learn, in this case, the underlying causal model generating our observations. It is well-known that adding explicit structure can have many benefits on the models that we train—such as ensuring certain invariances in the model [119], or providing tractability guarantees when answering certain queries [30]—but we believe that there is potential to push this idea even further, making the structure of the neural network a first-class citizen in DL research, and helping us overcome many of the existing challenges in DL such as data-hunger, or the lack of generalization. In this regard, we believe that Causal NFs and identifiability theory open an interesting path towards achieving this goal.

Causal generative models

On one side, Causal NFs offer for the first time a principled connection between DL models and causal inference, enabling neural networks to be used for solving complex causal reasoning tasks. However, while general, we made a number of assumptions to get to the results presented in [Part III](#) that do not suit every use case, and loosening these could be quite important for a number of scenarios. Obvious extensions of Causal NFs would be to study cases where we have hidden confounders, loosen the monotonic assumption for the family of causal models, or to allow for probabilistic and soft interventions. Other more challenging research venues would be those that enable the application of Causal NFs in high-dimensional domains such as images and videos, for which we could study the application of other generative models (such as, *e.g.*, diffusion models [75]), or extending Causal NFs to work on the latent space, adopting a Causal Component Analysis perspective [105].

Beyond causality

Moreover, it is quite conceivable to detach Causal NFs from its causal nature, as the core essence of the derived results come from the structure of the network. Following this idea, Causal NFs could be easily extended to other settings for which we also have relationships across variables similar to that of cause-and-effect. One clear example is propositional logic, where we have logic rules which we can embed in the network architecture if we want the model to reason in a certain way, *e.g.*, if we had $P \rightarrow Q$, we can interpret *modus ponens* (*i.e.*, if P implies Q , and P is true, then Q is true), similar to a generative process where P causes Q .

[136] Muandet (2022), ‘Impossibility of Collective Intelligence.’ [↗](#)

[156] Piggins (2019), ‘Collective Choice and Social Welfare—Expanded Edition.’

[119] Lyle, Wilk, Kwiatkowska, Gal and Bloem-Reddy (2020), ‘On the benefits of invariance in neural networks.’ [↗](#)

[30] Choi, Vergari and Van den Broeck (2020), ‘Probabilistic circuits: A unifying framework for tractable probabilistic models.’

[105] Liang, Kekic, Kügelgen, Buchholz, Besserve, Gresele and Schölkopf (2023), ‘Causal Component Analysis.’ [↗](#)

Part V.

Appendix



Additional Material for Chapter 5

A.

A.1 Proofs

A.1 Proofs	147
A.2 Stackelberg games	147
A.3 Experiments	149

Proposition A.1 Suppose $g_k := L_k \circ h_k$ is an infinitely differentiable real-valued function, and let us call $\mathbf{G}_k = \nabla_{\mathbf{Z}} g_k(\mathbf{Z})$ its derivative with respect to \mathbf{Z} , for every $k \in \{1, 2, \dots, K\}$. If $\cos(\mathbf{G}_i, \mathbf{G}_j) > -1/(K-1)$ pairwise, then there exists a small-enough step size $\epsilon > 0$ such that, for all k , we have that $L_k(h_{\phi_k}(\mathbf{Z} - \epsilon \cdot C \sum_k \mathbf{U}_k), \mathbf{Y}_k) < L_k(h_{\phi_k}(\mathbf{Z}), \mathbf{Y}_k)$, where $\mathbf{U}_k := \mathbf{G}_k / \|\mathbf{G}_k\|$ and $C \geq 0$.

Proof. Since g_k is infinitely differentiable, we can take the first-order Taylor expansion of g_k around \mathbf{Z} , for any k , evaluated at $\mathbf{Z} - \epsilon \mathbf{V}$ for a given vector \mathbf{V} :

$$g_k(\mathbf{Z} - \epsilon \mathbf{V}) = g_k(\mathbf{Z}) - \epsilon \langle \mathbf{G}_k, \mathbf{V} \rangle + o(\epsilon). \quad (\text{A.1})$$

In our case, $\mathbf{V} = C \sum_k \mathbf{U}_k$ with $C \geq 0$, then:

$$\begin{aligned} g_k(\mathbf{Z} - \epsilon \mathbf{V}) - g_k(\mathbf{Z}) &= -\epsilon \cdot C \|\mathbf{G}_k\| \sum_i \langle \mathbf{U}_k, \mathbf{U}_i \rangle + o(\epsilon) \\ &= -\epsilon \cdot C \|\mathbf{G}_k\| \left(1 + \sum_{i \neq k} \langle \mathbf{U}_k, \mathbf{U}_i \rangle \right) + o(\epsilon). \end{aligned} \quad (\text{A.2})$$

Since $\|\mathbf{U}_k\| = 1$ for all $k \in \{1, 2, \dots, K\}$, it holds that

$$-1 \leq \cos(\mathbf{U}_k, \mathbf{U}_i) = \langle \mathbf{U}_k, \mathbf{U}_i \rangle \leq 1. \quad (\text{A.3})$$


If $\cos(\mathbf{G}_k, \mathbf{G}_i) > -1/(K-1)$ for all $i \neq k$, then we have that $1 + \sum_{i \neq k} \langle \mathbf{U}_k, \mathbf{U}_i \rangle > 0$ and $g_k(\mathbf{Z} - \epsilon \mathbf{V}) < g_k(\mathbf{Z})$ for a small enough $\epsilon > 0$.

□

A.2 Stackelberg games and RotoGrad

In game theory, a Stackelberg game [58] is an asymmetric game where two players play alternately. One of the players—whose objective is to blindly minimize their loss function—is known as the follower, \mathcal{F} . The other player is known as the leader, \mathcal{L} . In contrast to the follower, the leader attempts to minimize their own loss function, but with the advantage of knowing which will be the best response to their move by the follower. The problem can be written as

$$\begin{aligned} \text{Leader: } \min_{x_l \in X_l} \{ \mathcal{L}(x_l, x_f) \mid x_f \in \arg \min_{y \in X_f} \mathcal{F}(x_l, y) \}, \\ \text{Follower: } \min_{x_f \in X_f} \mathcal{F}(x_l, x_f), \end{aligned} \quad (\text{A.4})$$

[58] Fiez, Chasnov and Ratliff (2020), ‘Implicit Learning Dynamics in Stackelberg Games: Equilibria Characterization, Convergence Analysis, and Empirical Study.’


where $x_l \in X_l$ and $x_f \in X_f$ are the actions taken by the leader and follower, respectively.

While traditionally one assumes that players make perfect alternate moves in each step of [Equation A.4](#), *gradient-play* Stackelberg games assume instead that players perform simultaneous gradient updates,

$$\begin{aligned} x_l^{t+1} &= x_l^t - \alpha_l^t \nabla_{x_l} \mathcal{L}(x_l, x_f), \\ x_f^{t+1} &= x_f^t - \alpha_f^t \nabla_{x_f} \mathcal{L}(x_l, x_f), \end{aligned} \quad (\text{A.5})$$

where α_l and α_f are the learning rates of the leader and follower, respectively.

An important concept in game theory is that of an equilibrium point, *i.e.*, a point in which both players are satisfied with their situation, meaning that there is no available move immediately improving any of the players' scores, so that none of the players is willing to perform additional actions/updates. Specifically, in this thesis we focus on the following definition introduced by Fiez et al. [58]:

Definition A.1 (differential Stackelberg equilibrium) A pair of points $x_l^* \in X_l$, and $x_f^* \in X_f$, where $x_f^* = r(x_l^*)$ is implicitly defined by $\nabla_{x_f} \mathcal{F}(x_l^*, x_f^*) = 0$, is a differential Stackelberg equilibrium point if $\nabla_{x_l} \mathcal{L}(x_l^*, r(x_l^*)) = 0$, and $\nabla_{x_l}^2 \mathcal{L}(x_l^*, r(x_l^*))$ is positive definite.

Note that, when the players manage to reach such an equilibrium point, both of them are in a local optimum. Here, we make use of the following result, introduced by Fiez et al. [58], to provide theoretical convergence guarantees to an equilibrium point:

Proposition A.2 In the given setting, if the leader's learning rate goes to zero at a faster rate than the follower's, that is, $\alpha_l(t) = o(\alpha_f(t))$, where $\alpha_i(t)$ denotes the learning rate of player i at step t , then they will asymptotically converge to a differential Stackelberg equilibrium point almost surely.


In other words, as long as the follower learns faster than the leader, they will end up in a situation where both are satisfied. Fiez et al. [58] extended this result to the finite-time case, showing that the game will end close to an equilibrium point with high probability.

As we can observe, the Stackelberg formulation in [Equation A.4](#) is really similar to RotoGrad's formulation in [Equation 5.5](#). Even more, the update rule in [Equation A.5](#) is quite similar to that one shown in [Algorithm 5.3](#). Therefore, it is sensible to cast RotoGrad as a Stackelberg game. One important but subtle bit about this link regards the extra information used by the leader. In our case, this extra knowledge explicitly appears in [Equation 5.4](#) in the form of the follower's gradient, g_{nk} , which is the direction the follower will follow and, as it is performing first-order optimization by assumption, this gradient directly encodes the follower's response.

Thanks to the Stackelberg formulation in [Equation 5.5](#) we can make use of [Proposition A.2](#) and, thus, draw theoretical guarantees on the training stability and convergence. In other words, we can say that performing

training steps as those described in [Algorithm 5.3](#) will stably converge as long as the leader is asymptotically the slow learner, *i.e.* $\alpha_l^t = o(\alpha_f^t)$.

In practice, however, the optimization procedure proposed by Fiez et al. [58] requires computing the gradient of a gradient, thus incurring a significant overhead. Instead, we use Gradient Ascent-Descent (GDA), which only computes partial derivatives and enjoys similar guarantees [86], as we empirically showed in the manuscript.

[86] Jin, Netrapalli and Jordan (2020), ‘What is Local Optimality in Nonconvex-Nonconcave Minimax Optimization?’ 

A.3 Experiments

A.3.1 Experimental setups . . . 149

A.3.2 Additional results . . . 154

A.3.1 Experimental setups

Here, we discuss common settings across all experiments. Refer to specific sections further below for details concerning single experiments.

Computational resources. All experiments were performed on a shared cluster system with two NVIDIA DGX-A100. Therefore, all experiments were run with (up to) 4 cores of AMD EPYC 7742 CPUs and, for those trained on GPU (CIFAR10, CelebA, and NYUv2), a single NVIDIA A100 GPU. All experiments were restricted to 12 GB of RAM.

Loss normalization. Similar as in the gradient case studied in this work, magnitude differences between losses can make the model overlook some tasks. To overcome this issue, here we perform loss normalization, that is, we normalize all losses by their value at the first training iteration (so that they are all 1 in the 1st iteration). To account for some losses that may quickly decrease at the start, after the 20th iteration, we instead normalize losses dividing by their value at that iteration.

Checkpoints. For the single training of a model, we select the parameters of the model by taking those that obtained the best validation error after each training epoch. That is, after each epoch we evaluate the linearly-scalarized validation loss, $\sum_k L_k$, and use the parameters that obtained the best error during training. This can be seen as an extension of early-stopping where, instead of stopping, we keep training until reaching the maximum number of epochs hoping to keep improving.

Baselines. We have implemented all baselines according to the original paper descriptions, except for PCGrad, which we apply to the gradients with respect to the feature \mathbf{z} (instead of the shared parameters θ , as in the original paper). Note that this is in accordance with recent works [27, 112], which also use this implementation of PCGrad in the feature space. This way, all competing methods modify gradients with respect to the same variables and, as backpropagation performs the sum of gradients at the last shared representation \mathbf{z} , PCGrad can be applied to reduce conflict at that level.

Hyperparameter tuning. Model-specific hyperparameters were mostly selected by a combination of: **i)** using values described in prior works; and **ii)** empirical validation on the vanilla case (without any gradient-modifiers) to verify that the combinations of hyperparameters work. To select method-specific hyperparameters we performed a grid search, choosing those combinations of values that performed the best with respect to validation error.

Specifically, we took $\alpha \in \{0, 0.5, 1, 2\}$ and $\mathbf{R}_k \in \mathbb{R}^{m \times m}$ with $m \in \{0.25d, 0.5d, 0.75d, d\}$ (restricting ourselves to $m \leq 1500$) for RotoGrad. Regarding the learning rate of RotoGrad and GradNorm, we performed a grid search considering $\eta_{\text{rot}} \in \{0.05\eta, 0.1\eta, 0.5\eta, \eta, 2\eta\}$, where η_{rot} and η are the learning rates of RotoGrad or GradNorm, and the network, respectively.

1: That is, better than the vanilla multitask learning (MTL) optimization without the \mathbf{R}_k matrices.

Statistical test. For the tabular data, we highlight those results that are significantly better than those from the multitask baseline.¹ To find these values, we run a paired one-sided Student’s t-test across each method and the baseline. For those metrics for which higher is better, our null hypothesis is that the method’s performance is equal or lower than the baseline, and for those for which lower is better, the null hypothesis is that the method’s performance is equal or greater than the baseline. We use a significance level of $\alpha = 0.05$.

Notation. In this section, we use the following to describe different architectures: [Conv- F - C] denotes a convolutional layer with filter size F and C number of channels; [Max] denotes a max-pool layer of filter size and stride 2, and [Dense- H] a dense layer with output size H .

Illustrative examples

Losses and metrics. Both illustrative experiments use **mean squared error (MSE)** as both loss and metric. Regarding the specific form of φ in Equation 5.6, the avocado-shaped experiments uses

$$\varphi((x, y), s) = (x - s)^2 + 25y^2, \quad (\text{A.6})$$

while the non-convex second experiment uses

$$\varphi((x, y), s) = -\frac{\sin(3x + 4.5s)}{x + 1.5s} - \frac{\sin(3y + 4.5s)}{y + 1.5s} + |x + 1.5s| + |y + 1.5s| \quad (\text{A.7})$$

Model. As described in the main manuscript, we use a single input $x \in \mathbb{R}^2$ picked at random from a standard normal distribution, and drop all task-specific network parameters (that is, $h_k(\mathbf{r}_k) = \mathbf{r}_k$). As backbone, we take a simple network of the form $\mathbf{z} = \mathbf{W}_2 \max(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1, 0) + \mathbf{b}_2$ with $\mathbf{b}_1 \in \mathbb{R}^{10}$, $\mathbf{b}_2 \in \mathbb{R}^2$, and $\mathbf{W}_1, \mathbf{W}_2^\top \in \mathbb{R}^{10 \times 2}$.

Hyperparameters (convex case). We train the model for 100 epochs. As network optimizer we use stochastic gradient descent (SGD) with a learning rate of 0.01. For the rotations, we use RAdam [113] with a

learning rate of 0.5 (for visualization purposes we need a high learning rate, in such a simple scenario it still converges) and exponential decay with decaying factor 0.99999.

Hyperparameter (non-convex case). For the second experiment, we train the model for 400 epochs and, once again, use SGD as the network optimizer with a learning rate of 0.015. For the rotations, we use RAdam [113] with a learning rate of 0.1 and an exponential decay of 0.99999.

MNIST and SVHN

Datasets. We use two modified versions of MNIST [102] and SVHN [145] in which each image has two digits, one on each side of the image. In the case of MNIST, both of them are merged such that they form an overlapped image of 28×28 , as shown in Figure A.1a. Since SVHN contains backgrounds, we simply paste two images together without overlapping, obtaining images of size 32×64 , as shown in Figure A.1b. Moreover, we transform all SVHN samples to grayscale.

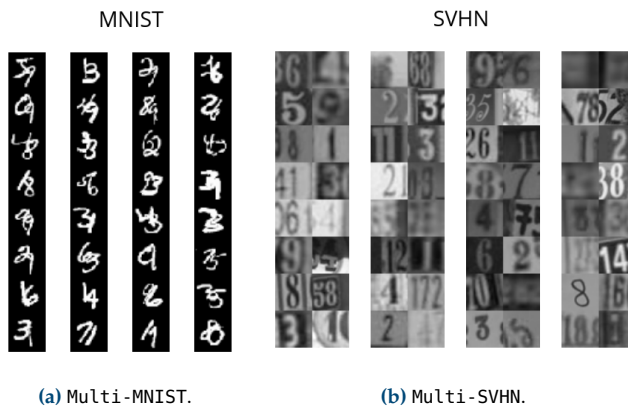



Figure A.1: Samples extracted from the modified MNIST and SVHN datasets.

Tasks, losses, and metrics. In order to further clarify the setup used, here we describe in detail each task. Specifically, we have:

- ▶ **Left digit** classification. Loss: **negative log-likelihood (NLL)**. Metric: accuracy (ACC).
- ▶ **Right digit** classification. Loss: NLL. Metric: ACC.
- ▶ **Parity** of the product of digits, *i.e.*, whether the product of both digits gives an odd number (binary-classification). Loss: **binary cross-entropy (BCE)**. Metric: F1-score (F1).
- ▶ **Sum** of both digits (regression). Loss: MSE. Metric: MSE.
- ▶ **Active pixels** in the image, *i.e.*, predict the number of pixels with values higher than 0.5, where we use pixels lying in the unit interval (regression). Loss: MSE. Metric: MSE.

Model. Our backbone is an adaption from the original LeNet [101] model. Specifically, we use:

- **MNIST:** [Conv-5-10] [Max] [ReLU] [Conv-5-20] [Max] [Dense-50] [ReLU] [BN],
- **SVHN:** [Conv-5-10] [Max] [ReLU] [Conv-5-20] [Max] [Conv-5-20] [Dense-50] [ReLU] [BN],

[80] Ioffe and Szegedy (2015), ‘Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.’ 

where [BN] refers to Batch Normalization [80]. Depending on the type of task, we use a different head. Specifically, we use:

- **Regression:** [Dense-50] [ReLU] [Dense-1],
- **Classification:** [Dense-50] [ReLU] [Dense-10] [Log-Softmax],
- **Binary-classification:** [Dense-1] [Sigmoid].

Model hyperparameters. For both datasets, we train the model for 300 epochs using a batch size of 1024. For the network parameters, we use RAdam [113] with a learning rate of $1e-3$.

Methods hyperparameters. In Tables A.1 and A.2 we show the results of GradNorm with: **i)** MNIST, with $R_k, \alpha = 0$; **ii)** MNIST without $R_k, \alpha = 0.5$; **iii)** SVHN with $R_k, \alpha = 1$; and **iv)** SVHN without $R_k, \alpha = 2$. We train RotoGrad with full-size rotation matrices ($m = d$). Both methods use RAdam with learning rate $5e-4$ and exponential decay of 0.9999.

CIFAR10

Dataset. We use CIFAR10 [97] as dataset, with 40 000 instances as training data and the rest as testing data. Additionally, every time we get a sample from the dataset we: **i)** crop the image by a randomly selected square of size $3 \times 32 \times 32$; **ii)** randomly flip the image horizontally; and **iii)** standardize the image channel-wise using the mean and standard deviation estimators obtained on the training data.

Model. We take as backbone ResNet-18 [74] without pre-training, where we remove the last linear and pool layers. In addition, we add a Batch Normalization layer. For each task-specific head, we simply use a linear layer followed by a sigmoid function, that is, [Dense-1] [Sigmoid].

Losses and metrics. We treat each class (out of ten) as a binary-classification task where we use BCE and F1-score as loss and metric, respectively.

Model hyperparameters. We use a batch size of 128 and train the model for 500 epochs. For the network parameters, we use as optimizer SGD with learning rate of 0.01, Nesterov momentum of 0.9, and a weight decay of $5e-4$. Additionally, we use for the network parameters a cosine learning-rate scheduler with a period of 200 iterations.

Methods hyperparameters. Results shown in Tables 5.5 and A.3 use $\alpha = 0$ and $\alpha = 0.5$ for GradNorm with and without R_k , respectively, and we use RAdam [113] as optimizer with learning rate 0.001 and an exponential decay factor of 0.99995 for both GradNorm and RotoGrad.

NYUV2

Setup. In contrast with the rest of experiments, for the NYUV2 experiments shown in Section 5.4, instead of writing our own implementation, we slightly modified the open-source code provided by Liu et al. [115] at <https://github.com/lorenmt/mtan> (commit 268c5c1). We therefore use the exact same setting as them—and refer to their paper and code for further details, with the addition of using data augmentation for the experiments which, although not described in the paper, is included in the repository as a command-line argument. We will provide along this work a diff file to include all gradient-modifier methods into the aforementioned code.

Methods hyperparameters. For the results shown in Table 5.4 we use GradNorm with $\alpha = 0$ and RotoGrad with rotations R_k of size 1024. We use a similar optimization strategy as the rest of parameters, using Adam [92] with learning rate $5e-5$ (half the one of the network parameters) and where we halve this learning rate every 100 iterations.

CelebA

Dataset. We use CelebA [117] as dataset with usual splits. We resize each sample image so that they have size $3 \times 64 \times 64$.

Losses and metrics. We treat each class (out of 40) as a binary-classification task where we use BCE and F1-score as loss and metric, respectively.

ResNet-18 model. As with CIFAR10, we use as backbone ResNet-18 [74] without pre-training, where we remove the last linear and pool layers. In addition, we add a batch-normalization layer. For each task-specific head, we use a linear layer followed by a sigmoid function, that is, [Dense-1][Sigmoid].

ResNet-18 hyperparameters. We use a batch size of 256 and train the model for 100 epochs. For the network parameters, we use RAdam [113] as optimizer with learning rate 0.001 and exponential decay of 0.99995 applied every 2400 iterations.

Convolutional model. For the second architecture, we use a convolutional network as backbone, [Conv-3-64][BN][Max][Conv-3-128][BN][Conv-3-128][BN][Max][Conv-3-256][BN][Conv-3-256][BN][Max][Conv-3-512][BN][Dense-512][BN]. For the task-specific heads, we take a simple network of the form [Dense-128][BN][Dense-1][Sigmoid].

Convolutional hyperparameters. We use a batch size of 8 and train the model for 20 epochs. For the network parameters, we use RAdam [113] as optimizer with learning rate 0.001 and exponential decay of 0.96 applied every 2400 iterations.

Methods hyperparameters. Results shown in Tables 5.6 and A.4 use GradNorm with: **i)** convolutional network with R_k , $\alpha = 0$; **ii)** convolutional network without R_k , $\alpha = 1$; **iii)** residual network with R_k , $\alpha = 0.5$; and **iv)** residual network without R_k , $\alpha = 1$. For RotoGrad, we rotate 256 and 1536 elements of z for the convolutional and residual networks. As optimizer, we use RAdam [113] with learning rate $5e-6$ and an exponential decay factor of 0.99995 for both GradNorm and RotoGrad. Note that for these experiments we omit MGDA-UB [182] as it is computationally prohibitive in comparisons with other methods. In single-seed experiments, we however observed that it does not perform too well (specially in the convolutional network).

A.3.2 Additional results

Multi-MNIST and Multi-SVHN

Table A.1: Test performance (median and standard deviation) on two set of unrelated tasks on Multi-MNIST and Multi-SVHN, across 10 different runs.

Method	MNIST		SVHN	
	Digits $\text{avg}_k \Delta_k \uparrow$	Act Pix MSE \downarrow	Digits $\text{avg}_k \Delta_k \uparrow$	Act Pix MSE \downarrow
STL	0.00 ± 0.00	0.01 ± 0.01	0.00 ± 0.00	0.17 ± 0.06
Vanilla	-1.43 ± 3.24	0.14 ± 0.05	4.78 ± 0.88	3.04 ± 2.53
GradDrop	-1.30 ± 1.82	0.16 ± 0.04	5.34 ± 0.92	2.99 ± 2.59
PCGrad	-1.22 ± 2.81	0.13 ± 0.01	5.01 ± 0.65	2.70 ± 2.25
MGDA-UB	-29.14 ± 9.23	0.06 ± 0.00	-4.36 ± 6.72	1.00 ± 0.57
GradNorm	0.86 ± 1.93	0.09 ± 0.04	5.24 ± 0.89	4.12 ± 9.46
IMTL-G	2.12 ± 1.46	0.07 ± 0.02	5.94 ± 0.99	1.70 ± 1.05
RotoGrad	1.55 ± 2.22	0.08 ± 0.03	6.08 ± 0.48	1.61 ± 2.72

We reuse the experimental setup from Subsection 5.4.1—now using the original LeNet [101] and a multitask-version of SVHN [145]—in order to evaluate how disruptive the orthogonal image-related task is for different methods. We can observe (Table A.1) that the effect of the image-related task is more disruptive in MNIST, in which MGDA-UB utterly fails. Direction-aware methods (GradDrop and PCGrad) do not improve the vanilla results, whereas IMTL-G, GradNorm, and RotoGrad obtain the best results.

We also provide the complete results for all metrics in Table A.2. In the case of MNIST, we can observe that both regression tasks tend to be quite disruptive. GradNorm, IMTL-G, and RotoGrad manage to improve over all tasks while maintaining good performance on the rest of tasks. MGDA-UB, however, focuses on the image-related task too much and overlooks other tasks. In SVHN we observe a similar behaviour. This time, all methods are able to leverage positive transfer and improve their results on the parity and sum tasks, obtaining similar task improvement results. Yet, the image-related task is more disruptive than before, showing bigger differences between methods. Again, MGDA-UB completely focuses on this task, but now is able to not overlook any task while doing so. Regarding

Table A.2: Complete results (median and standard deviation) of different competing methods on MNIST/SVHN on all tasks, see [side note 1](#) and [Appendix A.3.2](#).

		Left digit Acc. \uparrow	Right digit Acc. \uparrow	Product parity F1 \uparrow	Sum digits MSE \downarrow	$\text{avg}_k \Delta_k \uparrow$	Act. Pix. MSE \downarrow
MNIST	STL	95.70 \pm 0.20	94.05 \pm 0.16	92.09 \pm 0.76	1.90 \pm 0.10	0.00 \pm 0.00	0.01 \pm 0.01
	Without R_k						
	Vanilla [†]	94.94 \pm 0.20	93.26 \pm 0.27	93.07 \pm 0.48	2.10 \pm 0.17	-3.26 \pm 3.12	0.11 \pm 0.01
	GradDrop	95.33 \pm 0.39	93.55 \pm 0.29	93.32 \pm 0.54	2.14 \pm 0.07	-2.52 \pm 1.63	0.13 \pm 0.02
	PCGrad	95.07 \pm 0.39	93.28 \pm 0.18	93.34 \pm 0.51	2.14 \pm 0.19	-3.36 \pm 3.86	0.12 \pm 0.02
	MGDA-UB	94.46 \pm 1.04	92.23 \pm 1.54	83.89 \pm 1.84	2.50 \pm 0.60	-10.80 \pm 1045	0.06 \pm 0.02
	GradNorm	95.19 \pm 0.37	93.70 \pm 0.31	93.31 \pm 0.39	2.06 \pm 2871	-1.81 \pm 3799	0.09 \pm 7.46
	IMTL-G	95.28 \pm 0.38	93.84 \pm 0.21	93.24 \pm 0.49	1.91 \pm 6.61	-0.01 \pm 8218	0.07 \pm 2.05
	With R_k						
	Vanilla	95.13 \pm 0.20	93.41 \pm 0.17	93.54 \pm 0.50	1.99 \pm 0.17	-1.43 \pm 3.24	0.14 \pm 0.05
	GradDrop	95.14 \pm 0.16	93.47 \pm 0.12	93.59 \pm 0.32	2.00 \pm 0.06	-1.30 \pm 1.82	0.16 \pm 0.04
	PCGrad	95.04 \pm 0.26	93.36 \pm 0.30	93.49 \pm 0.30	1.98 \pm 0.13	-1.22 \pm 2.81	0.13 \pm 0.01
	MGDA-UB	89.99 \pm 2.21	86.76 \pm 1.18	79.24 \pm 2.83	3.65 \pm 0.42	-29.14 \pm 9.23	0.06 \pm 0.00
	GradNorm	95.28 \pm 0.18	93.56 \pm 0.25	93.56 \pm 0.57	1.86 \pm 0.07	0.86 \pm 1.93	0.09 \pm 0.04
	IMTL-G	95.47 \pm 0.27	93.79 \pm 0.31	93.56 \pm 0.57	1.73 \pm 0.09	2.12 \pm 1.46	0.07 \pm 0.02
	RotoGrad	95.45 \pm 0.19	93.83 \pm 0.19	93.22 \pm 0.35	1.85 \pm 0.13	1.55 \pm 2.22	0.08 \pm 0.03
SVHN	STL	85.05 \pm 0.45	84.58 \pm 0.24	77.47 \pm 1.13	5.84 \pm 0.14	0.00 \pm 0.00	0.17 \pm 0.06
	Without R_k						
	Vanilla [†]	84.18 \pm 0.30	84.18 \pm 0.38	80.11 \pm 0.85	4.81 \pm 0.06	5.14 \pm 0.83	2.75 \pm 3.17
	GradDrop	84.38 \pm 0.29	84.48 \pm 0.41	80.11 \pm 0.69	4.69 \pm 0.12	5.68 \pm 1.05	1.91 \pm 0.86
	PCGrad	84.22 \pm 0.31	84.23 \pm 0.21	79.92 \pm 0.79	4.69 \pm 0.09	5.50 \pm 0.75	2.26 \pm 0.85
	MGDA-UB	84.61 \pm 0.75	84.38 \pm 0.45	77.44 \pm 1.44	4.47 \pm 0.18	5.99 \pm 1.48	0.66 \pm 0.75
	GradNorm	84.23 \pm 0.33	84.13 \pm 0.30	79.40 \pm 0.87	4.92 \pm 0.07	4.60 \pm 1.01	4.30 \pm 2.18
	IMTL-G	84.60 \pm 0.45	84.39 \pm 0.37	79.63 \pm 1.10	4.57 \pm 0.13	5.81 \pm 0.85	2.47 \pm 1.65
	With R_k						
	Vanilla	84.11 \pm 0.48	84.11 \pm 0.40	79.83 \pm 0.79	4.84 \pm 0.10	4.78 \pm 0.88	3.04 \pm 2.53
	GradDrop	84.23 \pm 0.35	84.33 \pm 0.40	80.10 \pm 0.83	4.73 \pm 0.09	5.34 \pm 0.92	2.99 \pm 2.59
	PCGrad	84.21 \pm 0.21	84.26 \pm 0.38	79.64 \pm 0.48	4.84 \pm 0.06	5.01 \pm 0.65	2.70 \pm 2.25
	MGDA-UB	77.05 \pm 5.44	78.00 \pm 5.04	71.76 \pm 4.32	5.27 \pm 0.56	-4.36 \pm 6.72	1.00 \pm 0.57
	GradNorm	84.37 \pm 0.34	84.30 \pm 0.46	79.97 \pm 0.75	4.72 \pm 0.13	5.24 \pm 0.89	4.12 \pm 9.46
	IMTL-G	84.23 \pm 0.34	84.23 \pm 0.39	79.77 \pm 1.04	4.51 \pm 0.12	5.94 \pm 0.99	1.70 \pm 1.05
	RotoGrad	84.60 \pm 0.50	84.44 \pm 0.45	79.14 \pm 0.96	4.45 \pm 0.10	6.08 \pm 0.48	1.61 \pm 2.72

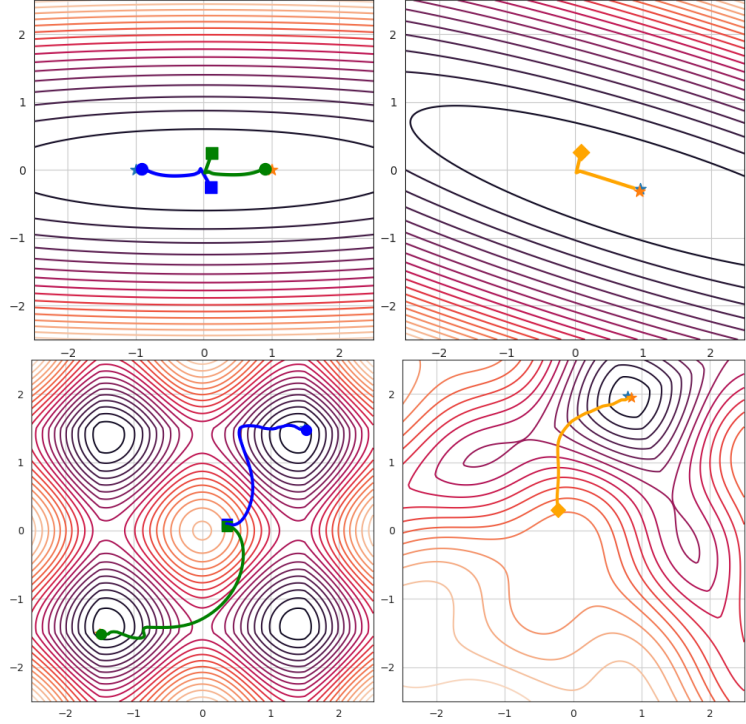
the other methods, all of them improved their results with respect to the vanilla case, with RotoGrad and GradNorm obtaining the second-best results.

