On Fairness, Invariance and Memorization in Machine Decision and Deep Learning Algorithms

A dissertation submitted towards the degree
Doctor of Engineering
of the Faculty of Mathematics and Computer Science of
Saarland University

by
Till Speicher

Saarbrücken, 2024

Date of Colloquium:	February 24th, 2025
Dean of Faculty:	Prof. Dr. Roland Speicher
Chair of the Committee:	Prof. Dr. Isabel Valera
Reporters:	Prof. Dr. Krishna P. Gummadi Prof. Dr. Adish Singla Dr. Mariya Toneva
Academic Assistant:	Dr. Soumi Das

Abstract

As learning algorithms become more capable, they are used to tackle an increasingly large spectrum of tasks. Their applications range from understanding images, speech and natural language to making socially impactful decisions, such as about people's eligibility for loans and jobs. Therefore, it is important to better understand both the consequences of algorithmic decisions and the mechanisms by which algorithms arrive at their outputs. Of particular interest in this regard are fairness when algorithmic decisions impact people's lives and the behavior of deep learning algorithms, the most powerful but also opaque type of learning algorithm.

To this end, this thesis makes two contributions: First, we study fairness in algorithmic decision-making. At a conceptual level, we introduce a metric for measuring unfairness in algorithmic decisions based on inequality indices from the economics literature. We show that this metric can be used to decompose the overall unfairness for a given set of users into between- and within-subgroup components and highlight potential tradeoffs between them, as well as between fairness and accuracy. At an empirical level, we demonstrate the necessity for studying fairness in algorithmically controlled systems by exposing the potential for discrimination that is enabled by Facebook's advertising platform. In this context, we demonstrate how advertisers can target ads to exclude users belonging to protected sensitive groups, a practice that is illegal in domains such as housing, employment and finance, and highlight the necessity for better mitigation methods.

The second contribution of this thesis is aimed at better understanding the mechanisms governing the behavior of deep learning algorithms. First, we study the role that invariance plays in learning useful representations. We show that the set of invariances possessed by representations is of critical importance in determining whether they are useful for downstream tasks, more important than many other factors commonly considered to determine transfer performance. Second, we investigate memorization in large language models, which have recently become very popular. By training models to memorize random strings, we uncover a rich and surprising set of dynamics during the memorization process. We find that models undergo two phases during memorization, that strings with lower entropy are harder to memorize, that the memorization dynamics evolve during repeated memorization and that models can recall tokens in random strings with only a very restricted amount of information.

Zusammenfassung

Lernalgorithmen werden immer leistungsfähiger und damit zunehmend zur Bewältigung eines immer breiteren Aufgabenspektrums eingesetzt. Ihre Anwendungen reichen vom Verstehen von Bildern und Sprache bis hin zum Treffen von Entscheidungen, die das Leben von Menschen direkt beeinflussen, wie z.B. über die Vergabe von Krediten und Arbeitsplätzen. Daher ist es wichtig, sowohl die Konsequenzen algorithmischer Entscheidungen als auch die Mechanismen, durch die Algorithmen zu ihren Ergebnissen gelangen, besser zu verstehen. Besonders relevant in diesem Zusammenhang sind ein besseres Verständnis der Fairness algorithmischer Entscheidungen, sowie des Verhaltens von Deep-Learning-Algorithmen, der leistungsfähigsten, aber auch undurchsichtigsten Art von Lernalgorithmen.

Zu diesem Zweck leistet diese Arbeit zwei Beiträge: Zum Ersten untersuchen wir Fairness in algorithmischen Entscheidungen. Auf einer konzeptionellen Ebene führen wir eine Metrik zur Erfassung von Unfairness in algorithmischen Entscheidungen ein, die auf Ungleichverteilungskoeffizienten aus dem Bereich der Ökonomie basiert. Wir zeigen, wie diese Metrik verwendet werden kann, um die Gesamtheit der Unfairness in einer Gruppe von Nutzern in zwei Komponenten zu zerlegen: Unfairness zwischen verschiedenen Subgruppen und Unfairness innerhalb der Subgruppen. Diese beiden Unfairness-Komponenten können miteinander im Konflikt stehen, und auch im Konflikt zur Genauigkeit des Algorithmus. Auf einer praktischen Ebene untersuchen wir das Potenzial zur Diskriminierung, das die Werbeplattform von Facebook bietet. Wir zeigen, wie Werbetreibende Anzeigen schalten können, die Nutzer ausschließen, die zu geschützten, sensiblen Gruppen gehören, eine Praxis, die beim Bewerben von Wohnungen, Arbeitsplätzen und Finanzprodukten illegal ist. Unsere Ergebnisse zeigen, dass bessere Methoden, um Diskriminierung entgegenzuwirken, notwendig sind.

Der zweite Teil dieser Arbeit entwickelt ein besseres Verständnis der Mechanismen, die das Verhalten von Deep-Learning-Algorithmen steuern. Zunächst untersuchen wir die Rolle, die Invarianz beim Lernen nützlicher Repräsentationen von Daten spielt. Wir zeigen, dass welchen Datentransformationen gegenüber Repräsentationen invariant sind, von entscheidender Bedeutung dafür ist, ob sie für bestimmte Aufgaben geeignet sind, wichtiger als viele andere Faktoren, von denen üblicherweise angenommen wird, dass sie den Nutzen von Repräsentationen beeinflussen. Des Weiteren untersuchen wir, wie sich Sprachmodelle (Large Language Models) Daten merken. Wir trainieren Modelle darauf, zufällige Zeichenketten auswendig zu lernen, und beobachten dabei eine Reihe interessanter und überraschender Prozesse während des Lernvorgangs. Modelle durchlaufen zwei Phasen während des Auswendiglernens, es ist für sie schwieriger, sich Zeichenketten mit niedrigerer Entropie zu merken, das wiederholte Auswendiglernen unterschiedlicher Zeichenketten verändert die Lerndynamiken, und nach dem Auswendiglernen können Modelle Zeichen mit nur sehr eingeschränktem Kontext abrufen.

Acknowledgements

I would like to thank my PhD advisor, Krishna P. Gummadi, for his valuable guidance, support, and feedback throughout my PhD. I would also like to thank my colleagues at the Networked Systems group as well as the co-authors I've had the privilege to work with, especially Abhisek Dash, Ayan Majumdar, Bishwamittra Ghosh, Camila Kolling, Evimaria Terzi, Hoda Heidari, Johnnatan Messias, Junaid Ali, Nina Grgić-Hlača, Mohammad Aflah Khan, Muhammad Bilal Zafar, Sepehr Mousavi, Qinyuan Wu, Soumi Das, Vabuk Pahari, Vedant Nanda, and also my other colleagues at MPI-SWS.

I am also grateful for the support I have received from the helpful MPI staff, particularly Annika Meiser, Christian Klein, Claudia Richter, Gretchen Gravelle, Maria-Louise Albrecht, Rose Hoberman, Sarah Naujoks, and Tobias Kaufmann. I would also like to give a shout-out to Carina Schmitt for always being fast to help me with technical issues.

Finally, I would like to thank my friends, my partner, Ashmi, for her encouragement during my PhD, my family, Mechthild, Ewald and Moritz, for their unwavering support.

Contents

Chapter 1

Introduction

As algorithmic decision systems, in particular learning-based ones, are becoming increasingly capable, they are adopted in an ever wider spectrum of domains. Their applications range from image and natural language processing all the way to making decisions in life-affecting scenarios such as criminal risk prediction Angwin et al. [2016], Berk et al. [2017] and credit risk assessments Furletti [2002]. The breadth and impact of these applications has prompted research into the properties, working mechanisms and consequences of using these systems on a broad scale. Among the most important concerns are the potential for unfairness when algorithmic decision systems can affect people's lives Barocas and Selbst [2016a], Podesta et al. [2014], Romei and Ruggieri [2014], as well as understanding the mechanisms that determine the behavior of learning algorithms.

In this thesis, we address open problems in both of these areas. For fairness, we propose ways to measure unfairness in algorithmic decisions and practically demonstrate the existence of potential unfairness on Facebook's advertising platform. To better understand the behavior of learning algorithms, we study deep learning (DL) models, which are seeing increasingly widespread adoption but are also notoriously difficult to interpret. In particular, we focus on the role of invariance in representation learning as well as the dynamics of memorization in large language models (LLMs).

1.1 Thesis Contributions

This thesis makes contributions in four areas of research. We introduce each of them separately.

1.1.1 Measuring Unfairness Using Inequality Indices

As algorithmic decision making systems are increasingly used in life-affecting scenarios such as criminal risk prediction [Angwin et al., 2016, Berk et al., 2017] and credit risk assessments [Furletti, 2002], concerns have risen about the potential unfairness of these decisions to certain social groups or individuals [Barocas and Selbst, 2016a, Podesta et al., 2014, Romei and Ruggieri, 2014]. In response, a number of works have proposed learning mechanisms for fair decision making by imposing additional constraints or *conditions* [Dwork et al., 2012, Feldman et al., 2015, Hardt et al., 2016, Joseph et al., 2016, Zafar et al., 2017a,b].

In the first part of this thesis, we focus on a simple yet foundational question about unfairness of algorithms: *Given two unfair algorithms, how should we determine which of the two is more unfair?* Prior works on algorithmic fairness largely focus on formally defining *conditions* for fairness, but do not precisely define suitable *measures* for unfairness. That is, they can answer the binary question: *is an algorithm fair or unfair?*, but do not have a principled way to answer the nuanced question: *if an algorithm is unfair, how unfair is it?*

Figure 1.1 illustrates the questions we seek to answer through an example of two binary classifiers $\mathcal{C}_1$ and $\mathcal{C}_2$, whose decisions affect 10 individuals belonging to 3 different groups. The

Individuals		i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}
Groups		g_1	g_1	g_2	g_2	g_2	g_2	g_3	g_3	g_3	g_3
True Labels		1	0	0	1	0	0	1	0	1	1
Predicted Labels	$\mathcal{C}_1$	1	0	0	0	1	1	1	0	1	0
	$\mathcal{C}_2$	0	1	1	0	0	0	0	1	1	1

	$\mathcal{C}_1$				$\mathcal{C}_2$			
	g1	**g2**	**g3**	**Fair?**	**g1**	**g2**	**g3**	**Fair?**
FPR	0.00	0.67	0.00	✗	1.00	0.33	1.00	✗
FNR	0.00	1.00	0.33	✗	1.00	1.00	0.33	✗
FDR	0.00	1.00	0.00	✗	1.00	1.00	0.33	✗
FOR	0.00	0.50	0.50	✗	1.00	0.33	1.00	✗
AR	0.50	0.50	0.50	✓	0.50	0.25	0.75	✗
Acc.	1.00	0.25	0.75	✗	0.00	0.50	0.50	✗
Individual Fairness	Rejects $\frac{2}{5}$ deserving users Accepts $\frac{2}{5}$ undeserving users				Rejects $\frac{3}{5}$ deserving users Accepts $\frac{3}{5}$ undeserving users			

Figure 1.1: **[Example of two unfair classifiers.]** [Top] A set of ten users along with their true labels, $y \in \{0, 1\}$, and predicted labels, $\hat{y} \in \{0, 1\}$, by two classifiers $\mathcal{C}_1$ and $\mathcal{C}_2$. Label 1 represents a *more desirable* outcome (*e.g.*, receiving a loan) than label 0. The users belong to three different groups: g_1 (red), g_2 (green), and g_3 (blue). [Bottom] Fairness of the classifiers according to various group- and individual-level metrics. The group-level metrics are false positive rate (FPR), false negative rate (FNR), false discovery rate (FDR), false omission rate (FOR), acceptance rate in desirable class (AR), accuracy (Acc.). The individual-level metric requires individuals deserving similar outcomes (i.e., with similar true lables) to receive similar outcomes (i.e., receive similar predicted labels), according to the Lipschitz condition Dwork et al. [2012]. The table also shows information about whether the classifier is fair w.r.t. the corresponding conditions or not. The fairness conditions are described in detail in Figure 1.2. We note that while both $\mathcal{C}_1$ and $\mathcal{C}_2$ are individually unfair —they violate the Lipschitz condition— according to our unfairness measure, the unfairness of $\mathcal{C}_1$ is 0.2 whereas the unfairness of $\mathcal{C}_2$ is 0.3. Hence, $\mathcal{C}_2$ is more individually unfair than $\mathcal{C}_1$. One can similarly quantify and compare unfairness based on other fairness notions described in Figure 1.2.

figure shows that both $\mathcal{C}_1$ and $\mathcal{C}_2$ yield unequal false positive/negative rates across the 3 groups and are thus unfair at the level of groups—which set of unequal false positive/negative rates ($\mathcal{C}_1$'s or $\mathcal{C}_2$'s) are more unfair? Similarly, both $\mathcal{C}_1$ and $\mathcal{C}_2$ violate our individual-level fairness condition for "treating individuals deserving similar outcomes similarly", but do so in different ways—whose violation of our individual fairness condition is more unfair?

We argue that how we address the unfairness measurement question has significant practical consequences. First, several studies have observed that satisfying multiple fairness conditions at the same time is infeasible [Chouldechova, 2016, Corbett-Davies et al., 2017, Kearns et al., 2017, Kleinberg et al., 2017]. Hence in practice, designers often need to select the *least unfair* algorithm from a feasible set of unfair algorithms. Second, when training fair learning models, practitioners face a tradeoff between accuracy and fairness [Flores et al., 2016, Kleinberg et al., 2017]. These tradeoffs rely on model-specific fairness measures (i.e., proxies chosen for computational tractability) that do not *generalize* across different models. Consequently, they cannot be used to compare accuracy-unfairness tradeoffs of models trained using different fair learning algorithms. Finally, designers of fair learning models make a number of *ad hoc or implicit* choices about fairness measures without explicit justification; for instance, it is unclear why in many

	Fairness Notion		Fairness Condition	Benefit Function TP	TN	FP	FN
Group Fairness	Parity Mistreatment	Accuracy	Equal accuracy for all groups	1	1	0	0
		Equal Opportunity	Equal FPR for all groups	n/a	1	0	n/a
			Equal FNR for all groups	1	n/a	n/a	0
		Well-calibration	Equal FDR for all groups	1	n/a	0	n/a
			Equal FOR for all groups	n/a	1	n/a	0
	Parity Impact / Statistical Parity		Equal acceptance rate for all groups	1	0	1	0
Individual Fairness	Our proposal		Individuals *deserving* similar outcomes *receive* similar outcomes	1	1	2	0

Figure 1.2: **[A summary of different fairness notions and their corresponding fairness conditions.]** We also show a benefit function that we use to compute inequality under each of the fairness conditions. Since all outcomes of a classifier can be decomposed into true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), the benefit function needs to assign a benefit score to each of these prediction types under any given fairness notion. For example, under statistical parity, which requires equal acceptance rates for all groups, we assign a benefit of "1" to TP and FP, and a benefit of "0" to TN and FN. "n/a" under an entry shows that these points are not considered under the corresponding fairness notion (*e.g.*, equality of FPR requires considering only the points with negative true labels, *i.e.*, $y = 1$).

previous works [Dwork et al., 2012, Hardt et al., 2016, Zafar et al., 2017a,b], the relative sizes of the groups in the population are not considered in estimating unfairness—even though these quantities matter when estimating accuracy.

In this thesis, we propose to quantify unfairness using *inequality indices* that have been extensively studied in economics and social welfare [Atkinson, 1970, Cowell and Kuga, 1981, Kakwani, 1980]. Traditionally, inequality indices such as Coefficient of Variation [Abdi, 2010], Gini [Bellù and Liberati, 2006, Gini, 1912], Atkinson [Atkinson, 1970], Hoover [Long and Nucci, 1997], and Theil [Theil, 1967], have been proposed to quantify how unequally incomes are distributed across individuals and groups in a population. Our interest in using these indices is rooted in the *well-justified* axiomatic basis for their designs. Specifically, we argue that many axioms satisfied by inequality indices such as *anonymity*, *population invariance*, *progressive transfer preference*, and *subgroup decomposability* are appealing properties for unfairness measures to satisfy. Thus, inequality indices are naturally well-suited as measures for algorithmic unfairness.