Illustrative examples

We complement the illustrative figures shown in [Figure 5.3](#) by providing, for each example, an illustration of the effect of RotoGrad shown as an active and passive transformation. In an active transformation ([Figure A.2](#) left), points in the space are the ones modified. In our case, this means that we rotate feature z , obtaining r_1 and r_2 , while the loss functions remain the same. In other words, for each z we obtain a task-specific feature r_k that optimizes its loss function. In contrast, a passive transformation ([Figure A.2](#) right) keeps the points unaltered while applying the transformation to the space itself. In our case, this translates to rotating the optimization landscape of each loss function (now we have $L_k \circ R_k$ instead of L_k), so that our single feature z has an easier job at optimizing both tasks. In the case of RotoGrad, we can observe in both right figures that both optima lie in the same point, as we are aligning task gradients.

Besides the two regression experiments shown in [Section 5.3](#), we include

Figure A.2: Level plots showing the illustrative examples of Figure 5.3 for RotoGrad. **Top:** Convex case. **Bottom:** Non-convex case. **Left:** Active transformation (trajectories of r_k and the level plot of $L_1 + L_2$). **Right:** Passive transformation (trajectory of z and level plot of $(L_1 \circ R_1) + (L_2 \circ R_2)$).



here an additional experiment where we test RotoGrad in the worst-case scenario of gradient conflict, *i.e.*, one in which task gradients are opposite to each other. To this end, we solve a 2-task binary classification problem where, as dataset, we take 1000 samples from a 2D Gaussian mixture model with two clusters; $y_{nk} = 1$ if x_n was sampled from cluster k ; and $y_{nk} = 0$ otherwise. We use as model a logistic regression model of the form $y_k = W_2 \max(W_1 x + b_1, 0) + b_2$ with $b_1 \in \mathbb{R}^2$, $b_2 \in \mathbb{R}$, $W_1 \in \mathbb{R}^{2 \times 2}$, and $W_2 \in \mathbb{R}^{1 \times 2}$. Because rotations in 1D are ill-posed,² here we add task parameters to increase the dimensionality of z and make all parameters shared, so that there is still no task-specific parameters. To avoid a complete conflict where $\nabla_z L_1 + \nabla_z L_2 = 0$, we randomly flip the labels for the second tasks with 5% probability. Figure A.3 shows that, in this extreme scenario, RotoGrad is able to learn both tasks by aligning gradients, *i.e.*, by learning that one rotation is the inverse of the other $R_1 = R_2^\top$.

2: Only the identity is a proper rotation.

Training stability

While we showed in Subsection 5.4.1 only the results for the sum-of-digits task as they were nice and clear, here we show in Figure A.4 the results of those same experiments in Subsection 5.4.1 for all the different tasks. The same discussion from the main manuscript can be carried out for all metrics. Additionally, we can observe that the vanilla case (learning rate zero) completely overlooks the image-related task (Active pixels) while performing the best in the parity task.

Additionally, let us clarify what we mean here with stability, as in the main manuscript we mainly talked about convergence guarantees. In these experiments we test the convergence guarantees of the experiments in terms of training stability, meaning the variance of the obtained results across different runs. The intuition here is that, since the model does

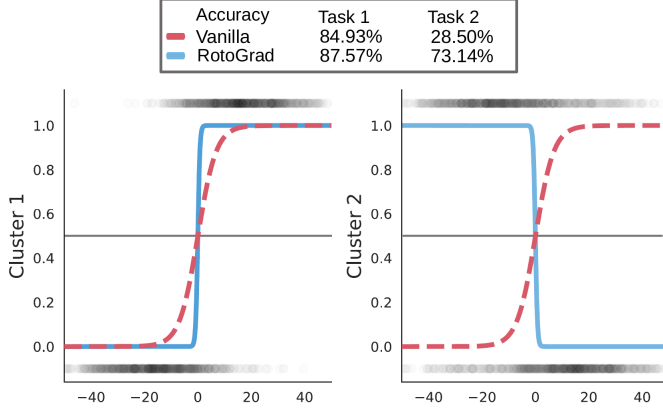


Figure A.3: Logistic regression for opposite classification tasks. Test data is plotted scattered as gray dots. RotoGrad learns both opposite rotations $R_1 = R_2^\top$.

not converge, we should expect some wriggling learning curves during training and, as we take the model with the best validation error, the individual task metrics should have bigger variance (*i.e.*, less stability) across runs.

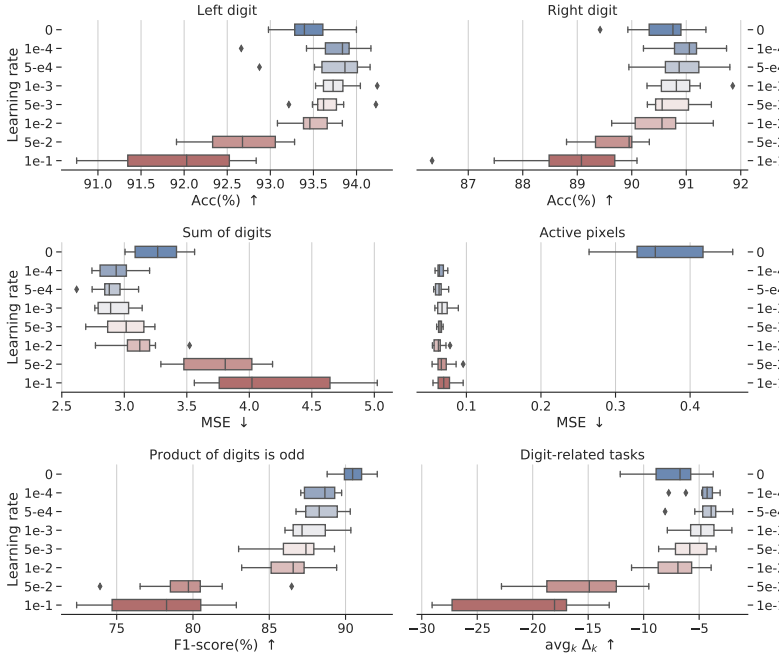


Figure A.4: RotoGrad's performance on all tasks for the experiments in Subsection 5.4.1 for all metrics. We can observe training instabilities/stiffness on all tasks as we highly increase/decrease RotoGrad's learning rate, as discussed in the main manuscript.

CIFAR10 and CelebA

For the sake of completeness, we present in Table A.3 and Table A.4 the same tables as in Section 5.4, but with more statistics of the results. For CIFAR10, we now included in Table A.3 the minimum task improvement across tasks and, while noisier, we can still observe that RotoGrad also improve this statistic. The standard deviation of the task improvement across tasks is, however, not too informative. In the case of CelebA, we added in Table A.4 the maximum F1-score across tasks and, similar to the last case, it is not too informative, as all methods achieve almost perfect f1-score in one of the classes. We also include the training times for some baselines, showing that RotoGrad stays on par with them.

Table A.3: Complete task-improvement statistics in CIFAR10 for all competing methods and RotoGrad with different dimensionality for R_k , see Section 5.4.

Method	d	$\min_k \Delta_k \uparrow$	$\text{med}_k \Delta_k \uparrow$	$\text{avg}_k \Delta_k \uparrow$	$\text{std}_k \Delta_k \downarrow$	$\max_k \Delta_k \uparrow$
Vanilla [†]	0	-0.81 ± 0.37	1.90 ± 0.53	2.58 ± 0.54	3.38 ± 0.94	11.14 ± 3.35
RotoGrad	64	-1.70 ± 0.81	1.79 ± 0.57	2.90 ± 0.49	3.98 ± 0.62	13.16 ± 2.40
RotoGrad	128	-1.12 ± 0.36	2.25 ± 1.07	2.97 ± 1.08	3.84 ± 0.87	12.64 ± 3.56
RotoGrad	256	0.17 ± 1.01	2.16 ± 0.72	3.68 ± 0.68	3.83 ± 0.74	14.01 ± 3.22
RotoGrad	512	-0.43 ± 0.76	3.67 ± 1.40	4.48 ± 0.99	4.23 ± 0.82	15.57 ± 3.99
Without R_k	Vanilla [†]	-0.81 ± 0.37	1.90 ± 0.53	2.58 ± 0.54	3.38 ± 0.94	11.14 ± 3.35
	GradDrop	-0.73 ± 0.33	2.80 ± 0.20	3.41 ± 0.45	4.08 ± 0.34	13.58 ± 1.50
	PCGrad	-1.52 ± 0.98	1.95 ± 0.87	2.86 ± 0.81	3.74 ± 0.69	12.01 ± 3.19
	MGDA-UB	-7.27 ± 1.36	-1.21 ± 0.74	-1.75 ± 0.43	3.24 ± 0.55	3.67 ± 0.98
	GradNorm	-0.35 ± 0.59	2.45 ± 0.66	3.23 ± 0.35	4.02 ± 0.33	14.25 ± 1.35
	IMTL-G	-0.39 ± 0.82	1.97 ± 0.29	2.73 ± 0.27	3.25 ± 0.75	10.20 ± 2.98
With R_k ($d = 512$)	Vanilla	-0.85 ± 0.58	3.10 ± 1.29	3.12 ± 0.79	4.05 ± 0.56	14.23 ± 2.86
	GradDrop	-1.49 ± 0.78	3.27 ± 1.61	3.54 ± 1.10	4.11 ± 0.56	13.88 ± 2.95
	PCGrad	-1.44 ± 0.58	2.67 ± 0.88	3.29 ± 0.46	3.90 ± 0.37	13.44 ± 1.86
	MGDA-UB	-3.59 ± 1.48	0.57 ± 0.62	0.21 ± 0.67	2.44 ± 0.52	4.78 ± 2.15
	GradNorm	-0.79 ± 1.28	3.10 ± 1.01	3.21 ± 1.04	3.41 ± 0.86	10.88 ± 4.73
	IMTL-G	-1.29 ± 0.52	1.81 ± 0.87	3.02 ± 0.69	3.81 ± 0.21	12.76 ± 1.77
	RotoGrad	-0.43 ± 0.76	3.67 ± 1.40	4.48 ± 0.99	4.23 ± 0.82	15.57 ± 3.99

Table A.4: Complete F1-score statistics and training hours in CelebA for all competing methods and two different architectures/settings (median over 5 runs), see Section 5.4. For the convolutional network we use $m = 256$, and $m = 1536$ for the residual network.

Method		Convolutional ($d = 512$)						ResNet-18 ($d = 2048$)					
		Task F1-scores (%) \uparrow						Task F1-scores (%) \uparrow					
		\min_k	med_k	avg_k	$\text{std}_k \downarrow$	\max_k	Hours	\min_k	med_k	avg_k	$\text{std}_k \downarrow$	\max_k	Hours
Without R_k	Vanilla [†]	1.62	53.39	58.49	24.26	96.97	7.62	15.45	63.04	62.85	22.09	96.58	1.49
	GradDrop	2.63	52.32	57.33	25.27	96.72	8.53	13.31	64.37	63.95	20.93	96.59	1.60
	PCGrad	2.69	54.60	56.87	25.75	97.04	34.05	13.61	62.45	62.74	21.60	96.64	5.75
	GradNorm	2.17	52.98	56.91	24.72	96.84	20.93	17.42	62.49	62.62	21.93	96.55	3.61
	IMTL-G	0.00	14.81	31.90	33.58	93.31	9.46	9.87	62.22	62.03	22.47	96.51	1.73
With R_k	Vanilla	4.24	49.85	55.33	26.03	96.88	16.29	19.71	63.56	63.23	21.16	96.55	9.33
	GradDrop	3.18	50.07	54.43	27.21	96.80	17.20	12.33	62.40	62.74	21.74	96.65	9.41
	PCGrad	1.44	53.05	54.72	27.61	96.90	41.79	14.71	63.65	62.61	22.22	96.59	13.72
	GradNorm	2.08	52.53	56.71	24.57	96.96	30.02	9.05	60.20	60.78	22.31	96.38	11.36
	IMTL-G	0.00	37.00	42.24	33.46	94.34	18.05	17.11	61.68	60.72	22.80	96.44	9.52
	RotoGrad	4.59	55.02	57.20	24.75	96.79	27.20	9.96	63.84	62.81	21.80	96.45	6.68

NYUv2

Complementing the results shown in Section 5.4, we show in Table A.5 the results obtained combining RotoGrad with all other existing methods (rows within RotoGrad +), for gradient scaling methods we only apply the rotation part of RotoGrad. Results show that RotoGrad helps improve/balance all other methods, which is specially true for those methods that heavily overlook some tasks. Specifically, MGDA-UB stops overlooking the semantic segmentation and depth estimation tasks, while PCGrad and GradDrop stop completely overlooking the surface normal loss. Note that we also show in Table A.5 the training times of each method, and RotoGrad stays on par with non-extended methods in training time. As mentioned in Appendix A.3.1, due to cluster overload, some times were deceptively high (specifically those baselines with R_k) as we had to run them on different machines, and were omitted to avoid confusion.

Table A.5: Results for different methods on the NYUv2 dataset with a SegNet model. RotoGrad obtains the best performance in segmentation and depth tasks on all metrics, while significantly improving the results on normal surfaces with respect to the vanilla case.

Method	Relative improvements \uparrow			Segmentation \uparrow		Depth \downarrow		Normal Surfaces						Time \downarrow
	Δ_S	Δ_D	Δ_N	mIoU	Pix Acc	Abs.	Rel.	Angle Dist. \downarrow		Within $t^\circ \uparrow$			h	
								Mean	Median	11.25	22.5	30		
STL	0.0	0.0	0.0	39.21	64.59	0.70	0.27	25.09	19.18	30.01	57.33	69.30	8.90	
RotoGrad +	GradDrop	1.2	12.6	-7.5	40.26	65.63	0.63	0.24	26.33	21.08	26.47	53.38	66.05	3.94
	PCGrad	0.0	19.7	-8.3	39.08	64.68	0.54	0.21	26.41	21.29	26.13	52.99	65.72	3.89
	MGDA-UB	2.5	23.2	-8.1	39.32	65.48	0.54	0.21	26.43	21.22	26.16	53.16	66.07	3.85
	GradNorm	1.1	21.4	-7.7	39.08	65.43	0.54	0.21	26.44	21.42	26.17	52.59	65.52	3.84
	IMTL-G	1.7	21.2	-6.9	40.13	65.17	0.55	0.21	26.20	21.06	26.69	53.39	66.04	3.96
With R_k ($m = 1024$)	Rotate	3.3	20.5	-6.6	39.63	66.16	0.53	0.21	26.12	20.93	26.85	53.76	66.50	3.82
	Scale	-0.3	20.0	-7.9	38.89	65.94	0.54	0.22	26.47	21.24	26.24	53.04	65.81	3.87
	RotoGrad	1.8	24.0	-6.1	39.32	66.07	0.53	0.21	26.01	20.80	27.18	54.02	66.53	3.83
	Vanilla	-2.7	20.6	-25.7	38.05	64.39	0.54	0.22	30.02	26.16	20.02	43.47	56.87	3.81
	GradDrop	-0.9	14.0	-25.2	38.79	64.36	0.59	0.24	29.80	25.81	19.88	44.08	57.54	4.01
	PCGrad	-2.7	20.5	-26.3	37.15	63.44	0.55	0.22	30.06	26.18	19.58	43.51	56.87	3.89
	MGDA-UB	-31.2	-0.7	0.6	21.60	51.60	0.77	0.29	24.74	18.90	30.32	57.95	69.88	3.85
	GradNorm	-0.6	19.5	-10.5	37.22	63.61	0.54	0.22	26.68	21.67	25.95	52.16	64.95	3.85
	IMTL-G	-0.3	17.6	-7.5	38.38	64.66	0.54	0.22	26.38	21.35	26.56	52.84	65.69	3.99
	Vanilla [†]	-0.9	16.8	-25.0	37.11	63.98	0.56	0.22	29.93	25.89	20.34	43.92	57.39	3.46
Without R_k	GradDrop	-0.1	15.7	-27.0	37.51	63.62	0.59	0.23	30.15	26.33	19.32	43.15	56.59	3.55
	PCGrad	-0.5	20.0	-24.6	38.51	63.95	0.55	0.22	29.79	25.77	20.61	44.22	57.64	3.51
	MGDA-UB	-32.2	-8.2	1.5	20.75	51.44	0.73	0.28	24.70	18.92	30.57	57.95	69.99	3.52
	GradNorm	2.2	20.6	-10.2	39.29	64.80	0.53	0.22	26.77	21.88	25.39	51.78	64.76	3.50
	IMTL-G	1.9	21.4	-6.7	39.94	65.96	0.55	0.21	26.23	21.14	26.77	53.25	66.22	3.61

Additional Material for Chapter 6

B.

B.1 Experimental details

B.1 Experimental details . . . 161

B.1.1 Hyperparameter optimization

B.1.1 Hyperparameters . . . 161
B.1.2 Network 161
B.1.3 Tasks 162
B.1.4 Datasets 162
B.1.5 Marker size 163

In [Chapter 6](#), we have attempted to perform an exhaustive hyperparameter search (including regularization methods) to avoid misconducts identified in prior works [99, 220]. Therefore, for every experiment,¹ we use BOHB [55] to perform a hyperparameter search, a state-of-the-art Bayesian gray-box hyperparameter tuning method. Moreover, we run each configuration three times² as we observed that the network was occasionally initialized close to the optimum of a task, biasing the hyperparameter selection.

1: *I.e.*, for every combination of benchmark gradient-conflict approach.

2: *I.e.*, a trial on BOHB is the average result of running three seeds.

Specifically, we tune those hyperparameters that affect training (*e.g.*, learning rate), but not those affecting the architecture (*e.g.*, the number of hidden layers). We also tune the hyperparameters from the gradient-conflict methods, and the amount of regularization via weight decay. Every experiment has the same computational resources (a dedicated DGX A100 GPU) and time budget (6 h).

After tuning, we repeat each experiment 30 times, fixing the randomness across methods to provide fair comparisons, and, in each run, take the best model according to validation error during training (a variation of early stopping quite popular in [multitask learning \(MTL\)](#)). We fix the batch size to 1024 in every experiment, and consider the following search space:

Hyperparameter	Method	Distribution	Min	Max
Learning rate	All	Log-uniform	1e−5	5e−2
Learning rate scheduler	All	Boolean	No	Yes
Exponential decay	All	Log-uniform	1e−7	1
	GradNorm			
Method-specific, α	CAGrad	Uniform	0	2
	GradVac			
RotoGrad’s learning rate	RotoGrad	Log-Uniform	1e−5	5e−2

On average, each experiment tried 50 different configurations.

B.1.2 Network

Notation. We use the following notation: [Conv- F - C] denotes a convolutional layer with kernel size F and C number of channels; [Max] denotes a max-pool layer of kernel size and stride 2; [Dense- H] a dense layer with output size H ; and [ReLU], [Sigmoid], and [Log-Softmax], their corresponding activation functions.

Table B.1: Summary of the tasks considered for the experiments of Chapter 6. C denotes the number of classes of the base dataset.

Task	Type	Loss	Metric	Expression	Predicts
Left	C-Class.	NLL	Acc.	y_L	Left digit.
Right	C-Class.	NLL	Acc.	y_R	Right digit.
Odd	2-Class.	BCE	F1	$y_L y_R \bmod 2$	Parity of the product of both digits.
Density	Regression	MSE	MSE	$\sum_{ij} \mathbf{1}_{\{x_{ij} > 0.5\}}$	Number of “active” pixels.
Number	Regression	MSE	MSE	$C y_L + y_R$	Number made by reading both digits together.
Both	C^2 -Class.	NLL	Acc.	$C y_L + y_R$	Number made by reading both digits together.

Architecture. Following the initial setup from Sener and Koltun [182], we implement the backbone of our model with a LeNet backbone [101] and, in order to modify the number of parameters of the backbone, we adopt the same strategy as Ruchte and Grabocka [167] and introduce a channel multiplier parameter, c . Namely, the backbone is described as:

Backbone: [Conv-5-10 c] [Max] [ReLU] [Conv-5-20 c] [Max] [Dense- d] [ReLU] .

Depending on the task type, we use a slightly different head:

Classification: [Dense-50] [ReLU] [Dense-C] [Log-Softmax] ,
 Binary classification: [Dense-1] [Sigmoid] ,
 Regression: [Dense-50] [ReLU] [Dense-1] .

B.1.3 Tasks

We consider the **negative log-likelihood (NLL)** for classification tasks, **binary cross-entropy (BCE)** for binary classification tasks, and **mean squared error (MSE)** for regression tasks. Regarding the task metrics, we consider accuracy (Acc.) for classification tasks, F1-score (F1) for binary classification tasks, and MSE for regression tasks. We show in Table B.1 a summary of the considered tasks:

As mentioned in the main text, we rescale the prediction range of regression tasks to make them invariant to the number of classes and the image size. Specifically, instead of predicting, say, a number $C y_L + y_R$ for the **N** task, we predict $(C y_L + y_R)/C^2$, and for the task **D** we predict $\frac{1}{784} \sum_{ij} \mathbf{1}_{x_{ij}}$.

It is also important to note that the tasks are defined with respect to the class number, and not the MNIST digit. This way, we can replace MNIST with another dataset without further changes.

B.1.4 Datasets

We consider datasets that are (almost) perfect drop-in replacements of the original MNIST [102] dataset. In particular, every dataset considered contains single-channel images of size 28×28 px. We summarize the considered datasets in Table B.2. All datasets contain images with the same dimensions and number of channels, to reduce to the minimum the number of modifications required to use them. Moreover, as we have normalized the **N** task, it should be invariant to the number of classes in the dataset (disregarding the dimension of the classification head, as pointed out at the end of Subsection 6.3.3).

Table B.2: Summary of the tasks considered for the experiments of Chapter 6. C denotes the number of classes of the base dataset.

Dataset name	No. Classes (C)	Training size	Composition
MNIST [102]	10	60 000	Handwritten digits.
FMNIST [219]	10	60 000	Clothes images.
KMNIST [31]	10	60 000	Handwritten kanjis.
Letter [33]	37	88 800	Handwritten letters.
Balanced [33]	47	112 800	Handwritten digits and letters.

B.1.5 A note on the marker size

To compute the marker size of Figure 6.11, where the results are not directly comparable as we change the dataset across experiments, we resort to a slight variation of the robust ranking, R_R , introduced in Subsection 6.2.1. Instead of computing R_R based on the raw task metrics, we compute them with respect to their relative improvement *w.r.t.* to single task learning (STL). That is, we perform the rankings necessary for R_R on the relative task performances, Δ_k , rather on the metrics, M_k . While this is not ideal because of the incomparability issues, note that still we are aggregating the relative improvement metrics across the same quantity, yet different baselines, so that the results should still be sensible.

Additional Material for Chapter 8

C.

C.1 Basic properties of L -smoothness

In this section, we prove two basic properties of L -smooth functions, as we used them on the main text. Specifically:

Proposition C.1 If a function $\ell(\boldsymbol{\eta})$ is L_i -smooth with respect to $\eta_i \in \boldsymbol{\eta} \in \mathbb{R}^I$, for every $i \in \{1, 2, \dots, I\}$, then ℓ is $\sum_i L_i$ -smooth w.r.t. $\boldsymbol{\eta}$.

Proof. Consider two arbitrary $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^I$. Then, we have by assumption that $\|\partial_{\eta_i} \ell(\boldsymbol{a}) - \partial_{\eta_i} \ell(\boldsymbol{b})\| \leq L_i \|\boldsymbol{a} - \boldsymbol{b}\|$ for $i \in \{1, 2, \dots, I\}$ and

$$\|\nabla_{\boldsymbol{\eta}} \ell(\boldsymbol{a}) - \nabla_{\boldsymbol{\eta}} \ell(\boldsymbol{b})\| \leq \sum_i \|\partial_{\eta_i} \ell(\boldsymbol{a}) - \partial_{\eta_i} \ell(\boldsymbol{b})\| \leq \sum_i L_i \|\boldsymbol{a} - \boldsymbol{b}\|. \quad (\text{C.1})$$

□

Proposition C.2 If two functions $\ell_1(\boldsymbol{\eta})$ and $\ell_2(\boldsymbol{\eta})$ are L_1 -smooth and L_2 -smooth with respect to $\boldsymbol{\eta}$, respectively, then $\ell_1 + \ell_2$ is $L_1 + L_2$ -smooth with respect to $\boldsymbol{\eta}$.

Proof. Consider two arbitrary $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^I$. Then,

$$\begin{aligned} & \|\nabla_{\boldsymbol{\eta}}(\ell_1 + \ell_2)(\boldsymbol{a}) - \nabla_{\boldsymbol{\eta}}(\ell_1 + \ell_2)(\boldsymbol{b})\| \\ &= \|(\nabla_{\boldsymbol{\eta}} \ell_1(\boldsymbol{a}) - \nabla_{\boldsymbol{\eta}} \ell_1(\boldsymbol{b})) + (\nabla_{\boldsymbol{\eta}} \ell_2(\boldsymbol{a}) - \nabla_{\boldsymbol{\eta}} \ell_2(\boldsymbol{b}))\| \\ &\leq \|\nabla_{\boldsymbol{\eta}} \ell_1(\boldsymbol{a}) - \nabla_{\boldsymbol{\eta}} \ell_1(\boldsymbol{b})\| + \|\nabla_{\boldsymbol{\eta}} \ell_2(\boldsymbol{a}) - \nabla_{\boldsymbol{\eta}} \ell_2(\boldsymbol{b})\| \\ &\leq L_1 \|\boldsymbol{a} - \boldsymbol{b}\| + L_2 \|\boldsymbol{a} - \boldsymbol{b}\| = (L_1 + L_2) \|\boldsymbol{a} - \boldsymbol{b}\|. \end{aligned} \quad (\text{C.2})$$

□

C.2 Exponential family

We here provide the proofs of [Proposition 8.1](#) on how to scale the exponential family, as well as a reference table with the properties of some common distributions of the manuscript. As a reminder, the exponential family is characterized for having the form

$$p(x; \boldsymbol{\eta}) = h(x) \exp [\boldsymbol{\eta}^\top T(x) - A(\boldsymbol{\eta})], \quad (\text{C.3})$$

where $\boldsymbol{\eta}$ are the natural parameters, $T(x)$ the sufficient statistics, $h(x)$ the base measure, and $A(\boldsymbol{\eta})$ the log-partition function, which ensures that the distribution integrates to one.

To ease the task of transforming the natural ($\boldsymbol{\eta}$) and canonical ($\boldsymbol{\theta}$) parameters, we provide in [Table C.1](#) a cheat-sheet with the relationship between

C.1 L -smoothness	165
C.2 Exponential family	165
C.3 Description of Lip-std	167
C.4 L -smoothness after std	170
C.5 Experimental setup	174
C.6 Additional results	176

Table C.1: Relationship between the traditional parameters, θ , and the natural parameters, η , and how to scale the latter (see Proposition 8.1 in the main text) for different distributions of the exponential family.

Likelihood	θ	$T(x)$	$\theta \mapsto \eta$	$\eta \mapsto \theta$	$x \mapsto \tilde{x}$	$f(\omega)$	$\eta \mapsto \tilde{\eta}$
Normal	μ	x	$\frac{\mu}{\sigma^2}$	$\frac{-\eta_1}{2\eta_2}$	ωx	ω	$\frac{\eta_1}{\omega}$
	σ^2	x^2	$\frac{-1}{2\sigma^2}$	$\frac{-1}{\eta_2}$		ω^2	$\frac{\eta_2}{\omega^2}$
Log-normal	μ	$\log x$	$\frac{\mu}{\sigma^2}$	$\frac{-\eta_1}{2\eta_2}$	x^ω	ω	$\frac{\eta_1}{\omega}$
	σ^2	$(\log x)^2$	$\frac{-1}{2\sigma^2}$	$\frac{-1}{\eta_2}$		ω^2	$\frac{\eta_2}{\omega^2}$
Gamma	α	$\log x$	$\alpha - 1$	$\eta_1 + 1$	ωx	1	η_1
	β	x	$-\beta$	$-\eta_2$		ω	$\frac{\eta_2}{\omega}$
Inv. Gaussian	μ	x	$-\frac{\lambda}{2\mu^2}$	$\sqrt{\frac{\eta_2}{\eta_1}}$	ωx	ω	$\frac{\eta_1}{\omega}$
	λ	$\frac{1}{x}$	$-\frac{\lambda}{2}$	$-2\eta_2$		$\frac{1}{\omega}$	$\omega\eta_2$
Inv. Gamma	α	$\log x$	$-\alpha - 1$	$-\eta_1 - 1$	ωx	1	η_1
	β	$\frac{1}{x}$	$-\beta$	$-\eta_2$		$\frac{1}{\omega}$	$\omega\eta_2$
Exponential	λ	x	$-\lambda$	$-\eta_1$	ωx	ω	$\frac{\eta_1}{\omega}$
Rayleigh	σ	$\frac{x^2}{2}$	$\frac{-1}{\sigma^2}$	$\sqrt{\frac{1}{-\eta_1}}$	ωx	ω^2	$\frac{\eta_1}{\omega}$
Bernoulli	p	x	$\log \frac{p}{1-p}$	$\frac{1}{1+e^{-\eta_1}}$	-	-	-
Poisson	λ	x	$\log \lambda$	e^{η_1}	-	-	-

them some common distributions, and how their natural parameters scale as a function of the scaling factor ω .

Regarding the relation between scaled and original variables within the exponential family, we now prove a more general version of Proposition 8.1 from the main text.

Proposition C.3 Let $p(x; \eta)$ be a density function of the exponential family with $x \in X \subset \mathbb{R}$ and $\eta \in \mathbb{R}^I$. Assume a bijective function $\tilde{x} : X \times \mathbb{R}^+ \rightarrow X$ such that, given $\omega \in \mathbb{R}^+$, it defines the function (and random variable (R.V.)) $\tilde{x}_\omega := \tilde{x}(x, \omega)$. If all sufficient statistics factorize as $T_i(\tilde{x}_\omega) = f_i(\omega)T_i(x) + g_i(\omega)$, then if we define $\tilde{\eta}$ such that $\eta = f(\omega) \odot \tilde{\eta}$, where $f = [f_1 \ f_2 \ \dots \ f_I]$ and \odot is the Hadamard product, we have

$$\partial_{\tilde{\eta}_i}^j \log p(\tilde{x}_\omega; \tilde{\eta}) = f_i(\omega)^j \partial_{\eta_i}^j \log p(x; \eta) \quad \text{for } j \in \mathbb{N}, \quad (\text{C.4})$$

where $\partial_{\tilde{\eta}_i}^j$ denotes the j -th-partial derivative with respect to $\tilde{\eta}_i$.

Proof. First, we relate the normalizing constants $A(\tilde{\eta})$ and $A(\eta)$ of $\log p(\tilde{x}_\omega; \tilde{\eta})$ and $\log p(x; \eta)$, respectively:

$$\begin{aligned}
 A(\tilde{\eta}) &= \log \int h(\tilde{x}_\omega) \exp [\tilde{\eta}^\top T(\tilde{x}_\omega)] d\tilde{x}_\omega \\
 &= \sum_i g_i(\omega) \tilde{\eta}_i + \log \int h(\tilde{x}_\omega) \exp [\eta^\top T(x)] d\tilde{x}_\omega \\
 &= \sum_i g_i(\omega) \tilde{\eta}_i + \log \int \frac{h(\tilde{x}_\omega)}{h(x)} h(x) \exp [\eta^\top T(x) \pm A(\eta)] \partial_x \tilde{x}_\omega(x) dx
 \end{aligned}$$

$$= \sum_i g_i(\omega) \tilde{\eta}_i + A(\boldsymbol{\eta}) + \log \mathbb{E}_{p(x; \boldsymbol{\eta})} \left[\frac{h(\tilde{x}_\omega)}{h(x)} \partial_x \tilde{x}_\omega(x) \right]. \quad (\text{C.5})$$

We can safely divide by $h(x)$ above, since it is the Radon-Nikodym derivative $\frac{dH(x)}{dx}$ and we can assume that is non-zero almost everywhere in the domain of the likelihood.

Second, we relate $p(\tilde{x}_\omega; \tilde{\boldsymbol{\eta}})$ and $p(x; \boldsymbol{\eta})$ using a similar calculation:

$$\begin{aligned} p(\tilde{x}_\omega; \tilde{\boldsymbol{\eta}}) &= h(\tilde{x}_\omega) \exp [\tilde{\boldsymbol{\eta}}^\top \mathbf{T}(\tilde{x}_\omega) - A(\tilde{\boldsymbol{\eta}})] \\ &= \frac{h(\tilde{x}_\omega)}{h(x)} \frac{h(x) \exp [\boldsymbol{\eta}^\top \mathbf{T}(x) - A(\boldsymbol{\eta})]}{\mathbb{E}_{p(x; \boldsymbol{\eta})} \left[\frac{h(\tilde{x}_\omega)}{h(x)} \partial_x \tilde{x}_\omega(x) \right]} \\ &= \frac{h(\tilde{x}_\omega)}{h(x)} \frac{p(x; \boldsymbol{\eta})}{\mathbb{E}_{p(x; \boldsymbol{\eta})} \left[\frac{h(\tilde{x}_\omega)}{h(x)} \partial_x \tilde{x}_\omega(x) \right]}. \end{aligned} \quad (\text{C.6})$$

By defining everything that is not $p(x; \boldsymbol{\eta})$ in the previous equation as $\psi(x, \omega)$ we have that:

$$\log p(\tilde{x}_\omega; \tilde{\boldsymbol{\eta}}) = \log p(x; \boldsymbol{\eta}) + \log \psi(x, \omega). \quad (\text{C.7})$$

Now, for the case $j = 1$ we just need to use the chain rule and the fact that $\psi(x, \omega)$ does not depend on η_i :

$$\begin{aligned} \partial_{\tilde{\eta}_i} \log p(\tilde{x}_\omega; \tilde{\boldsymbol{\eta}}) &= \partial_{\tilde{\eta}_i} [\log p(x; \boldsymbol{\eta}) + \log \psi(x, \omega)] \\ &= \partial_{\tilde{\eta}_i} \eta_i \cdot \partial_{\eta_i} \log p(x; \boldsymbol{\eta}) \\ &= f_i(\omega) \cdot \partial_{\eta_i} \log p(x; \boldsymbol{\eta}). \end{aligned} \quad (\text{C.8})$$

And we can prove the case $j > 1$ by induction:

$$\begin{aligned} \partial_{\tilde{\eta}_i}^j \log p(\tilde{x}_\omega; \tilde{\boldsymbol{\eta}}) &= \partial_{\tilde{\eta}_i} \left[\partial_{\tilde{\eta}_i}^{j-1} \log p(\tilde{x}_\omega; \tilde{\boldsymbol{\eta}}) \right] \\ &= f_i^{j-1}(\omega) \cdot \partial_{\tilde{\eta}_i} \eta_i \cdot \partial_{\eta_i} \left[\partial_{\tilde{\eta}_i}^{j-1} \log p(x; \boldsymbol{\eta}) \right] \\ &= f_i^j(\omega) \cdot \partial_{\eta_i}^j \log p(x; \boldsymbol{\eta}). \end{aligned} \quad (\text{C.9})$$

□

C.3 Full description of Lipschitz standardization

In this section we provide a complete description of Lipschitz standardization. Following the naming convention introduced in [Section 8.5](#), Lipschitz standardization would be denoted by `lip-gamma`, *i.e.*, the algorithm is composed of two different components: **i)** applying the Gamma trick, and **ii)** scaling each distribution properly using our Lipschitz criterion ([Equation 8.14](#) from the main text). We proceed by describing each individual component.

C.3.1 Gamma Trick	168
C.3.2 Workflow example . . .	169
C.3.3 Lipschitz scaling	170

C.3.1 Gamma Trick

As introduced in [Subsection 8.4.1](#) from the main text, the Gamma Trick is a sequence of steps that allow us to model discrete variables using a set of Gamma distributions. Here, we describe all of these steps separately.

Preprocessing. The first step is to analyse which of the given likelihoods need to be replaced to transform the data $x \mapsto \bar{x}$ during preprocessing. In practice, we have a set ‘testing’ and ‘training’ likelihoods, as well as mappings between them to help us identify which of the training likelihoods correspond to which of the testing likelihoods. Depending on the likelihood, we proceed as follows:

- ▶ **Continuous.** One-to-one correspondence. Append the likelihood to the set of training likelihoods, and do not alter the data.
- ▶ **Bernoulli.**
 1. Make sure that $x \in \{1, 2\}$.
 2. Sample $\epsilon_{dn} \sim \text{Beta}(1.1, 30)$, one per sample.
 3. Transform the training data such that $\bar{x}_{dn} = x_{dn} + \epsilon_{dn}$.
 4. Again, one-to-one correspondence. We append a Gamma to the set of training likelihoods.
- ▶ **Poisson.** Exactly as in the Bernoulli case, but making sure that the data lies on the natural numbers.
- ▶ **M-class categorical.**
 1. Transform the data to a one-hot encoding (if it was not already), so that now $x_n = [x_{0n} \ x_{1n} \ \dots \ x_{Mn}]$.
 2. One-to-many correspondence. For each of the classes we perform the exact same steps described for the Bernoulli distribution. Note that now we need to keep track that these new M Gamma distributions correspond to the d -th modality.

Afterwards, all the modalities are scaled accordingly.

Training. During training, we just need to use the preprocessed data and the training likelihoods as usual.

Testing. During testing we need to distinguish two cases:

- ▶ **Data for the model.** As the model expects data with the same properties as the training data, we need to apply the same preprocessing as to the training data, which we can easily do on the fly.
- ▶ **Data for the likelihood evaluation.** When evaluating the likelihood, we need to use the original testing data, that is, without the preprocessing described above. Moreover, note that the testing likelihoods are the original ones.

Recovering the original parameters. As described in the main text, we need to come back to the original space whenever we evaluate the model, and that means recovering the original parameters from the ones obtained by the model. Specifically, assuming that the model outputs $\tilde{\eta}$, we scaled them back to $\bar{\eta}$ if the Gamma trick was applied to that modality, then:

- **Continuous.** Nothing is necessary, that is, $\eta = \bar{\eta}$.
- **Bernoulli.**
 1. We transform the natural parameters to the canonical ones, *i.e.*, $\bar{\alpha} = \bar{\eta}_1 + 1$ and $\bar{\beta} = -\bar{\eta}_2$.
 2. We compute the mean, *i.e.*, $\mu = \bar{\alpha} / \bar{\beta} - \mathbb{E}[\epsilon]$.
 3. We obtain the parameter p by making sure that μ fulfils the constraints, *i.e.*, $p = \max(0, \min(1, \mu))$.
- **Poisson.** Same steps as for the Bernoulli case, but for the last one we recover the rate parameter by doing $\lambda = \max(\delta, \mu)$, where $0 < \delta \ll 1$ ensures that $\lambda > 0$.
- **Categorical.**
 1. For each of the M classes, we recover the parameters μ_m as described in the Bernoulli case, but omitting the last step.
 2. We obtain the parameters π_m of the distribution by normalizing the μ_m , *i.e.*, $\pi_m = \mu_m / \sum_i \mu_i$.

C.3.2 Illustrative example of data workflow

We provide a simple example that shows how data is transformed and used throughout the entire process. Assume that we have two input dimensions, $D = 2$, whose distributions are assumed to be normal $x_1 \sim \mathcal{N}(\mu, \sigma)$ and categorical with 3 classes $x_2 \sim \text{Cat}(\pi[\pi_1 \ \pi_2 \ \pi_3])$, respectively. Let us further suppose that we want to use **lip-gamma**, *i.e.*, Lipschitz-standardization combined with the Gamma trick. Then, we would not alter the first variable $\bar{x}_1 = x_1 \sim \mathcal{N}(\mu, \sigma)$, but substitute x_2 with $\bar{x}_{2j} = x_{2j} + \epsilon_j \sim \Gamma(\bar{\alpha}_j, \bar{\beta}_j)$, where $j \in \{1, 2, 3\}$ are the indexes of the new variables, $x_{2j} \sim \text{Bern}(p_j)$ models the j -th element of x_2 when considered as a one-hot encoding, and $\epsilon_j \sim \text{Beta}(1.1, 30)$ is the (independent) additive noise variables.

Now we can transform all variables, obtaining the new scaled variables $\tilde{x}_1 = \omega_1 \bar{x}_1 \sim \mathcal{N}(\tilde{\mu}, \tilde{\sigma})$ and $\tilde{x}_{2j} = \omega_{2j} \bar{x}_{2j} \sim \Gamma(\tilde{\alpha}, \tilde{\beta})$ for $j \in \{1, 2, 3\}$. After training, or whenever we need to evaluate the model in non-training data, we must return to the original probabilistic model x_1, x_2 . When recovering the \bar{x} variables, we need to use **Proposition 8.1** so that $\bar{\eta}_i = f_i(\omega) \odot \tilde{\eta}_i$, where we have obtained $\tilde{\eta}_i$ as the output of our model. To finally recover the original variables, x_1, x_2 , we do not need to do anything to x_1 since $x_1 = \bar{x}_1$. For the second variable, we obtain $x_{2j} \sim \text{Bern}(p_j)$ as

$$p_j = \max(0, \min(1, \mathbb{E}[\bar{x}_{2j}] - \mathbb{E}[\epsilon_j])) = \max(0, \min(1, \bar{\alpha}_j / \bar{\beta}_j - 0.035)),$$

and finally recover $x_2 \sim \text{Cat}(\pi)$ with $\pi = [\frac{p_1}{p_1+p_2+p_3} \ \frac{p_2}{p_1+p_2+p_3} \ \frac{p_3}{p_1+p_2+p_3}]$.

C.3.3 Lipschitz scaling criterion

We describe now how to implement Lipschitz standardization to find the weights according to

$$\omega_d^* := \arg \min_{\omega_d} \left(\sum_{i=1}^{I_d} \tilde{L}_{di}(\omega_d) - L^* \right)^2, \quad (\text{C.10})$$

for the distributions considered in the main manuscript. Here, we assume that every distribution is continuous or the Gamma trick has been applied, otherwise, discrete variables are ignored as they cannot be scaled. Regarding the target smoothness L^* , as mentioned in the main paper, it is set to $1/(D\alpha)$ in the general case, where D is the number of likelihoods that can be scaled, and α the initial learning rate set by the practitioner. For the particular case of a M -class categorical distribution under with the Gamma trick, the target smoothness for the new M Gamma distributions is set to $1/(DM\alpha)$.

1: *I.e.*, to solve Equation 8.14.

2: Exploiting the fact that we have a continuous increasing function with respect to the weight ω .

Algorithm C.1: Pseudocode to compute the Lipschitz-standardization weights for a given continuous distribution.

In Algorithm C.1 we show the pseudocode to find the optimal weight¹ for a single modality. In short, the algorithm computes the Hessian at the **maximum likelihood estimation (MLE)** η estimators using either automatic differentiation or closed-form expressions, and then optimize the equation above using any suitable optimization method, *e.g.*, root-finding methods. Some important remarks: **i)** in this case, computing the Hessian matrix is cheap; **ii)** the derivatives are computed once, and we avoid recalculating derivatives using Equation 8.13; **iii)** we rely on numerical methods to find ω , but it is easy to show that Equation 8.14 has a single positive solution for the considered distributions;² and **iv)** to ensure that the optimizer finds a positive ω , we parametrize it with a softplus function, turning Equation 8.14 into an unconstrained optimization problem.

```

1  function: HessianAtMLE
2  input: data  $X = \{x_n\}_{n=1}^N$ 
3  begin
4       $\hat{\eta} \leftarrow \text{MLE}_p(X)$ 
5       $\ell \leftarrow \sum_n \log p(x_n; \hat{\eta})$                                 # Evaluate the log-likelihood
6       $H \leftarrow \text{Hessian}(\ell, \eta)$                                 # Hessian w.r.t. the natural parameters
7      return  $H$ 
8  end
9
10 function: FindWeight
11 input: data  $X$ , target smoothness  $L^*$ 
12 begin
13      $H \leftarrow \text{HessianAtMLE}(X)$ 
14      $\text{goal}(\omega) \leftarrow (\|H \odot f(\omega) \mathbf{1}^\top\|_{2,1} - L^*)^2$ 
15     return  $\omega \leftarrow \text{solve}(\text{goal})$                                 # E.g., using Brent's method
16 end

```

C.4 L -smoothness after standardizing

C.4.1 Gamma distribution . . . 174

Remark C.1 We use throughout the fact that $\partial_{\eta_i} A(\eta) = \mathbb{E}[T_i(x)]$ for every $i \in \{1, 2, \dots, I\}$ as long as the distribution belongs to the exponential family.

In this section, we compute the local L -smoothness around a point after standardizing the data for some common distributions. To this end, we first compute the Hessian of $\log p(x; \eta)$ *w.r.t.* η . Then, we can compute the L -smoothness after standardizing in two equivalent ways: **i)** using Equation 8.13; or **ii)** evaluating the second derivatives on the standardized

parameters. We choose the latter as it is simpler in this case, whereas the former enable an algorithm that reuses computations.

(Log-)Normal distribution.

$$\mathbb{E}[T_1(x)] = \mathbb{E}[x] = \mu = \frac{-\eta_1}{2\eta_2}, \quad (\text{C.11})$$

$$\mathbb{E}[T_2(x)] = \mathbb{E}[x^2] = \mu^2 + \sigma^2 = \frac{\eta_1^2}{4\eta_2^2} + \frac{-1}{2\eta_2} = \frac{\eta_1^2 - 2\eta_2}{4\eta_2^2}, \quad (\text{C.12})$$

$$H_{11} = -\partial_{\eta_1} \frac{-\eta_1}{2\eta_2} = -\frac{-1}{2\eta_2} = -\sigma^2, \quad (\text{C.13})$$

$$\begin{aligned} H_{22} &= -\partial_{\eta_2} \frac{\eta_1^2 - 2\eta_2}{4\eta_2^2} = \frac{-1}{4} \frac{-2\eta_2^2 - 2\eta_2(\eta_1^2 - 2\eta_2)}{\eta_2^4} \\ &= -\frac{\eta_2 - \eta_1^2}{2\eta_2^3} = -2\sigma^2(\sigma^2 + 2\mu^2), \end{aligned} \quad (\text{C.14})$$

$$H_{12} = -\partial_{\eta_1} \frac{\eta_1^2 - 2\eta_2}{4\eta_2^2} = -\frac{\eta_1}{2\eta_2^2} = -2\mu\sigma^2. \quad (\text{C.15})$$

Therefore, we have that

$$L_1 = \sqrt{\sigma^4 + 4\mu^2\sigma^4} \quad \text{and} \quad L_2 = \sqrt{4\mu^2\sigma^4 + 4\sigma^4(\sigma^2 + 2\mu^2)^2}, \quad (\text{C.16})$$

around the point $[\mu \ \sigma^2]$. After standardizing the data, we have that $\tilde{\mu} = \mu/\sigma$ and $\tilde{\sigma}^2 = 1$, resulting in

$$\tilde{L}_1^{\text{std}} = \sqrt{1 + 4(\mu/\sigma)^2} \quad \text{and} \quad \tilde{L}_2^{\text{std}} = 2\sqrt{(\mu/\sigma)^2 + (1 + 2(\mu/\sigma)^2)^2}. \quad (\text{C.17})$$

Gamma distribution.

$$\begin{aligned} \mathbb{E}[T_1(x)] &= \alpha - \log \beta + \log \Gamma(\alpha) + (1 - \alpha)\psi(\alpha) \\ &= \eta_1 + 1 - \log(-\eta_2) + \log \Gamma(\eta_1 + 1) - \eta_1\psi(\eta_1 + 1), \end{aligned} \quad (\text{C.18})$$

$$\mathbb{E}[T_2(x)] = \mathbb{E}[x] = \frac{\alpha}{\beta} = \frac{\eta_1 + 1}{-\eta_2}, \quad (\text{C.19})$$

$$\begin{aligned} H_{11} &= -\partial_{\eta_1} [\eta_1 + 1 - \log(-\eta_2) + \log \Gamma(\eta_1 + 1) - \eta_1\psi(\eta_1 + 1)] \\ &= -1 + \psi(\eta_1 + 1) - \psi(\eta_1 + 1) + \eta_1\psi^{(1)}(\eta_1 + 1) \\ &= -1 + \eta_1\psi^{(1)}(\eta_1 + 1) = (\alpha - 1)\psi^{(1)}(\alpha) - 1, \end{aligned} \quad (\text{C.20})$$

$$H_{22} = -\partial_{\eta_2} \frac{\eta_1 + 1}{-\eta_2} = -\frac{\eta_1 + 1}{\eta_2^2} = -\alpha/\beta^2 = -\mathbb{V}[x], \quad (\text{C.21})$$

$$H_{12} = \frac{-1}{-\eta_2} = -1/\beta. \quad (\text{C.22})$$

So that

$$L_1 = \sqrt{[1 + (\alpha - 1)\psi^{(1)}(\alpha)]^2 + 1/\beta^2} \quad \text{and} \quad (\text{C.23})$$

$$L_2 = \sqrt{\mathbb{V}[x]^2 + 1/\beta^2}. \quad (\text{C.24})$$

Remark C.2 We denote the Hessian as \mathbf{H} , and use the fact that

$$\mathbf{H} = \begin{bmatrix} \partial_{\eta_1} \mathbb{E}[T_1(x)] & \partial_{\eta_2} \mathbb{E}[T_1(x)] \\ \partial_{\eta_1} \mathbb{E}[T_2(x)] & \partial_{\eta_2} \mathbb{E}[T_2(x)] \end{bmatrix},$$

and $H_{12} = H_{21}$.

After standardizing, we obtain $\tilde{\alpha} = \alpha$, $\tilde{\beta} = \sqrt{\alpha}$, and $\mathbb{V}[\tilde{x}] = 1$. Thus:

$$L_1^{\text{std}} = \sqrt{[1 + (1 - \alpha)\psi^{(1)}(\alpha)]^2 + 1/\alpha^2} \quad \text{and} \quad L_2^{\text{std}} = \sqrt{1 + 1/\alpha}, \quad (\text{C.25})$$

i.e., \tilde{L}_1^{std} is a function dominated by $\psi^{(1)}(\alpha)$ for $\alpha < 1$ and by $1/\sqrt{\alpha}$ otherwise, and \tilde{L}_2^{std} is dominated by $1/\sqrt{\alpha}$ for $\alpha < 1$ and by 1 otherwise.

Exponential distribution. If $x \sim \text{Exp}(\lambda)$, then $x \sim \Gamma(1, \lambda)$, and we can use the previous result to state that $L_1 = \mathbb{V}[x]$ and $\tilde{L}_1^{\text{std}} = 1$.

Rayleigh distribution. This distribution has parameter $\sigma > 0$, sufficient statistic $T_1(x) = x^2/2$, and natural parameter $\eta_1 = -1/\sigma^2$. Using the fact that its raw moments are of the form $\mathbb{E}[x^i] = \sigma^i 2^{i/2} \Gamma(1 + \frac{i}{2})$, we have:

$$\mathbb{E}[T_1(x)] = \frac{1}{2} \mathbb{E}[x^2] = \frac{1}{2} \sigma^2 2 \Gamma(2) = \sigma^2 = \frac{-1}{\eta_1}, \quad (\text{C.26})$$

$$H_{11} = -\mathbb{E}[T_1(x)] = \frac{1}{\eta_1^2} = \sigma^4. \quad (\text{C.27})$$

Therefore, $L_1 = \sigma^4$. After standardizing, $\mathbb{V}[\tilde{x}] = \frac{4-\pi}{2} \sigma^2 = 1 \Rightarrow \tilde{\sigma}^2 = \frac{2}{4-\pi}$ and $\tilde{L}_1^{\text{std}} = \left(\frac{2}{4-\pi}\right)^2 \approx 5.428$.

Inverse Gaussian distribution. It has parameters $\mu, \lambda > 0$, sufficient statistics $T_1(x) = x$, $T_2(x) = 1/x$, and natural parameters $\eta_1 = \frac{-\lambda}{2\mu^2}$, $\eta_2 = \frac{-\lambda}{2}$. Thus, we have that

$$\mathbb{E}[T_1(x)] = \mathbb{E}[x] = \mu = \sqrt{\eta_2/\eta_1}, \quad (\text{C.28})$$

$$\mathbb{E}[T_2(x)] = \mathbb{E}\left[\frac{1}{x}\right] = \frac{1}{\mu} + \frac{1}{\lambda} = \sqrt{\frac{\eta_1}{\eta_2}} - \frac{1}{2\eta_2}, \quad (\text{C.29})$$

$$\begin{aligned} H_{11} &= -\partial_{\eta_1} \sqrt{\frac{\eta_2}{\eta_1}} = -\sqrt{\eta_2} \partial_{\eta_1} \frac{1}{\sqrt{\eta_1}} = -\frac{-1}{2} \sqrt{\frac{\eta_2}{\eta_1}} \frac{1}{\eta_1} \\ &= -\sqrt{\frac{\eta_2}{\eta_1}} \frac{\eta_2}{\eta_1} \frac{1}{-2\eta_2} = -\mu^3/\lambda, \end{aligned} \quad (\text{C.30})$$

$$H_{12} = -\partial_{\eta_2} \sqrt{\frac{\eta_2}{\eta_1}} = \frac{-1}{2} \frac{1}{\sqrt{\eta_1 \eta_2}} = \sqrt{\frac{\eta_2}{\eta_1}} \frac{1}{-2\eta_2} = \mu/\lambda, \quad (\text{C.31})$$

$$\begin{aligned} H_{22} &= -\partial_{\eta_2} \left(\sqrt{\frac{\eta_1}{\eta_2}} - \frac{1}{2\eta_2} \right) = \frac{1}{2} \sqrt{\frac{\eta_1}{\eta_2}} \frac{1}{\eta_2} - \frac{1}{2\eta_2^2} \\ &= -\frac{1 - \sqrt{\eta_1 \eta_2}}{2\eta_2^2} = -\frac{2\mu + \lambda}{\mu\lambda^2}. \end{aligned} \quad (\text{C.32})$$

Therefore,

$$L_1 = \sqrt{\frac{\mu^6 + \mu^2}{\lambda^2}} = \frac{\mu\sqrt{\mu^4 + 1}}{\lambda} \quad \text{and} \quad (\text{C.33})$$

$$L_2 = \sqrt{\mu^2/\lambda^2 + \left(\frac{2\mu + \lambda}{\mu\lambda^2}\right)^2}. \quad (\text{C.34})$$

After standardizing, we have $\mathbb{V}[\tilde{x}] = \tilde{\mu}^3/\tilde{\lambda} = 1$, so that $\omega = \sqrt{\lambda/\mu^3}$. Also, $\tilde{\lambda} = -2\tilde{\eta}_2 = -2\eta_2\omega = \lambda\omega$ and $\tilde{\mu} = \sqrt{\tilde{\eta}_2/\tilde{\eta}_1} = \sqrt{\eta_2/\eta_1}\sqrt{\omega^2} = \mu\omega$. Therefore, $\tilde{\lambda}^{\text{std}} = \lambda\sqrt{\lambda/\mu^3}$ and $\tilde{\mu}^{\text{std}} = \mu\sqrt{\lambda/\mu^3}$. Finally,

$$\begin{aligned}\tilde{L}_1^{\text{std}} &= \frac{\mu\sqrt{\mu^4\left(\frac{\lambda^2}{\mu^6}\right) + 1}}{\lambda} = \frac{\mu}{\lambda}\sqrt{\left(\frac{\lambda}{\mu}\right)^2 + 1} \\ &= \frac{\mu}{\lambda}\sqrt{\left(\frac{\mu}{\lambda}\right)^{-2} + 1},\end{aligned}\quad (\text{C.35})$$

$$\begin{aligned}\tilde{L}_2^{\text{std}} &= \sqrt{\frac{\mu^2}{\lambda^2} + \left(\frac{(2\mu + \lambda)\omega}{\mu\lambda^2\omega^3}\right)^2} = \sqrt{\frac{\mu^2}{\lambda^2} + \left(\frac{(2\mu + \lambda)}{\mu\lambda^2}\right)^2 \frac{1}{\omega^4}} \\ &= \sqrt{\frac{\mu^2}{\lambda^2} + \left(\frac{2\mu + \lambda}{\mu\lambda^2}\right)^2 \frac{\mu^6}{\lambda^2}} = \sqrt{\left(\frac{\mu}{\lambda}\right)^2 + \left(2 + \frac{\lambda}{\mu}\right)^2 \left(\frac{\mu}{\lambda}\right)^6}.\end{aligned}\quad (\text{C.36})$$

Note that both constants vanish (grow) as μ vanishes (grows).

Inverse Gamma distribution. This distribution has parameters $\alpha, \beta > 0$, sufficient statistics $T_1(x) = \log x$, $T_2(x) = 1/x$, and natural parameters $\eta_1 = -\alpha - 1$, $\eta_2 = -\beta$.

$$\begin{aligned}\mathbb{E}[T_1(x)] &= \alpha + \log \beta + \log \Gamma(\alpha) - (1 + \alpha)\psi(\alpha) \\ &= -\eta_1 - 1 + \log(-\eta_2) + \log \Gamma(-\eta_1 - 1) + \eta_1\psi(-\eta_1 - 1),\end{aligned}\quad (\text{C.37})$$

$$\begin{aligned}H_{11} &= \partial_{\eta_1} [\eta_1 + 1 - \log(-\eta_2) - \log \Gamma(-\eta_1 - 1) - \eta_1\psi(-\eta_1 - 1)] \\ &= 1 + \psi(-\eta_1 - 1) - \psi(-\eta_1 - 1) + \eta_1\psi^{(1)}(-\eta_1 - 1) \\ &= 1 - (\alpha + 1)\psi^{(1)}(\alpha),\end{aligned}\quad (\text{C.38})$$

$$\begin{aligned}\mathbb{E}[T_2(x)] &= \mathbb{E}[1/x] = \frac{\alpha}{\beta} = \frac{\eta_1 + 1}{\eta_2}, \\ H_{22} &= -\partial_{\eta_2} \frac{\eta_1 + 1}{\eta_2} = \frac{\eta_1 + 1}{\eta_2^2} = -\frac{\alpha}{\beta^2},\end{aligned}\quad (\text{C.39})$$

$$H_{12} = \frac{-1}{\eta_2} = \frac{1}{\beta}.\quad (\text{C.40})$$

Therefore,

$$L_1 = \sqrt{(1 - (\alpha + 1)\psi^{(1)}(\alpha))^2 + 1/\beta^2},\quad (\text{C.41})$$

$$L_2 = \sqrt{1/\beta^2 + \alpha^2/\beta^4}.\quad (\text{C.42})$$

After standardizing, we obtain (for $\alpha > 2$):

$$\mathbb{V}[x] = \frac{\beta^2}{(\alpha - 1)^2(\alpha - 2)} = 1 \Rightarrow \beta^2 = (\alpha - 1)^2(\alpha - 2),\quad (\text{C.43})$$

$$\tilde{L}_1^{\text{std}} = \sqrt{(1 - (\alpha + 1)\psi^{(1)}(\alpha))^2 + \frac{1}{(\alpha - 1)^2(\alpha - 2)}},\quad (\text{C.44})$$

$$\tilde{L}_2^{\text{std}} = \sqrt{\frac{1}{(\alpha - 1)^2(\alpha - 2)} + \frac{\alpha^2}{(\alpha - 1)^4(\alpha - 2)^2}}.\quad (\text{C.45})$$

Something interesting about these last two estimators is that both explode as α approaches 2, and both vanish as it gets far from it.

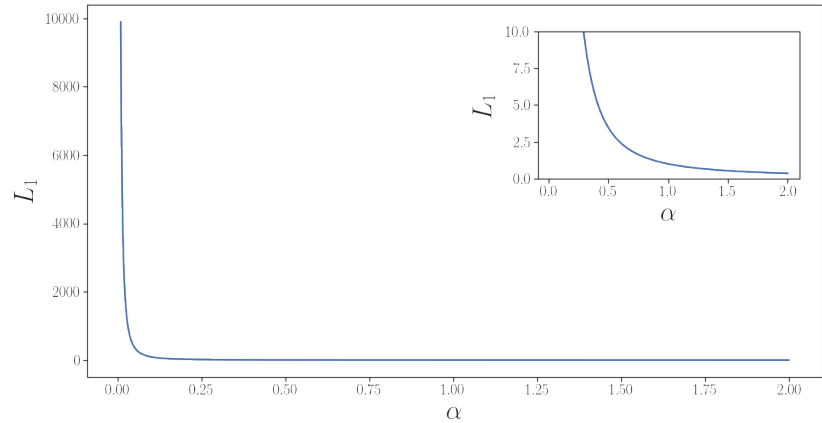


Figure C.1: Plot of L_1 for the Gamma distribution as a function of α .

C.4.1 Scale-invariant smoothness of the Gamma distribution

In section 8.4.1 we introduced the Gamma trick, which acts as an approximation of discrete distributions. One subtle detail is that we assumed discrete data to lay in the natural numbers. The reason behind it is that, for the approximation to be accurate, it is beneficial that the data is a bit far from zero. We justified the statement above by noting that the second derivative of a Gamma log-likelihood *w.r.t.* the first natural parameter, *i.e.*, H_{11} , rapidly decreases as the data moves away from zero.

As computed before in Equation C.18, one part of L_1 is scale-invariant and of the form $1 + (1 - \alpha)\psi^{(1)}(\alpha)$. Figure C.1 shows a plot of the formula above as a function of α . It is easy to observe that as the shape parameter grows the value of (our approximation to) L_1 drastically decreases.

Finally, by supposing that discrete data are natural numbers, the mode is at least one, which in practice means that the value for α is bigger than 1 (usually close to 10), thus ensuring that the value of (our approximation to) L_1 mostly depends on the scale-dependent parameter β .

C.5 Details on the experimental setup

C.5.1 Model descriptions . . . 174

C.5.2 Experimental setup . . . 176

In this section, we describe the experimental setup for each of the experiments we conduct in the main text.

C.5.1 Model descriptions

Mixture model. Here we give a deeper description on the implementation details used for the models in the experiments. See Subsection 7.3.1 for an introduction to mixture models.

To ensure that the parameters predicted by the model fulfil the domain restriction of each particular distribution, we perform the following transformation:


$$\eta_i(\bar{\eta}_i) = \begin{cases} \text{softplus}(\bar{\eta}_i) + m + \epsilon & \text{if } m < \eta_i, \\ -(\text{softplus}(\bar{\eta}_i) + M + \epsilon) & \text{if } \eta_i < M, \end{cases} \quad (\text{C.46})$$

where $0 < \epsilon \ll 1$ (in practice, $\epsilon = 1e-8$), and where we never have double-bounded constraints.

The only hyperparameter for the mixture model is the number of clusters, M . For the experiments, we use $M = 5$ for the Breast, Wine, and spam datasets, and $M = 10$ otherwise.

As mentioned in [Subsection 7.3.1](#), to learn the parameters of the discrete latent variables, we change the categorical distribution by a GumbelSoftmax distribution [81] during training, with a temperature that updates every 20 epochs as

$$\text{temp} = \max(0.001, e^{-0.001\text{epoch}}). \quad (\text{C.47})$$

[81] Jang, Gu and Poole (2017), ‘Categorical Reparameterization with Gumbel-Softmax.’ 

Matrix factorization. Similar to the mixture model, we refer to [Subsection 7.3.2](#) for an introduction to the matrix factorization model.

There some details that have to be noted. First, the variance of the local parameters is shared among instances and treated as a regular model parameter. Similarly, only the first parameter, η_1 , of each distribution is learnt using [black-box variational inference \(BBVI\)](#). The remaining parameters are treated as regular model parameters. The same transformations as in the mixture model are performed to the parameters in order to fulfil their particular domain requirements.

When it comes to experiments, the only hyperparameter is the latent size. We set it to half the number of dimensions for each dataset (before applying any trick to the data that may increase the number of dimensions).

Variational autoencoder (VAE). See [Subsection 7.3.3](#) for an introduction to VAEs. For the experiments, we follow a basic architecture with the following components:

- ▶ **Encoder.** 3-layer neural network with hyperbolic tangents as activation functions.
- ▶ **Decoder.** 4-layer neural network with ReLU as activation functions.

Other relevant implementation details are the following:

- ▶ We assume normal latent variables with a standard normal as prior.
- ▶ Hidden layers have 256 neurons.
- ▶ The latent size is set to the 75 % of the data number of dimensions (before preprocessing).
- ▶ Layers are initialized using a Xavier uniform policy.

Specific details about the encoder:

- ▶ As we have to avoid using the missing data (as we use it to evaluate the model performance), we implement an input-dropout layer as in [heterogeneous-incomplete VAE \(HI-VAE\)](#) [143].
- ▶ To guarantee a well-behaved training across all methods, we put a batch-normalization layer at the beginning of the encoder. Note that this does not interfere with the goal of this work, which is about the evaluation of the loss function.

[143] Nazabal, Olmos, Ghahramani and Valera (2020), ‘Handling incomplete heterogeneous data using VAEs.’

- In order to obtain the parameters of the variational approximation, μ_z and σ_z , we pass the output of the encoder through two linear layers, one for the mean and another for the log-scale. The latter is transformed to the scale via a softplus function.

Specific details about the decoder:

- The size of the output is set to the number of parameters to learn. Each one being transformed accordingly to fulfil their distributional restrictions, as done with the previous models.

C.5.2 Experimental setup

For the experiments in the main text, we use Adam [92] as optimizer with a learning rate of 0.001 for all models except MF, which is set to 0.01. The batch size is set to 1024 in all cases, and we train 400 epochs for the bigger datasets (letter, Adult, and Credit), while we train 2000 epochs for the intermediate ones (Wine, and spam), and 3000 epochs for the the Breast dataset. Moreover, we automate the process of choosing a likelihood as described in Section 7.1 based on basic properties of the data. Table C.3 shows a summary with the description of the datasets.

Table C.3: Types of variables per modality, and number of samples.