Our core idea is to use existing inequality indices in order to measure *how* ***unequally*** the outcomes of an algorithm **benefit** different individuals or groups in a population. This requires us to define a benefit function that maps the algorithmic output for each individual to a non-negative real number. By adapting the benefit function according to the desired fairness condition, we show that inequality indices can be applied generally to quantify unfairness across all the proposed fairness conditions shown in Figure 1.2. Since we quantify inequality of algorithmic outcomes, our measure is independent of the specifics of any learning model and can be used to compare unfairness of different algorithms.

We consider a family of inequality indices called *generalized entropy indices*, which includes Coefficient of Variation and Theil index as special cases. Generalized entropy indices have a useful property called *subgroup decomposability*. For any division of the population into a set of non-overlapping groups, the property guarantees that our unfairness measure over the entire population can be decomposed as the sum of a *between-group* unfairness component (computed imagining that all individuals in a group receive the group's mean benefit) and a *within-group*

unfairness component (computed as a weighted sum of inequality in benefits received by individuals within each group). Thus, inequality indices not only offer a *unifying* approach to quantifying unfairness at the levels of both individuals and groups, but they also reveal previously overlooked *tradeoffs* between individual-level and group-level fairness.

Further, the decomposition enables us to: (i) quantify how unfair an algorithm is along various sensitive attribute-based groups within a population (e.g., groups based on race, gender or age) and (ii) account for the "gerrymandered" unfairness affecting structured subgroups constructed from "intersecting" the sensitive attribute-groups (e.g., groups like young white women or old black men) [Kearns et al., 2017]. Our empirical evaluations show that existing fair learning methods [Hardt et al., 2016, Zafar et al., 2017a], while successful in eliminating between-group unfairness, (a) may be targeting only a small fraction of the overall unfairness in the decision making algorithms and (b) can result in an increase in within-group unfairness, which paradoxically can lead to training algorithms whose overall unfairness is worse than those trained using traditional learning methods.

To summarize the contributions of this part of the thesis: (i) we propose inequality-indices based unfairness measures that offer a justified and generalizable framework to compare the fairness of a variety of algorithmic predictors against one another, (ii) we theoretically characterize and empirically illustrate the tradeoffs between individual fairness when measured using inequality indices and the prediction accuracy, and (iii) we study the relationship between individual- and group-level unfairness, showing that recently proposed learning mechanisms for mitigating (between-)group unfairness can lead to high within-group unfairness and consequently, high individual unfairness.

1.1.2 Potential for Discrimination in Targeted Advertising

A lot of recent work has focused on detecting instances of discrimination in online services ranging from discriminatory pricing on e-commerce and travel sites like Staples [Mikians et al., 2012] and Hotels.com [Hannák et al., 2014] to discriminatory prioritization of service requests and offerings from certain users over others in crowdsourcing and social networking sites like TaskRabbit [Hannák et al., 2017]. In this part of the thesis, we focus on the potential for discrimination in *online advertising*, which underpins much of the Internet's economy. Specifically, we focus on *targeted advertising*, where ads are shown only to a subset of users that have attributes (features) selected by the advertiser. Targeted ads stand in contrast to non-targeted ads, such as banner ads on websites, that are shown to all users of the sites, independent of their attributes.

The targeted advertising ecosystem comprises of (i) *advertisers*, who decide which users an ad should (not) be shown to; (ii) *ad platforms*, such as Google and Facebook, that aggregate data about their users and make it available to advertisers for targeting; and (iii) *users* of ad platforms that are consumers of the ads. The potential for discrimination in targeted advertising arises from the ability of an advertiser to use the extensive personal (demographic, behavioral, and interests) data that ad platforms gather about their users to target their ads. An intentionally malicious—or unintentionally ignorant—advertiser could leverage such data to preferentially target (i.e., include or exclude from targeting) users belonging to certain sensitive social groups (e.g., minority race, religion, or sexual orientation).

Recently, the Facebook ad platform was the target of intense media scrutiny [Angwin and Parris Jr., 2016] and a civil rights lawsuit for allowing advertisers to target ads with an attribute named "ethnic affinity." After clarifying that a user's "ethnic affinity" does not represent the user's ethnicity, but rather represents how interested the user is in content related to different ethnic communities, Facebook agreed to not allow ads related to housing, employment, and finan-

cial services be targeted using the attribute [Facebook, 2017] and renamed it to "multicultural affinity."[1]

In this chapter, we conduct a systematic study of the potential for discriminatory advertising on the Facebook advertisement platform. We focus on Facebook because it is one of the largest online advertising platforms in terms of number of users reached by ads, the number of advertisers, and the amount of personal data gathered about the users that is made available to advertisers. Furthermore, Facebook is an innovator in introducing new methods for targeting users, such as *custom audience*[2] and *look-alike audience*[3] targeting that are then subsequently adopted by other online social media and social networking platforms like Twitter,[4] Pinterest,[5] LinkedIn,[6] and YouTube.[7] Thus, many of our findings may also be applicable to these other online ad targeting platforms as well.

Our study here is driven by the following high-level question: *What are all the different ways in which a Facebook advertiser, out of malice or ignorance, can target users in a discriminatory manner (i.e., include or exclude users based on their sensitive attributes like race)?*

To answer this question, we begin by proposing an intuitive measure to quantify discrimination in targeted ads. We then systematically investigate three different targeting methods (attribute-based targeting, PII-based targeting, and look-alike audience targeting) offered by Facebook for their ability to enable discriminatory advertising. At a high-level, we find that all three methods enable advertisers to run highly discriminatory ads. Worse, we show that the existing solution approaches of banning the use of certain attributes like "ethnic affinity" in targeting is not only inadequate, but does not even apply in two out of the three ad targeting methods.

While our findings primarily serve to demonstrate the perniciousness of the problem of discriminatory advertising in today's ad platforms, it also lays the foundations for solving (i.e., detecting and mitigating) ad discrimination.

1.1.3 Understanding the Importance of Invariance in Learned Representations

Many learning problems are increasingly solved by adapting pretrained (foundation) models to downstream tasks [Bommasani et al., 2021]. To decide when and how pretrained models can be transferred to new tasks, it is important to understand the factors that determine the transfer performance of their representations. The literature on transfer learning has proposed a number of factors that influence transfer performance. Among them are for instance the accuracy of a model on its training dataset, the architecture and size of the model and the size of the training dataset [Huh et al., 2016, Kolesnikov et al., 2020, Kornblith et al., 2019]. One surprising recent result is that models robust to adversarial attacks, *i.e.* invariant to certain ϵ-ball perturbations, exhibit higher downstream performance on transfer tasks than non-robust ones, despite achieving lower performance on their training dataset [Salman et al., 2020]. Other invariances, such as to textures, have also been found to boost performance [Geirhos et al., 2018a]. These findings suggest *invariance* as another important factor that can influence transfer performance.

---

[1] Unfortunately, Facebook was found half a year later to still accept discriminatory ads, despite the fixes it claims were put in place [Angwin et al., 2017a].

[2] `https://www.facebook.com/business/help/170456843145568`

[3] `https://www.facebook.com/business/help/164749007013531`

[4] `https://business.twitter.com/en/targeting/tailored-audiences.html`

[5] `https://business.pinterest.com/en/blog/new-targeting-tools-make-pinterest-ads-even-more-effective`

[6] `https://business.linkedin.com/marketing-solutions/ad-targeting/matched-audiences`

[7] `https://support.google.com/youtube/answer/2454017`

A model can be said to be invariant to some transformation if its output or representations do not change in response to applying the transformation to its input. Invariance has been recognized as an important property of models and their representations and consequently has received a lot of attention [Bloem-Reddy and Teh, 2020, Cohen et al., 2019, Ericsson et al., 2021b, Lyle et al., 2020]. Most of this work, however, focuses on the role that invariance plays for a specific task. In the case of transfer learning, on the other hand, there are two tasks involved: the task on which the model is trained and the task to which it is transferred. The effect of invariance, both learned during pretraining and required by the downstream task, on the *transfer performance* of representations has received less attention and is not fully understood yet.

One reason why the relationship between invariance and transfer performance has not been more thoroughly explored is that doing so is challenging, especially when only using common real-world datasets. To investigate invariance at a fine-grained level, it is necessary to know the different ways in which inputs to a model differ, in order to determine how those differences relate to changes in representations. This, however, is not possible with typical real-world datasets such as CIFAR-10 [Krizhevsky et al., 2009], ImageNet [Russakovsky et al., 2015] or VTAB [Zhai et al., 2020]. For example, the CIFAR-10 dataset contains images of cats, but using this dataset to assess whether a model is invariant to the position or pose of cats is not possible, since no position or other information is available beyond class labels.

Therefore, to study invariance carefully, we introduce a family of synthetic datasets, Transforms-2D, that allows us to precisely control the differences and similarities between inputs in a model's training and test sets. Using these datasets, we explore the importance of invariance in achieving high transfer performance, as well as how transferable invariance is to new tasks. Concretely, we make the following contributions:

- We introduce a family of synthetic datasets called Transforms-2D, which allows us to carefully control the transformations acting on inputs. By using these datasets, we are able to train models to exhibit specific invariances in their representations and to evaluate their performance on transfer tasks that require specific invariances. We also use them as the basis for measuring invariance to input transformations.

- We investigate the connection between invariance and downstream performance and compare it to other factors commonly studied in the transfer learning literature, such as the number of training samples, the model architecture, the relationship of training and target classes, and the relationship of training- and target-task performance. We find that while these other factors play a role in determining transfer performance, sharing the invariances of the target task is often as or more important. We further show how undesirable invariance can harm the transfer performance of representations.

- We explore the transferability of invariance between tasks and find that in most cases, models can transfer a high degree of learned invariance to out of distribution tasks, which might help explain the importance of invariance for transfer performance.

- While our observations are derived from experiments on synthetic data, we validate them on real-world datasets and find that similar trends hold in these settings.

1.1.4 Studying Memorization in LLMs with Random Strings

The potential for large language models (LLMs) to memorize training data has many important implications. Our goal in this chapter is to create a framework that enables us to better understand, measure and distinguish memorization from other phenomena (*e.g.*, in-context learning) that influence the generation (recollection) of the next token in LLMs. Such distinction is

particularly challenging in an era where models are trained on internet-scale corpora and even open-weight models often have no documentation about the training data they used.

We study memorization by creating a "clean" experimental setting, where we train LLMs to memorize *random strings*. Random strings allow us to study memorization in isolation from other phenomena. Random strings i) guarantee that models have not seen the data during pre-training, ii) ensure that models have to memorize the data in order to achieve low loss (*i.e.* there is no other way to predict tokens better than a distribution-based guess), and iii) give us precise control over all aspects of the data, such as string length, alphabet size, and entropy. Achieving all of these properties with natural language would not be possible. Note that privacy sensitive data in the real-world looks often similar to random strings (e.g., private keys, phone numbers, etc.). Thus, our results are relevant to the memorization abilities of LLMs on such data.

The goal of our experiments is to unveil phenomena that happen when LLMs memorize data. Therefore, in our experiments, we *repeatedly* expose models to random strings since, intuitively, repetition is associated with memorization. Through this process, we can understand the *dynamics* of memorization, *i.e.* when memorization "kicks-in" over the period of repeated exposition to the same or different random strings and how the model behaves.

In order to obtain a comprehensive understanding of the memorization process in LLMs, we perform an extensive set of experiments over both pretrained and untrained models from different families (Pythia, Phi, Llama, GPT and OPT) with parameter counts spanning more than two orders of magnitude. We use random data generated from different distributions, vocabularies and string structures. We train models on random strings appearing in isolation as well as in the context of natural language data.

In all cases we make the same observations: with repeated exposure to the same random string, models memorize the random strings and we consistently observe two phases in this process: the *Guessing-Phase* and the *Memorisation-Phase* phase. In the *Guessing-Phase* a model learns the probability distribution of the tokens in the string; in the *Memorisation-Phase* it memorizes the next token based on prefixes and this is when memorization actually happens. We further analyse the memorization process, motivated by the following questions:

Q1: *Are some strings easier to memorize than others?* To address this question, we experiment with strings of different alphabet sizes and different entropy over the distribution of their tokens. We find that the entropy of the distribution of tokens in the random strings affects the two-phase dynamics: in the *Guessing-Phase*, strings with lower entropy are predicted better, however, in the subsequent *Memorisation-Phase* strings with higher entropy are memorized faster.

Q2: *What information do models need in order to recall memorized tokens?* We address this question by investigating the role of *exact prefixes* in next-token recollection. We find that as the number of repeated exposures to the random string increases, the shorter the prefix needed for next-token recollection become. Somewhat surprisingly, we also find that the exact prefix alone is not sufficient. *Global context*, *i.e.*, knowledge of the probability distribution of the tokens in the string, significantly increases next-token recollection accuracy.

Q3: *How do models behave when asked to memorize different random strings sequentially?* We also train models to *sequentially* memorize different random strings – one at a time. Our results indicate that models forget old random strings when they get repeatedly exposed to new ones. However, as models get exposed to more random strings they forget less, and they memorize new strings faster.

Contrary to related work on memorization that aims to quantify privacy risks, we focus on understanding and revealing the intricacies of the memorization process in LLMs. Some of the phenomena we unveil as we tackle the above questions are surprising and unexpected. Many have significant implications for quantifying memorization, understanding how memorization works, and estimating the risks of memorising different types of training data, including uncovering

new threats with priming models for memorising data. For many of the phenomena we do not have clear explanations as of why they happen but we feel it is important to report them to the community as they rule out certain theories related to memorization and give rise to new ones. We also think that our findings can motivate further studies that will increase our understanding of memorization.

1.2 Thesis Outline

Due to the diversity of the topics covered in this thesis, it is structured into four self-contained chapters.

- In Chapter 2, we discuss our unified approach to measuring unfairness in algorithmic decision systems using inequality indices.
- In Chapter 3, we investigate the potential for discrimination in targeted advertising on Facebook.
- In Chapter 4, we study the role representational invariance plays for the transfer performance.
- In Chapter 5, we study the memorization in large language models, using random strings.

We discuss the respective related work separately in each chapter.

1.3 Publications

The work in this thesis is based on the following publications:

- The work on measuring unfairness using inequality indices described in Chapter 2 is published as "A Unified Approach to Quantifying Algorithmic Unfairness: Measuring Individual & Group Unfairness via Inequality Indices", **T. Speicher**, H. Heidari, N. Grigic-Hlaca, K. P. Gummadi, A. Singla, A. Weller, and M. B. Zafar. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018* [Speicher et al., 2018b].
- The work on exposing the potential for discrimination in targeted advertising described in Chapter 3 is published as "Potential for Discrimination in Online Targeted Advertising", **T. Speicher**, M. Ali, G. Venkatadri, F. N. Ribeiro, G. Arvanitakis, F. Benevenuto, K. P. Gummadi, P. Loiseau, and A. Mislove. In *Proceedings of the 2018 Conference on Fairness, Accountability, and Transparency, FAccT 2018* [Speicher et al., 2018a].
- The work on understanding the impact of invariance on the transfer performance of representations described in Chapter 4 is accepted as "Understanding the Role of Invariance in Transfer Learning", **T. Speicher**, V. Nanda, and K. P. Gummadi. In *Transactions on Machine Learning Research, TMLR 2024*.
- The work on understanding memorization in LLMs in Chapter 5 is complete and currently under review.

Chapter 2

Measuring Unfairness in Algorithmic Decisions via Inequality Indices

Discrimination via algorithmic decision-making has received considerable attention. Prior work largely focuses on defining *conditions* for fairness, but does not define satisfactory *measures* of algorithmic unfairness. In this part of the thesis, we focus on the following question: Given two unfair algorithms, how should we determine which of the two is more unfair?