Dataset	Credit	Adult	Wine	spam	letter	Breast
Continuous	13	3	11	57	0	0
Poisson	1	2	1	0	16	9
Categorical	10	7	1	1	1	1
No. samples	30 000	32 000	7000	4600	20 000	700

C.6 Additional experimental results

In this section, we show complementary results from those show in the main text. First, Figure C.2 depicts the same results as Figure 8.5, but grouping by missing-values percentages instead. Second, we plot in Figures C.3 and C.4 per-modality bar plots of the normalized missing imputation error, for the Credit and letter datasets, respectively. These figures further validate the argument of `lip-gamma` not overlooking any

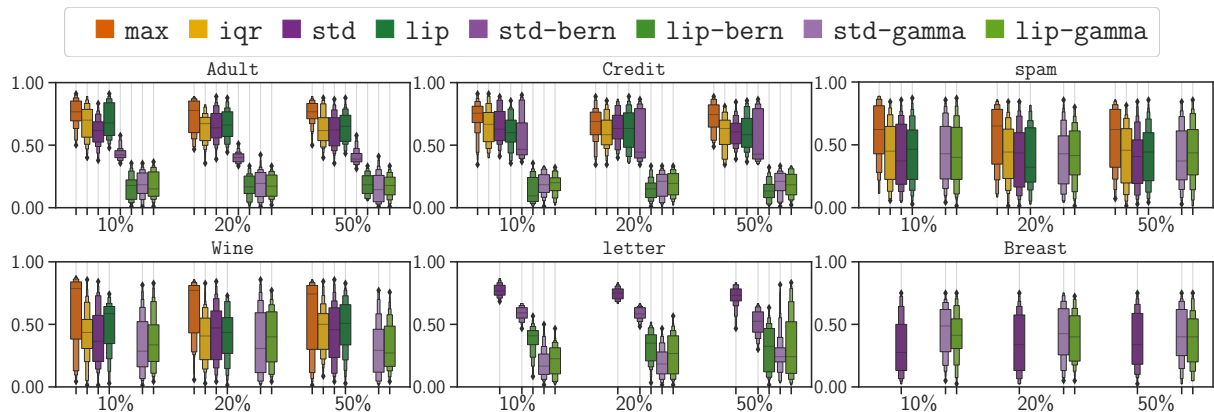


Figure C.2: Mean ranking across methods (lower is better) for different datasets and missing-values percentages on VAEs models.

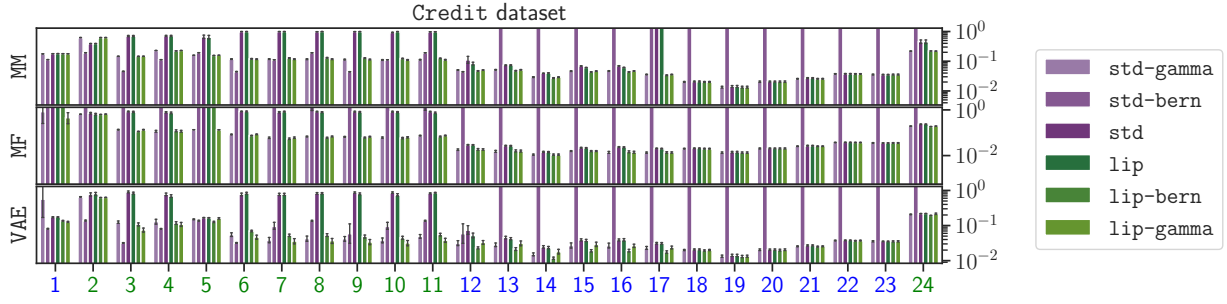


Figure C.3: Per-modality normalized missing imputation error on the Credit dataset (lower is better). Y-axis is in log-scale.

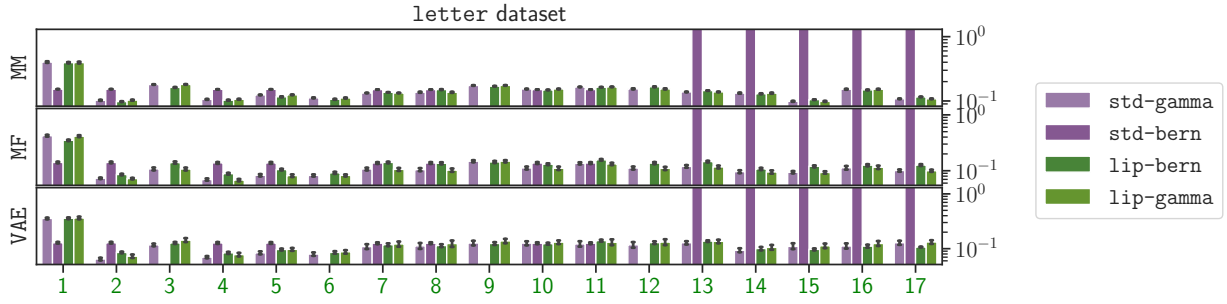


Figure C.4: Per-modality normalized missing imputation error on the letter dataset (lower is better). Y-axis is in log-scale.

variable significantly, unlike any other considered method. Finally, we present the results in tabular form, divided by type of variable (discrete *v.s.* continuous) and model type (MM, MF and VAE). Tables C.4 to C.6 show the results obtained with, respectively, 10 %, 20 %, and 50 % of missing data. Significant differences *w.r.t.* `std` are highlighted.

As discussed in Section 8.5, applying Lipschitz standardization results in the best results on imputation error across all datasets, while not overlooking any of the dimensions. We can also observe how this improvement mainly manifests on discrete random variables when the Bernoulli and Gamma tricks are applied. However, the case where properly learning the discrete distributions translates to an improvement on all dimensions can also occur, as in the Credit dataset.

Finally, there is an important aspect that qualitatively differentiates `lip-gamma` from any other method. The objective of Lipschitz-standardize every modality is to encourage an impartial learning process, and in cases with high heterogeneity, such as Credit and Adult, the downstream effects become clearer. For example, this can be readily observe by checking that `lip-gamma` never spikes in any of the bar plots or tables shown in this thesis, *i.e.*, Figures 8.7, C.3 and C.4 and Tables C.4 to C.6. Remarkably, `lip-gamma` still achieves good performance even when half the data is missing during training.

Table C.4: Missing imputation error per type of variable for different methods, datasets, and models, with a 10 % of missing data. Highlighted cells are significantly different results *w. r. t.* **std** according to a t-test with $\alpha = 0.05$.

Imputation error		Discrete modalities			Continuous modalities		
		MM	MF	VAE	MM	MF	VAE
Credit	max	0.773 ± 0.022	∞	0.720 ± 0.055	0.134 ± 0.051	0.056 ± 0.002	0.038 ± 0.002
	iqr	0.777 ± 0.025	8.486 ± 0.022	0.719 ± 0.036	0.058 ± 0.009	∞	0.044 ± 0.010
	std [†]	0.770 ± 0.028	4.448 ± 0.007	0.712 ± 0.024	0.055 ± 0.001	∞	0.042 ± 0.003
	std-bern	0.191 ± 0.002	0.196 ± 0.048	0.135 ± 0.002	0.047 ± 0.001	∞	0.037 ± 0.016
	std-gamma	0.189 ± 0.005	0.143 ± 0.003	0.123 ± 0.006	0.045 ± 0.001	0.043 ± 0.032	0.126 ± 0.280
	lip	0.777 ± 0.020	0.779 ± 0.148	0.683 ± 0.036	0.054 ± 0.001	∞	0.040 ± 0.002
	lip-bern	0.195 ± 0.004	0.133 ± 0.001	0.123 ± 0.001	0.044 ± 0.002	∞	0.029 ± 0.002
	lip-gamma	0.189 ± 0.005	0.143 ± 0.002	0.112 ± 0.002	0.045 ± 0.001	0.045 ± 0.025	0.033 ± 0.003
Adult	max	0.645 ± 0.003	0.618 ± 0.051	0.694 ± 0.037	0.089 ± 0.000	0.089 ± 0.000	0.078 ± 0.005
	iqr	0.601 ± 0.004	0.671 ± 0.038	0.702 ± 0.036	0.087 ± 0.001	0.081 ± 0.001	0.072 ± 0.003
	std [†]	0.600 ± 0.002	0.622 ± 0.052	0.706 ± 0.022	0.087 ± 0.001	0.081 ± 0.001	0.071 ± 0.002
	std-bern	0.242 ± 0.001	0.218 ± 0.012	0.152 ± 0.002	0.087 ± 0.003	0.100 ± 0.004	0.087 ± 0.004
	std-gamma	0.229 ± 0.004	0.182 ± 0.003	0.125 ± 0.003	0.087 ± 0.003	0.087 ± 0.003	0.503 ± 0.001
	lip	0.638 ± 0.003	0.651 ± 0.031	0.708 ± 0.037	0.088 ± 0.000	0.082 ± 0.000	0.072 ± 0.002
	lip-bern	0.231 ± 0.004	0.167 ± 0.002	0.131 ± 0.005	0.087 ± 0.003	0.094 ± 0.003	0.072 ± 0.002
	lip-gamma	0.228 ± 0.004	0.187 ± 0.010	0.122 ± 0.002	0.087 ± 0.003	0.094 ± 0.009	0.085 ± 0.005
Wine	max	0.110 ± 0.007	0.352 ± 0.110	0.114 ± 0.063	0.111 ± 0.001	0.274 ± 0.075	0.069 ± 0.000
	iqr	0.099 ± 0.005	0.092 ± 0.002	0.086 ± 0.008	0.093 ± 0.001	0.148 ± 0.170	0.071 ± 0.003
	std [†]	0.099 ± 0.005	0.090 ± 0.002	0.089 ± 0.008	0.093 ± 0.001	0.198 ± 0.337	0.073 ± 0.002
	std-gamma	0.099 ± 0.003	0.090 ± 0.002	0.087 ± 0.005	0.092 ± 0.001	0.208 ± 0.376	0.073 ± 0.001
	lip	0.099 ± 0.004	0.098 ± 0.004	0.089 ± 0.008	0.093 ± 0.001	0.200 ± 0.250	0.071 ± 0.001
	lip-gamma	0.099 ± 0.003	0.092 ± 0.003	0.088 ± 0.011	0.093 ± 0.001	0.250 ± 0.443	0.071 ± 0.002
spam	max	0.158 ± 0.018	0.081 ± 0.012	0.232 ± 0.122	0.054 ± 0.001	0.054 ± 0.001	∞
	iqr	0.149 ± 0.022	0.081 ± 0.007	0.086 ± 0.016	0.054 ± 0.001	0.054 ± 0.001	∞
	std [†]	0.144 ± 0.021	0.080 ± 0.007	0.094 ± 0.012	0.054 ± 0.001	0.054 ± 0.001	0.050 ± 0.002
	std-gamma	0.167 ± 0.033	0.082 ± 0.008	0.090 ± 0.010	0.054 ± 0.001	0.054 ± 0.001	0.050 ± 0.001
	lip	0.142 ± 0.021	0.083 ± 0.007	0.091 ± 0.018	0.054 ± 0.001	0.054 ± 0.001	0.051 ± 0.003
	lip-gamma	0.165 ± 0.036	0.080 ± 0.009	0.088 ± 0.020	0.054 ± 0.001	0.054 ± 0.001	0.050 ± 0.002
letter	std [†]	0.210 ± 0.008	0.190 ± 0.001	0.183 ± 0.005	-	-	-
	std-bern	0.158 ± 0.001	0.146 ± 0.008	0.119 ± 0.002	-	-	-
	std-gamma	0.150 ± 0.002	0.108 ± 0.001	0.098 ± 0.000	-	-	-
	lip-bern	0.149 ± 0.002	0.125 ± 0.000	0.114 ± 0.002	-	-	-
	lip-gamma	0.149 ± 0.002	0.106 ± 0.001	0.105 ± 0.005	-	-	-
Breast	std [†]	0.198 ± 0.005	0.212 ± 0.006	0.183 ± 0.006	-	-	-
	std-gamma	0.201 ± 0.005	0.200 ± 0.007	0.201 ± 0.007	-	-	-
	lip-gamma	0.200 ± 0.005	0.199 ± 0.006	0.198 ± 0.008	-	-	-

Table C.5: Missing imputation error per type of variable for different methods, datasets, and models, with a 20 % of missing data. Highlighted cells are significantly different results *w. r. t.* **std** according to a t-test with $\alpha = 0.05$.

Imputation error		Discrete modalities			Continuous modalities		
		MM	MF	VAE	MM	MF	VAE
Credit	max	0.805 \pm 0.018	∞	0.739 \pm 0.047	0.110 \pm 0.015	0.056 \pm 0.003	0.038 \pm 0.002
	iqr	0.803 \pm 0.021	9.938 \pm 0.015	0.689 \pm 0.016	0.054 \pm 0.001	∞	0.042 \pm 0.002
	std [†]	0.805 \pm 0.023	∞	0.707 \pm 0.034	0.055 \pm 0.001	∞	0.046 \pm 0.007
	std-bern	0.189 \pm 0.001	1.722 \pm 0.002	0.139 \pm 0.002	0.047 \pm 0.000	∞	0.034 \pm 0.001
	std-gamma	0.186 \pm 0.004	0.146 \pm 0.002	0.133 \pm 0.007	0.045 \pm 0.001	0.039 \pm 0.019	0.037 \pm 0.003
	lip	0.809 \pm 0.014	4.661 \pm 0.005	0.703 \pm 0.017	0.053 \pm 0.001	∞	0.042 \pm 0.002
	lip-bern	0.192 \pm 0.002	0.410 \pm 0.453	0.133 \pm 0.001	0.044 \pm 0.001	∞	0.030 \pm 0.001
	lip-gamma	0.185 \pm 0.004	0.147 \pm 0.001	0.124 \pm 0.002	0.046 \pm 0.001	0.035 \pm 0.007	0.033 \pm 0.003
Adult	max	0.644 \pm 0.002	0.630 \pm 0.054	0.671 \pm 0.040	0.090 \pm 0.001	0.090 \pm 0.001	0.073 \pm 0.001
	iqr	0.601 \pm 0.003	0.656 \pm 0.030	0.702 \pm 0.024	0.089 \pm 0.001	0.084 \pm 0.003	0.071 \pm 0.001
	std [†]	0.602 \pm 0.004	0.633 \pm 0.023	0.701 \pm 0.024	0.089 \pm 0.001	0.084 \pm 0.002	0.082 \pm 0.034
	std-bern	0.242 \pm 0.003	0.225 \pm 0.014	0.162 \pm 0.002	0.086 \pm 0.001	0.099 \pm 0.003	0.089 \pm 0.003
	std-gamma	0.230 \pm 0.003	0.188 \pm 0.005	0.163 \pm 0.033	0.087 \pm 0.002	0.087 \pm 0.002	∞
	lip	0.635 \pm 0.007	0.647 \pm 0.025	0.702 \pm 0.025	0.090 \pm 0.001	0.083 \pm 0.001	0.073 \pm 0.001
	lip-bern	0.231 \pm 0.002	0.180 \pm 0.004	0.149 \pm 0.002	0.087 \pm 0.002	0.094 \pm 0.001	0.075 \pm 0.003
	lip-gamma	0.230 \pm 0.002	0.193 \pm 0.006	0.142 \pm 0.002	0.087 \pm 0.002	0.092 \pm 0.004	0.082 \pm 0.002
Wine	max	0.118 \pm 0.009	0.281 \pm 0.120	0.125 \pm 0.048	0.112 \pm 0.000	0.235 \pm 0.069	0.073 \pm 0.000
	iqr	0.105 \pm 0.006	0.101 \pm 0.002	0.087 \pm 0.004	0.094 \pm 0.001	0.109 \pm 0.029	0.074 \pm 0.001
	std [†]	0.107 \pm 0.007	0.099 \pm 0.001	0.089 \pm 0.002	0.094 \pm 0.001	0.113 \pm 0.048	0.076 \pm 0.002
	std-gamma	0.101 \pm 0.006	0.099 \pm 0.002	0.092 \pm 0.004	0.093 \pm 0.001	0.121 \pm 0.078	0.076 \pm 0.002
	lip	0.106 \pm 0.006	0.103 \pm 0.007	0.091 \pm 0.013	0.094 \pm 0.001	0.156 \pm 0.092	0.074 \pm 0.001
	lip-gamma	0.103 \pm 0.006	0.099 \pm 0.002	0.093 \pm 0.006	0.094 \pm 0.001	0.285 \pm 0.518	0.074 \pm 0.001
spam	max	0.176 \pm 0.025	0.089 \pm 0.014	0.222 \pm 0.097	0.055 \pm 0.001	0.055 \pm 0.001	∞
	iqr	0.185 \pm 0.034	0.086 \pm 0.012	0.093 \pm 0.011	0.055 \pm 0.001	0.055 \pm 0.001	∞
	std [†]	0.186 \pm 0.035	0.088 \pm 0.012	0.094 \pm 0.007	0.055 \pm 0.001	0.055 \pm 0.001	0.060 \pm 0.018
	std-gamma	0.168 \pm 0.022	0.099 \pm 0.009	0.100 \pm 0.011	0.055 \pm 0.001	0.055 \pm 0.001	∞
	lip	0.182 \pm 0.034	0.089 \pm 0.011	0.100 \pm 0.014	0.055 \pm 0.001	0.055 \pm 0.001	0.055 \pm 0.011
	lip-gamma	0.169 \pm 0.030	0.096 \pm 0.010	0.106 \pm 0.024	0.055 \pm 0.001	0.055 \pm 0.001	0.051 \pm 0.001
letter	std [†]	0.210 \pm 0.007	0.193 \pm 0.000	0.188 \pm 0.004	-	-	-
	std-bern	0.159 \pm 0.001	0.157 \pm 0.009	0.126 \pm 0.002	-	-	-
	std-gamma	0.151 \pm 0.001	0.114 \pm 0.001	0.111 \pm 0.003	-	-	-
	lip-bern	0.150 \pm 0.001	0.131 \pm 0.000	0.120 \pm 0.002	-	-	-
	lip-gamma	0.151 \pm 0.001	0.112 \pm 0.001	0.123 \pm 0.003	-	-	-
Breast	std [†]	0.196 \pm 0.004	0.224 \pm 0.021	0.183 \pm 0.004	-	-	-
	std-gamma	0.196 \pm 0.006	0.200 \pm 0.002	0.201 \pm 0.004	-	-	-
	lip-gamma	0.197 \pm 0.006	0.200 \pm 0.002	0.196 \pm 0.007	-	-	-

Table C.6: Missing imputation error per type of variable for different methods, datasets, and models, with a 50 % of missing data. Highlighted cells are significantly different results *w. r. t.* **std** according to a t-test with $\alpha = 0.05$.

Imputation error		Discrete modalities			Continuous modalities		
		MM	MF	VAE	MM	MF	VAE
Credit	max	0.833 ± 0.025	∞	0.764 ± 0.051	∞	0.057 ± 0.001	0.045 ± 0.004
	iqr	0.831 ± 0.024	∞	0.709 ± 0.028	∞	∞	0.045 ± 0.003
	std [†]	0.829 ± 0.037	∞	0.709 ± 0.045	∞	∞	0.046 ± 0.003
	std-bern	0.188 ± 0.000	∞	0.157 ± 0.001	0.047 ± 0.000	∞	0.035 ± 0.001
	std-gamma	0.191 ± 0.003	0.160 ± 0.002	0.163 ± 0.007	0.046 ± 0.001	0.165 ± 0.295	0.037 ± 0.002
	lip	0.838 ± 0.037	∞	0.685 ± 0.051	∞	∞	0.046 ± 0.004
	lip-bern	0.194 ± 0.002	∞	0.154 ± 0.001	0.044 ± 0.001	∞	0.033 ± 0.001
	lip-gamma	0.192 ± 0.003	0.161 ± 0.003	0.150 ± 0.003	0.046 ± 0.001	0.080 ± 0.092	0.036 ± 0.003
Adult	max	0.642 ± 0.001	0.654 ± 0.048	0.681 ± 0.041	0.089 ± 0.000	0.089 ± 0.000	0.075 ± 0.003
	iqr	0.600 ± 0.001	0.668 ± 0.033	0.685 ± 0.039	0.088 ± 0.000	0.088 ± 0.006	0.072 ± 0.004
	std [†]	0.600 ± 0.001	0.667 ± 0.052	0.666 ± 0.057	0.088 ± 0.000	0.086 ± 0.002	0.071 ± 0.003
	std-bern	0.243 ± 0.001	0.242 ± 0.008	0.200 ± 0.001	0.086 ± 0.001	0.100 ± 0.003	0.087 ± 0.004
	std-gamma	0.239 ± 0.002	0.209 ± 0.001	0.210 ± 0.017	0.087 ± 0.000	0.090 ± 0.008	0.098 ± 0.009
	lip	0.636 ± 0.005	0.667 ± 0.031	0.690 ± 0.015	0.089 ± 0.000	0.088 ± 0.008	0.072 ± 0.001
	lip-bern	0.242 ± 0.002	0.210 ± 0.001	0.194 ± 0.002	0.087 ± 0.001	0.096 ± 0.001	0.083 ± 0.003
	lip-gamma	0.239 ± 0.002	0.212 ± 0.002	0.192 ± 0.002	0.087 ± 0.000	0.098 ± 0.003	0.081 ± 0.003
Wine	max	0.155 ± 0.020	0.264 ± 0.102	0.131 ± 0.020	0.116 ± 0.001	0.273 ± 0.038	0.087 ± 0.001
	iqr	0.122 ± 0.005	0.148 ± 0.005	0.116 ± 0.007	0.098 ± 0.002	0.132 ± 0.002	0.091 ± 0.002
	std [†]	0.122 ± 0.005	0.145 ± 0.004	0.118 ± 0.010	0.098 ± 0.002	0.131 ± 0.002	0.092 ± 0.003
	std-gamma	0.121 ± 0.006	0.134 ± 0.004	0.121 ± 0.006	0.097 ± 0.001	0.129 ± 0.001	0.091 ± 0.002
	lip	0.123 ± 0.004	0.160 ± 0.016	0.115 ± 0.010	0.098 ± 0.002	0.186 ± 0.048	0.090 ± 0.002
	lip-gamma	0.121 ± 0.005	0.140 ± 0.007	0.112 ± 0.006	0.098 ± 0.001	0.204 ± 0.053	0.088 ± 0.001
spam	max	0.183 ± 0.018	0.127 ± 0.003	0.328 ± 0.132	0.055 ± 0.000	0.055 ± 0.000	∞
	iqr	0.187 ± 0.027	0.118 ± 0.003	0.149 ± 0.017	0.055 ± 0.000	0.055 ± 0.000	∞
	std [†]	0.188 ± 0.027	0.118 ± 0.004	0.144 ± 0.011	0.055 ± 0.000	0.055 ± 0.000	∞
	std-gamma	0.195 ± 0.048	0.129 ± 0.006	0.149 ± 0.013	0.055 ± 0.000	0.055 ± 0.000	∞
	lip	0.188 ± 0.028	0.122 ± 0.004	0.142 ± 0.007	0.055 ± 0.000	0.056 ± 0.002	0.058 ± 0.012
	lip-gamma	0.191 ± 0.049	0.131 ± 0.007	0.147 ± 0.009	0.055 ± 0.000	0.056 ± 0.002	0.054 ± 0.000
letter	std [†]	0.210 ± 0.004	0.207 ± 0.000	0.192 ± 0.002	-	-	-
	std-bern	0.165 ± 0.001	0.164 ± 0.005	0.145 ± 0.001	-	-	-
	std-gamma	0.154 ± 0.001	0.145 ± 0.000	0.154 ± 0.010	-	-	-
	lip-bern	0.153 ± 0.001	0.155 ± 0.001	0.144 ± 0.003	-	-	-
	lip-gamma	0.153 ± 0.001	0.144 ± 0.000	0.166 ± 0.011	-	-	-
Breast	std [†]	0.207 ± 0.004	0.251 ± 0.008	0.201 ± 0.005	-	-	-
	std-gamma	0.208 ± 0.007	0.210 ± 0.004	0.213 ± 0.005	-	-	-
	lip-gamma	0.209 ± 0.006	0.211 ± 0.004	0.205 ± 0.006	-	-	-

Additional Material for Chapter 9

D.

D.1 General algorithm

D.1 General algorithm	181
D.2 Analysis by data type . .	182
D.3 Model descriptions . . .	184
D.4 Experimental details . .	187

In [Section 9.2](#) of the main text, we gradually introduced the impartiality block, starting with a simple example, and adapting it as we were facing different challenges. Here, we introduce the impartiality block in a general and flexible way, so that it could be easier for the reader to understand how to apply it to the tailored models explained in the main manuscript, as well as how to apply the impartiality block on their own models.

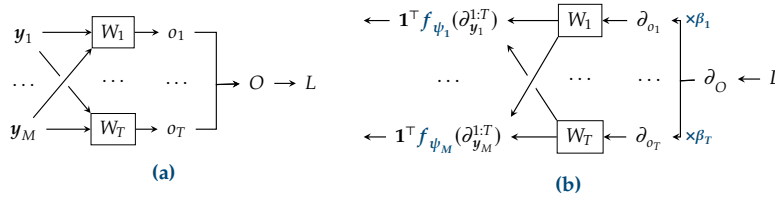


Figure D.1: General sketches of a (a) forward and (b) modified backward pass of an impartiality block. Note that here we are using the shorthand $\partial_x^{1:K}$ for a sequence of K gradients *w. r. t.* x .

[Algorithm D.1](#) shows the general formulation, and [Figure D.1](#) the forward and backward passes. To detach the block from its original presentation, we have adopted here a general notation for the different elements of the block, as well as allow for multiple entries. In this way, we would like to emphasize that the key aspect of the impartiality block is its structure, and not the variables that appear within it. That is, [Algorithm D.1](#) can be applied to any impartiality block, independently of whether the input is an intermediate feature,¹ or the features of a neural network.²

- 1: Such as in the blocks related with [LI](#), see [Section 9.2](#)
- 2: Such as in the blocks related with [DEL](#), see [Subsection 9.3.2](#).

```

1 function: ImpartialBackward(inputs:  $y_{1:M}$ , heads:  $W_{1:T}$ , output:  $O$ )
2 input: output gradient,  $\nabla_O L$ 
3 begin
4   for  $t = 1, 2, \dots, T$  do
5      $\partial_{o_t} \leftarrow \beta_t \nabla_{o_t} O \nabla_O L$            # Re-weigh the gradient at the heads
6      $\nabla_{W_t} L \leftarrow \nabla_{W_t} o_t \cdot \partial_{o_t}$        # Compute head gradients (if any)
7
8     # Per-modality gradients wrt. the common inputs
9     for  $m = 1, 2, \dots, M$  do
10       $\partial_{y_m}^t \leftarrow \nabla_{y_m} o_t \cdot \partial_{o_t}$ 
11    done
12  done
13
14  # Apply MTL methods and return the gradients
15  return  $\mathbf{1}^\top f_{\psi_1}(\partial_{y_1}^{1:T}), \mathbf{1}^\top f_{\psi_2}(\partial_{y_2}^{1:T}), \dots, \mathbf{1}^\top f_{\psi_M}(\partial_{y_M}^{1:T})$ 
16 end

```

Algorithm D.1: General impartial backward pass within the impartiality block.

With the re-formulation of the impartiality block, we provide now a summary of the impartiality blocks presented in the main text:

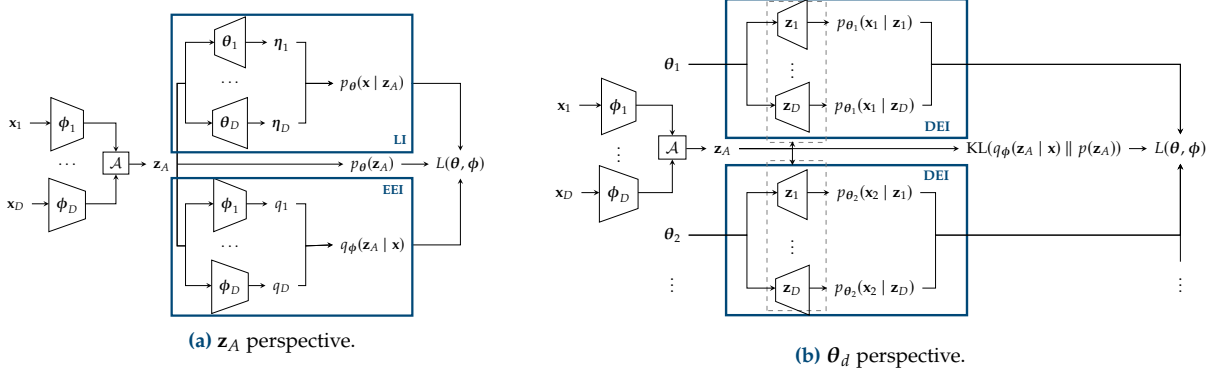


Figure D.2: Forward pass of the mixture-based models from two different points of view: (a) shows the perspective where z is the active role, where for each z_A we find two impartiality blocks; in (b) we show the perspective of the decoder parameters, where they play the active role, and now we can observe that there is an impartiality block result of evaluating each decoder in all expert samples. Here, the dashed block simply indicates that the sample z_A gets distributed into that set of blocks to be evaluated. Note that we explicitly show only 2 out of D blocks for DEI.

Model	Goal	#	Backward call
VAE	LI	1	<code>ImpartialBackward(\mathbf{y}; $\omega_{1:D}$; $\boldsymbol{\eta}$)</code>
IWAE	LI	1	<code>ImpartialBackward(\mathbf{y}; $\omega_{1:D}$; $\boldsymbol{\eta}$)</code>
DReG	LI	2	<code>ImpartialBackward(\mathbf{y}; $\omega_{1:D}$; $\boldsymbol{\eta}$)</code>
HI - VAE	LI	2	<code>ImpartialBackward(\mathbf{y}; $\omega_{1:D}$; $\boldsymbol{\eta}$)</code> <code>ImpartialBackward(\mathbf{s}; $\omega_{1:D}$; $\boldsymbol{\eta}$)</code>
Mixture	LI	M	<code>ImpartialBackward(\mathbf{z}_A; $\boldsymbol{\theta}_{1:D}$; $p_{\theta}(\mathbf{x} \mathbf{z}_A)$)</code>
Mixture	EEI	M	<code>ImpartialBackward(\mathbf{z}_A; ϕ_A; $q_{\phi}(\mathbf{z}_A \mathbf{x})$)</code>
Mixture	DEI	D	<code>ImpartialBackward($\boldsymbol{\theta}_d$; \mathbf{z}_A; L)</code>

Unfortunately, we could not show the third source of impartiality blocks for the mixture-base models in Subsection 9.4.2. We present the complete computational graph for this type of models in Figure D.2, with two different ways of drawing the computational graph that unveil all the impartiality blocks.

D.2 Analysis by data type

In this section, we attempt to mathematically sketch the results obtained in Table 9.3 of the main text. To this end, we are going to compute the expected value of the squared norm of the gradient *w.r.t.* each of the likelihoods, *i.e.*, we estimate $\mathbb{E}_{\mathbf{x}_d}[\|\nabla_{\boldsymbol{\eta}_d} \log p_{\theta_d}(\mathbf{x}_d; \boldsymbol{\eta}_d)\|^2]$. We make the extra assumption that we evaluate on the parameters that explain the random variable (*R. V.*), *i.e.*, that \mathbf{x}_d follows the density $p_{\theta_d}(\mathbf{x}_d; \boldsymbol{\eta}_d)$. We break down this informal proof in two steps:

Computing the expected squared norms. We first take advantage that all considered distributions are part of the exponential distribution,³ and find a general formula valid for all of them. As a reminder, the exponential family, with natural parameters $\boldsymbol{\eta} \in \mathbb{R}^I$, is a family of distributions which is characterized by having a density function of the form

$$p(\mathbf{x}; \boldsymbol{\eta}) = h(\mathbf{x}) \exp [\boldsymbol{\eta}^\top T(\mathbf{x}) - A(\boldsymbol{\eta})]. \quad (\text{D.1})$$

3: Introduced in Section 7.2.

Using this general expression, we can compute the value of interest:

$$\ln p(x; \boldsymbol{\eta}) = \mathbf{T}(x)^\top \boldsymbol{\eta} - A(\boldsymbol{\eta}) + C(x), \quad (\text{D.2})$$

$$\partial_{\eta_i} \ln p(x; \boldsymbol{\eta}) = T_i(x) - \partial_{\eta_i} A(\boldsymbol{\eta}) = T_i(x) - \mathbb{E}[T_i(x)], \quad (\text{D.3})$$

and therefore

$$\mathbb{E}_{x_d} [\|\nabla_{\boldsymbol{\eta}_d} \log p_{\boldsymbol{\theta}_d}(x_d; \boldsymbol{\eta}_d)\|^2] = \sum_{i=1}^I \mathbb{E}_{x_d} [(T_i(x) - \mathbb{E}[T_i(x)])^2], \quad (\text{D.4})$$

where we have used the fact that $\partial_{\eta_i} A(\boldsymbol{\eta}) = \mathbb{E}[T_i(\mathbf{x})]$.