Our core idea is to use existing inequality indices from economics to measure how unequally the outcomes of an algorithm benefit different individuals or groups in a population. We propose a general framework that enables us to quantify unfairness both at the individual and the group level. Using this approach we find overlooked tradeoffs between different fairness notions: using our proposed measures, the *overall individual-level* unfairness of an algorithm can be decomposed into a *between-group* and a *within-group* component. However, we demonstrate that minimizing exclusively the between-group component — which is what many earlier methods do — may, in fact, increase the within-group, and hence the overall unfairness. We characterize and illustrate the tradeoffs between our measures of (un)fairness and the prediction accuracy.

Relevant publication: The work in this chapter has been published as "A Unified Approach to Quantifying Algorithmic Unfairness: Measuring Individual & Group Unfairness via Inequality Indices", **T. Speicher**, H. Heidari, N. Grigic-Hlaca, K. P. Gummadi, A. Singla, A. Weller, and M. B. Zafar. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018* [Speicher et al., 2018b].

2.1 Measuring Algorithmic Unfairness via Inequality Indices

We first formally describe the setup of a fairness-aware machine learning task; then proceed to show that by defining an appropriate benefit function, existing inequality indices can be applied across the board to quantify algorithmic unfairness. We describe important properties (axioms) which we suggest a reasonable measure of algorithmic unfairness must satisfy. We end this section by comparing our proposed approach with previous work.

2.1.1 Setting

We consider the standard supervised learning setting: A learning algorithm receives a training data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ consisting of n instances, where $\mathbf{x}_i \in \mathcal{X}$ specifies the feature vector[1] for an individual i (*e.g.*, $\mathbf{x}_i$ could consist of individual i's age, gender, and previous number of arrests in a criminal risk prediction task) and $y_i \in \mathcal{Y}$ is the outcome for this individual (*e.g.*, whether

[1] Throughout the chapter, we use boldface notation to indicate a vector.

or not they commit a bail violation). Unless specified otherwise, we assume $\mathcal{X} \subseteq \mathbb{R}^M$, where M denotes the number of features. If $\mathcal{Y}$ is a finite set of labels (*e.g.*, $\mathcal{Y} = \{0, 1\}$), the learning task is called classification; if $\mathcal{Y}$ is continuous (*i.e.*, $\mathcal{Y} = \mathbb{R}$), it is called regression. In this chapter, we will focus on *binary classification*, but our work extends to multiclass classification and regression, as well.

We assume certain features (*e.g.*, gender or race) are considered *sensitive*. Sensitive features specify an individual's membership in socially salient groups (*e.g.*, women or African-Americans). For simplicity of exposition, we assume there is just one sensitive feature. However, the discussion can be extended to account for multiple sensitive features. We denote the sensitive feature for each individual i as $z_i \in \mathcal{Z} = \{1, 2, \ldots, K\}$. Note that z_i may or may not be part of the feature vector $\mathbf{x}_i$. One can define partitions of the dataset $\mathcal{D}$ based on the sensitive feature, that is, $\mathcal{D}_z = \{(\mathbf{x}_i, y_i) \mid z_i = z\}$. We refer to each partition $\mathcal{D}_z$ of the data as a sensitive feature group.

The goal of a learning algorithm is to use the training data to fit a *model* (or hypothesis) that accurately predicts the label for a new instance. A model $\theta : \mathcal{X} \to \mathcal{Y}$ receives the feature vector corresponding to a new individual and makes a prediction about his/her label. Let $\mathbf{\Theta}$ be the hypothesis class consisting of all the models from which the learning algorithm can choose. A learning algorithm receives $\mathcal{D}$ as the input; then utilizes the data to select a model $\theta \in \mathbf{\Theta}$ that minimizes some notion of loss. For instance, in classification the (0-1) loss of a model θ on the training data $\mathcal{D}$ is defined as $L(\theta) = \sum_{i=1}^{n} |y_i - \hat{y}_i|$ where $\hat{y}_i = \theta(\mathbf{x}_i)$. The learning algorithm outputs $\theta^* \in \mathbf{\Theta}$ that minimizes the loss, *i.e.*, $\theta^* = \operatorname{argmin} L(\theta)$.

2.1.2 Unfairness as Inequality in Benefits

The core idea of our proposal is to quantify the unfairness of an algorithm by measuring how **unequally** the outcomes of the algorithm **benefit** different individuals or groups in a population. While intuitive, our proposal raises two key questions: (i) how should we map algorithmic predictions received by individuals or groups to **benefits**? and (ii) given a set of benefits received by individuals or groups, how should we quantify **inequality** in the benefit distribution? We now tackle the first question, related to defining a benefit function for an individual given an outcome. In Section 2.1.3, we propose inequality indices as the answer to the second question.

Our choice of the benefit function will be dictated by the type of fairness notion we wish to apply on the task at hand. Figure 1.2 summarizes the different fairness notions that have been defined in prior works and their corresponding benefit functions. We now explain the choice of our benefit functions for the different fairness notions in the context of binary classification. Formally, let $y_i \in \mathcal{Y} = \{0, 1\}$ indicate the true label for individual i. We assume that labels in the training data reflect *ground truth*, and thus, y_i is the label *deserved* by individual i. Let $\hat{y}_i \in \{0, 1\}$ be the label the algorithm assigns to individual i.

Intuitively, the algorithmic benefit an individual i receives, b_i, should capture the *desirability* of outcome $\hat{y}_i$ for the individual. The desirability of an individual's outcome may be determined taking into account the individual's own preferences or the broader societal good. For instance, consider the criminal risk prediction example, where the positive label ($\hat{y} = 1$) indicates a low risk of criminal behavior and the negative label ($\hat{y} = 0$) indicates a high risk of criminal behavior. An individual defendant would clearly prefer the former outcome over the latter. However, from a social good perspective, accurate outcomes ($\hat{y} = y$) would be more desirable than inaccurate outcomes. Furthermore, amongst the inaccurate outcomes, one might wish to distinguish between the desirability of false positives (where a high risk person is released) and false negatives (where an low risk person is withheld).

In our binary classification scenario, where all outcomes can be decomposed into true positives ($\hat{y} = 1, y = 1$), true negatives ($\hat{y} = 0, y = 0$), false positives ($\hat{y} = 1, y = 0$), and false negatives

($\hat{y} = 0, y = 1$), the choice of our benefit function crucially determines the relative desirability of these different types of outcomes and captures different notions of fairness. For instance, the notion of *parity mistreatment* considers accurate outcomes as more desirable than inaccurate ones – so we choose a benefit function that assigns higher value ($b_i = 1$) to true positives and true negatives and a lower value ($b_i = 0$) to false positives and false negatives. In contrast, the notion of *parity impact* considers a positive label outcome as more desirable than a negative label outcome – so we adapt the benefit function to assign higher value ($b_i = 1$) to true positives and false positives and a lower value ($b_i = 0$) to true negatives and false negatives. To capture *group fairness*, once we define the benefits for all individuals, $\mathbf{b} = (b_1, \cdots, b_n)$, we can define the benefit for a subset/group g of the population, denoted by μ_g, as the mean value of the benefits received by individuals in the group: $\mu_g = \frac{1}{|g|} \sum_{i \in g} b_i$.

To capture *individual fairness*, we propose defining the benefit function of an individual i as the *discrepancy* between i's preference for the outcome i truly deserves (*i.e.*, y_i), and i's preference for the outcome the learning algorithm assigns (*i.e.*, $\hat{y}_i$). As an illustration, in this work we consider a benefit function that assigns the highest value ($b_i = 2$) for false positives (i.e., individuals that receive the advantageous positive label undeservedly), moderate values ($b_i = 1$) for true positives and true negatives (i.e., individuals that receive the labels they deserve) and lowest value ($b_i = 0$) for false negatives (i.e., individuals that receive the disadvantageous negative label despite deserving the positive label). More precisely, we compute the benefit for individual i as follows:

$$b_i = \hat{y}_i - y_i + 1. \tag{2.1}$$

We make two observations about the values of the benefit functions for different types of outcomes. First, while different fairness notions specify a preference ordering for different types of outcomes (i.e., true positives, false positives, true negatives, and false negatives), the absolute benefit values could be specified differently. The choice of benefit values would depend on the context and task at hand and the difficulty of determining them may vary in practice. Second, as many existing measures of inequality in benefits are limited to handling non-negative values, we need to ensure that $b_i \geq 0$ for $i = 1, \cdots, n$ and that there exists $j \in [n]$ such that $b_j > 0$.

Our proposal is to measure the *overall individual-level* unfairness of an algorithm by plugging b_i's (as defined above) into an existing inequality index (such as generalized entropy—to be defined shortly). Throughout the rest of the chapter, we will use the terms "overall unfairness" and "individual unfairness" interchangeably, to refer to our proposed measure. Our approach can be further generalized to measuring (un)fairness beyond supervised learning tasks (e.g. for unsupervised tasks, such as clustering or ranking)—this only requires the specification a proper notion of benefit for individuals given their *relative* outcomes within the population. We leave a careful exploration of this direction for future, and focus on supervised learning tasks in the current work.

Next, we discuss how we can generally quantify the unfairness of an algorithm as the degree to which it distributes benefit unequally across individuals using inequality indices.

2.1.3 Axioms for Measuring Inequality

Borrowing insights from the rich body of work on the axiomatic characterization of inequality indices in economics and social science [Atkinson, 1970, Cowell and Kuga, 1981, Kakwani, 1980, Kolm, 1976a,b, Litchfield, 1999, Sen, 1973], we argue that many axioms satisfied by inequality indices are appealing properties for measures of algorithmic unfairness. Therefore, inequality indices are naturally well-suited as measures for algorithmic unfairness. In this section, we briefly overview these axioms.

b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}
1	1	1	2	0	0	1	1	1	2
Overall individual-level unfairness $= I(b_1, \ldots, b_{10})$									

μ_{g_1}	μ_{g_1}	μ_{g_2}	μ_{g_2}	μ_{g_2}	μ_{g_2}	μ_{g_3}	μ_{g_3}	μ_{g_3}	μ_{g_3}
1	1	0.75	0.75	0.75	0.75	1.25	1.25	1.25	1.25
Between-group unfairness $= I(\mu_{g_1}, \mu_{g_1}, \mu_{g_2}, \mu_{g_2}, \mu_{g_2}, \mu_{g_2}, \mu_{g_3}, \mu_{g_3}, \mu_{g_3}, \mu_{g_3})$									

b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}
0	2	1	0	1	1	1	1	1	2
Within-group unf. $= I(b_1, b_2)$		**Within-group unf.** $= I(b_3, b_4, b_5, b_6)$				**Within-group unf.** $= I(b_7, b_8, b_9, b_{10})$			

Figure 2.1: **[The set of ten users along with the benefit that each user receives from classifier $\mathcal{C}_1$ in Figure 1.1.]** The overall individual-level unfairness of the classifier can be computed as the inequality I over the benefits received by the users. Overall unfairness can be decomposed into two components: 1) between-group unfairness is computed as the inequality between (weighted) average group benefits for a given group, and 2) within-group unfairness which is a weighted sum of within-group inequality.

Suppose society consists of n individuals, where $b_i \geq 0$ denotes the benefit individual i receives as the result of being subject to algorithmic decision making. An inequality measure, $I : \bigcup_{n=1}^{\infty} \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0}$, maps any benefit distribution/vector $\mathbf{b}$ to a non-negative real number $I(\mathbf{b})$. A benefit vector $\mathbf{b}$ is considered less unfair (*i.e.*, more fair) than $\mathbf{b}'$ if and only if $I(\mathbf{b}) < I(\mathbf{b}')$.

Many inequality indices previously studied satisfy the following four principles:

- **Anonymity:** The measure does not depend on any characteristics of the individuals other than their benefit, and is independent of who earns each level of benefit. Formally:

$$I(b_1, b_2, \cdots, b_n) = I\left(b_{(1)}, b_{(2)}, \cdots, b_{(n)}\right),$$

  where $(b_{(1)}, b_{(2)}, \ldots, b_{(n)})$ is the benefit vector $(b_1, b_2, \cdots, b_n)$ sorted in ascending order.

- **Population invariance:** The measure is independent of the size of the population under consideration. More precisely, let $\mathbf{b}' = \langle \mathbf{b}, \cdots, \mathbf{b} \rangle \in \mathbb{R}^{nk}_{\geq 0}$ be a k-replication of $\mathbf{b}$. Then $I(\mathbf{b}) = I(\mathbf{b}')$.

- **Transfer principle:** Transferring benefit from a high-benefit to a low-benefit individual must decrease inequality. More precisely for any $1 \leq i < j \leq n$ and $0 < \delta < \frac{b_{(j)} - b_{(i)}}{2}$,

$$I(b_{(1)}, \cdots, b_{(i)} + \delta, \cdots, b_{(j)} - \delta, \cdots, b_{(n)}) < I(\mathbf{b}).$$

  Note that the transfer should not reverse the relative position of the two individuals i and j. The transfer principle is sometimes called the Pigou-Dalton principle [Dalton, 1920, Pigou, 1912].

- **Zero-Normalization:** The measure is minimized when every individual receives the same level of benefit. That is, for any $b \in \mathbb{R}_{\geq 0}$, $I(b, b, \cdots, b) = 0$.

In addition to the above four principles satisfied by many inequality indices, we also focus on the following property which is important for our purposes. **Subgroup decomposability**

is a structural property of some inequality measures requiring that for any partition G of the population into groups, the measure, $I(\mathbf{b})$, can be expressed as the sum of a *"between-group component"* $I^G_\beta(\mathbf{b})$ (computed by assigning to each person in a subgroup $g \in G$ the subgroup's mean benefit μ_g) and a *"within-group component"* $I^G_\omega(\mathbf{b})$ (a weighted sum of subgroup inequality levels): [2]

$$I(\mathbf{b}) = I_\beta(\mathbf{b}) + I_\omega(\mathbf{b}).$$

See Figure 2.1 for an illustration of this property.

While not all inequality measures satisfy the decomposability property (*e.g.*, the Gini Index does not), the property has been studied extensively in economics, as it allows economists to compare patterns and dynamics of inequality in different subpopulations (*e.g.*, racial minorities Conceição and Ferreira [2000]).

Our measure of unfairness. For quantifying algorithmic unfairness, in this work, we focus on a family of inequality indices called *generalized entropy indices*. For a constant $\alpha \notin \{0, 1\}$, the generalized entropy of benefits $b_1, b_2, \cdots, b_n$ with mean benefit μ is defined as follows:

$$\mathcal{E}^\alpha(b_1, b_2, \cdots, b_n) = \frac{1}{n\alpha(\alpha-1)} \sum_{i=1}^{n} \left[\left(\frac{b_i}{\mu} \right)^\alpha - 1 \right]. \tag{2.2}$$

One can interpret generalized entropy as a measure of information theoretic *redundancy* in data. Generalized entropy satisfies the earlier properties of anonymity, population-invariance, the Pigou-Dalton transfer principle, and zero-normalization. Further it is subgroup decomposable Cowell and Kuga [1981], and also *scale-invariant*.[3] In fact, Shorrocks [1980] show that generalized entropy is the only differentiable family of inequality indices that satisfies population- and scale-invariance. Our interest in this family of inequality indices is motivated by this result and by our aim of understanding the trade-offs between individual and group-level unfairness.

2.1.4 Comparison with Previous Work

Existing notions of algorithmic fairness can be divided into two distinct categories: *group* and *individual* fairness.