We can now simply plug in the specific values for the sufficient statistics for each of the likelihoods:

	$\partial_{\eta_1} \ln p(x; \boldsymbol{\eta})$	$\partial_{\eta_2} \ln p(x; \boldsymbol{\eta})$	$\mathbb{E}[\ \nabla_{\boldsymbol{\eta}} \ln p(\mathbf{x}; \boldsymbol{\eta})\ ^2]$
Normal	$x - \mu$	$x^2 - (\mu^2 + \sigma^2)$	$\sigma^2 + 4\mu^2\sigma^2$
Log-normal	$\ln x - \mu$	$(\ln x)^2 - (\mu^2 + \sigma^2)$	$\sigma^2 + 4\mu^2\sigma^2$
Poisson	$x - \lambda$		λ
Categorical	$\mathbf{1}_{\{x=i\}} - \pi_i$		$\sum_{i=1}^I \mathbb{E}[(\mathbf{1}_{\{x=i\}} - \pi_i)^2]$

For each likelihood above, we have used the usual notation for their traditional parameters. Moreover, note that the moments are not well-defined for the categorical distribution. Instead, we just compute the average over the entire dataset.

Bounding the norms under our data pipeline. Once that we have rough estimates of the expected squared norms of the gradients for each likelihood, we need to connect them with the experiments in [Subsection 9.4.1](#). Specifically, we need to take into account the preprocessing and the datasets themselves. We use the `Adult` dataset as an example:

- **Normal.** We standardize the data, such that $\mu = 0$ and $\sigma = 1$. Therefore, $\mathbb{E}[\|\nabla_{\boldsymbol{\eta}} \ln p(\mathbf{x}; \boldsymbol{\eta})\|^2] \approx 1$.
- **Log-normal.** We standardize (only scaling) in log-space. In `Adult`, the biggest log-normal distribution lies in the range $[15, 22]$, and thus $\mu \approx 1$, $\sigma < 1$, and $\mathbb{E}[\|\nabla_{\boldsymbol{\eta}} \ln p(\mathbf{x}; \boldsymbol{\eta})\|^2] \approx 1$.
- **Poisson.** Since it is discrete, we do not standardize. Count data can be quite large, reaching in `Adult` a maximum value of 100. Thus, $\mathbb{E}[\|\nabla_{\boldsymbol{\eta}} \ln p(\mathbf{x}; \boldsymbol{\eta})\|^2] \gg 1$ in `Adult`.
- **Categorical.** Again, we do not standardize, as it is discrete. However, it is relatively simple to see that $0 \leq \mathbb{E}[\|\nabla_{\boldsymbol{\eta}} \ln p(\mathbf{x}; \boldsymbol{\eta})\|^2] \leq I$ since $0 \leq \pi_i \leq 1$ and $\mathbf{1}_{\{x=i\}} \in \{0, 1\}$. However, the number of classes I is usually small, and the gradient is bounded by I during the entire training, while in the other cases this is not the case.

Therefore, using these rough calculations, we can expect the values of $\mathbb{E}[\|\nabla_{\boldsymbol{\eta}} \ln p(\mathbf{x}; \boldsymbol{\eta})\|^2]$ to follow the following order:

$$\text{Categorical} < \text{Normal} \approx \text{Log-normal} \ll \text{Poisson}.$$

Now, if we compute the difference between normalized errors in [Table 9.3](#), we obtain that our approach improves the error across types in an order similar to the reverse of the one above:

	Cat	$\log \mathcal{N}$	\mathcal{N}	Poiss
Vanilla	0.158	0.064	0.041	0.058
Ours	0.065	0.057	0.039	0.083
Improv.	0.092 >	0.008 \approx	0.002 >	-0.025

D.3 Model descriptions

D.3.1 VAE	184
D.3.2 IWAE	185
D.3.3 DReG	185
D.3.4 HVAE	186
D.3.5 Mixture models	186

In this section we explain the implementation details for each model, please refer to the original papers for a detailed explanation of each model. We follow the notation described in Table D.2.

D.3.1 Variational autoencoder (VAE)

We implement the original **variational autoencoder (VAE)** [93] assuming the following probabilistic model:

$$\begin{aligned}
 \text{Prior:} \quad & p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I}), \\
 \text{Likelihood:} \quad & p_{\theta}(\mathbf{x} | \mathbf{z}) = \prod_d p_d(\mathbf{x}_d | \boldsymbol{\eta}_d(\mathbf{z}; \boldsymbol{\theta})), \\
 \text{Posterior approx.:} \quad & q_{\phi}(\mathbf{z} | \mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}; \boldsymbol{\phi}), \boldsymbol{\sigma}(\mathbf{x}; \boldsymbol{\phi})).
 \end{aligned}$$

Here $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are modelled by the encoder, and all $\boldsymbol{\eta}_d$ are jointly modelled by the decoder.

These two neural networks are of the following form:

Encoder: [Dropout - 10 %] [BN] [Dense - h] [Tanh] [Dense - h] [Tanh] [Dense - h] [Tanh] [Dense - $2d$]
Decoder: [Dense - h] [ReLU] [Dense - h] [ReLU] [Dense - h] [ReLU] [Dense - D']

4: E.g., the variance has to be positive.

Additionally, we make sure that each parameter fulfils its distributional constraints⁴ by passing it through a softplus function when necessary. It is also important to note that, while we parametrize the latent space using the mean and standard deviation, we parametrize the parameters of the likelihoods using their natural parameters.

Table D.2: Notation to describe the architectures in the appendix.

Notation	Description
D	Number of features.
D'	Total number of likelihood parameters.
d	Latent size.
h	Hidden size.
[Dense - h]	Linear layer with output of size h .
[Conv - k - s - p]	Convolutional layer with kernel size k , stride s and padding p .
[ConvT - k - s - p]	Transposed convolutional layer with kernel size k , stride s and padding p .
[Dropout - 10 %]	Dropout Srivastava et al. [192] with 10 % of dropping probability.
[ReLU]	Rectified linear unit activation function.
[Tanh]	Hyperbolic tangent activation function.
[Sigmoid]	Sigmoid activation function.


Loss. We use the negative **evidence lower bound (ELBO)** as training loss:

$$\text{ELBO}(\mathbf{x}, \theta, \phi) := \mathbb{E}_{q_\phi} [\log p_\theta(\mathbf{x} | \mathbf{z})] - \text{KL}(q_\phi(\mathbf{z}) \parallel p(\mathbf{z})). \quad (\text{D.5})$$

Imputation. We impute data by taking the modes of $q_\phi(\mathbf{z} | \mathbf{x})$ and $p_d(\mathbf{x}_d; \eta_d(\mathbf{z}; \theta))$.

D.3.2 Importance weighted autoencoder (IWAE)

Importance weighted autoencoders (IWAEs) [15] differ from VAEs only on the training loss.

[15] Burda, Grosse and Salakhutdinov (2016), ‘Importance Weighted Autoencoders.’ 


Loss. Instead of maximizing the ELBO, IWAE maximizes a tighter loss that makes use of K *i.i.d.* samples from \mathbf{z} :


$$\text{IWAE}(\mathbf{x}, \theta, \phi) := \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_K \sim q_\phi} \left[\log \frac{1}{K} \sum_k \frac{p_\theta(\mathbf{x} | \mathbf{z}_k) p(\mathbf{z}_k)}{q_\phi(\mathbf{z}_k | \mathbf{x})} \right]. \quad (\text{D.6})$$

For all the results shown in **Table 9.2** we set the number of importance samples to $K = 20$.

D.3.3 Doubly reparametrized gradient estimator (DReG)

Rainforth et al. [158] showed that the gradient estimators produced by IWAE have some undesired properties that could hamper properly learning the encoder parameters. Later, Tucker et al. [202] provided a simple way of solving these issues using the reparametrization trick a second time. Again, the model is identical to a VAE, but which is optimized with two different losses: one for the encoder, and one for the decoder. We use $K = 20$ importance samples.

[158] Rainforth, Kosiorek, Le, Maddison, Igl, Wood and Teh (2018), ‘Tighter Variational Bounds are Not Necessarily Better.’ 

[202] Tucker, Lawson, Gu and Maddison (2019), ‘Doubly Reparameterized Gradient Estimators for Monte Carlo Objectives.’ 

Encoder loss. For one importance sample \mathbf{z}_k , let us define

$$\omega_k := \frac{p_\theta(\mathbf{x} | \mathbf{z}_k) p(\mathbf{z}_k)}{q_\phi(\mathbf{z}_k)}, \quad \text{and} \quad \tilde{\omega}_k := \frac{\omega_k}{\sum_i \omega_i}, \quad (\text{D.7})$$

such that $\sum_k \tilde{\omega}_k = 1$. Then, the loss is defined as

$$\text{DReG}^{\text{enc}}(\mathbf{x}, \theta, \phi) := \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_K \sim q_\phi} \left[\sum_k \tilde{\omega}_k^2 \log \omega_k \right], \quad (\text{D.8})$$

where we consider $\tilde{\omega}_k$ to be a constant value (*i.e.*, we do not back-propagate through it), and we compute the derivative *w.r.t.* ϕ only through \mathbf{z} (*i.e.*, we do not compute the partial derivative *w.r.t.* ϕ).

Decoder loss. Similarly, we optimize the parameters of the decoder by maximizing the following loss (same assumptions on $\tilde{\omega}_k$ and ϕ):

$$\text{DReG}^{\text{dec}}(\mathbf{x}, \theta, \phi) := \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_K \sim q_\phi} \left[\sum_k \tilde{\omega}_k \log \omega_k \right]. \quad (\text{D.9})$$

[143] Nazabal, Olmos, Ghahramani and Valera (2020), ‘Handling incomplete heterogeneous data using VAEs.’

D.3.4 HIVAE

We have faithfully re-implemented the original version of **heterogeneous-incomplete VAE (HI-VAE)** [143], using the same architecture and parameter count, as well as implementing the proposed normalization and denormalization layers. Therefore, results between HI-VAE and the rest of the models in Table 9.2 are not completely comparable. We refer the reader to Subsection 7.3.3 for an introduction to the model.

We fix the size of each latent variable to 10, with a hidden size of $h = 5D$, just as in the original paper.

Loss. We maximize the ELBO as originally proposed:

$$\text{ELBO}(\mathbf{x}, p_{\theta}, q_{\phi}) := \mathbb{E}_{\mathbf{z}, \mathbf{s} \sim q_{\phi}} \left[\log \frac{p_{\theta}(\mathbf{x}, \mathbf{z}, \mathbf{s})}{q_{\phi}(\mathbf{z}, \mathbf{s})} \right]. \quad (\text{D.10})$$

D.3.5 Mixture-based VAEs

For the mixture-based models, we have followed the same architecture and setups as the ones used by Sutter et al. [199]. When it comes to different models, we only have changed the way we sample the modalities \mathbf{z}_A by changing the selection of \mathcal{A} , but the architectures are equal to the ones used in previous literature.

Therefore, we here describe the architecture for all the models at once, as they differ on the loss function and the experts, which does not modify the underlying network. We assume the following probabilistic model for the MNIST-SVHN-Text experiments:

$$\begin{aligned} \text{Prior:} \quad & p(\mathbf{z}) = \mathcal{N}(0, I), \\ & \text{Laplace}(\mathbf{x}_M \mid \mu(\mathbf{z}; \boldsymbol{\theta}), 0.75) \\ \text{Likelihood:} \quad & p_{\theta}(\mathbf{x} \mid \mathbf{z}) = \text{Laplace}(\mathbf{x}_S \mid \mu(\mathbf{z}; \boldsymbol{\theta}), 0.75), \\ & \text{Cat}(\mathbf{x}_T \mid \pi(\mathbf{z}; \boldsymbol{\theta})) \\ \text{Posterior approx.:} \quad & q_{\phi}(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}(\mu(\mathbf{x}; \boldsymbol{\phi}), \sigma(\mathbf{x}; \boldsymbol{\phi})), \end{aligned}$$

where variables are properly transformed to meet their constraints. We consider the following encoders and decoders for each modality:

MNIST:

Encoder: [Dense- h] [ReLU] [Dense- h] [ReLU] [Dense- $2d$]
Decoder: [Dense- h] [ReLU] [Dense- h] [ReLU] [Dense- $2D$] [Sigmoid]

SVHN:

Encoder: [Conv-4-2-1] [ReLU] [Conv-4-2-1] [ReLU] [Conv-4-2-1]
[ReLU] [Conv-4-1-0]
Decoder: [ConvT-4-1-0] [ReLU] [ConvT-4-2-1] [ReLU] [ConvT-4-2-1]
[ReLU] [Conv-4-2-1] [Sigmoid]

where the last convolutional layer of the encoder is repeated twice, one for each parameter of the posterior approximation.

Text:

Encoder: [Conv-1-1-0] [ReLU] [Conv-4-2-1] [ReLU] [Conv-4-2-0]
[ReLU] [Dense- $2d$]
Decoder: [Dense- D] [ConvT-4-1-0] [ReLU] [ConvT-4-2-1]
[ReLU] [Conv-1-1-0]

Experimental setup. For each experiment, we train the model for 30 epochs and a batch size of 128. We use AMSGrad [161] with a learning rate of 0.001. Regarding the variational loss, we use $K = 30$ importance samples for all losses (when using the ELBO, we instead use those samples for the Monte Carlo estimator of the outer expectation). For evaluation, we take the model parameters with the highest validation error (10 % of the training data) during training, and report all the metrics with respect to a test set.

D.4 Experimental details

D.4.1 Heterogeneous exps. . . 187

D.4.2 Multimodal exps. . . . 190

D.4.3 Additional results . . . 191

D.4.1 Heterogeneous experiments

Dataset descriptions

Likelihood selection. We choose the likelihood based on the properties of each modality, as we described in Section 7.1.

Datasets. For the experiments shown in Subsection 9.4.1, we use 12 different heterogeneous and homogeneous datasets. First, we took Adult, Credit, Wine, Bank, ElNino, Magic, and BooNE datasets from the UCI repository [48]. Then, we included from the R package datasets [157] the following datasets: Diam., IMDB, HI, rwm5yr, and labour.

Table D.3 provides the statistics per dataset in terms of sizes and number of likelihoods. It is important to remark that the IMDB and Adult datasets contain NaNs values (each only in two of the features). We replace them by non-NaN values and ignore them during training and evaluation using boolean masks.

Preprocessing. When parsing the dataset, we centre all real-valued features. We further standardize real-valued features, computing their (training) standard deviation and dividing the data by this quantity. We also divide by the standard deviation for positive real-valued features

Dataset	N	D	Real	Positive	Count	Categorical
Adult	32561	12	0	3	1	7
Credit	30000	24	6	7	1	10
Wine	6497	13	0	11	1	1
Diam.	53940	10	7	0	0	3
Bank	41188	21	10	0	0	11
IMDB	28819	23	4	1	10	8
HI	22272	12	5	1	0	6
rwm5yr	19609	16	0	2	3	11
labour	15992	9	3	0	2	4
ElNino	178080	12	12	0	0	0
Magic	19020	11	11	0	0	0
BooNE	130065	43	43	0	0	0

Table D.3: Datasets description. The first two columns describe number of instances, N , and number of features, D . The next columns describe the number of data types per dataset. Note that the last three datasets are homogeneous, and thus only have real variables.

[143] Nazabal, Olmos, Ghahramani and Valera (2020), ‘Handling incomplete heterogeneous data using VAEs.’

(but in the log-space, as we assume a log-normal likelihood). These last two steps are omitted for HI-VAE, since it uses its own normalization layer as described by Nazabal et al. [143]. We also treat non-negative as positive real-valued features by adding a negligible value of $1e-20$. Finally, we make sure that the support of count, binary, and categorical features are in accordance to that of the library used during implementation by removing their minimum value in the case of binary and categorical features, and 1 in the case of count features.

Additionally, we performed some extra preprocessing to the IMDB and Bank datasets. In the IMDB dataset, there are ten features that contain rating percentages of users to the movies, ranging from 0 to 100, at intervals of 0.5. We convert each of them into discrete features starting from one by performing $x'_d = 2x_d + 1$ to each of these features, treating them afterwards as count data. As for the Bank dataset, we remove the uninformative dimension 12th as a data cleaning step.

Experimental settings

We train all experiments using Adam as optimizer, with a learning rate of 0.001 for all models, a batch size of 128, and we train for 400 epochs with the all datasets (except for Wine with 2000 epochs). For HI-VAE, we set the batch size to 1000 and the number of epochs to 2000 as in the original paper. We randomly split the data into training (70 %), validation (10 %), and testing (20 %).

We set the latent size of \mathbf{z} , to 50 % of the number of features of the dataset, and the hidden size of each layer to 50 for all the experiments, except for those of the Bank dataset which are set to 100.

Metric. For numerical features (real, positive, and count data) we compute the normalized root mean squared error:

$$\text{err}(d) = \frac{1}{D} \frac{\|x_d - \tilde{x}_d\|_2}{\max(x_d) - \min(x_d)}, \quad (\text{D.11})$$

where \tilde{x} is the model prediction. For the case of nominal features (categorical and binary data) we use the error rate as reconstruction error:

$$\text{err}(d) = \frac{1}{N} \sum_{n=1}^N \mathbf{1}_{\{x_{nd} \neq \tilde{x}_{nd}\}}. \quad (\text{D.12})$$

The final metric shown in Table 9.2 is the average across dimensions, $\text{err} = \frac{1}{D} \sum_d \text{err}(d)$.

Model selection. In order to make fair comparisons, for each model and dataset we first tuned the hyperparameters (e.g., hidden/latent/batch size, number of epochs) for the Vanilla implementations (i.e., without modifying the backward pass). To this end, we ran grid searches and averaged the validation metric over 5 random seeds, just as in Table 9.2, choosing the set of hyperparameters that performed the best in terms of reconstruction error during validation. Note that all these hyperparameters (including optimization hyperparameters such as learning rate) are shared across all methods of the same setting. Additionally, we verified

Dataset	VAE-ELBO	VAE-IWAE	VAE-DReG	HI - VAE
Adult	IMTL - G	IMTL - G	IMTL - G-PG	GN-PG($\alpha = 1$)
Credit	IMTL - G	IMTL - G-GD	IMTL - G-GD	GN($\alpha = 1$)
Wine	GN($\alpha = 0$)	GN-PG($\alpha = 0$)	GN($\alpha = 0$)	GN($\alpha = 0$)
Diam.	IMTL - G	IMTL - G	IMTL - G-PG	GN($\alpha = 0$)
Bank	GN($\alpha = 0$)	GN-GD($\alpha = 0$)	GN($\alpha = 0$)	MGDA-PG
IMDB	GN-GD($\alpha = 0$)	GN($\alpha = 0$)	GN-PG($\alpha = 0$)	GN-PG($\alpha = 0$)
HI	GN-GD($\alpha = 0$)	GN($\alpha = 0$)	GN-PG($\alpha = 0$)	MGDA
rwm5yr	GN($\alpha = 1$)	GN-GD($\alpha = 1$)	GN($\alpha = 1$)	MGDA-PG
labour	GN($\alpha = 0$)	GN($\alpha = 1$)	GN-PG($\alpha = 0$)	GN($\alpha = 0$)
ElNino	IMTL - G	IMTL - G-PG	IMTL - G-GD	GN($\alpha = 0$)
Magic	GN($\alpha = 1$)	IMTL - G	GN($\alpha = 1$)	IMTL - G
BooNE	IMTL - G-PG	IMTL - G	GN-PG($\alpha = 0$)	MGDA-PG

Table D.4: The best MTL methods chosen by cross-validation. GN, GD, PG, and MGDA stand for GradNorm, GradDrop, PCGrad, and MGDA-UB, respectively.

that the vanilla models were performing well by visually inspecting the marginal reconstructions.

Selecting the MTL algorithm. For the heterogeneous experiments we trained all the possible combinations between the following magnitude-aware algorithms: **i)** doing nothing; **ii)** GradNorm [26]; **iii)** MGDA-UB [182]; **iv)** IMTL-G [112]; and direction-aware algorithms: **i)** doing nothing; **ii)** GradDrop [27]; **iii)** PCGrad [227]. This amounts to a total of 12 combinations, plus the hyperparameters of the algorithms. In this case, we only tune the α parameter from GradNorm between the values zero and one. Then, similar to model selection, we chose the best algorithm by averaging over 5 random seeds and taking the combination of methods that performed the best in terms of reconstruction error in the validation set, see Table D.4. In general, it was enough to focus on the median to select the best combination. However, some combinations had outliers, and we chose those having a good balance between median, mean, and standard deviation.

Statistical test. In order to compare the performance of the proposed method with the baseline, we employ the corrected paired t-test [139]. The usual paired t-test assumes that the data used to perform the test is independently sampled, which usually does not hold in the machine learning as we sample the training and test data from the same distribution. As a consequence, paired t-test might suggest statistical significance between the compared models, whereas there is no such significance (type I error). Corrected paired t-test considers the dependency of the sampled data, correcting the variance of the differences of the paired samples in the two testing models.

Data generation. To generate the data for the experiments in Subsection 9.4.1, we followed the same approach as Ghosh et al. [65] and made use of post-hoc **gaussian mixture models (GMMs)** to approximate the aggregated posterior, $\mathbb{E}_{\mathbf{x}}[q_{\phi}(\mathbf{z} | \mathbf{x})]$. After training the VAEs, we use the latent space generated from the training data and fit a GMM (with 100 components) on that data. Next, we use this GMM to sample a dataset with as many samples as the test data.


[65] Ghosh, Sajjadi, Vergari, Black and Schölkopf (2020), ‘From Variational to Deterministic Autoencoders.’ 

Table D.5: Test reconstruction errors (mean and standard deviation) of different models and losses for the baseline and our framework.

Dataset	Method	VAE-ELBO	VAE-IWAE	VAE-DReG	HI - VAE
Adult	Vanilla	0.21 ± 0.01	0.22 ± 0.02	0.24 ± 0.01	0.13 ± 0.00
	Ours	0.11 ± 0.02	0.12 ± 0.02	0.19 ± 0.08	0.09 ± 0.02
Credit	Vanilla	0.13 ± 0.00	0.14 ± 0.02	0.14 ± 0.01	0.15 ± 0.09
	Ours	0.04 ± 0.00	0.05 ± 0.01	0.08 ± 0.01	0.06 ± 0.01
Wine	Vanilla	0.09 ± 0.00	0.08 ± 0.00	0.08 ± 0.00	0.13 ± 0.01
	Ours	0.07 ± 0.01	0.07 ± 0.00	0.07 ± 0.00	0.11 ± 0.02
Diam.	Vanilla	0.19 ± 0.01	0.18 ± 0.01	0.18 ± 0.00	0.11 ± 0.02
	Ours	0.13 ± 0.02	0.12 ± 0.01	0.14 ± 0.01	0.01 ± 0.01
Bank	Vanilla	0.20 ± 0.00	0.20 ± 0.00	0.19 ± 0.00	0.13 ± 0.02
	Ours	0.04 ± 0.00	0.10 ± 0.05	0.11 ± 0.04	0.10 ± 0.01
IMDB	Vanilla	0.09 ± 0.02	0.10 ± 0.02	0.10 ± 0.02	0.08 ± 0.00
	Ours	0.05 ± 0.04	0.05 ± 0.04	0.06 ± 0.04	0.10 ± 0.09
HI	Vanilla	0.17 ± 0.01	0.16 ± 0.00	0.15 ± 0.00	0.11 ± 0.00
	Ours	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.11 ± 0.01
rwm5yr	Vanilla	0.11 ± 0.01	0.09 ± 0.00	0.10 ± 0.00	0.04 ± 0.01
	Ours	0.03 ± 0.00	0.03 ± 0.01	0.03 ± 0.00	0.02 ± 0.00
labour	Vanilla	0.11 ± 0.00	0.10 ± 0.00	0.10 ± 0.00	0.10 ± 0.00
	Ours	0.06 ± 0.00	0.07 ± 0.00	0.08 ± 0.01	0.07 ± 0.00
ElNino	Vanilla	0.10 ± 0.01	0.09 ± 0.00	0.08 ± 0.00	0.10 ± 0.01
	Ours	0.07 ± 0.01	0.06 ± 0.01	0.07 ± 0.00	0.02 ± 0.00
Magic	Vanilla	0.06 ± 0.00	0.05 ± 0.00	0.05 ± 0.00	0.06 ± 0.00
	Ours	0.06 ± 0.00	0.05 ± 0.00	0.05 ± 0.00	0.03 ± 0.00
BooNE	Vanilla	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00
	Ours	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00

Additional experimental results

In addition to the results presented in the main text, we present in [Table D.5](#) the same table as [Table 9.2](#) but showing also the standard deviation of the results. Moreover, we show in [Figures D.3](#) and [D.4](#) the complete pair plots for the HI and labour datasets.

D.4.2 Multimodal experiments

Experiment details

For the multimodal experiments on MNIST-SVHN-Text, we have followed the same setup (including hyperparameters) as Sutter et al. [199]. We differ from their setups in that, in order to provide a fair comparison between losses, we always employ $K = 30$ samples from \mathbf{z} , whether they are used as importance samples (IWAE, SIWAE) or used for the Monte-Carlo approximation of the expected value *w.r.t.* \mathbf{z} . Also, we do model selection using a validation dataset (10 % of the training data), and use a test set to obtain all the results presented in this work. Following Shi et al. [185], we use the Sticking-The-Landing estimator (STL) [164] for all losses. In short, this estimator simply omits the partial derivatives of the posterior approximation *w.r.t.* the encoder parameters. Note that Shi et al. [185] did not mention this estimator, but they rather talk about the

[199] Sutter, Daunhawer and Vogt (2021), ‘Generalized Multimodal ELBO.’ [↗](#)

DReG loss [202]. However, due to a bug in their code, they effectively compute the STL estimator in their experiments.

Selecting MTL algorithm. Since the number of impartiality blocks is large, and the training times are considerably longer than for the heterogeneous experiments, here we keep performing cross-validation, but this time we substitute grid-selection by hand-picked hyperparameters options that we observed to perform better than others (for example, we replaced IMTL-G [112] by CAGrad [111], as it was really clear by looking at the logs that IMTL-G was not working at all). Instead of looking for a specific algorithm for each of the impartiality blocks, we assume the same algorithm for all of them (same hyperparameters, but different parameters) and only cross-validate by using the modified backward pass on the blocks associated with the different goals in an incremental way (*i.e.*, as presented in Table 9.7 in the main text).

Choosing the best algorithm in the multimodal setup is more complicated, as we care about different metrics (coherence and latent classification) at different levels (self and cross metrics), for each modality. We group all metrics in metric-type pairs (*e.g.*, latent-classification-self), and within each group, we group them by the expert/modality they are testing (*e.g.*, cross latent classification for the first expert tests all other latent samples in the classifier of the first expert). For each metric, we compute a value $I(x, y)$, where x is the value obtained by the algorithm, and y the value obtained by the baseline, and take the average of each sequence recursively until obtaining a single number. We use the relative improvement $I(x, y) = \text{avg } \Delta(x, y) = \frac{x-y}{y}$ to compare the different metrics, choosing the method that obtains the best improvement, averaged across experts/modalities and metrics. For MVAE, we noticed that the metrics tend to oscillate and there are important trade-offs in performance. Therefore, for this model we adopt a more conservative approach and use $I(x, y) = \mathbf{1}_{x \geq y}$, to choose the algorithm that, on average, improves the most number of metrics.

D.4.3 Additional experimental results

In this section, we have included the complete results for the experiments with the MNIST-SVHN-Text dataset. Specifically, we present: the reconstruction coherence results for all the three losses (Table D.6); the self and cross coherence results in tabular form for the three losses (Tables D.7 to D.9), including extra information like the training times, the specific MTL algorithms used, and the goals for which we apply them; and the log-likelihoods conditioned on different modalities (Tables D.10 to D.12), showing standard deviations as space permits, thus showing the high variance that the vanilla approach shows at times (for example, MoPoE in Table D.12). Finally, we present the parallel coordinate plots for the three models (Figure D.5).

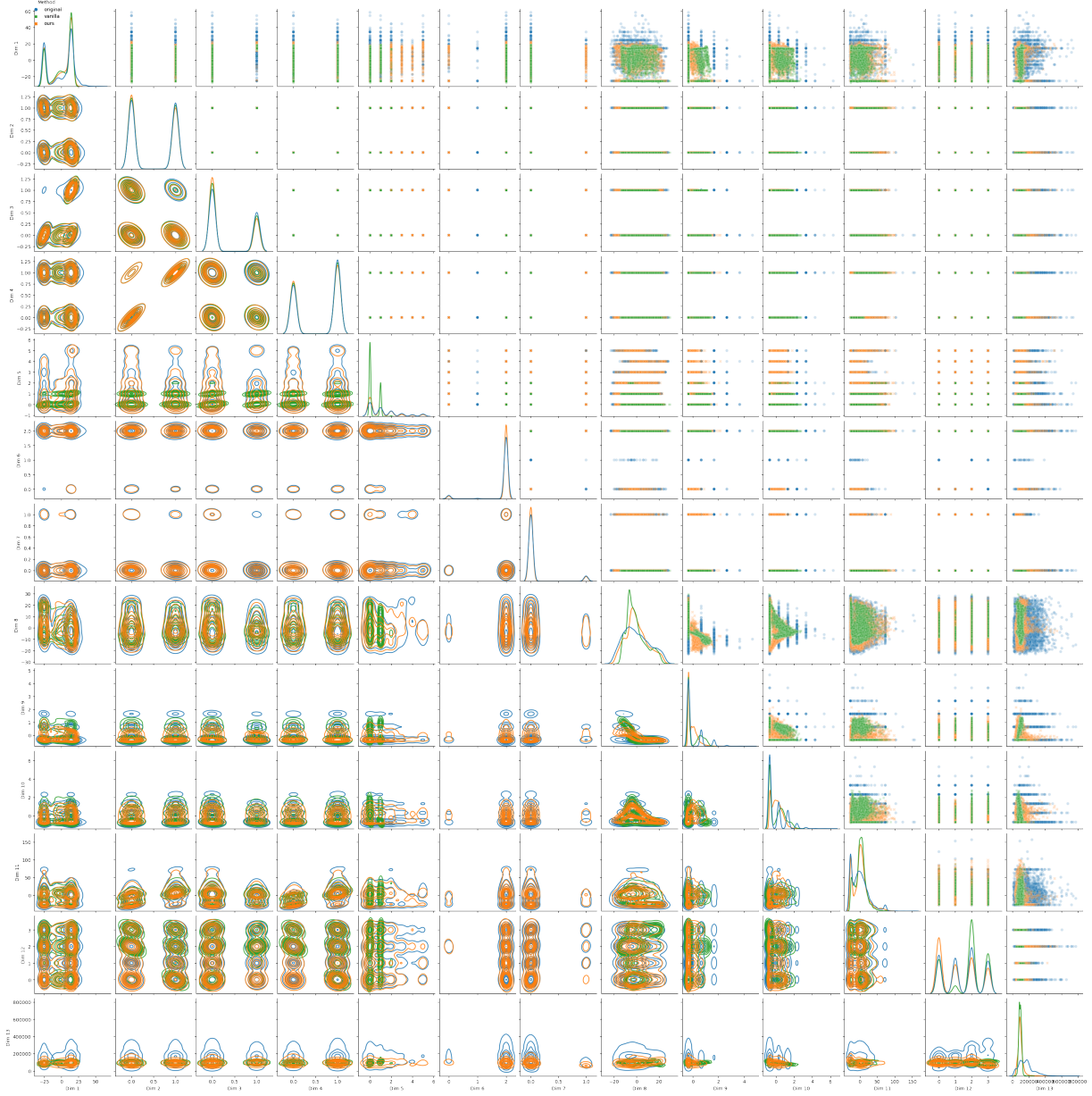


Figure D.3: Pair plot of all the dimensions of HI, generated from different VAEs. Diagonal show the marginals, upper-diagonals scatter plots, and lower-diagonals kernel density estimates. The VAE trained with our approach is able to generate faithful samples.

Table D.6: Reconstruction coherence ($A \in \{M, S, T\}$) for each modality, model, and dataset.