Group fairness. Group fairness notions require that given a classifier θ, a certain group-conditional quality metric $q_z(\theta)$ is the same for all sensitive feature groups. That is:

$$q_z(\theta) = q_{z'}(\theta) \quad \forall z, z' \in \mathcal{Z}.$$

Different choices for $q_z(.)$ have led to different namings of the corresponding group fairness notions (see *e.g.*, statistical parity Corbett-Davies et al. [2017], Dwork et al. [2012], Kleinberg et al. [2017], disparate impact Feldman et al. [2015], Zafar et al. [2017b], equality of opportunity Hardt et al. [2016], calibration Kleinberg et al. [2017], and disparate mistreatment Zafar et al. [2017a]). Generally, these notions cannot guarantee fairness at the individual level, or when groups are further refined (see Kearns et al. Kearns et al. [2017] for an illustrative example).

Existing group fairness notions are similar to the between-group component of fairness that we propose. However, these notions usually do not take into account the size of different groups, whereas our between-group measure considers the proportion of the groups relative to the total population as illustrated in Figure 2.1. For example consider a population divided into two

[2] When the partition G we are referring to is clear from the context, we drop the superscript G to simplify notation.

[3] A measure I is scale-invariant if for any constant $c > 0$, $I(c\mathbf{b}) = I(\mathbf{b})$.

groups A and B containing 70% and 30% of the population with the negative ground truth label respectively. Using generalized entropy with $\alpha = 2$, a classifier C_1 achieving a false positive rate of 0.8 on A and 0.6 on B has a between-group inequality of 0.06, whereas a classifier C_2 with false positive rates of 0.6 on A and 0.8 on B results in a lower between-group inequality of 0.04. However, when considering a group fairness measure based on differences in false positive rates between A and B, C_1 and C_2 would be equally fair.

Individual fairness. Dwork et al. [2012] first formalized the notion of individual fairness for classification tasks using Lipschitz conditions on the classifier outcomes. Their notion of individual fairness requires that two individuals who are similar with respect to the task at hand, receive similar classification outcomes. Dwork et al.'s definition is, therefore, formalized in terms of a similarity function between individuals. For instance, in practice given two individuals with feature values $\mathbf{x}$ and $\mathbf{x}'$, and suitable distance functions $D_{\mathcal{X}}$ and $D_{\mathcal{Y}}$ (defined over $\mathcal{X} \times \mathcal{X}$ and $\Delta(\mathcal{Y}) \times \Delta(\mathcal{Y})$, respectively), Dwork et al.'s notion for individual fairness requires the following condition to hold:

$$D_{\mathcal{Y}}\big(p(\hat{y} = 1|\mathbf{x}), p(\hat{y} = 1|\mathbf{x}')\big) < D_{\mathcal{X}}(\mathbf{x}, \mathbf{x}').$$

Due to its dependence on the individual feature vectors $\mathbf{x}$, Dwork et al.'s notion of individual fairness does not satisfy the anonymity principle.

Furthermore, Dwork et al.'s notion of individual fairness only provides a 'yes/no' answer to whether fairness *conditions* are satisfied, but does not provide a meaningful *measure* of algorithmic fairness when considered *independent of prediction accuracy*. We further illustrate this point with two examples: First, by this definition a model that assigns the same outcome to everyone is considered fair, regardless of people's merit for different outcomes (e.g. awarding pretrial release to every defendant is considered fair, even though only some of them—those who appear for subsequent hearings and don't commit a crime[4]—deserve to be awarded the pretrial release). Second, the definition does not take into account the difference in social desirability of various outcomes. For instance, if one flips the (binary) labels predicted by a fair classifier, the resulting classifier will be considered equally fair (e.g. a classifier that awards pretrial release to a defendant if and only if they go on to violate the release criteria is considered fair!). The measure we propose in equations 2.1 and 2.2 addresses these issues by offering a merit-based metric of fairness that seeks to *equalize* the *benefit* individuals receive as the result of being subject to algorithmic decision making.

Finally, we remark that there has been interest in a similar axiomatic approach to methods for algorithmic interpretability Lundberg and Lee [2017], Sundararajan et al. [2017].

2.2 Theoretical Characterization

In this section, we characterize the conditions under which there is a tradeoff between accuracy and our notion of algorithmic fairness. Further, we shed light on the relationship between our notion of fairness and existing group measures, precisely connecting the two when the inequality index in use is *additively decomposable* (see Subramanian [2011] and the references therein). At a high level, we show that group unfairness is one piece of a larger puzzle: overall unfairness may be regarded as a combination of unfairness *within*- and *between*-groups. As the number of groupings increases, with each becoming smaller (eventually becoming single individuals), the between-group component grows to be an increasingly large part of the overall unfairness.

[4]For making the pretrial release decisions, these two are the main criteria that the judges or the algorithms try to assess Berk et al. [2017], Summers and Willis [2010].

2.2.1 Accuracy vs. Individual Fairness

We begin by observing that the fairness optimal classifier is perfectly fair if and only if the accuracy optimal classifier is perfectly accurate. Given a classifier θ and training data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, let $I_\mathcal{D}(\theta)$ specify the individual unfairness of θ on $\mathcal{D}$, that is, $I_\mathcal{D}(\theta) = I(b_1^\theta, \cdots, b_n^\theta)$ where $b_i^\theta = 1 + \theta(\mathbf{x}_i) - y_i$. $L_\mathcal{D}(\theta)$ is the empirical loss of θ on $\mathcal{D}$.

Proposition 2.2.1. *Suppose $I(.)$ is a zero-normalized inequality index and $\mathbf{\Theta}$ is closed under complements.*[5] *For any training data set $\mathcal{D}$, there exists a classifier $\theta \in \mathbf{\Theta}$ for which $I_\mathcal{D}(\theta) = 0$ if and only if there exists a classifier θ' for which $L_\mathcal{D}(\theta') = 0$.* [6]

Proposition 2.2.1 may seem to suggest that our notion of fairness is entirely in harmony with prediction accuracy: by simply minimizing prediction error, unfairness will be automatically eliminated. While this is true in the special case of fully separable data (or when we have access to an oracle with 0 prediction error), it is not true in general. The following result shows that under broad conditions, the fairness optimal classifier may not coincide with the accuracy optimal classifier. For training data set $\mathcal{D}$, let $\theta_\mathcal{D}^A$ be the accuracy optimal classifier, and $\theta_\mathcal{D}^F$ be the fairness optimal classifier:

$$\theta_\mathcal{D}^A = \arg\min_\theta L_\mathcal{D}(\theta) \text{ and } \theta_\mathcal{D}^F = \arg\min_\theta I_\mathcal{D}(\theta).$$

Proposition 2.2.2. *Suppose $I(.)$ satisfies the transfer principle and is population- and scale-invariant. If there exists a feature vector $\tilde{\mathbf{x}}$ such that $0 < \mathbb{P}[y = 1|\mathbf{x} = \tilde{\mathbf{x}}] < 0.5$, then there exists a training data set $\mathcal{D}$ for which $\theta_\mathcal{D}^B \not\equiv \theta_\mathcal{D}^F$.*

Even though the fairness optimal and accuracy optimal classifiers do not necessarily coincide, one might wonder if the fairness optimal classifier always results in near-optimal accuracy. In fact, it does not. Example 1 in Appendix A.1 shows that the accuracy of the fairness optimal classifier can be arbitrarily worse than that of the accuracy optimal classifier.

2.2.2 Individual vs. Group Fairness

Next, we focus on additive-decomposability and show how this property allows us to establish formally the existence of tradeoffs between individual- and group-level (un)fairness. Suppose we partition the population into $|G|$ disjoint subgroups, where subgroup $g \in G$ consists of n_g individuals with the benefit vector $\mathbf{b}^g = (b_1^g, \cdots, b_{n_g}^g)$ and mean benefit μ_g. Each partition could, for instance, correspond to a sensitive feature group (*e.g.*, $g = 1$ consists of all African-American defendants and $g = 2$, all white defendants). One can re-write the Generalized Entropy as follows:

$$\begin{aligned}
\mathcal{E}^\alpha(b_1, b_2, \cdots, b_n) &= \sum_{g=1}^{|G|} \frac{n_g}{n} \left(\frac{\mu_g}{\mu}\right)^\alpha \mathcal{E}^\alpha(\mathbf{b}^g) \\
&\quad + \sum_{g=1}^{|G|} \frac{n_g}{n\alpha(\alpha-1)} \left[\left(\frac{\mu_g}{\mu}\right)^\alpha - 1\right] \\
&= \mathcal{E}_\omega^\alpha(\mathbf{b}) + \mathcal{E}_\beta^\alpha(\mathbf{b}).
\end{aligned}$$

Note that imposing a constraint on a decomposable inequality measure, such as $\mathcal{E}^\alpha(\mathbf{b})$, guarantees both within-group and between-group inequality are bounded. Existing notions of group fairness, however, capture only the *between-group* component (when $|G| = 2$, the between-group unfairness

[5] In the context of a binary classification task with $\mathcal{Y} = \{0, 1\}$, $\mathbf{\Theta}$ is closed under complements if for any $\theta \in \mathbf{\Theta}$, also $1 - \theta \in \mathbf{\Theta}$.

[6] Proofs can be found in Appendix A.1.

is minimized if and only if the two groups receive the same treatment on average). The problem with imposing a constraint on the between-group component ($\mathcal{E}^{\alpha}_{\beta}(\mathbf{b})$) alone, is that it may drive up the within-group component, $\mathcal{E}^{\alpha}_{\omega}(\mathbf{b})$. In fact, we show that if an individual-fairness optimal classifier is not group-fairness optimal, then optimizing for group fairness alone will certainly increase unfairness within groups sufficiently so as to raise the overall (individual) unfairness.

Formally, minimizing our notion of individual unfairness while guaranteeing a certain level of accuracy corresponds to the following optimization:[7]

$$\min_{\theta \in \mathbf{\Theta}} \quad I_{\beta}(\theta) + I_{\omega}(\theta) \qquad \text{s.t.} \quad L(\theta) \geq \delta, \tag{2.3}$$

while minimizing only between-group unfairness corresponds to:

$$\min_{\theta \in \mathbf{\Theta}} \quad I_{\beta}(\theta) \qquad \text{s.t.} \quad L(\theta) \geq \delta. \tag{2.4}$$

Let $\theta^*(\delta)$ be an optimal solution for optimization (2.3)—if there are multiple optimal solutions, pick one with the lowest I_{β}. Let $\theta^*_{\beta}(\delta)$ be any optimal solution for optimization (2.4). The following holds.

Proposition 2.2.3. *Suppose $I(.)$ is additively decomposable. For any $\delta \in [0,1]$, if $I_{\beta}\left(\theta^*_{\beta}(\delta)\right) \neq I_{\beta}\left(\theta^*(\delta)\right)$, then $I_{\omega}\left(\theta^*_{\beta}(\delta)\right) > I_{\omega}\left(\theta^*(\delta)\right)$ and $I\left(\theta^*_{\beta}(\delta)\right) > I\left(\theta^*(\delta)\right)$.*

Next, we show that the contribution of the between-group component to overall inequality, i.e. I_{β}/I, depends—among other things—on the granularity of the groups. In particular, the between-group contribution increases with intersectionality: I_{β} is lower when computed over just race (African-Americans vs. Caucasians), and is higher when computed over the intersection of race and gender (female African-Americans, ..., male Caucasians). More precisely, suppose G, G' are two partitions of the population into disjoint groups. Let $G \times G'$ specify the Cartesian product of the two partitions: for $g \in G, g' \in G'$, $i \in (g, g')$ if and only if $i \in g$ and $i \in g'$. It is easy to show the following result.

Proposition 2.2.4. *Suppose $I(.)$ is zero-normalized and additively decomposable. Suppose G, G' are two different partitions of the population into disjoint groups. For any benefit distribution $\mathbf{b}$, $I^{G}_{\beta}(\mathbf{b}) \leq I^{G \times G'}_{\beta}(\mathbf{b})$.*

If one continues refining the groups, eventually every individual will be in their own group and the between-group unfairness becomes equivalent to the overall individual unfairness. This offers a framework to interpolate between group and individual fairness.

When the number of groups is small and people within each group receive highly unequal benefits, the contribution of the between-group component to overall unfairness is small, and narrowing down attention to reducing I_{β} alone may result in fairness gerrymandering Kearns et al. [2017]: while it is often easy to reduce group unfairness in this case, doing so will affect the overall unfairness in unpredictable ways—potentially making the within-group unfairness worse. On the other hand, as the number of groups increases or the treatment of people within a group becomes more uniform, the role that the between-group component plays in overall unfairness grows – but, as noted by Kearns et al. Kearns et al. [2017], it also becomes computationally harder to control and limit the between-group unfairness.

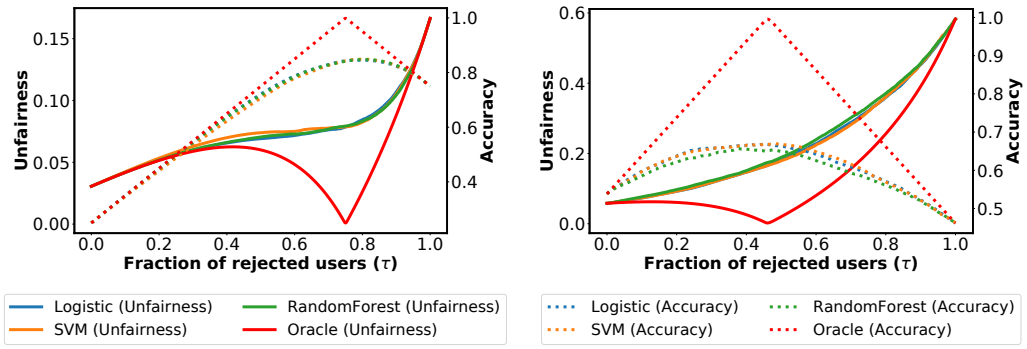
[7] For simplicity, we are dropping the training data set $\mathcal{D}$ from the subscripts.

Proposition 2.2.5. *Suppose $I(.)$ is zero-normalized and additively decomposable. Suppose $I(\mathbf{b}) \neq 0$. For any partition G of the population to disjoint subgroups, $0 \leq \frac{I_\beta^G(\mathbf{b})}{I(\mathbf{b})} \leq 1$. Further, there exist benefit distributions $\mathbf{b}$ and $\mathbf{b}'$ such that $\frac{I_\beta^G(\mathbf{b})}{I(\mathbf{b})} = 1$ and $\frac{I_\beta^G(\mathbf{b}')}{I(\mathbf{b}')} = 0$.*

Implication for practitioners: individual or group unfairness? We saw that the contribution of the group component to overall unfairness is a nuanced function of the granularity with which the groups are defined, as well as the unfairness within each group. If our goal is to reduce overall unfairness, note that existing fair learning models that exclusively focus on reducing between-group unfairness would help only when between-group unfairness accounts for a large part of the overall unfairness. Our measures of unfairness present a framework to examine this condition.

2.3 Empirical Analysis

In this section, we empirically validate our theoretical propositions from Section 2.2 on multiple real-world datasets. Specifically, in Section 2.3.1, our goal is to shed light on tradeoffs between overall individual-level unfairness and accuracy. In Section 2.3.2, we explore how the overall unfairness decomposes along the lines of sensitive attribute groups. We use the subgroup-decomposability of our proposed unfairness measures to study finer-grained fairness-accuracy tradeoffs at the levels of between-group and within-group unfairness. In Section 2.3.3, we empirically explore how methods to control between-group unfairness affect other unfairness components.



(a) Adult dataset (b) COMPAS dataset

Figure 2.2: [**Overall unfairness (solid lines—$\mathcal{E}^2(\mathbf{b})$ in Eq. 2.5) and accuracy (dotted lines) as a function of the decision ranking threshold (τ) for various classifiers.**] The positive and negative class ratio in the Adult dataset is about 0.25 : 0.75, hence the 1.00 accuracy point corresponds to $\tau = 0.75$. Similarly, the positive and negative class ratio in the COMPAS dataset is about 0.55 : 0.45, hence the 1.00 accuracy point corresponds to $\tau = 0.45$. For oracle, the optimal point for accuracy corresponds to minimal unfairness; this doesn't hold for other classifiers.