		ELBO			IWAE			SIWAE		
	x_d	M	S	T	M	S	T	M	S	T
MVAE	Vanilla	97.53	88.26	99.30	97.27	87.19	98.76	97.37	87.47	98.83
	Ours	97.85	89.65	99.64	98.28	89.01	99.93	97.42	87.63	99.20
MMVAE	Vanilla	86.01	45.59	89.17	85.25	84.03	88.66	58.95	61.27	63.27
	Ours	89.42	45.83	91.54	87.55	86.87	90.93	74.85	73.89	81.09
MoPoE	Vanilla	95.72	85.86	98.01	95.82	87.55	97.93	75.10	67.16	76.61
	Ours	96.50	93.60	99.14	97.29	92.93	99.00	96.91	89.01	99.28

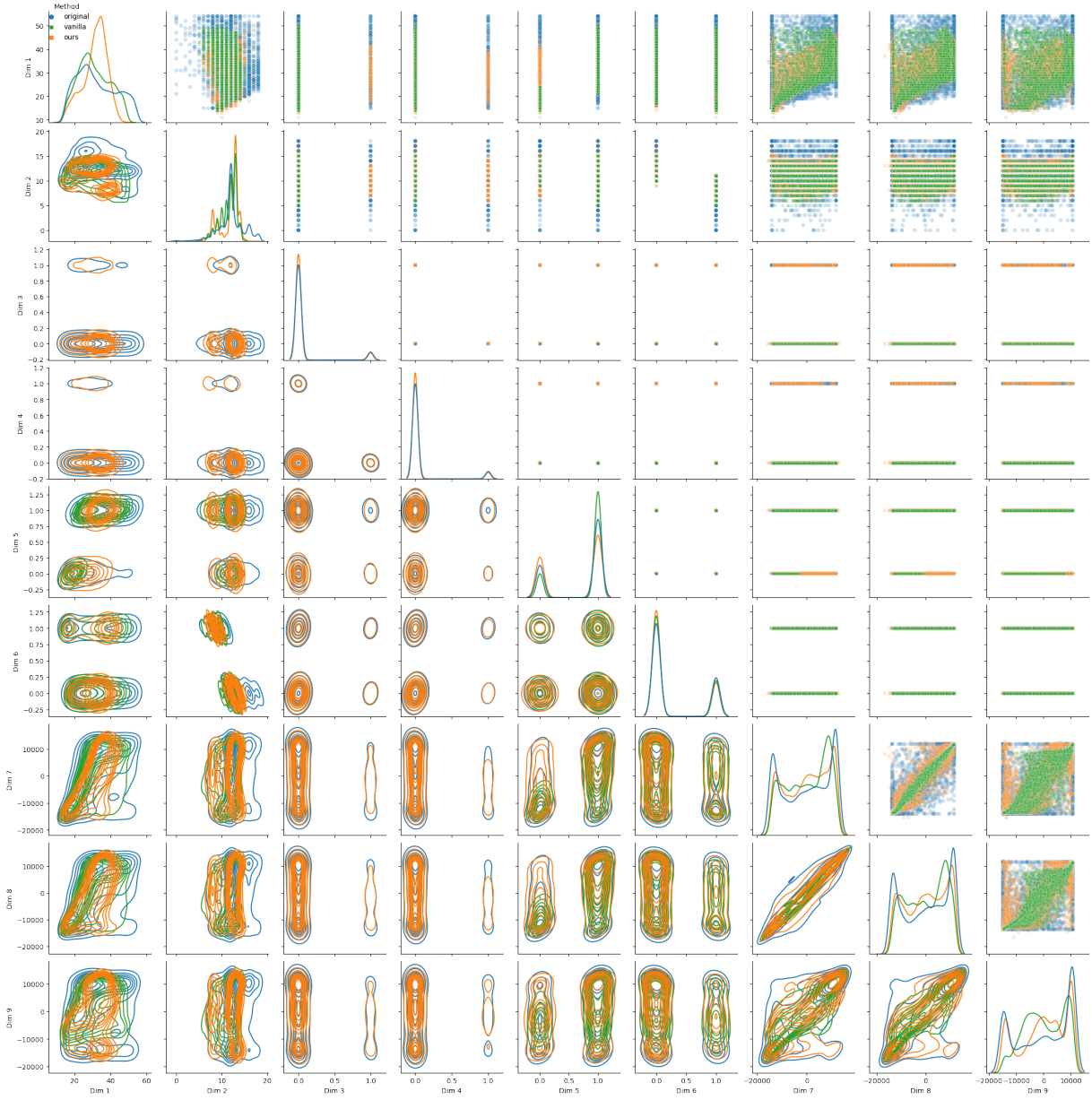


Figure D.4: Pair plot of all the dimensions of labour, generated from different VAEs. Diagonal show the marginals, upper-diagonals scatter plots, and lower-diagonals kernel density estimates. The VAE trained with our approach is able to generate faithful samples.

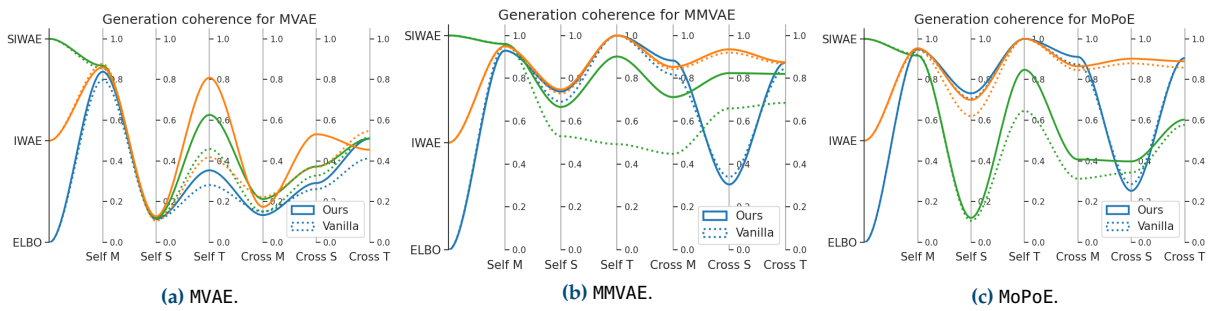


Figure D.5: Generation coherence results for all models and losses. In general, our framework improves all metrics *w. r. t.* the baseline.

Table D.7: Self and cross generation coherence (%) results for different models on MNIST-SVHN-Text, trained using ELBO and averaged over 5 different seeds. Models trained with our framework are able to sample more coherent modalities. CA stands for CAGrad.

x_d	A				Self coherence			Cross coherence									Time h
		LI	EEI	DEI	M	S	T	S	M	ST	M	S	MT	M	T	MS	
					M	S	T										
MVAE	Vanilla	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	80.30	12.63	25.77	11.12	18.22	19.49	43.29	18.50	16.90	51.82	11.65	54.71	3.82
	CA ($\alpha = 0.4$)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	85.12	12.34	34.64	10.70	12.51	16.83	44.94	22.36	29.45	61.41	12.26	69.77	4.82
MMVAE	Vanilla	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	95.23	68.25	99.99	62.93	99.92	81.43	31.06	37.42	34.24	96.27	71.29	83.79	10.07
	CA ($\alpha = 10$)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	92.62	73.84	99.99	76.01	99.59	87.80	29.11	34.46	31.78	95.27	79.34	87.31	13.06
MoPoE	Vanilla	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	94.52	71.21	99.99	66.80	99.98	98.12	19.70	33.25	31.55	96.79	77.04	97.15	23.24
	CA ($\alpha = 10$)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	94.64	73.36	100.00	74.76	99.98	98.42	15.84	32.84	31.03	96.03	78.61	96.67	29.41

Table D.8: Self and cross generation coherence (%) results for different models on MNIST-SVHN-Text, trained using IWAE and averaged over 5 different seeds. Models trained with our framework are able to sample more coherent modalities. CA stands for CAGrad.

x_d	A				Self coherence			Cross coherence									Time h
		LI	EEI	DEI	M	S	T	S	M	ST	M	S	MT	M	T	MS	
					M	S	T										
MVAE	vanilla	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	87.33	11.70	37.49	10.76	26.63	29.48	54.10	25.16	29.51	70.24	11.38	72.33	3.80
	GN ($\alpha = 0.0$)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	85.73	12.68	79.19	11.18	19.45	22.05	49.37	55.83	54.48	59.56	11.79	63.69	4.24
MMVAE	vanilla	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	94.70	68.13	99.99	62.34	98.79	80.55	86.72	97.29	92.01	96.83	69.23	83.02	10.08
	CA ($\alpha = 0.4$)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	94.96	73.66	99.99	68.49	99.25	83.85	88.99	97.97	93.49	96.27	76.55	86.40	11.96
MoPoE	vanilla	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	94.44	63.16	99.97	58.20	99.01	99.24	80.75	94.76	88.62	96.51	66.13	96.03	23.27
	CA ($\alpha = 10.0$)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	95.41	69.54	99.99	61.75	99.06	99.07	81.23	96.47	92.98	96.64	73.33	96.16	29.35

Table D.9: Self and cross generation coherence (%) results for different models on MNIST-SVHN-Text, trained using SIWAE and averaged over 5 different seeds. Models trained with our framework are able to sample more coherent modalities. CA stands for CAGrad.

x_d	A				Self coherence			Cross coherence									Time h
		LI	EEI	DEI	M	S	T	S	M	ST	M	S	MT	M	T	MS	
					M	S	T										
MVAE	Vanilla	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	82.06	12.08	36.67	10.34	17.12	19.19	49.99	19.31	31.19	62.50	10.82	64.25	3.81
	CA ($\alpha = 0.4$)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	86.89	12.89	59.15	10.99	20.92	30.34	54.83	31.82	24.97	71.87	11.10	72.74	4.77
MMVAE	Vanilla	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	95.90	48.30	53.02	28.43	52.52	40.45	84.44	51.08	67.77	96.80	39.96	68.38	10.07
	CA ($\alpha = 0.4$)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	95.90	58.20	88.70	49.33	79.32	64.30	87.29	76.17	81.71	96.70	57.86	77.28	12.30
MoPoE	Vanilla	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	92.32	11.60	69.05	10.13	51.02	34.67	41.93	46.39	51.58	85.19	10.57	67.54	23.29
	GN ($\alpha = 0.5$)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	90.99	12.00	83.82	10.63	62.75	52.08	28.19	46.91	43.34	79.64	10.81	90.33	24.58

Table D.10: Log-likelihood of the joint generative model, conditioned on the variational posterior of subsets of the modalities. Results report on ELBO as loss function and are averaged over 5 different seeds. Each log-likelihood is divided by the dimensionality of its modality before adding them up, to better reflect improvement across modalities.

		$\mathbb{X} M$	$\mathbb{X} S$	$\mathbb{X} T$	$\mathbb{X} MS$	$\mathbb{X} MT$	$\mathbb{X} ST$	$\mathbb{X} MST$
MVAE	Van.	-8.61 ± 1.18	-10.26 ± 1.14	-8.17 ± 1.30	-8.18 ± 1.26	-1.51 ± 0.11	-7.41 ± 1.38	-1.02 ± 0.11
	Ours	-8.07 ± 0.88	-9.89 ± 0.62	-6.94 ± 1.19	-7.44 ± 0.64	-1.42 ± 0.05	-6.12 ± 1.69	-0.95 ± 0.00
MMVAE	Van.	-2.30 ± 0.21	-2.18 ± 0.09	-1.19 ± 0.00	-2.24 ± 0.11	-1.75 ± 0.10	-1.68 ± 0.04	-1.89 ± 0.07
	Ours	-2.59 ± 0.40	-2.31 ± 0.13	-1.19 ± 0.00	-2.45 ± 0.23	-1.89 ± 0.20	-1.75 ± 0.06	-2.03 ± 0.15
MoPoE	Van.	-1.93 ± 0.01	-2.06 ± 0.04	-1.19 ± 0.00	-1.76 ± 0.00	-1.18 ± 0.00	-1.04 ± 0.00	-1.03 ± 0.05
	Ours	-1.92 ± 0.03	-2.09 ± 0.05	-1.19 ± 0.00	-1.75 ± 0.02	-1.17 ± 0.01	-1.03 ± 0.00	-1.00 ± 0.01

Table D.11: Log-likelihood of the joint generative model, conditioned on the variational posterior of subsets of the modalities. Results report on IWAE as loss function and are averaged over 5 different seeds. Each log-likelihood is divided by the dimensionality of its modality before adding them up, to better reflect improvement across modalities.

		$\mathbb{X} M$	$\mathbb{X} S$	$\mathbb{X} T$	$\mathbb{X} MS$	$\mathbb{X} MT$	$\mathbb{X} ST$	$\mathbb{X} MST$
MVAE	Van.	-8.62 ± 0.40	-10.68 ± 0.98	-6.78 ± 1.66	-8.28 ± 0.47	-1.83 ± 0.14	-6.08 ± 1.36	-1.15 ± 0.03
	Ours	-8.26 ± 0.29	-9.51 ± 0.25	-2.83 ± 0.54	-7.74 ± 0.15	-1.35 ± 0.03	-1.96 ± 0.33	-0.94 ± 0.00
MMVAE	Van.	-1.98 ± 0.06	-2.74 ± 0.32	-1.27 ± 0.00	-2.36 ± 0.15	-1.62 ± 0.03	-2.00 ± 0.16	-1.99 ± 0.10
	Ours	-2.44 ± 0.03	-2.45 ± 0.12	-1.26 ± 0.00	-2.44 ± 0.06	-1.85 ± 0.01	-1.86 ± 0.06	-2.05 ± 0.04
MoPoE	Van.	-1.97 ± 0.00	-2.55 ± 0.12	-1.27 ± 0.00	-5.85 ± 0.52	-1.25 ± 0.00	-1.05 ± 0.00	-1.02 ± 0.00
	Ours	-2.01 ± 0.01	-2.61 ± 0.11	-1.27 ± 0.00	-7.07 ± 0.06	-1.24 ± 0.01	-1.05 ± 0.00	-1.01 ± 0.00

Table D.12: Log-likelihood of the joint generative model, conditioned on the variational posterior of subsets of the modalities. Results report on SIWAE as loss function and are averaged over 5 different seeds. Each log-likelihood is divided by the dimensionality of its modality before adding them up, to better reflect improvement across modalities.

		$\mathbb{X} M$	$\mathbb{X} S$	$\mathbb{X} T$	$\mathbb{X} MS$	$\mathbb{X} MT$	$\mathbb{X} ST$	$\mathbb{X} MST$
MVAE	Van.	-8.82 ± 0.63	-10.13 ± 0.29	-7.17 ± 1.73	-8.52 ± 0.74	-2.23 ± 1.31	-5.94 ± 1.35	-1.15 ± 0.02
	Ours	-8.41 ± 0.14	-10.52 ± 0.53	-5.65 ± 1.76	-8.13 ± 0.17	-1.46 ± 0.07	-5.49 ± 1.87	-1.01 ± 0.01
MMVAE	Van.	-2.01 ± 0.08	-4.04 ± 0.43	-3.25 ± 0.94	-3.02 ± 0.20	-2.63 ± 0.49	-3.64 ± 0.25	-3.10 ± 0.18
	Ours	-2.34 ± 0.36	-3.22 ± 1.28	-3.03 ± 1.07	-2.78 ± 0.59	-2.69 ± 0.54	-3.12 ± 0.43	-2.86 ± 0.22
MoPoE	Van.	-6.29 ± 2.75	-10.02 ± 0.43	-3.80 ± 1.48	-7.51 ± 0.93	-1.59 ± 0.35	-4.97 ± 2.68	-3.15 ± 3.78
	Ours	-8.01 ± 0.42	-10.16 ± 0.55	-2.96 ± 1.11	-7.26 ± 0.28	-1.34 ± 0.01	-2.30 ± 0.91	-0.99 ± 0.02

Additional Material for Chapters 11 and 12


E.

E.1 Interventions in previous works

E.1 Previous do-operators . .	197
E.2 Details & extra results . .	198
E.3 Fairness use-case	205

In this section, we put our implementation of the do-operator (see [Section 11.5](#)) into context and describe how the models considered in [Section 12.4](#), namely CAREFL [\[91\]](#) and VACA [\[174\]](#), propose to perform interventions with their models.

CAREFL [\[91\]](#). Two different algorithms were proposed to sample from an interventional distribution in CAREFL: **i)** a sequential algorithm which mimics the usual implementation of the do-operator with the recursive representation of the **structural causal model (SCM)**; and **ii)** a parallel algorithm that samples the counterfactual in a single call. While the first algorithm works, the parallel one is the one actually implemented in their official codebase, and it only works when intervening on root nodes.

[\[91\]](#) Khemakhem, Monti, Leech and Hyvärinen (2021), ‘Causal Autoregressive Flows.’ 

This second algorithm, if we were to compute $do(x_i := \alpha)$, is described as follows:¹ **i)** sample \mathbf{u} from $P_{\mathbf{u}}$; **ii)** set u_i to the i -th value obtained by applying the flow, T_{θ} , to an observation with $x_i = \alpha$ and $x_j = 0$ for $j \neq i$; and **iii)** return the value obtained by T_{θ}^{-1} with \mathbf{u} .

¹: See Alg. 2 in [\[91\]](#).

While this algorithm resembles the one proposed in this thesis, this implementation does not have into account that the value of u_i to fix α *does depend* on the observed values of its parents, which is clear by looking at the linear illustrative example from [Figure 11.1](#). As a consequence, the algorithm only works when the node has no parents, which is why we replaced it by the one we proposed for the comparisons in [Section 12.4](#) and [Appendix E.2](#).

VACA [\[174\]](#). Based on **graph neural networks (GNNs)**, the approach for intervening on VACA is reminiscent to the traditional implementation of the do-operator. Specifically, the authors propose to sever those edges in every layer of the GNN whose endpoints fall in the path generating the intervened variable, so that the ancestors have no way to influence it by design. While the previous statement is true: ancestors cannot influence the intervened variable nor its descendants, here we argue that this process would require us to ‘recalibrate’ the model, as the middle computations after an intervention change in more complex ways than simply removing the ancestors from the structural equation, while keeping the rest unchanged. To see this, consider the following non-linear triangular SCM:

$$\begin{cases} x_1 = u_1 \\ x_2 = x_1^2 u_2 \\ x_3 = 2x_1 + \frac{x_2}{x_1} + \frac{x_2}{x_1^2} + u_3 \end{cases} \quad (\text{E.1})$$

which VACA could learn with two layers as follows:

$$\begin{cases} z_1 = u_1 \\ z_2 = u_1 u_2 \\ z_3 = u_1 + u_2 + u_3 \end{cases} \quad \begin{cases} x_1 = z_1 \\ x_2 = z_1 z_2 \\ x_3 = z_1 + z_2 + z_3 \end{cases} \quad (\text{E.2})$$

Now, if we were to compute $do(x_2 := \alpha)$, the real SCM would yield:

$$\begin{cases} x_1 = u_1 \\ x_2 = \alpha \\ x_3 = 2x_1 + \frac{\alpha}{x_1} + \frac{\alpha}{x_1^2} + u_3 \end{cases} \quad (\text{E.3})$$

while VACA would yield:

$$\begin{cases} z_1 = u_1 \\ z_2 = \alpha \\ z_3 = u_1 + \alpha + u_3 \end{cases} \quad \begin{cases} x_1 = z_1 \\ x_2 = \alpha \\ x_3 = z_1 + \alpha + z_3 \end{cases} \implies \begin{cases} x_1 = u_1 \\ x_2 = \alpha \\ x_3 = 2x_1 + 2\alpha + u_3 \end{cases} \quad (\text{E.4})$$

where we can clearly see that the expression for x_3 is not the correct one. In contrast, our **causal normalizing flow (Causal NF)** would keep the structural equations as they are, and set u_2 to α/x_1^2 , yielding the correct value.

E.2 Experimental details and extra results

E.2.1 Base distribution	201
E.2.2 Flow architecture	201
E.2.3 Additional SCMs	202

In this section, we complement the description of the experimental section from **Chapter 12**, and provide the reader with additional results in the following subsections. First, we describe the details common to every experiment, and delve into the specifics of each experiment in their respective subsections.

Hardware. Every individual experiment shown in this paper ran on a single CPU with 8 GB of RAM. To run all experiments, we used a local computing cluster with an automatic job assignment system, so we cannot ensure the specific CPU used for each particular experiment. However, we know that every experiment used one of the following CPUs picked randomly given the demand when scheduled: AMD EPYC 7702 64-Core Processor, AMD EPYC 7662 64-Core Processor, Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz, or Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz.

Training and evaluation methodology. For every experiment with synthetic SCMs, we generated a dataset with 20 000 training samples, 2500 validation samples, and 2500 test samples. We ran every model for 1000 epochs, and the results shown in the manuscript correspond to the test set evaluation at the last epoch. For the optimization, we used Adam [92] with an initial learning rate of 0.001, and reduced the learning rate with a decay factor of 0.95 when it reaches a plateau longer than 60 epochs. For hyperparameter tuning, we always perform a grid search with similar budget, and select the best hyperparameter combination

according to validation loss, reporting always results from the test dataset in the manuscript. Every experiment is repeated 5 times, and we show averages and standard deviations.

Datasets. This section provides all the information of the SCMs employed in the empirical evaluation of [Chapter 12](#). The exogenous variables always follow a standard normal distribution $\mathcal{N}(0, 1)$, except for LargeBD, where a uniform distribution $\mathcal{U}(0, 1)$ is used instead. Subsequently, we define the 12 SCMs employed (encompassing both linear and non-linear equations) and we additionally provide their causal graph in [Figure E.1](#).

Let us first define the softplus operation as $s(x) = \log(1.0 + e^x)$. Then:

3-Chain_{lin}:

$$x_1 = f_1(u_1) = u_1 \quad (\text{E.5})$$

$$x_2 = f_2(x_1, u_2) = 10 \cdot x_1 - u_2 \quad (\text{E.6})$$

$$x_3 = f_3(x_2, u_3) = 0.25 \cdot x_2 + 2 \cdot u_3 \quad (\text{E.7})$$

3-Chain_{nlin}:

$$x_1 = f_1(u_1) = u_1 \quad (\text{E.8})$$

$$x_2 = f_2(x_1, u_2) = e^{x_1/2} + u_2/4 \quad (\text{E.9})$$

$$x_3 = f_3(x_2, u_3) = \frac{(x_2 - 5)^3}{15} + u_3 \quad (\text{E.10})$$

4-Chain_{lin}:

$$x_1 = f_1(u_1) = u_1 \quad (\text{E.11})$$

$$x_2 = f_2(x_1, u_2) = 5 \cdot x_1 - u_2 \quad (\text{E.12})$$

$$x_3 = f_3(x_2, u_3) = -0.5 \cdot x_2 - 1.5 \cdot u_3 \quad (\text{E.13})$$

$$x_4 = f_4(x_3, u_4) = x_3 + u_4 \quad (\text{E.14})$$

5-Chain_{lin}:

$$x_1 = f_1(u_1) = u_1 \quad (\text{E.15})$$

$$x_2 = f_2(x_1, u_2) = 10 \cdot x_1 - u_2 \quad (\text{E.16})$$

$$x_3 = f_3(x_2, u_3) = 0.25 \cdot x_2 + 2 \cdot u_3 \quad (\text{E.17})$$

$$x_4 = f_4(x_3, u_4) = x_3 + u_4 \quad (\text{E.18})$$

$$x_5 = f_5(x_4, u_5) = -x_4 + u_5 \quad (\text{E.19})$$

Collider_{lin}:

$$x_1 = f_1(u_1) = u_1 \quad (\text{E.20})$$

$$x_2 = f_2(u_2) = 2 - u_2 \quad (\text{E.21})$$

$$x_3 = f_3(x_1, x_2, u_3) = 0.25 \cdot x_2 - 0.5 \cdot x_1 + 0.5 \cdot u_3 \quad (\text{E.22})$$

Fork_{lin}:

$$x_1 = f_1(u_1) = u_1 \quad (\text{E.23})$$

$$x_2 = f_2(u_2) = 2 - u_2 \quad (\text{E.24})$$

$$x_3 = f_3(x_1, x_2, u_3) = 0.25 \cdot x_2 - 1.5 \cdot x_1 + 0.5 \cdot u_3 \quad (\text{E.25})$$

$$x_4 = f_4(x_3, u_4) = x_3 + 0.25 \cdot u_4 \quad (\text{E.26})$$

Fork_{nlin}:

$$x_1 = f_1(u_1) = u_1 \quad (\text{E.27})$$

$$x_2 = f_2(u_2) = u_2 \quad (\text{E.28})$$

$$x_3 = f_3(x_1, x_2, u_3) = \frac{4}{1 + e^{-x_1 - x_2}} - x_2^2 + 0.5 \cdot u_3 \quad (\text{E.29})$$

$$x_4 = f_4(x_3, u_4) = \frac{20}{1 + e^{0.5 \cdot x_3^2 - x_3}} + u_4 \quad (\text{E.30})$$

LargeBD_{nlin}: Let us define

$$L(x, y) = s(x + 1) + s(0.5 + y) - 3.0, \quad (\text{E.31})$$

and let us call $\text{CDF}^{-1}(\mu, b, x)$ the quantile function of a Laplace distribution with location μ , scale b , and evaluated at x . Then:

$$x_1 = f_1(u_1) = s(1.8 \cdot u_1) - 1 \quad (\text{E.32})$$

$$x_2 = f_2(x_1, u_2) = 0.25 \cdot u_2 + L(x_1, 0) \cdot 1.5 \quad (\text{E.33})$$

$$x_3 = f_3(x_1, u_3) = L(x_1, u_3) \quad (\text{E.34})$$

$$x_4 = f_4(x_2, u_4) = L(x_2, u_4) \quad (\text{E.35})$$

$$x_5 = f_5(x_3, u_5) = L(x_3, u_5) \quad (\text{E.36})$$

$$x_6 = f_6(x_4, u_6) = L(x_4, u_6) \quad (\text{E.37})$$

$$x_7 = f_7(x_5, u_7) = L(x_5, u_7) \quad (\text{E.38})$$

$$x_8 = f_8(x_6, u_8) = 0.3 \cdot u_8 + (s(x_6 + 1) - 1) \quad (\text{E.39})$$

$$x_9 = f_9(x_7, x_8, u_9) = \text{CDF}^{-1} \left(-s \left(\frac{x_7 \cdot 1.3 + x_8}{3} + 1 \right) + 2, 0.6, u_9 \right) \quad (\text{E.40})$$

Simpson_{nlin}:

$$x_1 = f_1(u_1) = u_1 \quad (\text{E.41})$$

$$x_2 = f_2(x_1, u_2) = s(1 - x_1) + \sqrt{3/20} \cdot u_2 \quad (\text{E.42})$$

$$x_3 = f_3(x_1, x_2, u_3) = \tanh(2 \cdot x_2) + 1.5 \cdot x_1 - 1 + \tanh(u_3) \quad (\text{E.43})$$

$$x_4 = f_4(x_3, u_4) = \frac{x_3 - 4}{5} + 3 + \frac{1}{\sqrt{10}} \cdot u_4 \quad (\text{E.44})$$

Simpson_{symprod}:

$$x_1 = f_1(u_1) = u_1 \quad (\text{E.45})$$

$$x_2 = f_2(x_1, u_2) = 2 \cdot \tanh(2 \cdot x_1) + \frac{1}{\sqrt{10}} \cdot u_2 \quad (\text{E.46})$$

$$x_3 = f_3(x_1, x_2, u_3) = 0.5 \cdot x_1 \cdot x_2 + \frac{1}{\sqrt{2}} \cdot u_3 \quad (\text{E.47})$$

$$x_4 = f_4(x_1, u_4) = \tanh(1.5 \cdot x_1) + \sqrt{\frac{3}{10}} \cdot u_4 \quad (\text{E.48})$$

Triangle_{lin}:

$$x_1 = f_1(u_1) = u_1 + 1 \quad (\text{E.49})$$

$$x_2 = f_2(x_1, u_2) = 10 \cdot x_1 - u_2 \quad (\text{E.50})$$

$$x_3 = f_3(x_1, x_2, u_3) = 0.5 \cdot x_2 + x_1 + u_3 \quad (\text{E.51})$$

Triangle_{NLIN}:

$$x_1 = f_1(u_1) = u_1 + 1 \quad (\text{E.52})$$

$$x_2 = f_2(x_1, u_2) = 2 \cdot x_1^2 + u_2 \quad (\text{E.53})$$

$$x_3 = f_3(x_1, x_2, u_3) = \frac{20}{1 + e^{-x_2^2 + x_1}} + u_3 \quad (\text{E.54})$$

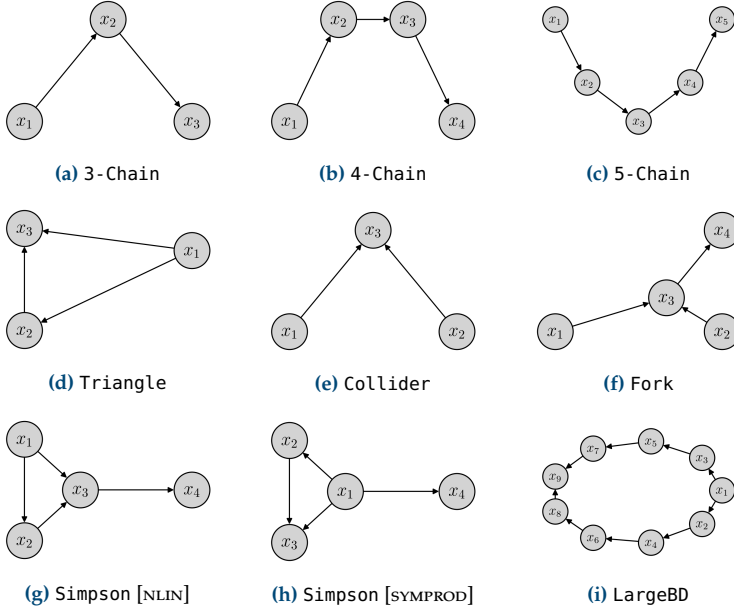


Figure E.1: Causal graph of the different SCMs considered in the experiments of Chapter 12.

E.2.1 Ablation: Base distribution

Hyperparameter tuning. While we fixed the flow to have a single **masked autoregressive flow (MAF)** [146] layer with ELU [32] activation functions, we determined through cross-validation the optimal number of layers and hidden units of the neural network within MAF. Specifically, we considered the following combinations, where $[a, b]$ represents two layers with a and b hidden units: $[16, 16, 16, 16]$, $[32, 32, 32]$, $[16, 16, 16]$, $[32, 32]$, $[32]$, $[64]$. As discussed at the start of the section, we report test results for the configuration with the best validation performance at the last epoch.

E.2.2 Ablation: Flow architecture

Hyperparameter Tuning. We cross-validate again the optimal number of layers and hidden units of the neural network of the unique layer of the Causal NF. We consider the following values, where $[a, b]$ represents two layers with a and b hidden units: $[32, 32, 32]$, $[16, 16, 16]$, $[32, 32]$, $[32]$, and $[64]$. As before, test results are reported for the configuration that achieved the best performance on the validation set at the final epoch.

E.2.3 Comparison: Additional non-linear SCMs

In this section, we complement the results from [Section 12.4](#) and provide a more extensive comparison of the proposed CausalNF, along with CAREFL [\[91\]](#) and VACA [\[174\]](#), on additional datasets.

2: See [\[174\]](#) for details.

3: Same format as before.

Hyperparameter Tuning. For VACA, we cross-validated the dropout rate with values $\{0.0, 0.1\}$, the GNN layer architecture with $\{\text{GIN}, \text{PNA}, \text{PNADisjoint}\}$,² and the number of layers in the network prior to the GNN with choices $\{1, 2\}$. For CAREFL, we cross-validated the number of layers in the flow, $\{1, 2, \dots, \text{diam } A\}$, and the number of layers and hidden units of the network composing the flow layers,³ $\{[16, 16, 16], [32, 32], [32], [64]\}$. For CausalNF, we used the abductive model with a single layer, and cross-validated the number of layers and hidden units in the network composing the layer of the flow with values $\{[16, 16, 16, 16], [32, 32, 32], [16, 16, 16], [32, 32], [32], [64]\}$. We report test results for the configuration with the best validation performance at the final epoch.

Results. [Table E.1](#) shows the performance of each model for all the considered datasets, further validating the conclusions drawn in the main manuscript: the proposed CausalNF consistently outperforms both CAREFL and VACA in terms of performance and computational efficiency. The performance of VACA is significantly inferior, and its computation time is higher, primarily due to the complexity of GNNs. Our CausalNF achieves similar performance to CAREFL in terms of observational fitting, while surpassing it on interventional and counterfactual estimation tasks. Additionally, CausalNF outperforms CAREFL in computational speed. This is to be expected since the optimal CAREFL architectures often have multiple layers, resulting in increased computation time. In contrast, CausalNF has a single layer, reducing computational complexity.

[Figure E.2](#) qualitative proofs the effectiveness of the proposed CausalNF in accurately modelling both observational and interventional distributions for the $\text{Simpson}_{\text{nl}_{\text{in}}}$ dataset. In this plot, blue represent the real distribution/samples, while orange represents the ones generated by CausalNF. [Figure E.2a](#) clearly shows that the model successfully captured the correlations among all variables in the observational distribution. Furthermore, [Figure E.2b](#) displays the interventional distribution obtained when we do $\text{do}(x_3 := -1.09)$, *i.e.*, when we intervene on the 25th empirical percentile of x_3 . Remarkably, CausalNF accurately learns the distribution of descendant variables, *i.e.*, x_4 , and effectively breaks any dependency between the ancestors of the intervened variable and x_4 . Additionally, [Figure E.3](#) shows a similar analysis for $5\text{-Chain}_{\text{lin}}$, when we perform $\text{do}(x_3 := 2.18)$ —which corresponds to intervening on the 75th percentile of x_3 —clearly showing that the correlations not involving the intervened path ($x_1 \rightarrow x_2$ and $x_4 \rightarrow x_5$) are preserved.

Table E.1: Comparison, on different SCMs, of the proposed CausalNF, VACA [174], and CAREFL [91] with the do-operator proposed in Section 11.5. The results are averaged over 5 different runs.

Dataset	Model	Performance			Time Evaluation (μ s)		
		Observ.	Interv.	Counter.	Training	Evaluation	Sampling
3-Chain LIN [174]	CausalNF	0.00 _{0.00}	0.05 _{0.01}	0.04 _{0.01}	0.41 _{0.06}	0.48 _{0.10}	0.76 _{0.06}
	CAREFL [†]	0.00 _{0.00}	0.20 _{0.13}	0.20 _{0.09}	0.68 _{0.24}	0.97 _{0.33}	1.94 _{0.77}
	VACA	4.44 _{1.03}	5.76 _{0.07}	4.98 _{0.10}	36.19 _{1.54}	28.33 _{0.72}	75.34 _{4.58}
3-Chain NLIN [174]	CausalNF	0.00 _{0.00}	0.03 _{0.01}	0.02 _{0.01}	0.52 _{0.06}	0.56 _{0.03}	1.02 _{0.05}
	CAREFL [†]	0.00 _{0.00}	0.05 _{0.02}	0.04 _{0.02}	0.60 _{0.22}	0.84 _{0.22}	1.66 _{0.41}
	VACA	12.82 _{1.00}	1.54 _{0.03}	1.32 _{0.02}	39.45 _{4.12}	30.93 _{2.30}	84.36 _{9.60}
4-Chain LIN	CausalNF	0.00 _{0.00}	0.07 _{0.02}	0.04 _{0.01}	0.56 _{0.08}	0.62 _{0.15}	1.54 _{0.40}
	CAREFL [†]	0.00 _{0.00}	0.16 _{0.07}	0.14 _{0.04}	0.70 _{0.28}	0.99 _{0.20}	2.85 _{0.54}
	VACA	13.14 _{0.73}	3.82 _{0.01}	3.72 _{0.05}	61.85 _{5.06}	49.31 _{4.11}	92.06 _{7.93}
5-Chain LIN	CausalNF	0.01 _{0.00}	0.12 _{0.02}	0.08 _{0.01}	0.62 _{0.19}	0.69 _{0.15}	1.91 _{0.44}
	CAREFL [†]	0.00 _{0.00}	0.47 _{0.23}	0.46 _{0.22}	0.79 _{0.41}	1.19 _{0.25}	4.21 _{0.87}
	VACA	17.31 _{0.84}	5.95 _{0.05}	6.06 _{0.08}	103.75 _{10.04}	80.81 _{11.06}	124.52 _{20.86}
Collider LIN [174]	CausalNF	0.00 _{0.00}	0.02 _{0.01}	0.01 _{0.00}	0.46 _{0.12}	0.56 _{0.11}	0.95 _{0.19}
	CAREFL [†]	0.00 _{0.00}	0.02 _{0.01}	0.01 _{0.00}	0.39 _{0.07}	0.45 _{0.05}	0.74 _{0.07}
	VACA	13.45 _{0.43}	0.22 _{0.01}	0.86 _{0.02}	37.22 _{3.55}	28.77 _{4.22}	71.21 _{6.73}
Fork LIN [22]	CausalNF	0.00 _{0.00}	0.03 _{0.01}	0.01 _{0.00}	0.52 _{0.05}	0.59 _{0.08}	1.57 _{0.57}
	CAREFL [†]	0.00 _{0.00}	0.04 _{0.01}	0.02 _{0.00}	0.60 _{0.17}	0.78 _{0.16}	2.39 _{1.06}
	VACA	8.75 _{0.73}	0.87 _{0.02}	1.43 _{0.02}	45.84 _{4.64}	34.66 _{2.39}	73.29 _{4.70}
Fork NLIN [22]	CausalNF	0.00 _{0.00}	0.07 _{0.02}	0.07 _{0.00}	0.63 _{0.16}	0.74 _{0.31}	1.84 _{0.84}
	CAREFL [†]	0.01 _{0.01}	0.11 _{0.04}	0.18 _{0.07}	0.57 _{0.17}	0.77 _{0.08}	1.96 _{0.17}
	VACA	5.09 _{0.60}	2.01 _{0.03}	3.19 _{0.06}	49.22 _{5.48}	42.13 _{2.95}	101.02 _{18.94}
LargeBD NLIN [63]	CausalNF	1.51 _{0.04}	0.02 _{0.00}	0.01 _{0.00}	0.52 _{0.10}	0.60 _{0.17}	3.05 _{0.66}
	CAREFL [†]	1.51 _{0.05}	0.05 _{0.01}	0.08 _{0.01}	0.84 _{0.47}	1.18 _{0.17}	8.25 _{1.29}
	VACA	53.66 _{2.07}	0.39 _{0.00}	0.82 _{0.02}	164.92 _{11.10}	137.88 _{15.72}	167.94 _{25.75}
Simpson NLIN [63]	CausalNF	0.29 _{0.01}	0.04 _{0.01}	0.02 _{0.00}	0.58 _{0.18}	0.63 _{0.26}	1.57 _{0.64}
	CAREFL [†]	0.29 _{0.01}	0.04 _{0.01}	0.03 _{0.00}	0.69 _{0.32}	1.02 _{0.26}	2.95 _{0.79}
	VACA	18.97 _{0.66}	0.60 _{0.00}	1.19 _{0.02}	54.76 _{7.46}	43.69 _{7.92}	87.01 _{16.33}
Simpson SYMPROD [63]	CausalNF	0.00 _{0.00}	0.07 _{0.01}	0.12 _{0.02}	0.59 _{0.17}	0.60 _{0.11}	1.51 _{0.30}
	CAREFL [†]	0.00 _{0.00}	0.10 _{0.02}	0.17 _{0.04}	0.49 _{0.15}	0.81 _{0.19}	1.91 _{0.33}
	VACA	13.85 _{0.64}	0.89 _{0.00}	1.50 _{0.04}	49.26 _{4.09}	37.78 _{3.41}	79.20 _{14.60}
Triangle LIN [174]	CausalNF	0.00 _{0.00}	0.24 _{0.05}	0.21 _{0.05}	0.54 _{0.05}	0.56 _{0.04}	1.05 _{0.07}
	CAREFL [†]	0.00 _{0.00}	0.15 _{0.06}	0.14 _{0.03}	0.60 _{0.20}	0.75 _{0.05}	1.50 _{0.10}
	VACA	3.82 _{0.69}	7.49 _{0.07}	7.22 _{0.17}	27.46 _{1.53}	21.61 _{1.00}	67.00 _{6.23}
Triangle NLIN [174]	CausalNF	0.00 _{0.00}	0.12 _{0.03}	0.13 _{0.02}	0.52 _{0.07}	0.58 _{0.07}	1.07 _{0.12}
	CAREFL [†]	0.00 _{0.00}	0.12 _{0.03}	0.17 _{0.03}	0.57 _{0.18}	0.83 _{0.26}	1.68 _{0.62}
	VACA	7.71 _{0.60}	4.78 _{0.01}	4.19 _{0.04}	28.82 _{1.21}	23.00 _{0.55}	70.65 _{3.70}

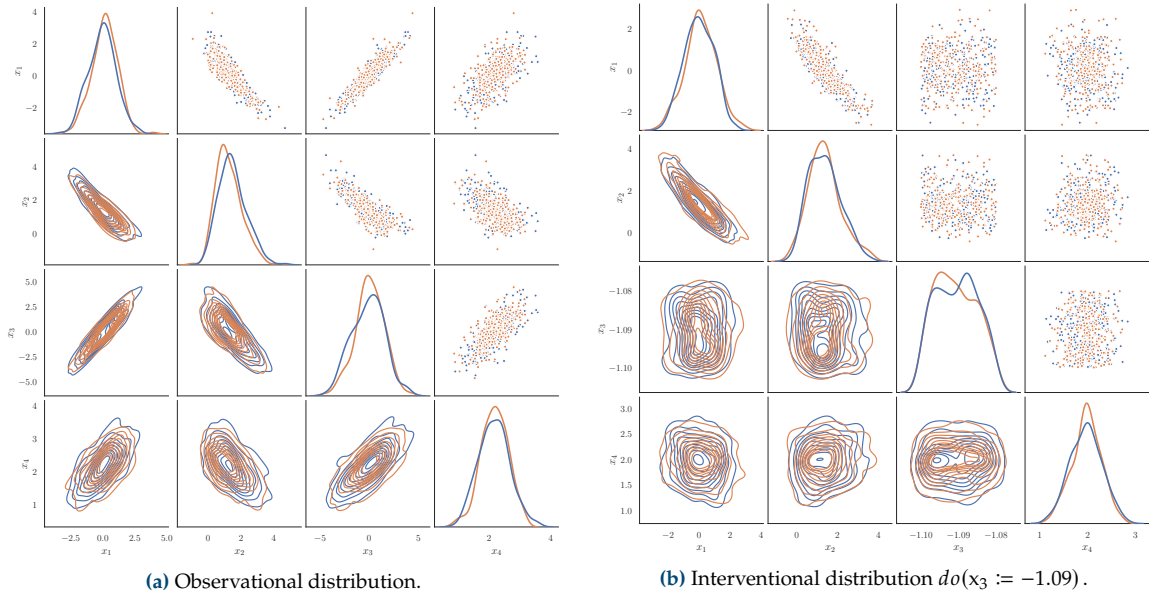


Figure E.2: Pair plot of real (in blue) and generated (in orange) data of $\text{Simpson}_{\text{nlin}}$. (a) Samples from the true and learnt observational distribution. (b) Samples from the true and learnt interventional distribution when $do(x_3 := -1.09)$. The plot illustrates that the dependency of x_4 on the ancestors of x_3 , namely x_1 and x_2 , is effectively broken.

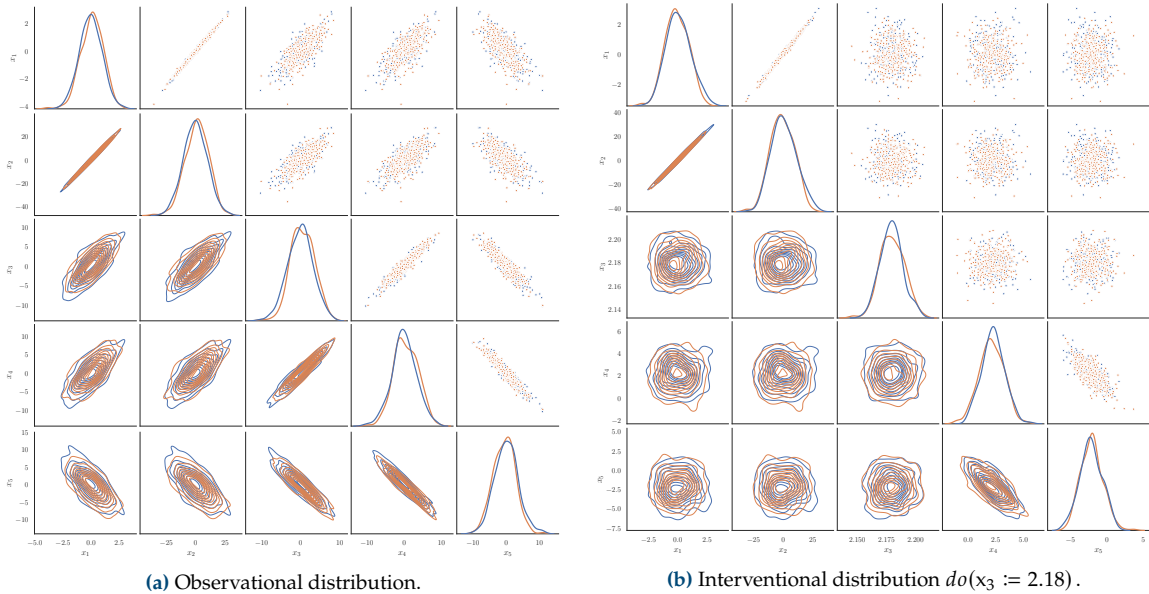


Figure E.3: Pair plot of real (in blue) and generated (in orange) data of $\text{5-Chain}_{\text{nlin}}$. (a) Samples from the true and learnt observational distribution. (b) Samples from the true and learnt interventional distribution when $do(x_3 := 2.18)$. The plot illustrates that the dependency of x_4 and x_5 on the ancestors of x_3 , namely x_1 and x_2 , is effectively broken.

E.3 Details on the fairness use-case

In this section, we provide additional details on the use-case of fairness auditing and classification using the German dataset [48], whose causal graph is shown in Figure E.4. Here, x_1 and x_2 correspond to the Credit amount and Repayment history variables, x_3 to x_5 to the Checking account, Savings, and Housing variables and, lastly, x_S and x_A represent the sensitive attribute Sex and the Age of loan applicant.

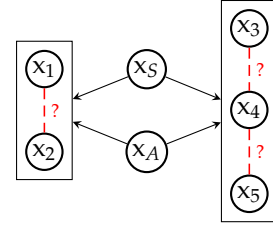


Figure E.4: Partial causal graph used for the Credit dataset [48]. Rectangles show the SCCs grouping different variables. Solid arrows represent causal relationships between SCCs, and dashed arrows represent an arbitrary order picked to learn the joint distribution of each SCC with an ANF. See Subsection 11.4.2 for an in-depth explanation on how to deal with partial causal graphs.

Training. For this section, we performed minimal hyperparameter tuning, and only tested a few combinations by hand. We decided to use a **neural spline flow (NSF)** [50] for the single layer of the Causal NF, which internally uses an feed-forward neural network with 3 layers, and 32 hidden units each. We use Adam [92] as the optimizer, with a learning rate of 0.01, along with a plateau scheduler with a decay factor of 0.9 and a patience parameter of 60 epochs. The training is performed for 1000 epochs, and the results are reported using 5-fold cross-validation with a 80 – 10 – 10 split for train, validation, and test data, respectively.

Results. On addition to the results from Section 12.5, Figure E.5 shows two pair plots from one of the 5 runs, chosen at random. The true empirical distribution is shown in blue, and the learnt distribution by the Causal NF is depicted in orange. Specifically, Figure E.5a illustrates the observational distribution, and Figure E.5b the interventional distribution, obtained when we intervene on the Sex variable and set it to 1, *i.e.*, $do(x_S := 1)$. We can observe that the Causal NF achieves a remarkable fit in both cases, demonstrating its capability to handle discrete data, and partial knowledge of the causal graph.

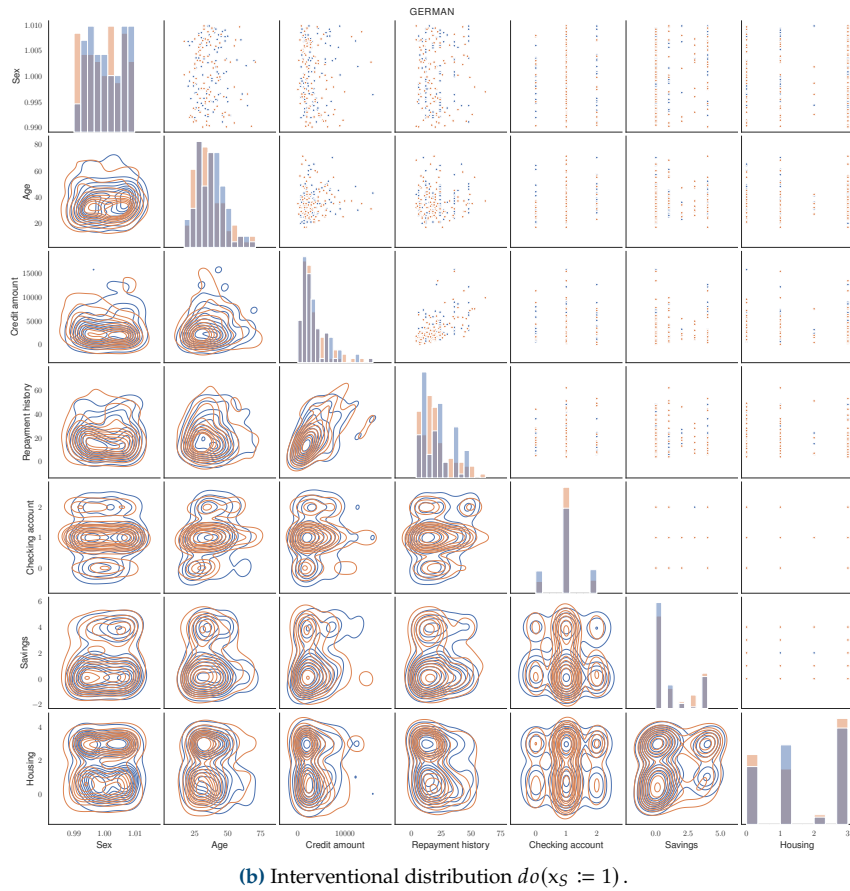
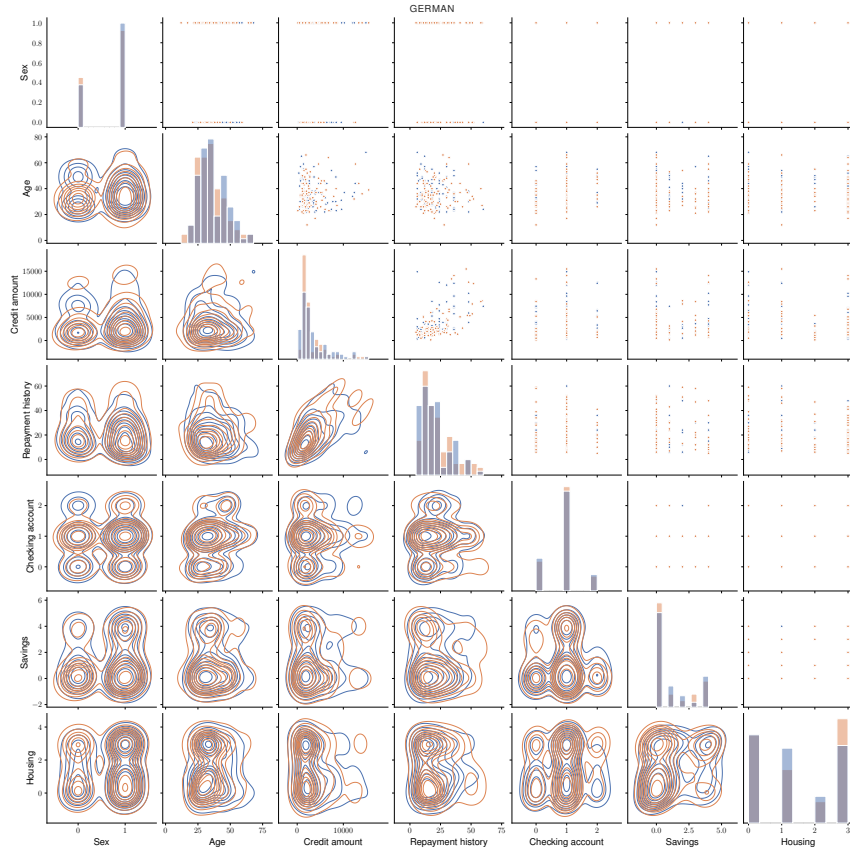


Figure E.5: Pair plot of real (in blue) and generated (in orange) data of the Credit dataset [48]. (a) Samples from the true and learnt observational distributions. (b) Samples from the true and learnt interventional distributions when $do(x_S := 1)$. The plot illustrates that the Causal NF is able to handle discrete data and correctly intervene on them.

Bibliography

- [1] Absil, P., Mahony, R. E., and Sepulchre, R., 'Optimization Algorithms on Matrix Manifolds.' *Princeton University Press*, 2008.
- [2] Ahmed, K., Teso, S., Chang, K., Broeck, G. V., and Vergari, A., 'Semantic Probabilistic Layers for Neuro-Symbolic Learning.' *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Edited by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho and A. Oh. 2022.
- [3] Ahn, C., Kim, E., and Oh, S., 'Deep Elastic Networks With Model Selection for Multi-Task Learning.' *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pages 6528–6537.
- [4] Amari, S., and Nagaoka, H., 'Methods of information geometry.' Volume 191. *American Mathematical Soc.*, 2000.
- [5] Badrinarayanan, V., Kendall, A., and Cipolla, R., 'SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation.' *IEEE Trans. Pattern Anal. Mach. Intell.* 39.12 (2017), pages 2481–2495.
- [6] Balgi, S., Peña, J. M., and Daoud, A., 'Personalized Public Policy Analysis in Social Sciences Using Causal-Graphical Normalizing Flows.' *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*. AAAI Press, 2022, pages 11810–11818.
- [7] Baltrušaitis, T., Ahuja, C., and Morency, L.-P., 'Multimodal machine learning: A survey and taxonomy.' *IEEE transactions on pattern analysis and machine intelligence* 41.2 (2018), pages 423–443.
- [8] Barrejón, D., Olmos, P. M., and Artés-Rodríguez, A., 'Medical data wrangling with sequential variational autoencoders.' *ArXiv preprint abs/2103.07206* (2021).
- [9] Belkin, M., Hsu, D., Ma, S., and Mandal, S., 'Reconciling modern machine-learning practice and the classical bias–variance trade-off.' *Proceedings of the National Academy of Sciences* 116.32 (2019), pages 15849–15854.
- [10] Betancourt, M., 'A conceptual introduction to Hamiltonian Monte Carlo.' *ArXiv preprint abs/1701.02434* (2017).
- [11] Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P. A., Horsfall, P., and Goodman, N. D., 'Pyro: Deep Universal Probabilistic Programming.' *J. Mach. Learn. Res.* 20 (2019), 28:1–28:6.
- [12] Bishop, C. M., 'Pattern Recognition and Machine Learning (Information Science and Statistics).' Berlin, Heidelberg: *Springer-Verlag*, 2006.
- [13] Bishop, C. M., and Bishop, H., 'Deep Learning: Foundations and Concepts.' *Springer*, 2024.
- [14] Boyd, S. P., and Vandenberghe, L., 'Convex optimization.' *Cambridge university press*, 2004.
- [15] Burda, Y., Grosse, R. B., and Salakhutdinov, R., 'Importance Weighted Autoencoders.' *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Edited by Y. Bengio and Y. LeCun. 2016.
- [16] Caruana, R., 'Multitask Learning: A Knowledge-Based Source of Inductive Bias.' *Machine Learning, Proceedings of the Tenth International Conference, University of Massachusetts, Amherst, MA, USA, June 27-29, 1993*. Edited by P. E. Utgoff. *Morgan Kaufmann*, 1993, pages 41–48.
- [17] Caruana, R., 'Multitask Learning'. PhD thesis. 1997.

- [18] Casado, M. L. ‘Trivializations for Gradient-Based Optimization on Manifolds.’ *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Edited by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox and R. Garnett. 2019, pages 9154–9164.
- [19] Casado, M. L., and Martínez-Rubio, D., ‘Cheap Orthogonal Constraints in Neural Networks: A Simple Parametrization of the Orthogonal and Unitary Group.’ *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Edited by K. Chaudhuri and R. Salakhutdinov. Volume 97. Proceedings of Machine Learning Research. PMLR, 2019, pages 3794–3803.
- [20] Casella, G., and Berger, R. L., ‘Statistical inference.’ *Cengage Learning*, 2021.
- [21] Chai, H., Yin, Z., Ding, Y., Liu, L., Fang, B., and Liao, Q., ‘A Model-Agnostic Approach to Mitigate Gradient Interference for Multi-Task Learning.’ *IEEE Transactions on Cybernetics* (2022), pages 1–14.
- [22] Chao, P., Blöbaum, P., and Kasiviswanathan, S. P., ‘Interventional and Counterfactual Inference with Diffusion Models.’ *ArXiv preprint abs/2302.00860* (2023).
- [23] Chatterjee, S., and Zielinski, P., ‘Making coherence out of nothing at all: measuring the evolution of gradient alignment.’ *ArXiv preprint abs/2008.01217* (2020).
- [24] Chen, A. Q., Shi, R., Gao, X., Baptista, R., and Krishnan, R. G., ‘Structured Neural Networks for Density Estimation and Causal Inference.’ *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Edited by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt and S. Levine. 2023.
- [25] Chen, H., Wang, X., Lan, X., Chen, H., Duan, X., Jia, J., and Zhu, W., ‘Curriculum-Listener: Consistency- and Complementarity-Aware Audio-Enhanced Temporal Sentence Grounding.’ *Proceedings of the 31st ACM International Conference on Multimedia* (2023).
- [26] Chen, Z., Badrinarayanan, V., Lee, C., and Rabinovich, A., ‘GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks.’ *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Edited by J. G. Dy and A. Krause. Volume 80. Proceedings of Machine Learning Research. PMLR, 2018, pages 793–802.
- [27] Chen, Z., Ngiam, J., Huang, Y., Luong, T., Kretschmar, H., Chai, Y., and Anguelov, D., ‘Just Pick a Sign: Optimizing Deep Multitask Models with Gradient Sign Dropout.’ *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin. 2020.
- [28] Chennupati, S., Sistu, G., Yogamani, S. K., and Rawashdeh, S. A., ‘MultiNet++: Multi-Stream Feature Aggregation and Geometric Loss Strategy for Multi-Task Learning.’ *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2019), pages 1200–1210.
- [29] Chiappa, S. ‘Path-Specific Counterfactual Fairness.’ *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pages 7801–7808.
- [30] Choi, Y., Vergari, A., and Van den Broeck, G., ‘Probabilistic circuits: A unifying framework for tractable probabilistic models.’ *UCLA*. URL: <http://starai.cs.ucla.edu/papers/ProbCirc20.pdf> (2020), page 6.
- [31] Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D., *Deep Learning for Classical Japanese Literature*. 2018. arXiv: [cs.CV/1812.01718](https://arxiv.org/abs/1812.01718) [[cs.CV](https://arxiv.org/abs/1812.01718)].
- [32] Clevert, D., Unterthiner, T., and Hochreiter, S., ‘Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs).’ *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Edited by Y. Bengio and Y. LeCun. 2016.
- [33] Cohen, G., Afshar, S., Tapson, J., and Van Schaik, A., ‘EMNIST: Extending MNIST to handwritten letters.’ *2017 international joint conference on neural networks (IJCNN)*. IEEE. 2017, pages 2921–2926.

- [34] Collins, L., Mokhtari, A., and Shakkottai, S., ‘Task-Robust Model-Agnostic Meta-Learning.’ *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin. 2020.
- [35] Cortes, C., and Vapnik, V., ‘Support-vector networks.’ *Machine learning* 20.3 (1995), pages 273–297.
- [36] Cortés, J. ‘Finite-time convergent gradient flows with applications to network consensus.’ *Autom.* 42.11 (2006), pages 1993–2000.
- [37] Couprie, C., Farabet, C., Najman, L., and LeCun, Y., ‘Indoor Semantic Segmentation using depth information.’ *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Conference Track Proceedings*. Edited by Y. Bengio and Y. LeCun. 2013.
- [38] Cox, D. R. ‘The regression analysis of binary sequences.’ *Journal of the Royal Statistical Society: Series B (Methodological)* 20.2 (1958), pages 215–232.
- [39] Cummings, R., Gupta, V., Kimpara, D., and Morgenstern, J., ‘On the Compatibility of Privacy and Fairness.’ *Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization. UMAP’19 Adjunct*. Larnaca, Cyprus: Association for Computing Machinery, 2019, pages 309–315.
- [40] D’Amour, A., Heller, K., Moldovan, D., Adlam, B., Alipanahi, B., Beutel, A., Chen, C., Deaton, J., Eisenstein, J., and Hoffman, M. D., ‘Underspecification presents challenges for credibility in modern machine learning.’ *Journal of Machine Learning Research* 23.226 (2022), pages 1–61.
- [41] Dance, H., and Bloem-Reddy, B., ‘Causal Inference with Cocycles.’ *ArXiv preprint abs/2405.13844* (2024).
- [42] Dandi, Y., Barba, L., and Jaggi, M., ‘Implicit Gradient Alignment in Distributed and Federated Learning.’ *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*. AAAI Press, 2022, pages 6454–6462.
- [43] Dayma, B. *DALL-E Mega - Training Journal*. 2022. URL: <https://wandb.ai/dalle-mini/dalle-mini/reports/DALL-E-Mega-Training-Journal--VmlldzoxODMxMDI2> (visited on 09/06/2022).
- [44] Désidéri, J.-A. ‘Multiple-gradient descent algorithm (MGDA) for multiobjective optimization.’ *Comptes Rendus Mathématique* 350.5 (2012), pages 313–318.
- [45] Désidéri, J.-A. ‘Multiple-gradient descent algorithm (MGDA) for multiobjective optimization.’ *Comptes Rendus Mathématique* 350.5-6 (2012), pages 313–318.
- [46] Dinh, A., Liu, D., and Xu, C., ‘PixelAsParam: A Gradient View on Diffusion Sampling with Guidance.’ *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*. Edited by A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato and J. Scarlett. Volume 202. Proceedings of Machine Learning Research. PMLR, 2023, pages 8120–8137.
- [47] Du, Y., Czarnecki, W. M., Jayakumar, S. M., Farajtabar, M., Pascanu, R., and Lakshminarayanan, B., ‘Adapting auxiliary losses using gradient similarity.’ *ArXiv preprint abs/1812.02224* (2018).
- [48] Dua, D., and Graff, C., ‘UCI Machine Learning Repository.’ 2017. URL: <http://archive.ics.uci.edu/ml>.
- [49] Duong, L., Cohn, T., Bird, S., and Cook, P., ‘Low Resource Dependency Parsing: Cross-lingual Parameter Sharing in a Neural Network Parser.’ *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Edited by C. Zong and M. Strube. Beijing, China: Association for Computational Linguistics, 2015, pages 845–850.
- [50] Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G., ‘Neural Spline Flows.’ *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Edited by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox and R. Garnett. 2019, pages 7509–7520.

- [51] Eberhardt, F., and Scheines, R., 'Interventions and Causal Inference.' *Philosophy of Science* 74.5 (2007), pages 981–995. (Visited on 14/05/2023).
- [52] Ektefaie, Y., Dasoulas, G., Noori, A., Farhat, M., and Zitnik, M., 'Geometric multimodal representation learning.' *ArXiv preprint abs/2209.03299* (2022).
- [53] Ektefaie, Y., Dasoulas, G., Noori, A., Farhat, M., and Zitnik, M., 'Multimodal learning with graphs.' *Nature Machine Intelligence* 5 (2022), pages 340–350.
- [54] Everitt, B. S. 'An introduction to finite mixture distributions.' *Statistical methods in medical research* 5.2 (1996), pages 107–127.
- [55] Falkner, S., Klein, A., and Hutter, F., 'BOHB: Robust and Efficient Hyperparameter Optimization at Scale.' *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Edited by J. G. Dy and A. Krause. Volume 80. Proceedings of Machine Learning Research. PMLR, 2018, pages 1436–1445.
- [56] Fan, D., Hou, Y., and Gao, C., 'Cf-vae: Causal disentangled representation learning with vae and causal flows.' *ArXiv preprint abs/2304.09010* (2023).
- [57] Farajtabar, M., Azizian, N., Mott, A., and Li, A., 'Orthogonal Gradient Descent for Continual Learning.' *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*. Edited by S. Chiappa and R. Calandra. Volume 108. Proceedings of Machine Learning Research. PMLR, 2020, pages 3762–3773.
- [58] Fiez, T., Chasnov, B., and Ratliff, L. J., 'Implicit Learning Dynamics in Stackelberg Games: Equilibria Characterization, Convergence Analysis, and Empirical Study.' *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Volume 119. Proceedings of Machine Learning Research. PMLR, 2020, pages 3133–3144.
- [59] Fifty, C., Amid, E., Zhao, Z., Yu, T., Anil, R., and Finn, C., 'Efficiently Identifying Task Groupings for Multi-Task Learning.' *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. Edited by M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang and J. W. Vaughan. 2021, pages 27503–27516.
- [60] Flennerhag, S., Rusu, A. A., Pascanu, R., Visin, F., Yin, H., and Hadsell, R., 'Meta-Learning with Warped Gradient Descent.' *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. *OpenReview.net*, 2020.
- [61] Fliege, J., and Svaiter, B. F., 'Steepest descent methods for multicriteria optimization.' *Mathematical methods of operations research* 51.3 (2000), pages 479–494.
- [62] Fort, S., Nowak, P. K., Jastrzebski, S., and Narayanan, S., 'Stiffness: A new perspective on generalization in neural networks.' *ArXiv preprint abs/1901.09491* (2019).
- [63] Geffner, T., Antorán, J., Foster, A., Gong, W., Ma, C., Kiciman, E., Sharma, A., Lamb, A., Kukla, M., Pawlowski, N., Allamanis, M., and Zhang, C., 'Deep End-to-end Causal Inference.' *ArXiv preprint abs/2202.02195* (2022).
- [64] Gendron, G., Witbrock, M., and Dobbie, G., 'A Survey of Methods, Challenges and Perspectives in Causality.' *ArXiv preprint abs/2302.00293* (2023).
- [65] Ghosh, P., Sajjadi, M. S. M., Vergari, A., Black, M. J., and Schölkopf, B., 'From Variational to Deterministic Autoencoders.' *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. *OpenReview.net*, 2020.
- [66] Glorot, X., and Bengio, Y., 'Understanding the difficulty of training deep feedforward neural networks.' *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Edited by Y. W. Teh and M. Titterton. Volume 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, 2010, pages 249–256.
- [67] Goodfellow, I. J., Bengio, Y., and Courville, A. C., 'Deep Learning.' *Adaptive computation and machine learning*. MIT Press, 2016.