Setup and Datasets. We use the Generalized Entropy index (cf. Eq. 2.2) with $\alpha = 2$ (in other words, half the squared coefficient of variation) to measure unfairness:

$$\mathcal{E}^2(b_1, b_2, \cdots, b_n) = \frac{1}{2 \times n} \sum_{i=1}^{n} \left[\left(\frac{b_i}{\mu} \right)^2 - 1 \right].$$

As noted in Section 2.1, the Generalized Entropy index can be further decomposed into between-group and within-group unfairness as:

$$\mathcal{E}^2(b_1, b_2, \cdots, b_n) = \quad \mathcal{E}^2_\omega(\mathbf{b}) + \mathcal{E}^2_\beta(\mathbf{b}). \tag{2.5}$$

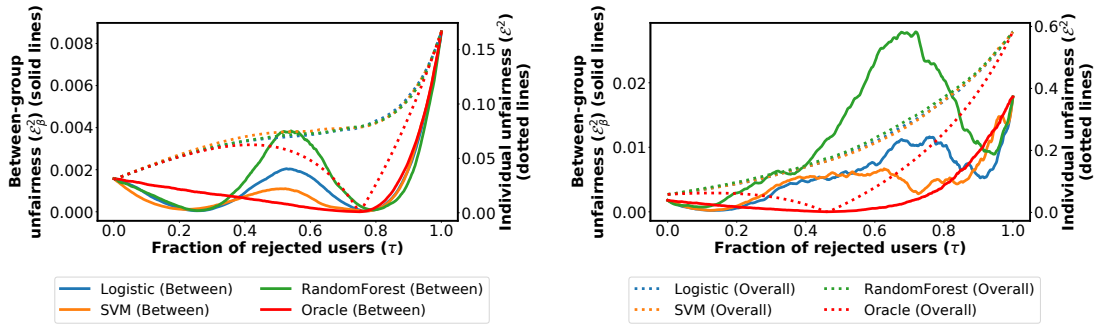
We will refer to the quantity $\mathcal{E}^2$ as *individual unfairness* or *overall unfairness* interchangeably, $\mathcal{E}^2_\beta$ as between-group unfairness, and $\mathcal{E}^2_\omega$ as within-group unfairness.

We experiment with two real-world datasets: (i) the *Adult income* dataset Adult [1996], and (ii) the *ProPublica COMPAS* dataset Larson et al. [2016]. Both datasets have received previous attention Corbett-Davies et al. [2017], Feldman et al. [2015], Zafar et al. [2017a,b], Zemel et al. [2013].

For the *Adult income* dataset, the task is to predict whether an individual earns more (positive class) or less (negative class) than 50,000 USD per year based on features like education level and occupation. We consider gender (female and male) and race (Black, White and Asian) as sensitive features. We filter out races (American Indian and Other) which constitute less than 1% of the dataset. After the filtering, the dataset consists of $44,434$ subjects and 11 features.

For the *ProPublica COMPAS* dataset, the task is to predict whether (negative class) or not (positive class) a criminal defendant would commit a crime within two years based on features like current charge degree or number of prior offenses. We use the same set of features as Zafar et al. Zafar et al. [2017a]. The sensitive features in this case are also gender (female and male) and race (Black, Hispanic, White). The dataset consists of $5,786$ subjects and 5 features.

For all experiments, we repeatedly split the data into 70%-30% train-test sets 10 times and report average statistics. All hyperparameters are validated using a further 70%-30% split of the train set into train and validation sets.



(a) Adult dataset (b) COMPAS dataset

Figure 2.3: **[Between-group unfairness (solid lines—$\mathcal{E}^2_\beta(\mathbf{b})$ in Eq. 2.5) and overall unfairness (dotted lines—$\mathcal{E}^2(\mathbf{b})$ in Eq. 2.5) as a function of the decision ranking threshold (τ) for various classifiers.]** Between-group unfairness $\mathcal{E}^2_\beta(\mathbf{b})$ only constitutes a small fraction of the overall unfairness $\mathcal{E}^2(\mathbf{b})$.

2.3.1 Fairness vs. Accuracy Tradeoffs

We begin by studying the tradeoff between the accuracy and the overall ***individual unfairness*** of a given classifier ($\mathcal{E}^2(\mathbf{b})$ in Eq. 2.5). We use three standard classifier models: logistic regression, support vector machine with RBF kernel (SVM), and random forest classifier. Results in Section 2.3.1 and Section 2.3.2 are computed by optimizing these classifiers for accuracy.

Each of the above models computes the *likelihood* of belonging to the positive class for every instance. We denote this likelihood by p_i for an individual i. We compare the fairness and

accuracy of these classifiers with that of an "oracle" that can perfectly predict the label for every instance (and assigns $p_i \in \{0, 1\}$). To predict a label in $\{0, 1\}$ for individuals, we first rank all instances (in increasing order) according to their p_i values with ties broken randomly; then we designate a decision ranking threshold $0 \leq \tau \leq 1$ and output a label of 1 for individual i if and only if $rank(i) \geq n\tau$, where $rank(i)$ denotes the rank of individual i in the sorted list. In other words, an increasing value of τ corresponds to the classifier rejecting more people from the sorted list (in order of their positive class likelihood p_i). As we vary τ, we expect both accuracy as well as unfairness of the resulting predictions to change as discussed below and shown in Figure 2.2.

For the oracle, as expected from Proposition 2.2.1, *a perfect accuracy corresponds to zero unfairness*: with an increasing τ, the accuracy increases while the unfairness decreases. After a certain optimal value of threshold τ (close to 0.75 in the Adult data and 0.45 in the COMPAS data), the trend reverses. We note that 0.75 and 0.45 represent the fraction of instances in the negative class in the respective datasets. Hence, at these optimal thresholds, all of the oracle's predictions are accurate (since the points are ranked based on their positive class likelihood) resulting in 0 unfairness.

However, for all other (non-oracle) classifiers, as expected via Proposition 2.2.2, the trend is very different: *the optimal threshold for (imperfect) accuracy is far from the optimal threshold for unfairness*. Moreover, with increasing τ, while unfairness continually increases, for accuracy we initially see an increase followed by a drop. We note that the overall unfairness is not always a monotone function of the decision ranking threshold as illustrated in Example 1.

2.3.2 Fairness Decomposability

As Figure 2.1 and Eq. 2.5 show, the individual unfairness of a predictor can be decomposed into between-group and within-group unfairness. In this part, we study the ***between-group unfairness*** component ($\mathcal{E}_\beta^2(\mathbf{b})$ in Eq. 2.5) as we change τ. To this end, we consider two sensitive features: gender and race. We split each of the datasets into all possible disjoint groups based on these sensitive features (*e.g.*, White women, Hispanic men, Black women).

Figure 2.3 shows between-group unfairness along with overall unfairness for different values of τ. We notice that for the Adult dataset, the *between-group unfairness follows a multi-modal trend*: it starts from a non-zero value at $\tau = 0$, falls to almost 0 for most classifiers (except for the oracle) at around $\tau = 0.2$, reaches a local peak again around $\tau = 0.5$, and finally completes another cycle to fall and then reach its maximum value at $\tau = 1.0$. The COMPAS dataset also shows a similar trend, albeit to a lesser extent.

Between-group unfairness and overall unfairness. Comparing the between-group unfairness and overall unfairness in Figure 2.3 reveals a very interesting insight: for the same value of τ, the between-group unfairness (solid lines) is a very small fraction of the overall unfairness (dotted lines). For example, considering the performance of the logistic regression classifier on the COMPAS dataset, the maximum value of the *overall unfairness* is close to 0.6 whereas the maximum value of the *between-group unfairness* is merely 0.01. We hypothesize that since the number of sensitive feature-based groups is much smaller than the number of all individuals in the dataset, the individual unfairness value dominates the between-group unfairness.

To test this hypothesis, we experiment with the following setup: We take the three sensitive features present in the COMPAS dataset, namely gender, race and age, and form sensitive feature groups based on all possible combinations of these features. For example, groups formed based on gender would be men and women; groups formed based on race would be Black, White and Hispanic; whereas groups formed based on gender as well as race would be Black men, Black women, White men and so on. For each of these sensitive feature combinations we plot in Figure 2.4 the percentage of contribution that the between-group unfairness has towards the

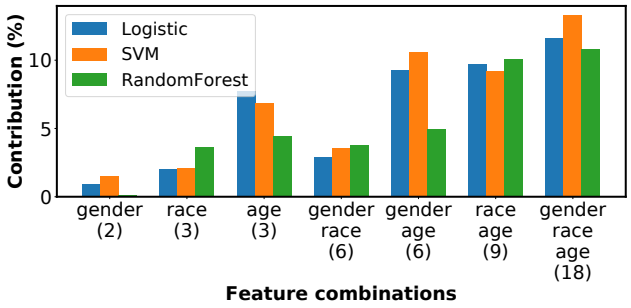


Figure 2.4: [**Between-group unfairness ($\mathcal{E}^2_\beta(\mathbf{b})$ in Eq. 2.5) as a fraction of the overall/individual unfairness ($\mathcal{E}^2(\mathbf{b})$ in Eq. 2.5) for various combinations of sensitive features.**] Numbers on the x-axis denote how many sensitive feature groups would be formed when using the corresponding sensitive feature set. Logistic regression, SVM and Random Forests achieve similar accuracies of 66%, 67% and 65% as well as similar overall individual unfairness of 0.145, 0.151 and 0.134 respectively on the ProPublica Compas dataset.

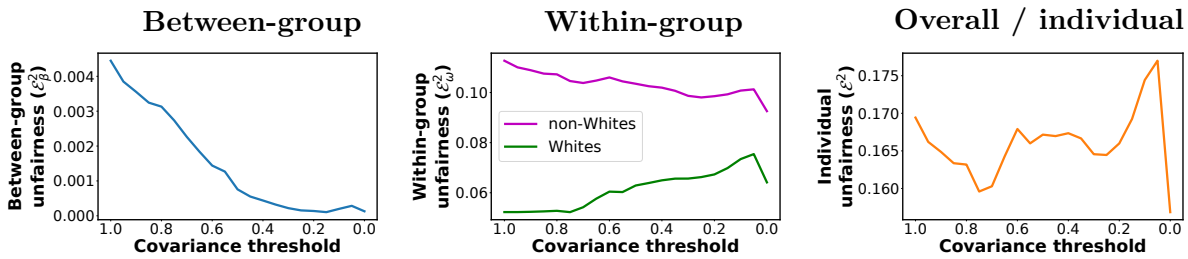


Figure 2.5: [**The effect of applying false negative rate constraints of Zafar et al. in order to minimize the between-group unfairness ($\mathcal{E}^2_\beta(\mathbf{b})$).**] The second plot shows that reducing the between-group unfairness leads to an increase in the within-group unfairness ($\mathcal{E}^2_\omega(\mathbf{b})$) for Whites. Moreover, for certain values of the covariance threshold, the overall unfairness ($\mathcal{E}^2(\mathbf{b})$) increases as compared to the unconstrained classifier.

overall unfairness. Figure 2.4 shows that *as the number of sensitive feature groups increases, the between-group unfairness contributes more and more towards the overall unfairness.* This result is also in line with the implications of Proposition 2.2.4.

Figure 2.4 also shows the following interesting insight: Even though the overall unfairness and accuracy of all the classifiers is very similar (cf. Figure 2.2), the random forest classifier leads of significantly smaller contribution of between-group unfairness as compared to other classifier (*e.g.*, gender, gender+age). In other words even for similar levels of accuracy and overall unfairness, *classifiers have very different between-group unfairness across different feature sets.*

Accuracy and between-group unfairness. We also study the between group unfairness in Figure 2.3 and corresponding accuracy in Figure 2.2. We notice that there is *no definitive correlation between the accuracy and the between-group unfairness*: In the Adult dataset, the highest level of accuracy (around $\tau = 0.8$) corresponds to one of the lowest values of between-group unfairness for all classifiers. However, this doesn't hold in the case of COMPAS dataset.

2.3.3 Interaction Between Different Types of Unfairness

In this section, we revisit the literature on fairness-aware machine learning and investigate how methods proposed to control between-group unfairness (which is what most existing methods

focus on Feldman et al. [2015], Hardt et al. [2016], Kamiran and Calders [2009], Kamishima et al. [2013], Zafar et al. [2017a,b], Zemel et al. [2013]) can affect the overall/individual and the within-group unfairness. Specifically, we study how the overall unfairness ($\mathcal{E}^2(\mathbf{b})$ in Eq. 2.5) and the within-group unfairness ($\mathcal{E}^2_\omega(\mathbf{b})$ in Eq. 2.5) would change when training a constrained classifier to minimize the between-group unfairness ($\mathcal{E}^2_\beta(\mathbf{b})$ in Eq. 2.5). Our study is motivated by the fact that while several methods focus on designing constraints to remove the between-group unfairness (*e.g.*, see Hardt et al. [2016], Zafar et al. [2017a], Zemel et al. [2013]), to the best of our knowledge, no prior work in fairness-aware machine learning has studied the effect of these constraints on the overall and the within-group unfairness.

To this end, we use the methodology proposed by Zafar et al. Zafar et al. [2017a] to remove the between-group unfairness based on false negative rates between different races (Whites and non-Whites) in the COMPAS dataset. Zafar et al. propose to remove the between-group unfairness by bounding the covariance between misclassification distance from the decision boundary and the sensitive feature value. The method operates by bounding the covariance of the unconstrained classifier by successive multiplicative factors between 1 and 0. A covariance multiplicative factor of 1 means that no fairness constraints are applied while training the classifier, whereas a factor of 0 means the tightest possible constraints are applied. As done by Zafar et al., we train several logistic regression classifiers to limit the between-group unfairness; each classifier is trained with a covariance multiplicative factor in the range $[1.00, 0.95, 0.90, \ldots, 0.05, 0.00]$.

Figure 2.5 shows the between-group unfairness, within-group unfairness, and overall/individual unfairness as the fairness constraints of Zafar et al. Zafar et al. [2017a] are tightened towards 0. The figure shows the following key insights: (i) *Reducing the between-group unfairness can in fact increase the within-group unfairness*: the within-group unfairness for Whites almost monotonically increases as the between-group unfairness is reduced. This observation also follows Proposition 2.2.3. (ii) *Reducing the between-group unfairness can exacerbate overall/individual unfairness*: As the between-group unfairness decreases between the covariance multiplicative factor of 0.8 to 0.6 (on the x-axis), the overall unfairness in fact goes up. These insights point to possible significant tensions between these different components of unfairness.

Summary of empirical analysis. Experiments on multiple real-world datasets performed in this section support the theoretical analysis of Section 2.2. The empirical (as well as the theoretical) analysis brings out the inherent tensions between fairness and accuracy, as well as between different (between- and within-group) components of fairness. These results point to potential for situations where optimizing for one type of fairness can exacerbate the other.

2.4 Conclusion

We proposed using inequality indices from economics as a principled way to compute the scalar degree of total unfairness of any algorithmic decision system. The approach is based on well-justified principles (axioms), and is general enough so that by varying the benefit function, we can capture all previous notions of algorithmic fairness conditions as special cases, while also admitting interesting generalizations. The resulting measures of total unfairness unify previous concepts of group and individual fairness, and allow us to study quantitatively the behavior of earlier methods to mitigate unfairness. These earlier methods typically worry only about between-group unfairness, which may be justified for legal reasons, or in order to redress particular social prejudices. However, we demonstrate that minimizing exclusively between-group unfairness may actually increase overall unfairness.

Chapter 3

Fairness Case Study: Discrimination in Facebook Advertising

Online targeted advertising platforms like Facebook have been criticized for allowing advertisers to discriminate against users belonging to sensitive groups, i.e., to exclude users belonging to a certain race or gender from receiving their ads. Such criticisms have led, for instance, Facebook to disallow the use of attributes such as ethnic affinity from being used by advertisers when targeting ads related to housing or employment or financial services.