- [68] Gorgolewski, K., Burns, C., Madison, C., Clark, D., Halchenko, Y., Waskom, M., and Ghosh, S., 'Nipype: A Flexible, Lightweight and Extensible Neuroimaging Data Processing Framework in Python.' *Frontiers in Neuroinformatics* 5 (2011).
- [69] Guo, P., Lee, C., and Ulbricht, D., 'Learning to Branch for Multi-Task Learning.' *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Volume 119. Proceedings of Machine Learning Research. PMLR, 2020, pages 3854–3863.
- [70] Guo, W., Wang, J., and Wang, S., 'Deep multimodal representation learning: A survey.' *IEEE Access* 7 (2019), pages 63373–63394.
- [71] Gutknecht, A. J., Wibral, M., and Makkeh, A., 'Bits and pieces: understanding information decomposition from part-whole relationships and formal logic.' *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 477.2251 (2021), page 20210110. eprint: <https://royalsocietypublishing.org/doi/pdf/10.1098/rspa.2021.0110>.
- [72] Hadsell, R., Rao, D., Rusu, A. A., and Pascanu, R., 'Embracing change: Continual learning in deep neural networks.' *Trends in cognitive sciences* 24.12 (2020), pages 1028–1040.
- [73] Han, J., Pei, J., and Kamber, M., 'Data mining: concepts and techniques.' *Elsevier*, 2011.
- [74] He, K., Zhang, X., Ren, S., and Sun, J., 'Deep Residual Learning for Image Recognition.' *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pages 770–778.
- [75] Ho, J., Jain, A., and Abbeel, P., 'Denoising Diffusion Probabilistic Models.' *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin. 2020.
- [76] Hornik, K., Stinchcombe, M., and White, H., 'Multilayer feedforward networks are universal approximators.' *Neural networks* 2.5 (1989), pages 359–366.
- [77] Hoyer, P. O., Janzing, D., Mooij, J. M., Peters, J., and Schölkopf, B., 'Nonlinear causal discovery with additive noise models.' *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*. Edited by D. Koller, D. Schuurmans, Y. Bengio and L. Bottou. Curran Associates, Inc., 2008, pages 689–696.
- [78] Huang, Z., Sang, Y., Sun, Y., and Lv, J., 'A neural network learning algorithm for highly imbalanced data classification.' *Information Sciences* 612 (2022), pages 496–513.
- [79] Hyvärinen, A. 'Painful intelligence: What AI can tell us about human suffering.' *ArXiv preprint abs/2205.15409* (2022).
- [80] Ioffe, S., and Szegedy, C., 'Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.' *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. Edited by F. R. Bach and D. M. Blei. Volume 37. JMLR Workshop and Conference Proceedings. JMLR.org, 2015, pages 448–456.
- [81] Jang, E., Gu, S., and Poole, B., 'Categorical Reparameterization with Gumbel-Softmax.' *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
- [86] Jin, C., Netrapalli, P., and Jordan, M. I., 'What is Local Optimality in Nonconvex-Nonconcave Minimax Optimization?' *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Volume 119. Proceedings of Machine Learning Research. PMLR, 2020, pages 4880–4889.
- [87] Kamiran, F., and Calders, T., 'Data preprocessing techniques for classification without discrimination.' *Knowledge and information systems* 33.1 (2012), pages 1–33.
- [88] Karimi, A., Kügelgen, B. J., Schölkopf, B., and Valera, I., 'Algorithmic recourse under imperfect causal knowledge: a probabilistic approach.' *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin. 2020.

- [89] Kendall, A., Gal, Y., and Cipolla, R., 'Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics.' *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018. IEEE Computer Society, 2018*, pages 7482–7491.
- [90] Khemakhem, I., Kingma, D. P., Monti, R. P., and Hyvärinen, A., 'Variational Autoencoders and Nonlinear ICA: A Unifying Framework.' *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]. Edited by S. Chiappa and R. Calandra. Volume 108. Proceedings of Machine Learning Research. PMLR, 2020*, pages 2207–2217.
- [91] Khemakhem, I., Monti, R. P., Leech, R., and Hyvärinen, A., 'Causal Autoregressive Flows.' *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event. Edited by A. Banerjee and K. Fukumizu. Volume 130. Proceedings of Machine Learning Research. PMLR, 2021*, pages 3520–3528.
- [92] Kingma, D. P., and Ba, J., 'Adam: A Method for Stochastic Optimization.' *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. Edited by Y. Bengio and Y. LeCun. 2015*.
- [93] Kingma, D. P., and Welling, M., 'Auto-Encoding Variational Bayes.' *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings. Edited by Y. Bengio and Y. LeCun. 2014*.
- [94] Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M., 'Improved Variational Inference with Inverse Autoregressive Flow.' *Advances in Neural Information Processing Systems*. Edited by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon and R. Garnett. Volume 29. Curran Associates, Inc., 2016.
- [95] Knothe, H. 'Contributions to the theory of convex bodies.' *Michigan Mathematical Journal* 4 (1957), pages 39–52.
- [96] Kocaoglu, M., Snyder, C., Dimakis, A. G., and Vishwanath, S., 'CausalGAN: Learning Causal Implicit Generative Models with Adversarial Training.' *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018*.
- [97] Krizhevsky, A., and Hinton, G., 'Learning multiple layers of features from tiny images.' (2009).
- [98] Kumar, A., and III, H. D., 'Learning Task Grouping and Overlap in Multi-task Learning.' *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012. icml.cc / Omnipress, 2012*.
- [99] Kurin, V., Palma, A. D., Kostrikov, I., Whiteson, S., and Mudigonda, P. K., 'In Defense of the Unitary Scalarization for Deep Multi-Task Learning.' *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022. Edited by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho and A. Oh. 2022*.
- [100] Kusner, M. J., Loftus, J. R., Russell, C., and Silva, R., 'Counterfactual Fairness.' *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA. Edited by I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan and R. Garnett. 2017*, pages 4066–4076.
- [101] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P., 'Gradient-based learning applied to document recognition.' *Proceedings of the IEEE* 86.11 (1998), pages 2278–2324.
- [102] LeCun, Y., Cortes, C., and Burges, C., 'MNIST handwritten digit database.' *ATT Labs [Online]* 2 (2010).
- [103] Levi, H., and Ullman, S., 'Multi-Task Learning by a Top-Down Control Network.' *ArXiv preprint abs/2002.03335* (2020), pages 2553–2557.
- [104] Li, X., and Gong, H., 'Robust Optimization for Multilingual Translation with Imbalanced Data.' *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual. Edited by M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang and J. W. Vaughan. 2021*, pages 25086–25099.

- [105] Liang, W., Kekic, A., Kügelgen, J., Buchholz, S., Besserve, M., Gresele, L., and Schölkopf, B., ‘Causal Component Analysis.’ *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Edited by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt and S. Levine. 2023.
- [106] Lin, B., Ye, F., Zhang, Y., and Tsang, I. W.-H., ‘Reasonable Effectiveness of Random Weighting: A Litmus Test for Multi-Task Learning.’ 2021.
- [107] Lin, X., Yang, Z., Zhang, Q., and Kwong, S., ‘Controllable pareto multi-task learning.’ *ArXiv preprint abs/2010.06313* (2020).
- [108] Lin, X., Zhen, H., Li, Z., Zhang, Q., and Kwong, S., ‘Pareto Multi-Task Learning.’ *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Edited by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox and R. Garnett. 2019, pages 12037–12047.
- [109] Lin, X., Baweja, H. S., Kantor, G., and Held, D., ‘Adaptive Auxiliary Task Weighting for Reinforcement Learning.’ *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Edited by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox and R. Garnett. 2019, pages 4773–4784.
- [110] Liu, B., Feng, Y., Stone, P., and Liu, Q., ‘FAMO: Fast Adaptive Multitask Optimization.’ *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Edited by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt and S. Levine. 2023.
- [111] Liu, B., Liu, X., Jin, X., Stone, P., and Liu, Q., ‘Conflict-Averse Gradient Descent for Multi-task learning.’ *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. Edited by M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang and J. W. Vaughan. 2021, pages 18878–18890.
- [112] Liu, L., Li, Y., Kuang, Z., Xue, J., Chen, Y., Yang, W., Liao, Q., and Zhang, W., ‘Towards Impartial Multi-task Learning.’ *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. *OpenReview.net*, 2021.
- [113] Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J., ‘On the Variance of the Adaptive Learning Rate and Beyond.’ *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. *OpenReview.net*, 2020.
- [114] Liu, S., James, S., Davison, A. J., and Johns, E., ‘Auto-Lambda: Disentangling Dynamic Task Relationships.’ *ArXiv preprint abs/2202.03091* (2022).
- [115] Liu, S., Johns, E., and Davison, A. J., ‘End-To-End Multi-Task Learning With Attention.’ *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. *Computer Vision Foundation / IEEE*, 2019, pages 1871–1880.
- [116] Liu, Y., Zhang, K., Ren, X., Huang, Y., Jin, J., Qin, Y., Su, R., Xu, R., and Zhang, W., ‘An Aligning and Training Framework for Multimodal Recommendations.’ *ArXiv preprint abs/2403.12384* (2024).
- [117] Liu, Z., Luo, P., Wang, X., and Tang, X., ‘Deep Learning Face Attributes in the Wild.’ *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. *IEEE Computer Society*, 2015, pages 3730–3738.
- [118] Long, M., Cao, Z., Wang, J., and Yu, P. S., ‘Learning Multiple Tasks with Multilinear Relationship Networks.’ *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Edited by I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan and R. Garnett. 2017, pages 1594–1603.
- [119] Lyle, C., Wilk, M., Kwiatkowska, M., Gal, Y., and Bloem-Reddy, B., ‘On the benefits of invariance in neural networks.’ *ArXiv preprint abs/2005.00178* (2020).
- [120] Ma, C., Tschitschek, S., Turner, R. E., Hernández-Lobato, J. M., and Zhang, C., ‘VAEM: a Deep Generative Model for Heterogeneous Mixed Type Data.’ *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin. 2020.

- [121] Mahapatra, D., and Rajan, V., 'Multi-Task Learning with User Preferences: Gradient Descent with Controlled Ascent in Pareto Optimization.' *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Volume 119. Proceedings of Machine Learning Research. PMLR, 2020, pages 6597–6607.
- [122] Maheshwari, G., and Perrot, M., 'Fairgrad: Fairness aware gradient descent.' *ArXiv preprint abs/2206.10923* (2022).
- [123] Majumdar, A., and Valera, I., 'CARMA: A practical framework to generate recommendations for causal algorithmic recourse at scale.' *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*. FAccT '24. Rio de Janeiro, Brazil: Association for Computing Machinery, 2024, pages 1745–1762.
- [124] Maninis, K., Radosavovic, I., and Kokkinos, I., 'Attentive Single-Tasking of Multiple Tasks.' *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pages 1851–1860.
- [125] Mao, Y., Wang, Z., Liu, W., Lin, X., and Xie, P., 'MetaWeighting: Learning to Weight Tasks in Multi-Task Learning.' *Findings of the Association for Computational Linguistics: ACL 2022*. Edited by S. Muresan, P. Nakov and A. Villavicencio. Dublin, Ireland: Association for Computational Linguistics, 2022, pages 3436–3448.
- [126] Matthews, R. 'Storks deliver babies ($p=0.008$).' *Teaching Statistics* 22.2 (2000), pages 36–38.
- [127] Mehrotra, N., Jyothi, A. A., Durand, T., He, J., Sigal, L., and Mori, G., 'A Variational Auto-Encoder Model for Stochastic Point Processes.' *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pages 3165–3174.
- [128] Messerli, F. H. 'Chocolate consumption, cognitive function, and Nobel laureates.' *N Engl J Med* 367.16 (2012), pages 1562–1564.
- [129] Michel, P., Ruder, S., and Yogatama, D., 'Balancing Average and Worst-case Accuracy in Multitask Learning.' *ArXiv preprint abs/2110.05838* (2021).
- [130] Miettinen, K. 'Nonlinear multiobjective optimization.' Volume 12. Springer Science & Business Media, 1999.
- [131] Misra, I., Shrivastava, A., Gupta, A., and Hebert, M., 'Cross-Stitch Networks for Multi-task Learning.' *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pages 3994–4003.
- [132] Mitchell, R., McKim, J., and Meyer, B., 'Design by contract, by example.' USA: Addison Wesley Longman Publishing Co., Inc., 2001.
- [133] Mitchell, T. M. 'The need for biases in learning generalizations.' (1980).
- [134] Morningstar, W. R., Vikram, S. M., Ham, C., Gallagher, A. G., and Dillon, J. V., 'Automatic Differentiation Variational Inference with Mixtures.' *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*. Edited by A. Banerjee and K. Fukumizu. Volume 130. Proceedings of Machine Learning Research. PMLR, 2021, pages 3250–3258.
- [135] Movahedi, S., Adabinejad, M., Imani, A., Keshavarz, A., Dehghani, M., Shakery, A., and Araabi, B. N., 'A-DARTS: Mitigating Performance Collapse by Harmonizing Operation Selection among Cells.' *ArXiv preprint abs/2210.07998* (2022).
- [136] Muandet, K. 'Impossibility of Collective Intelligence.' *ArXiv preprint abs/2206.02786* (2022).
- [137] Murphy, K. P. 'Probabilistic Machine Learning: An introduction.' MIT Press, 2022.
- [138] Murray, R. W., Swenson, B., and Kar, S., 'Revisiting Normalized Gradient Descent: Fast Evasion of Saddle Points.' *IEEE Trans. Autom. Control*. 64.11 (2019), pages 4818–4824.
- [139] Nadeau, C., and Bengio, Y., 'Inference for the generalization error.' *Machine learning* 52.3 (2003), pages 239–281.

- [140] Nasr-Esfahany, A., Alizadeh, M., and Shah, D., 'Counterfactual Identifiability of Bijective Causal Models.' *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*. Edited by A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato and J. Scarlett. Volume 202. Proceedings of Machine Learning Research. PMLR, 2023, pages 25733–25754.
- [141] Nathan Silberman, P. K., and Fergus, R., 'Indoor Segmentation and Support Inference from RGBD Images.' *ECCV*. 2012.
- [142] Navon, A., Shamsian, A., Achituve, I., Maron, H., Kawaguchi, K., Chechik, G., and Fetaya, E., 'Multi-Task Learning as a Bargaining Game.' *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*. Edited by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu and S. Sabato. Volume 162. Proceedings of Machine Learning Research. PMLR, 2022, pages 16428–16446.
- [143] Nazabal, A., Olmos, P. M., Ghahramani, Z., and Valera, I., 'Handling incomplete heterogeneous data using VAEs.' *Pattern Recognition* 107 (2020), page 107501.
- [144] Nesterov, Y. E. 'Introductory Lectures on Convex Optimization - A Basic Course.' Volume 87. Applied Optimization. Springer, 2004.
- [145] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y., 'Reading digits in natural images with unsupervised feature learning.' *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning* (2011).
- [146] Papamakarios, G., Murray, I., and Pavlakou, T., 'Masked Autoregressive Flow for Density Estimation.' *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Edited by I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan and R. Garnett. 2017, pages 2338–2347.
- [147] Papamakarios, G., Nalisnick, E. T., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B., 'Normalizing Flows for Probabilistic Modeling and Inference.' *J. Mach. Learn. Res.* 22 (2021), 57:1–57:64.
- [148] Parafita, Á., and Vitrià, J., 'Estimand-Agnostic Causal Query Estimation With Deep Causal Graphs.' *IEEE Access* 10 (2022), pages 71370–71386.
- [149] Pascal, L., Michiardi, P., Bost, X., Huet, B., and Zuluaga, M. A., 'Improved optimization strategies for deep multi-task networks.' *ArXiv preprint abs/2109.11678* (2021).
- [150] Pascanu, R., Mikolov, T., and Bengio, Y., 'On the difficulty of training recurrent neural networks.' *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*. Volume 28. JMLR Workshop and Conference Proceedings. JMLR.org, 2013, pages 1310–1318.
- [151] Pawlowski, N., Castro, D. C., and Glocker, B., 'Deep Structural Causal Models for Tractable Counterfactual Inference.' *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin. 2020.
- [152] Pearl, J. 'Causality.' Cambridge University Press, 2009.
- [153] Pearl, J. 'Causal inference in statistics: An overview.' (2009).
- [154] Pearl, J. 'The Do-Calculus Revisited.' *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, August 14-18, 2012*. Edited by N. de Freitas and K. P. Murphy. AUAI Press, 2012, pages 3–11.
- [155] Petkova, V. I., and Ehrsson, H. H., 'If I Were You: Perceptual Illusion of Body Swapping.' *PLOS ONE* 3.12 (2008), pages 1–9.
- [156] Piggins, A. 'Collective Choice and Social Welfare—Expanded Edition.' 2019.
- [157] R Core Team, *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2021.
- [158] Rainforth, T., Kosiorek, A. R., Le, T. A., Maddison, C. J., Igl, M., Wood, F., and Teh, Y. W., 'Tighter Variational Bounds are Not Necessarily Better.' *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, July 10-15, 2018*. Edited by J. G. Dy and A. Krause. Volume 80. Proceedings of Machine Learning Research. PMLR, 2018, pages 4274–4282.

- [159] Ramé, A., Dancette, C., and Cord, M., 'Fishr: Invariant Gradient Variances for Out-of-Distribution Generalization.' *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*. Edited by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu and S. Sabato. Volume 162. Proceedings of Machine Learning Research. PMLR, 2022, pages 18347–18377.
- [160] Ranganath, R., Gerrish, S., and Blei, D. M., 'Black Box Variational Inference.' *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*. Volume 33. JMLR Workshop and Conference Proceedings. JMLR.org, 2014, pages 814–822.
- [161] Reddi, S. J., Kale, S., and Kumar, S., 'On the Convergence of Adam and Beyond.' *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.
- [162] Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-maroon, G., Giménez, M., Sulsky, Y., Kay, J., Springenberg, J. T., Eccles, T., Bruce, J., Razavi, A., Edwards, A., Heess, N., Chen, Y., Hadsell, R., Vinyals, O., Bordbar, M., and Freitas, N., 'A Generalist Agent.' *Transactions on Machine Learning Research* abs/2205.06175 (2022). Featured Certification.
- [163] Robbins, H., and Monro, S., 'A stochastic approximation method.' *The annals of mathematical statistics* (1951), pages 400–407.
- [164] Roeder, G., Wu, Y., and Duvenaud, D., 'Sticking the Landing: Simple, Lower-Variance Gradient Estimators for Variational Inference.' *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Edited by I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan and R. Garnett. 2017, pages 6925–6934.
- [165] Romera-Paredes, B., Aung, H., Bianchi-Berthouze, N., and Pontil, M., 'Multilinear Multitask Learning.' *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*. Volume 28. JMLR Workshop and Conference Proceedings. JMLR.org, 2013, pages 1444–1452.
- [166] Rosenblatt, M. 'Remarks on a Multivariate Transformation.' *The Annals of Mathematical Statistics* 23.3 (1952), pages 470–472. (Visited on 21/04/2023).
- [167] Ruchte, M., and Grabocka, J., 'Multi-task problems are not multi-objective.' *ArXiv preprint* abs/2110.07301 (2021).
- [168] Ruchte, M., and Grabocka, J., 'Scalable pareto front approximation for deep multi-objective learning.' *2021 IEEE international conference on data mining (ICDM)*. IEEE. 2021, pages 1306–1311.
- [169] Ruder, S. 'An Overview of Multi-Task Learning in Deep Neural Networks.' *ArXiv preprint* abs/1706.05098 (2017).
- [170] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., 'Learning representations by back-propagating errors.' *Nature* 323 (1986), pages 533–536.
- [171] Sabour, S., Frosst, N., and Hinton, G. E., 'Dynamic Routing Between Capsules.' *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Edited by I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan and R. Garnett. 2017, pages 3856–3866.
- [172] Salakhutdinov, R., and Mnih, A., 'Probabilistic Matrix Factorization.' *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*. Edited by J. C. Platt, D. Koller, Y. Singer and S. T. Roweis. Curran Associates, Inc., 2007, pages 1257–1264.
- [173] Sanchez, P., and Tsaftaris, S. A., 'Diffusion Causal Models for Counterfactual Estimation.' *CLEaR*. 2022.
- [174] Sánchez-Martín, P., Rateike, M., and Valera, I., 'VACA: Designing Variational Graph Autoencoders for Causal Queries.' *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*. AAAI Press, 2022, pages 8159–8168.

- [175] Sankararaman, K. A., De, S., Xu, Z., Huang, W. R., and Goldstein, T., 'The Impact of Neural Network Overparameterization on Gradient Confusion and Stochastic Gradient Descent.' *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Volume 119. Proceedings of Machine Learning Research. PMLR, 2020, pages 8469–8479.
- [176] Santambrogio, F. 'Optimal transport for applied mathematicians.' *Birkäuser*, NY 55.58-63 (2015), page 94.
- [177] Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A., 'How Does Batch Normalization Help Optimization?' *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Edited by S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett. 2018, pages 2488–2498.
- [178] Schäffler, S., Schultz, R., and Weinzierl, K., 'Stochastic method for the solution of unconstrained vector optimization problems.' *Journal of Optimization Theory and Applications* 114 (2002), pages 209–222.
- [179] Schölkopf, B., and Kügelgen, J., 'From statistical to causal learning.' *ArXiv preprint abs/2204.00607* (2022).
- [180] Schulz, A., Vetter, J., Gao, R., Morales, D., Lobato-Ríos, V., Ramdya, P., Goncalves, P. J., and Macke, J. H., 'Modeling conditional distributions of neural and behavioral data with masked variational autoencoders.' *bioRxiv* (2024).
- [181] Sekhon, J. '271 The Neyman–Rubin Model of Causal Inference and Estimation Via Matching Methods'. *The Oxford Handbook of Political Methodology*. Oxford University Press, 2008. eprint: https://academic.oup.com/book/0/chapter/215142463/chapter-ag-pdf/44586764/book_28340_section_215142463.ag.pdf.
- [182] Sener, O., and Koltun, V., 'Multi-Task Learning as Multi-Objective Optimization.' *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Edited by S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett. 2018, pages 525–536.
- [183] Shamsian, A., Navon, A., Glazer, N., Kawaguchi, K., Chechik, G., and Fetaya, E., 'Auxiliary Learning as an Asymmetric Bargaining Game.' *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*. Edited by A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato and J. Scarlett. Volume 202. Proceedings of Machine Learning Research. PMLR, 2023, pages 30689–30705.
- [184] Shen, J., Zhen, X., Worring, M., and Shao, L., 'Variational Multi-Task Learning with Gumbel-Softmax Priors.' *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. Edited by M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang and J. W. Vaughan. 2021, pages 21031–21042.
- [185] Shi, Y., Narayanaswamy, S., Paige, B., and Torr, P. H. S., 'Variational Mixture-of-Experts Autoencoders for Multi-Modal Deep Generative Models.' *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Edited by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox and R. Garnett. 2019, pages 15692–15703.
- [186] Shi, Y., Paige, B., Torr, P. H. S., and Siddharth, N., 'Relating by Contrasting: A Data-efficient Framework for Multimodal Generative Models.' *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.
- [187] Shi, Y., Seely, J., Torr, P. H. S., Narayanaswamy, S., Hannun, A. Y., Usunier, N., and Synnaeve, G., 'Gradient Matching for Domain Generalization.' *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.
- [188] Simpson, E. H. 'The interpretation of interaction in contingency tables.' *Journal of the Royal Statistical Society: Series B (Methodological)* 13.2 (1951), pages 238–241.
- [189] Sinha, A., Chen, Z., Badrinarayanan, V., and Rabinovich, A., 'Gradient adversarial training of neural networks.' *ArXiv preprint abs/1806.08028* (2018).

- [190] Smith, S. L., Dherin, B., Barrett, D. G. T., and De, S., ‘On the Origin of Implicit Regularization in Stochastic Gradient Descent.’ *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net*, 2021.
- [191] Song, S., Lichtenberg, S. P., and Xiao, J., ‘SUN RGB-D: A RGB-D scene understanding benchmark suite.’ *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015. IEEE Computer Society*, 2015, pages 567–576.
- [192] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R., ‘Dropout: A Simple Way to Prevent Neural Networks from Overfitting.’ *Journal of Machine Learning Research* 15.56 (2014), pages 1929–1958.
- [193] Standley, T., Zamir, A. R., Chen, D., Guibas, L. J., Malik, J., and Savarese, S., ‘Which Tasks Should Be Learned Together in Multi-task Learning?’ *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Volume 119. Proceedings of Machine Learning Research. PMLR, 2020, pages 9120–9132.
- [194] Sun, G., Probst, T., Paudel, D. P., Popovic, N., Kanakis, M., Patel, J., Dai, D., and Gool, L. V., ‘Task Switching Network for Multi-task Learning.’ *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021. IEEE*, 2021, pages 8271–8280.
- [195] Sun, X., Panda, R., Feris, R., and Saenko, K., ‘AdaShare: Learning What To Share For Efficient Deep Multi-Task Learning.’ *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin. 2020.
- [196] Sun, Y., Li, J., and Xu, X., ‘Meta-GF: Training dynamic-depth neural networks harmoniously.’ *European Conference on Computer Vision*. Springer. 2022, pages 691–708.
- [197] Suteu, M., and Guo, Y., ‘Regularizing deep multi-task networks using orthogonal gradients.’ *ArXiv preprint abs/1912.06844* (2019).
- [198] Sutter, T. M., Daunhawer, I., and Vogt, J. E., ‘Multimodal Generative Learning Utilizing Jensen-Shannon-Divergence.’ *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin. 2020.
- [199] Sutter, T. M., Daunhawer, I., and Vogt, J. E., ‘Generalized Multimodal ELBO.’ *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net*, 2021.
- [200] Tarjan, R. ‘Depth-First Search and Linear Graph Algorithms.’ *SIAM Journal on Computing* 1.2 (1972), pages 146–160. eprint: <https://doi.org/10.1137/0201010>.
- [201] Thrun, S., and O’Sullivan, J., ‘Discovering Structure in Multiple Learning Tasks: The TC Algorithm.’ *Machine Learning, Proceedings of the Thirteenth International Conference (ICML ’96), Bari, Italy, July 3-6, 1996*. Edited by L. Saitta. Morgan Kaufmann, 1996, pages 489–497.
- [202] Tucker, G., Lawson, D., Gu, S., and Maddison, C. J., ‘Doubly Reparameterized Gradient Estimators for Monte Carlo Objectives.’ *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net*, 2019.
- [203] Vahdat, A., and Kautz, J., ‘NVAE: A Deep Hierarchical Variational Autoencoder.’ *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin. 2020.
- [204] Von Kügelgen, J., Mohamed, A., and Beckers, S., ‘Backtracking counterfactuals.’ *Conference on Causal Learning and Reasoning*. PMLR. 2023, pages 177–196.
- [205] Wang, P., Zhang, H., Zhu, M., Jiang, X., Qin, J., and Yuan, Y., ‘MGIML: Cancer Grading With Incomplete Radiology-Pathology Data via Memory Learning and Gradient Homogenization.’ *IEEE Transactions on Medical Imaging* 43.6 (2024), pages 2113–2124.
- [206] Wang, Z., Fan, X., Qi, J., Wen, C., Wang, C., and Yu, R., ‘Federated Learning with Fair Averaging.’ *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*. Edited by Z. Zhou. *ijcai.org*, 2021, pages 1615–1623.

- [207] Wang, Z., Lipton, Z. C., and Tsvetkov, Y., 'On Negative Interference in Multilingual Models: Findings and A Meta-Learning Treatment.' *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Edited by B. Webber, T. Cohn, Y. He and Y. Liu. Online: Association for Computational Linguistics, 2020, pages 4438–4450.
- [208] Wang, Z., Tsvetkov, Y., Firat, O., and Cao, Y., 'Gradient Vaccine: Investigating and Improving Multi-task Optimization in Massively Multilingual Models.' *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.
- [209] Waqas, A., Tripathi, A., Ramachandran, R., Stewart, P., and Rasool, G., 'Multimodal Data Integration for Oncology in the Era of Deep Neural Networks: A Review.' *ArXiv preprint abs/2303.06471* (2023).
- [210] Wehenkel, A., and Louppe, G., 'Graphical Normalizing Flows.' *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*. Edited by A. Banerjee and K. Fukumizu. Volume 130. Proceedings of Machine Learning Research. PMLR, 2021, pages 37–45.
- [211] Wen, Y., Li, Z., Xiang, Y., and Reker, D., 'Improving molecular machine learning through adaptive subsampling with active learning.' *Digital Discovery* 2 (4 2023), pages 1134–1142.
- [212] Woodward, J. 'Causation and Manipulability'. *The Stanford Encyclopedia of Philosophy*. Edited by E. N. Zalta and U. Nodelman. Summer 2023. Metaphysics Research Lab, Stanford University, 2023.
- [213] Wu, M., and Goodman, N. D., 'Multimodal Generative Models for Scalable Weakly-Supervised Learning.' *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Edited by S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett. 2018, pages 5580–5590.
- [214] Wu, S., Zhang, H. R., and Ré, C., 'Understanding and Improving Information Transfer in Multi-Task Learning.' *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.
- [215] Wu, Z., Dadu, A., Tustison, N. J., Avants, B. B., Nalls, M. A., Sun, J., and Faghri, F., 'Multimodal Patient Representation Learning with Missing Modalities and Labels.' *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.
- [216] Xi, Q., and Bloem-Reddy, B., 'Indeterminacy in Generative Models: Characterization and Strong Identifiability.' *International Conference on Artificial Intelligence and Statistics, 25-27 April 2023, Palau de Congressos, Valencia, Spain*. Edited by F. J. R. Ruiz, J. G. Dy and J. van de Meent. Volume 206. Proceedings of Machine Learning Research. PMLR, 2023, pages 6912–6939.
- [217] Xi, Q., Gonzalez, S., and Bloem-Reddy, B., 'Triangular Monotonic Generative Models Can Perform Causal Discovery.' *Causal Representation Learning Workshop at NeurIPS 2023*. 2023.
- [218] Xia, K. M., Pan, Y., and Bareinboim, E., 'Neural Causal Models for Counterfactual Identification and Estimation.' *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.
- [219] Xiao, H., Rasul, K., and Vollgraf, R., *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. 2017. arXiv: [cs.LG/1708.07747](https://arxiv.org/abs/1708.07747) [[cs.LG](https://arxiv.org/abs/1708.07747)].
- [220] Xin, D., Ghorbani, B., Gilmer, J., Garg, A., and Firat, O., 'Do Current Multi-Task Optimization Methods in Deep Learning Even Help?' *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Edited by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho and A. Oh. 2022.
- [221] Xu, W., Sun, H., Deng, C., and Tan, Y., 'Variational Autoencoder for Semi-Supervised Text Classification.' *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. Edited by S. P. Singh and S. Markovitch. AAAI Press, 2017, pages 3358–3364.
- [222] Yager, R. R. 'On the analytic representation of the Leximin ordering and its application to flexible constraint propagation.' *European Journal of Operational Research* 102.1 (1997), pages 176–192.

- [223] Yang, E., Pan, J., Wang, X., Yu, H., Shen, L., Chen, X., Xiao, L., Jiang, J., and Guo, G., 'AdaTask: A Task-Aware Adaptive Learning Rate Approach to Multi-Task Learning.' *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*. Edited by B. Williams, Y. Chen and J. Neville. AAAI Press, 2023, pages 10745–10753.
- [224] Yang, M., Liu, F., Chen, Z., Shen, X., Hao, J., and Wang, J., 'CausalVAE: Disentangled Representation Learning via Neural Structural Causal Models.' *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 2021, pages 9593–9602.
- [225] Yang, Y., and Hospedales, T. M., 'Deep Multi-task Representation Learning: A Tensor Factorisation Approach.' *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
- [226] Yang, Y., and Hospedales, T. M., 'Trace Norm Regularised Deep Multi-Task Learning.' *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017.
- [227] Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C., 'Gradient Surgery for Multi-Task Learning.' *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin. 2020.
- [228] Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C., 'Multi-Task Reinforcement Learning without Interference.' ().
- [229] Zafar, M. B., Valera, I., Gomez-Rodriguez, M., and Gummadi, K. P., 'Fairness Constraints: Mechanisms for Fair Classification.' *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*. Edited by A. Singh and X. (Zhu). Volume 54. Proceedings of Machine Learning Research. PMLR, 2017, pages 962–970.
- [230] Zečević, M., Dhami, D. S., Velivcković, P., and Kersting, K., 'Relating Graph Neural Networks to Structural Causal Models.' *ArXiv preprint abs/2109.04173* (2021).
- [231] Zeleny, M. 'Compromise programming.' *Multiple criteria decision making* (1973).
- [232] Zhang, Y., and Yeung, D.-Y., 'A Regularization Approach to Learning Task Relationships in Multitask Learning.' *ACM Trans. Knowl. Discov. Data* 8.3 (2014).
- [233] Zhou, Y., Wang, X., Chen, H., Duan, X., and Zhu, W., 'Intra- and Inter-Modal Curriculum for Multimodal Learning.' *Proceedings of the 31st ACM International Conference on Multimedia* (2023).