In this part of thesis, we show that such measures are far from sufficient and that the problem of discrimination in targeted advertising is much more pernicious. We argue that discrimination measures should be based on the targeted population and not on the attributes used for targeting. We systematically investigate the different targeting methods offered by Facebook for their ability to enable discriminatory advertising. We show that a malicious advertiser can create highly discriminatory ads without using sensitive attributes. Our findings call for exploring fundamentally new methods for mitigating discrimination in online targeted advertising.

Relevant publication: The work in this chapter is published as "Potential for Discrimination in Online Targeted Advertising", **T. Speicher**, M. Ali, G. Venkatadri, F. N. Ribeiro, G. Arvanitakis, F. Benevenuto, K. P. Gummadi, P. Loiseau, and A. Mislove. In *Proceedings of the 2018 Conference on Fairness, Accountability, and Transparency, FAccT 2018* [Speicher et al., 2018a].

3.1 Quantifying Ad Discrimination

We begin by outlining different ad targeting methods offered by Facebook. We then discuss the current approach to determining whether an ad is discriminatory and argue why it is inadequate. Finally, we propose a new and intuitive approach to quantify discrimination.

3.1.1 Methods for Targeted Ads

Facebook gathers and infers several hundreds of attributes for all of its users, covering their demographical, behavioral, and interest features[1] Andreou et al. [2018]. Some of those attributes, such as gender or race, are considered *sensitive* meaning targeting (i.e., including or excluding) people based on those attributes is restricted by law for certain types of advertisements (e.g., those announcing access to housing or employment or financial services Barocas and Selbst [2016b]).

Facebook allows advertisers to select their target audience in three ways:

1. *Attribute-based targeting:* Advertisers can select audiences that have (or do not have) a certain attribute (or a combination of attributes), e.g., select users who are "men", "aged 35", and are interested in "tennis."

[1] `https://www.facebook.com/business/learn/facebook-ads-choose-audience`

2. *PII-based (custom audience) targeting:* Advertisers can directly specify who should be targeted by providing a list of personally identifiable information (PII) such as phone numbers or email addresses.

3. *Look-alike audience targeting:* Advertisers can ask Facebook to target users who are similar to (i.e., "look like") their existing set of customers, specified using their PII.

3.1.2 Quantification Approaches

Next, we discuss three basic approaches to quantifying discrimination and their trade-offs.
1. Based on advertiser's intent: An intuitive (moralized) way to quantify discrimination would be to base it on the advertiser's intent. However, not only is such a measure challenging to operationalize (i.e., to measure from empirical observations), but it also overlooks the harmful effects of unintentionally discriminatory ads that may be placed by a well-meaning but ignorant or careless advertiser. In this chapter, we do not consider such approaches.
2. Based on ad targeting process: Another approach to determine whether an ad is discriminatory is based on the *process* used to target the ads. Any ads placed using the right process would be non-discriminatory (by definition), while those using a wrong process would be declared discriminatory (by definition). Existing approaches, such as those that determine whether an ad is discriminatory based on the use of sensitive attributes (e.g., "ethnic affinity") in targeting, fall under this category. As we show in this chapter, attempting to quantify discrimination based on the process (means or methods) of targeting is quite difficult when there exist multiple different processes for targeting users. Instead, in this work, we advocate for a third approach.
3. Based on targeted audience (outcomes): We propose to quantify discrimination based on the outcomes of the ad targeting process, i.e., the audience selected for targeting. Put differently, we do not take into account how users are being targeted but only who they are. Outcome-based approaches to quantifying discrimination have the advantage that they can be generally applied to all scenarios, independently of the employed method of targeting. We discuss one such method in the next section.

3.1.3 Outcome-based Discrimination

To formalize our discrimination measure, we will assume that an ad platform like Facebook keeps track of a database $\mathbf{D} = (u_i)_{i=1,\ldots,n}$ of user-records u_i where each user is represented by a vector of boolean attributes, i.e., $u_i \in \mathbb{B}^m$. We denote the sensitive attribute (e.g., race or gender) that we are interested in a particular situation by $s \in \{1, \ldots, m\}$ and its value for a user u by u_s. The corresponding sensitive group $\mathbf{S}$ is the set of all users that have the sensitive attribute, i.e., $\mathbf{S} = \{u \in \mathbf{D} \,|\, u_s = 1\}$.

To measure outcome- (i.e., targeted audience-) based discrimination, we define a metric for how discriminatory an advertiser's targeting is. It is inspired by the *disparate impact* measure that is frequently used to detect discrimination in selecting candidates from a pool of applicants in recruiting and housing allotment scenarios Barocas and Selbst [2016b].

Our key observation is that ad targeting, like recruiting, involves selecting the *target audience* (**TA**) from a much larger pool of *relevant audience* (**RA**). The relevant audience of an ad is the set of *all* users in the database $\mathbf{D}$ who would find the ad useful and interesting and thus might interact with it. Intuitively, the discrimination measure should capture the extent to which the target audience selection is *biased* based on sensitive group membership of relevant users.

We define the *representation ratio* measure for sensitive attribute s to capture how much more likely a relevant user u is to be targeted when having the sensitive attribute compared to not having it. More specifically, it is the ratio between the fraction of relevant audience with

attribute s that are selected for targeting and the fraction of relevant audience without attribute s that are selected, i.e.,

$$\text{rep\_ratio}_s(\mathbf{TA}, \mathbf{RA}) = \frac{|\,\mathbf{TA} \cap \mathbf{RA}_s|/|\,\mathbf{RA}_s|}{|\,\mathbf{TA} \cap \mathbf{RA}_{\neg s}|/|\,\mathbf{RA}_{\neg s}|}, \tag{3.1}$$

where $\mathbf{RA}_s = \{u \in \mathbf{RA} \mid u_s = 1\}$ and $\mathbf{RA}_{\neg s} = \{u \in \mathbf{RA} \mid u_s = 0\}$.

Based on the representation ratio we define a measure that we call *disparity in targeting*, defined for a sensitive attribute s as:

$$\text{disparity}_s(\mathbf{TA}, \mathbf{RA}) = \max\left(\text{rep\_ratio}_s(\mathbf{TA}, \mathbf{RA}), \frac{1}{\text{rep\_ratio}_s(\mathbf{TA}, \mathbf{RA})}\right). \tag{3.2}$$

Note that it is important to compute disparity based on the relevant audience **RA** because **RA** may have a very different composition in terms of attribute s than the whole database **D**. For example, an ad for men's clothes may have a relevant audience **RA** with a gender-ratio highly skewed towards men. A random selection of users from **RA** would be non-disparate with respect to **RA**, but might be highly disparate with respect to **D**. Similarly, for the same targeted audience (including mostly males), some ads could be non-discriminatory (e.g., ads for men's clothes) while others could be highly discriminatory (e.g., ads for high-paying jobs), depending on the corresponding relevant audience. Throughout the chapter, we implicitly assume that, for the sensitive attributes considered, the relevant audience has the same distribution as the global population; and we show that the advertiser can include or exclude certains groups based on the sensitive attribute—hence the ad targeting is discriminatory.

We propose to detect discriminatory targeting using our disparity measure as follows: we declare a targeting formula as discriminatory when its disparity for some sensitive attribute value group exceeds a certain threshold (i.e., the group is over- or under-represented). For instance, a reasonable threshold value may be 1.25, mimicking the popular "80%" disparate impact rule Biddle [2005], to declare a group over- or under-represented.

In addition to disparity, we would be interested in the recall of an ad, which quantifies how many of the relevant users with the sensitive attribute the discriminatory ad targets or excludes. It can be defined as

$$\text{recall}(\mathbf{TA}, \mathbf{RA}') = \frac{|\mathbf{TA} \cap \mathbf{RA}'|}{|\mathbf{RA}'|}, \tag{3.3}$$

where $\mathbf{RA}'$ might be the restriction of **RA** to $\mathbf{RA}_s$ or $\mathbf{RA}_{\neg s}$, depending on whether the discriminatory advertiser wants to target or exclude users with the sensitive attribute s.

3.2 PII-based Targeting

In this section, we show how the audience targeting mechanism based on personally identifiable information (PII) recently introduced by Facebook can be exploited by advertisers to covertly implement discriminatory advertising. We first briefly describe the PII-based audience targeting feature of Facebook; we then explain how this feature can be exploited to implement discriminatory advertising. Next, we explain how public data sources have data that advertisers can use to implement discriminatory advertising, and finally demonstrate the feasibility of such an attack by using information from public records to create audiences for advertising that are discriminatory.

PII-based audience selection: While Facebook traditionally allowed advertisers to select audiences to advertise to by specifying attributes of the audience (e.g., age, gender, etc.), Facebook recently introduced *custom audiences*. This feature allows advertisers to specify *exactly* which users they want to target by specifying personally identifying information (PII) that uniquely

identifies those users. Facebook allows 15 different types of PII to be used, including phone numbers, email addresses, and combinations of name with other attributes (such as date of birth or ZIP code). The advertiser uploads a file containing a list of PII; Facebook then matches these PII to Facebook accounts to create a custom audience.

Custom audiences can be viewed as implementing a *linking* function that allows advertisers to link the large amounts of external personal data available today with Facebook's user information. The linking function that custom audiences provide to advertisers is not a one-to-one function (i.e., advertisers cannot determine the exact Facebook account of a given person), but rather, it is an *aggregate function* that maps PII to a group of Facebook users. In the next section, we show that, despite this limitation, custom audiences can be abused to covertly implement discriminatory advertising by exploiting external data to create lists that selectively include only people with the sensitive attribute.

3.2.1 Potential for Discrimination

To implement discriminatory advertising using custom audiences, an advertiser could simply create a list of PII corresponding selectively to people who have the sensitive attribute, uploading this list of PII to create a custom audience, and then advertising to that custom audience. Since the advertiser does not upload the sensitive attribute (instead uploading only a list of PII), and since the advertising platform itself may not have the sensitive user attribute, such targeting becomes difficult to detect.

Most advertisers already possess significant amounts of customer information (e.g., customer data, information from data brokers); however, even if they do not have such data, there are many other sources of data—including public records, data brokers, and web data—that can be accessed for free or at low cost. We next describe public sources of data from which one can get sensitive attributes for large sets of people; we then demonstrate how these data sources can be used in combination with custom audiences to implement discriminatory advertising.

3.2.2 Public Data Sources

An increasing amount of information about people is publicly available; we now briefly discuss how advertisers could obtain large amounts of external personal information.

Race, age, and gender Most U.S. states release voter records that contain the personal information of all registered voters (names, phone numbers, addresses, etc) along with other sensitive attributes such as race, age, and gender. For example, date of birth and gender are available in the records released by 38 and 34 states, respectively Minkus et al. [2015]; race information is available in the records of eight states (North Carolina, New Mexico, Louisiana, Tennessee, Alabama, Georgia, Florida, and South Carolina) Ansolabehere and Hersh [2013]. Even when the race or gender is not available they can often be predicted with reasonable precision from other attributes Mislove et al. [2011]. For example, Tang et al. Tang et al. [2011] propose a technique to infer the gender of a person from their name with an accuracy of 96.3% while covering more than 95% of users. Other companies such as Catalist[2] aggregate voter records from states and infer missing values of gender (from the first name) and race (from the name and address); the resulting race attributes matched voters' self-reported race 91% of the time Ansolabehere and Hersh [2013].

Criminal history People with criminal records—even those who have completed their sentence—are often victims of discrimination. We quickly survey the U.S. and find that more than 40 states

[2] `https://www.catalist.us`

	Voter Records		Facebook Users		Validation of Custom Audience
Attribute	Number	Percent	Targetable	Targetable %	% matching sensitive attribute
Male	3,438,620	45.5%	6,500	65%	81.5%
Female	3,995,533	52.8%	7,000	70%	91.4%
White	5,303,383	70.1%	6,800	68%	83.8%
Black	1,694,220	22.4%	6,300	63%	82.5%
Asian	79,250	1.0%	6,600	66%	28.8%
Hispanic	163,236	2.2%	5,900	59%	50.8%
Age (18-34)	1,985,117	26.2%	7,100	71%	80.3%
Age (35-54)	2,496,648	33.0%	6,900	69%	79.7%
Age (55+)	3,068,745	40.6%	5,700	57%	61.4%

Table 3.1: **[Results of creating custom audiences using only users with certain attributes from the North Carolina voter records.]** For each sensitive attribute, we created and uploaded a custom audience of 10K random voters with that attribute. Shown is the total number of records per attribute, the number of Facebook users in the resulting *Targetable* custom audience, and the percentage of *Targetable* users who *match the sensitive attribute* as per Facebook's estimates.

in the U.S. make criminal records available online, and that 18 states offer free access to their state-wide criminal record databases; these records often contain significant amounts of personal information such as name, race, gender, and date of birth, along with the specific criminal record. Thus, advertisers can easily create custom audiences consisting only of users in this vulnerable population.

3.2.3 Discriminatory Audience Creation

We briefly demonstrate how it is possible to create discriminatory custom audiences on today's advertising platforms. Note that we *did not* actually advertise to these users or affect them in any way. Rather, our goal here is simply to demonstrate that using only public sources of data, advertisers can target protected classes and vulnerable populations with little effort.

We downloaded the public voter records from North Carolina,[3] giving us 7.5M records. Using data from the voter records, we then created custom audiences on Facebook for each sensitive attribute, selecting a random subset of 10K users from the voter file with each attribute. For example, we created a custom audience of women by uploading a list of 10K voters listed as female; we created a custom audience of white users by uploading a list of 10K voters listed as white. We created these custom audiences by uploading records containing the following fields: last name, first name, city, state, zip code, phone number, and country.

We then examine how many of these records match to Facebook accounts that can be targeted with advertisements, and then evaluate whether the created audiences are indeed discriminatory. Whenever we target an audience (either based on attributes, or by specifying a custom audience), Facebook provides an estimate of the number of users in the audience who can be targeted with advertisements; this estimate is called the *potential reach*.[4] We first target only the custom audiences created, without any additional targeting attributes specified, and use the potential reach estimate to measure how many records in the audience are *Targetable*.

[3] `http://dl.ncsbe.gov/index.html?prefix=data/`

[4] Facebook previously defined the potential reach as "the number of daily active people on Facebook that match the audience you defined through your audience targeting selections."

Finally, in order to validate that advertisements targeted to these custom audiences would indeed be discriminatory, we take each custom audience and then target users with the corresponding sensitive attribute (e.g., for the male voter records audience, we target the Male attribute); we then measure the potential reach, and use the potential reach to measure what percentage of the *Targetable* users in the audience actually have that sensitive attribute (according to Facebook). Ideally, the percentage of *Targetable* users with the sensitive attribute would be 100%; however, Facebook may not know the attributes of some users, may have errors in their matching algorithm, or there may be errors in the user-provided data, making this percentage smaller.

It is important to note that definitions in our data sources (the voter file and census data) do not always line up with the targeting options that Facebook presents. For race, Facebook does not provide race directly but instead provides "ethnic affinity"; this is the same targeting parameter by which Facebook was accused of allowing discriminatory advertising Angwin and Parris Jr. [2016].

Results The results of this experiment are shown in Table 3.1, and we make a number of interesting observations. *First*, the fraction of voter records that are *Targetable* (*i.e.*, online on a daily basis) is both significantly high (over 65% for most audiences we create) and fairly consistent across custom audiences. The only notable outliers are the Age (55+) audience, with only 57% matching.

Second, we observe that the fraction of the *Targetable* audience that matches the sensitive attribute, although it varies fairly widely across the different sensitive attributes, is consistently much higher than the fraction of the general adult population that has those sensitive attributes (assuming the voter records to be representative of the general adult population). In particular, for many sensitive attributes including gender, most races, and all ages, the percentage of *Targetable* audience that matches the sensitive attribute is higher than 80%. We suspect that the reason this fraction is low for the Asian attribute is due to the fact that race is an attribute that users typically do not upload to Facebook directly; however, we leave determining the source of this inconsistency to future work. We also note that even for these cases, the fraction of the *Targetable* audience that matches the sensitive attribute is significantly higher than the fraction of the voter records with the sensitive attribute. Taken together, our results show that advertisers can exploit public records to easily target discriminatory advertisements to a large number of people.

3.2.4 Summary

We explored the inherent risks that custom audiences induce for end users by allowing the linking of external information with Facebook's user data. We demonstrated the ease with which malicious advertisers could leverage the custom audience feature now present on advertising platforms like Facebook to implement discriminatory advertising. In fact, the wide variety of sources of public data available today means that even if an advertiser does not possess customer records of its own, it can easily find data sources to feed into custom audience creation.

3.3 Attribute-based Targeting

In this section, we examine how Facebook's attribute-based targeting mechanism can be used to launch discriminatory ads. First, we briefly explain how attribute-based targeting works and then examine the potential for abusing it.

Attribute-based audience selection: In brief, attribute-based targeting refers to the process of selecting an ad audience by specifying that recipients need to have a certain attribute or a combination of attributes; this is the traditional way of targeting ads on Facebook. For each

user in the US, Facebook tracks a list of over 1,100 binary attributes spanning demographic, behavioral and interest categories that we refer to as *curated attributes*. Additionally, Facebook tracks users' interests in entities such as websites, apps, and services as well as topics ranging from food preferences (e.g., pizza) to niche interests (e.g., space exploration). We refer to these as *free-form attributes*, as they number at least in hundreds of thousands. It is unclear how exactly Facebook infers these attributes, but from their own description[5] this information can be gathered in many different ways such as user activity on Facebook pages, apps and services, check-ins with Facebook, and accesses to external webpages that use Facebook ad technologies. Beyond specifying a target region, language, age and gender for their ad, advertisers can choose that an ad should be shown to people that have some of these curated or free-form attributes turned on or off.

3.3.1 Potential for Discrimination

The potential for discrimination on the Facebook ad platform was first publicly highlighted when researchers discovered the ability to exclude people based on their "ethnic affinity" (a curated attribute) when targeting ads related to housing Angwin and Parris Jr. [2016]. Facebook responded by banning the use of ethnic affinity attribute for certain types of ads Facebook [2017]. More recently, researchers discovered the ability to target people interested in or holding anti-semitic viewpoints via free-form attributes like "jew haters" Angwin et al. [2017b].

Race	Most inclusive	Most exclusive
Asian	US Politics: Liberal (8%, 2.76) Frequent travelers (15%, 2.70) Interest: Vegetarianism (7%, 2.23)	US Politics: Very Conservative (14%, 0.30) African American affinity (17%, 0.41) Interest: Country music (20%, 0.48)
Black	African American affinity (17%, 7.06) US Politics: Very Liberal (12%, 6.44) Interest: Online games (9%, 4.91)	US Politics: Very Conservative (14%, 0.18) US Politics: Conservative (17%, 0.22) Interest: Mountain biking (6%, 0.35)
Indian	Interest: Motorcycles (7%, 2.08) Interest: Online games (9%, 2.04) Interest: Ecotourism (6%, 1.96)	US Politics: Very Conservative (14%, 0.50) Away from hometown (22%, 0.51) Primary OS Mac OS X (7%, 0.56)
White	US Politics: Very Conservative (14%, 5.19) US Politics: Conservative (17%, 3.77) Interest: Hiking (11%, 2.27)	African American affinity (17%, 0.15) US Politics: Very Liberal (12%, 0.16) Interest: Online games (9%, 0.20)

Table 3.2: [**Most inclusive and exclusive curated attributes for each race.**] In parentheses are the recall and representation ratio for a population from North Carolina. These were obtained by uploading voter records filtered to contain only a single race, and then measuring the size of the subaudience targeted by each attribute. Attributes present in less than 5% of the population are not considered.

These findings raise several questions about the potential for discriminatory targeting using Facebook's curated and free-form attributes. First, given that ethnic affinity-based targeting was disallowed for its potential correlation with ethnicity (race) of users, are there other demographic, behavioral, or interest attributes that are similarly correlated, if not more? Second, given that there exist hundreds of thousands of free-form attributes, can malicious advertisers find *facially neutral* free-form attributes that disproportionately target or exclude users of a sensitive group. For example, an advertiser seeking to create an audience excluding certain ethnic groups may choose to select her target audience from users interested in particular news media sites or magazines.

[5] `https://www.facebook.com/ads/about/?entry_product=ad_preferences`

To answer these questions and understand how vulnerable the Facebook ad platform is to these kinds of *indirect discrimination*, we investigate how strongly curated attributes other than "ethnic affinity" correlate with ethnicity and whether free-form attributes that are facially unrelated to sensitive attributes can be used as proxies for sensitive attributes. We executed these experiments by automatically querying the Facebook ad interface for the number of people belonging (or not belonging) to sensitive groups that have a certain curated or free-form attribute.

3.3.2 Discriminatory Audience Creation

We now explore how both curated and free-form attributes are correlated with ethnicity.

Curated attributes: We conduct our analysis in the way described in Section 3.2.3. We use the custom audience mechanism to create groups of people from the North Carolina voter records that only contain particular ethnicities (White, African-American, Asian, and Hispanic). We then create sub-audiences by choosing to only target users matching each curated attribute and observe the size estimates of these sub-audiences. The percentage of users from each audience for whom Facebook inferred a curated attribute reveals how prevalent the attribute is within the audiences of different ethnicities.

The top three inclusive and exclusive attributes per ethnicity are shown in Table 3.2. The results point out that ethnic affinity is by far not the only and—in many cases—not even the most disparate feature with respect to ethnicity. For example, when targeting Asians on Facebook, it is more effective to do so based on political leaning or eating habits. The tradeoffs between representation ratio and recall for members outside the sensitive group, which an advertiser has to consider when aiming to exclude sensitive group members, can also be gauged from the table. In particular, there are a number of curated attributes with low representation ratio (i.e., high disparity), some of which achieve high recall for members not belonging to the sensitive group.

Free-form Attribute	**Potential Target (PT)**	**PT Audience (%)**	**US Audience (%)**
Marie Claire	Female	90%	54%
myGayTrip.com	Man interested in Man	38.6%	0.38%
BlackNews.com	African American affinity	89%	16%
Hoa hoc Tro Magazine	Asian American affinity	95%	3.4%
Nuestro Diario	Hispanic affinity	98%	16%

Table 3.3: [**Free-form attributes that may be used for discriminatory targeting.**] We show the percentage of the attribute audience that are members of the sensitive group as well as the fraction of the U.S. Facebook population that are members of the sensitive group, as a reference.

Topic	**Free-form attributes**
Religion	Islam (5.7M), Catholic Church (6.5M), Evangelicalism (5.6M)
LGBT	LGBT community(21M), Gay pride (13M), Same-sex marriage(4.2M)
Vulnerable people	Addicted (100K), REHAB (450K), AA (50K), Support group (610K)

Table 3.4: [**Examples of free-form attributes that can be targeted by advertisers.**] In the parenthesis, we show the number of audience that can be targeted or excluded with the attribute.

Free-form attributes: We begin our investigation by gathering an extensive (though not exhaustive) list of free-form attributes that are supported by the Facebook marketing API.[6] The API provides two useful calls that we exploit: i) given a piece of text, the API provides a list of free-form attributes that *match* the given text; and ii) given an attribute, the API provides a list of other *related* attribute suggestions. For instance, the list of related attributes for 'The New York Times' includes 'The Washington Post', 'The Wall Street Journal', and 'The Economist'.

We start with a seed set of names of news outlets extracted from three different sources: Google News Leskovec et al. [2009], List of Newspapers,[7] and the top 1,000 newspapers from Alexa.[8] We first identify around 3,000 free-form attributes that exactly match with the names of the news outlets. We then execute a snowball sampling on these attributes, using Facebook's related attribute suggestions recursively starting from them. This process resulted in retrieving nearly 240,000 free-form attributes.

We begin by trying to find attributes from the above set of 240,000 attributes that can be used to primarily target or exclude people belonging to sensitive groups. Table 3.3 shows example free-form attributes that could be exploited for discriminatory targeting. For example, the attribute 'Marie Claire' has an audience with 90% of women, a much larger fraction than the proportion of U.S. women in Facebook (54%). Similarly, the attribute 'myGayTrip.com' has an audience of 38.6% men interested in men, while only 0.38% of the U.S. population in Facebook consists of men interested in men. We also identified a number of attributes with very biased audiences in terms of racial affinities. For example, 'BlackNews.com' has an audience with 89% of the users with African American affinity (in contrast with 16% of African American affinity in the reference population), the audience of 'Hoa hoc Tro Magazine' is composed of 95% users with Asian American affinity, which corresponds to 28 times more in comparison with the reference population. Similarly, 'Nuestro Diario' has an audience with 98% of Hispanic affinity (16% on the reference population). These results suggest that a malicious advertiser could easily find free-form attributes to launch discriminatory ads based on gender, race, and sexual orientation.

More worryingly, some free-form attributes allow a malicious advertiser to target people based on their beliefs. Table 3.4 presents a few sensitive free-form attributes from our dataset along with their potential audience in the U.S. These attributes correspond to a large audience with specific religious beliefs, including 'islam' (5.7M), 'catholic church' (6.5M), and 'evangelicalism' (5.6M). Thus, although it is not possible to target religion using curated attributes, one can use free-form attribute targeting to narrow the audience to people who are interested in a specific religion. Finally, we note that it is possible to target or exclude gay and LGBT users (or people sympathetic to their causes) via attributes like 'LGBT community' (21M), 'Gay pride' (13M), 'Same-sex marriage' (4.2M), as well as groups of vulnerable people, including 'Addicted' (100K), 'REHAB' (450K), 'AA' (50K). While this last set of free-form attributes might be useful, for example, for an advertiser to prevent addicted people to receive ads about alcoholic beverages, a discriminatory advertiser could explicitly exclude them.

Using Facebook's attribute suggestions: We first investigate the free-form attributes suggested by Facebook to better understand the criteria used to select these suggestions. Table 3.5 shows the suggestions returned by the Facebook Marketing API (right column) given a free-form attribute (left column). We selected attributes associated with news outlets biased towards conservative audience to check whether their respective suggestions are also similarly biased. For

[6] `https://developers.facebook.com/docs/marketing-api`

[7] `http://www.listofnewspapers.com/`

[8] `https://www.alexa.com/topsites/category/Top/News/Newspapers`

Very Conservative - U.S. Facebook Population (13.9%)	
Input Attribute	**Attribute Suggestions**
Townhall.com (79.5%)	The Daily Caller (67.1%), RedState (84.3%), TheBlaze (59.6%), Hot Air (news site) (79.4%)
The American Spectator (70.7%)	The Daily Caller (67.1%), Townhall.com (79.4%), The American Conservative (85.2%), National Review (78.6%), Weekly Standard (72.2%), Human Events (53.3%), Commentary (34.5%), RedState (84.3%), Harper's Magazine (11.7%), U.S. News & World Report (18.6%)
The Patriot Post (70.7%)	American Patriot (68.4%), Patriot Nation (54.3%), Patriot Update (84.2%), NewsBusters.org (78.7%), Guns & Patriots (61.2%), RedEye (9.1%), America's Conservative Voice (74.4%)
American Thinker (67.5%)	National Review (78.6%), Fox Nation (75.3%)
The Cullman Times (63%)	Montgomery Advertiser (40.4%), The Huntsville Times (40.8%), The Tuscaloosa News (44.8%), al.com (42.5%)

Table 3.5: **[Suggestions for the most conservative news outlets.]** The leftcolumn shows a set of free-form attributes for conservative news outlets and the right column shows the corresponding free-form attributes suggested by Facebook. The percentage of very conservative users in the audience of each of these free-form attributes is shown in parentheses.

instance, almost 80% of the audience of Townhall.com[9] are "very conservative" Facebook users, whereas the average amount of very conservative U.S. users of Facebook is about 13.89%. We note that the audiences corresponding to most of the suggested attributes also exhibit a strong bias towards conservative audience. From all suggestions presented, only Harper's Magazine[10] and RedEye[11] have a less conservative audience in comparison with the U.S. distribution.

A malicious advertiser could exploit free-form attribute suggestions from Facebook in two different ways. First, a malicious advertiser could exploit the Facebook's attribute suggestions to discover attributes that are facially neutral, but are similarly biased as a given free-form attribute. For example, one suggestion from Facebook for 'myGayTrip.com' is the free-form attribute 'Matt Dallas', who is a gay actor.[12] 19.4% of the audience for 'Matt Dallas' are men interested in men, which is 51 times more than the U.S. distribution (0.38%). Thus, a malicious advertiser may use 'Matt Dallas' as a facially neutral proxy for targeting or excluding gay users.

Second, an advertiser can use the suggestion mechanism to search for extremely biased free-form attributes. For example, suppose an advertiser is interested in conservative leaning audiences and the most biased free-form attribute they know is 'Fox', with 37% of conservative audience. The advertiser can start with 'Fox' and keep choosing more and more conservative attribute suggestions until she reaches attributes with extreme conservative audience bias. Below, we show a sequence of suggested attributes, starting from 'Fox', that leads to 'The Sean Hannity Show', a free-form attribute with 95% conservative audience.

Fox (37%) → Fox News Channel (67%) → Sean Hannity (88%) → Mark Levin (93%) → Rush Limbaugh (93%) → The Rush Limbaugh Show (94%) → The Sean Hannity Show (95%).

[9] `https://www.facebook.com/townhallcom/`

[10] `https://www.facebook.com/HarpersMagazine/`

[11] `https://www.facebook.com/TheRedEye/`

[12] `https://en.wikipedia.org/wiki/Matt_Dallas`

3.3.3 Summary

In this section, we demonstrated that many curated attributes (beyond ethnic affinity) exhibit correlations with sensitive attributes like race, which makes them potential vectors for discrimination. We also investigated whether the free-form attribute targeting mechanism allows advertisers to target or exclude sensitive groups of users in a discriminatory manner. Specifically, we showed that advertisers can circumvent existing limitations on targeting users based on their interests in sensitive topics like religion and sexual orientation. Furthermore, we show that malicious advertisers can exploit Facebook's suggestions to discover new facially neutral free-form attributes that allow extremely biased targeting.

3.4 Look-Alike Audience Targeting

In this section, we show how the recently introduced look-alike audience targeting mechanism can be exploited by advertisers to covertly implement discriminatory advertising. We first briefly describe the look-alike audience targeting feature of Facebook; we then explain how this feature can be exploited to implement discriminatory advertising.

Look-alike audience selection: Recently, Facebook introduced the *look-alike audience* targeting feature to help advertisers reach people that are *similar to* (i.e., look like) their existing set of customers.[13] Look-alike audiences are a particularly useful feature for advertisers who have limited data about their customers and want to grow their customer base. Advertisers can use it to outsource the job of marketing (i.e., identifying the attributes of their potential customers and finding them) to Facebook.

To select look-alike audiences, advertisers need to first provide Facebook with information about their existing (initial) set of customers called the *source audience*. An advertiser can choose source audience users in a variety of ways, including by uploading their customers' PII (similar to creating a custom audience) or by specifying them to be the fans of their Facebook page.

After specifying a source audience, Facebook allows advertisers to specify a geographical region (either countries or groups of countries) from which the look-alike audience should be chosen. Facebook orders (ranks) all users in the geographical region based on their similarity to (i.e., how closely they look like) the source audience and allows advertisers to select look-alike audiences by specifying a percentile range (e.g., <2% or 2%-4%) over these ordered users from the geographical region's population. Thus, an advertiser can select X to Y percentile of closest matching users from any country's population to target. In practice, Facebook limits Y to 10%.

3.4.1 Potential for Discrimination

Our concern is that a malicious advertiser seeking to place discriminatory advertisements could exploit look-alike audiences as follows: they could start by creating a highly biased (highly discriminatory) source audience and use the look-alike audience feature to find a larger set of users that is similarly biased, effectively scaling the bias to much larger populations. Put differently, our concern is that when the source audience is discriminatory, its look-alike audience would also be discriminatory. In the following sections, we first investigate whether biases in source audience selection propagate to look-alike audience selection. Later, we show how an advertiser seeking to selectively target people of a particular race could simply create a small (in the order of a few thousands) but highly biased source audience consisting primarily of people of a particular

[13] `https://www.facebook.com/business/help/164749007013531`

race (as described in Section 3.2.3) and use it to effectively target a large (in the order of tens of millions) yet similarly—or worse, exaggeratedly—biased look-alike audience.

3.4.2 Bias in Look-alike Audience Selection

Over-represented Attributes	Under-represented Attributes
Source Audience	
African American affinity (5.52)	Asian American affinity (0.09)
US politics: very liberal (3.21)	Hispanic (Spanish dominant) affinity (0.09)
Liberal content engagement (2.98)	Expats: Mexico (0.11)
Interest: Gospel music (2.64)	Hispanic (all) affinity (0.18)
Interest: Dancehalls (2.51)	Expats: all countries (0.22)
2% Look-Alike Audience	
African American affinity (5.24)	Hispanic (Spanish dominant) affinity (0.10)
Liberal content engagement (4.16)	Expats: Mexico (0.13)
US politics: very liberal (3.29)	Asian American affinity (0.13)
Interest: Gospel music (3.07)	Hispanic (all) affinity (0.19)
Interest: Soul music (2.32)	Expats: all countries (0.24)
2–4% Look-Alike Audience	
African American affinity (5.06)	Asian American affinity (0.17)
Liberal content engagement (3.61)	Hispanic (Spanish dominant) affinity (0.18)
US politics: very liberal (3.37)	Expats: Mexico (0.19)
Interest: Gospel music (2.72)	Hispanic (all) affinity (0.29)
Interest: Dancehalls (2.54)	Expats: all countries (0.37)

Table 3.6: **[Top 5 most over-represented and under-represented attributes in a source audience of African Americans and its two closest look-alike audiences.]** In parentheses, we show the value of the representation bias of each attribute.

In this section, we construct several highly biased source audiences and check if and how the selection biases in source audience propagate to look-alike audiences.

Similar to what we did in Sections 3.2 and 3.3, we use the North Carolina voter database to construct several groups of 10,000 randomly selected people based on their ethnicity (Asian, Black, White, Hispanic), gender, political affiliation (registered Democrat or Republican) and age (18-24, 24-35, 35-54, 55+). We construct a source audience corresponding to each group and for each source audience, we ask Facebook to construct look-alike audiences from the US in five percentile ranges: closest matching 2%, 2-4%, 4-6%, 6-8%, and 8-10% of the US population. Note that the audiences in the different percentile ranges do not overlap with one another and each subsequent percentile range becomes less similar (i.e., less closely matching) to the source audience.

Each of the five look-alike audiences we create (for every source audience of 10,000 people) consist of approximately 4.2 million people, thus totaling to an approximate of 21.1 million unique people in the US. Thus, look-alike audiences allow expansion of the source audience by over three orders of magnitude. The key remaining question is whether the look-alike audience selection reflects the biases in source audience selection.

To capture the biases in our audience selection, we define a measure that we call *representation bias* of a target audience for every user attribute f maintained by Facebook. Simply put, representation bias captures how disproportionately an attribute is observed amongst the target

audience (**TA**) compared to the people in the geographic location from where the look-alike audience is being selected (the geographic location is the US in our scenario and we refer to people in the US as the relevant audience, **RA**). More formally, the representation bias of an attribute f in an audience is defined as

$$\text{rep\_bias}_f(\mathbf{TA}, \mathbf{RA}) = \frac{|\,\mathbf{TA}_f\,|}{|\,\mathbf{TA}\,|} \frac{|\,\mathbf{RA}\,|}{|\,\mathbf{RA}_f\,|}, \tag{3.4}$$

where similar to the representation ratio (Equation (3.1)), $\mathbf{TA}_f$ and $\mathbf{RA}_f$ are the subsets of people with attribute f in **TA** and **RA** respectively. We leave out attributes with very low prevalence in Facebook from our analysis (i.e., attributes for which $|\,\mathbf{RA}_f\,|/|\,\mathbf{RA}\,| < 0.01$).

Knowing the representation bias of each attribute allows us to construct a ranking of attributes from most to least biased; we refer to attributes at the top of the ranking as *over-represented* and those at the bottom to be *under-represented* in a target audience. Table 3.6 shows the top 5 over-represented and under-represented attributes for the source audience of African Americans and its 2% and 2–4% (the most similar two) look-alike audiences. The table shows that a majority of attributes that were found to be overrepresented in the source audience remain so for the look-alike audiences; similar behavior can be observed for the underrepresented attributes. These results—particularly the presence of multicultural affinity attributes—suggest that Facebook is using its extensive set of attributes to likely infer the biases that we introduced into our source audience. Moreover, it is propagating these biases to the selection of look-alike audiences, constantly over-representing African Americans and under-representing Hispanics and Asian Americans compared to their proportions in the national population.

To further validate our findings above, we take the top 10 over-represented and under-represented attributes in the source audience and computed their average rank in the look-alike audiences. We performed these computations for differently biased source audiences (selected along the basis of gender, age, ethnicity and political affiliation). Figure 3.1 shows how the average rank changes across the look-alike audiences given by Facebook. We can see that attributes that were most over- and under-represented in source audience tend to stay, on average, amongst the most over- and under-represented in the look-alike audiences, respectively. These results strengthen our inference that the look-alike audience feature in Facebook is able to both capture the biases in a source audience and propagate the biases to the larger audiences it helps construct.

3.4.3 Discriminatory Audience Creation

Having observed that the look-alike audience selection mimics the biases of the source audience selection, we now check whether the bias propagation is sufficiently strong to lead to discriminatory audience creation. To answer this question, we compute the disparity of the sensitive attribute on which the source audience was biased, and observe how disparate that attribute remains in the look-alike audiences made by Facebook. Note that since we are observing look-alike audiences built from source audiences where the sensitive attribute was severely exaggerated, we expect the disparity measure to reflect the disparity in favor of the attribute.

Figure 3.2 shows how source audiences that were disparate in favor of an ethnic group tend to produce look-alike audiences also disparate in favor of that ethnicity; although as the audiences become less similar, the disparity tapers off. Only one of these audiences, the 2% look-alike audience for White, has a disparity below 1.25, the threshold obtained from the 80% disparate impact rule Biddle [2005]. These results show that look-alike audiences selected using highly biased source audiences can be highly discriminatory.

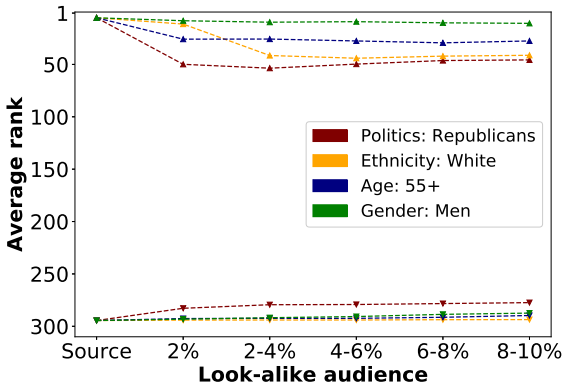


Figure 3.1: **[Comparison of the average ranks of top 10 over-represented and under-represented attributes in look-alike audiences built from different types of biased source audiences.]** Average ranks for over-represented attributes are indicated by upward triangles, downward triangles are used for the average ranks of under-represented attributes.

3.4.4 Summary

In this section, we investigated whether it is possible to start with a small discriminatory source audience and then leverage Facebook's look-alike audience feature to construct a considerably larger discriminatory audience. We show that in order to select a look-alike audience, Facebook tries to infer the attributes that distinguish the audience from the general population and propagates these biases in the selection of look-alike audiences. Such bias propagation can amplify the explicit (intentionally created) or implicit (unintentionally overlooked) biases in a source audience of a few thousand to a look-alike audience of tens of millions. As Facebook is actively involved in the selection of the look-alike audience, one might argue that Facebook needs to be more accountable for the selection of such a discriminatory audience.

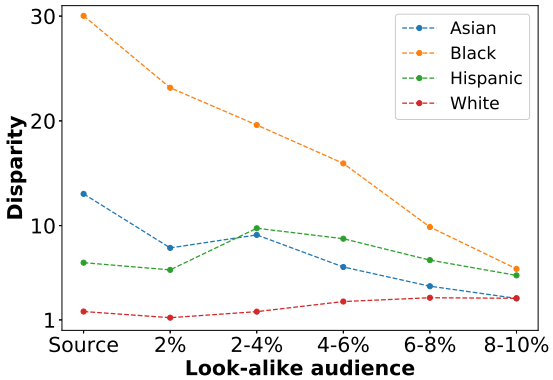


Figure 3.2: **[Disparity (in favor) for each ethnicity when the look-alike audiences are created from an audience biased towards that ethnicity.]** The results indicate that Facebook look-alike audiences are capable of reproducing the disparity in favor of a sensitive attribute from the source audience.

3.5 Concluding Discussion

Recently, concerns have been raised about the potential abuse of online advertising platforms to target ads related to housing, employment, and financial services only to users of a particular race, in violation of anti-discrimination laws. In this chapter, we set out to investigate the following high-level question: *can a malicious advertiser leverage the different targeting methods offered by platforms like Facebook to target users in a discriminatory manner?* At a high-level, our study makes the following contributions:

(i) We argue that the determination of whether a targeted ad is discriminatory should not be made based on the use or non-use of specific user attributes by advertisers. Rather, inspired by the notion of *disparate impact* Feldman et al. [2015], we propose a simple outcome-based measure for discriminatory targeting that is computed independently of the user attributes used in targeting.

(ii) Next, using public voter record data in the US, we conduct an empirical study demonstrating that several user attributes in Facebook, beyond the much-criticized "ethnic affinity," show strong positive and negative correlations with users belonging to different races. Worse, Facebook's related attribute suggestions can be exploited by advertisers to discover facially-neutral attributes that can be used for highly discriminatory audience targeting. Thus, simply banning certain attributes is insufficient to solve the problem.

(iii) Finally, we explore the vulnerability of two previously overlooked methods of targeting supported by Facebook namely, *PII-based (custom) audience targeting* and *look-alike audience targeting*. We show that both these methods can be exploited by a malicious advertiser to include or exclude users with certain sensitive features *at scale* (i.e., in the order of tens of millions of users).

Future work – Towards detecting and mitigating ad discrimination: Our study here has largely focussed on *understanding the problem* of discriminatory advertising rather than *proposing solutions* for detecting or mitigating discriminatory targeting. However, in the process, we lay the foundations for the future solutions. First, the discrimination measure proposed here could be used when designing procedures to detect discrimination in the future. Second, we argue that the look-alike audience selection feature also presents a promising solution to the problem of mitigating discrimination in audience selection. Specifically, ad platform providers could expand the targeted audience to include look-alike (most similar) users that belong to under-represented groups (rather than select all look-alike audience).

Chapter 4

The Role of Representational Invariance in Transfer Learning

Transfer learning is a powerful technique for knowledge-sharing between different tasks. Recent work has found that the representations of models with certain invariances, such as to adversarial input perturbations, achieve higher performance on downstream tasks. These findings suggest that invariance may be an important property in the context of transfer learning. However, the relationship of invariance with transfer performance is not fully understood yet and a number of questions remain. For instance, which invariances are important for transfer performance? What is the relationship of invariance with other factors of the pretraining task in increasing performance?

In this work, we systematically investigate the importance of representational invariance for transfer learning as well as how it interacts with other parameters during pretraining. To do so, we introduce a family of synthetic datasets that allows us to precisely control factors of variation both in training and test data. Using these datasets, we a) show that for learning representations with high transfer performance, invariance to the right transformations is more important than most other factors such as the number of training samples, the model architecture and the identity of the pretraining classes, b) show conditions under which invariance can harm the ability to transfer representations and c) introduce a measure to quantify invariance to certain transformations and use it to explore how transferable invariance is between tasks.

Relevant Publications: The work in this chapter is published as "Understanding the Role of Invariance in Transfer Learning", **T. Speicher**, V. Nanda, and K. P. Gummadi. In *Transactions on Machine Learning Research, TMLR 2024.*

4.1 Controlling and Evaluating Invariance in Representations

We begin by describing our approach to controlling for and evaluating the presence of invariance in representations.

4.1.1 Terminology

Notation. We operate in the standard supervised learning setting and denote by $\mathcal{D} = \{(\mathbf{x}_i, y_i)_{i=1}^N\}$ a dataset consisting of N examples $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ and associated labels $y \in \mathcal{Y} = \{1, \ldots, K\}$. The task is to find a function $g : \mathcal{X} \mapsto \mathcal{Y}$ that minimizes the empirical risk on $\mathcal{D}$. g is a neural network that is trained by minimizing the categorical cross-entropy loss over its predictions. For our purpose it is convenient to write g as $g = g_{cls}(g_{rep}(.))$ where $g_{rep} : \mathcal{X} \mapsto \mathcal{Z}$ maps inputs $\mathbf{x}$ to representations $\mathbf{z} \in \mathcal{Z} \subseteq \mathbb{R}^m$ and $g_{cls} : \mathcal{Z} \mapsto \mathcal{Y}$ maps representations to predictions $\hat{y}$. We will refer to g_{rep} simply as g if the meaning is clear from the context.

We primarily focus on representations at the penulatimate layer that are fixed after pretraining in this work. We study fixed representations, since retraining g_{rep} would change the invariance properties of the representations, and thus would not allow us to cleanly answer the question of how the invariance properties of representations affect their downstream performance. We focus on the penulatimate layer since its representations are most commonly used for transfer learning under the linear probing regime [Chen et al., 2020b, Kornblith et al., 2019, Salman et al., 2020].

Invariance. We are interested in the invariance of representations to transformations in a model's input. In the context of this work, we define invariance as follows: Given a transformation $t : \mathcal{X} \mapsto \mathcal{X}$, we say that model g is *invariant* to t if $g(t(x)) = g(x)$ for all $x \in \mathcal{X}$. We say that g is invariant to a set of transformations T if it is invariant to all transformations $t \in T$.

4.1.2 Constructing Invariant Representations

The main approaches to construct invariant representations are training with data augmentations [Antoniou et al., 2017, Benton et al., 2020, Chen et al., 2020a, Cubuk et al., 2018] and architectures that are equi- or invariant by construction [Cohen and Welling, 2016, Zhang, 2019]. The data augmentation approach randomly applies certain transformations to the input data during training. Since the transformations are independent from the training data, models trained this way have to become invariant to them in order to minimize their loss. Invariant architectures on the other hand build specific inductive biases into the model architecture, that make them equi- or invariant to certain transformations, such as rotation or translation [Worrall et al., 2017].

In this work, we choose data augmentations — *i.e.* input transformations — as the method to construct invariant representations. Our choice is motivated by the fact that a) training with data transformations is the most commonly used method to construct invariant representations in the literature, b) it is flexible and allows us to construct invariances to any type of transformation that we can define through code (as opposed to architectural invariance, which requires a different model design for each type of invariance) and c) it allows us to leverage the same mechanism we use to train invariant networks to also evaluate the invariance of representations. In Section 4.3 we show that training with input transformations indeed leads to representations that are invariant to those transformations.

4.1.3 Controlling Data Transformations via Synthetic Data

To both construct representations with known invariance properties and to evaluate them on tasks with known invariance requirements, we need to be able to control the transformations present in a model's training and test data. For instance, to determine whether the representations of a model are invariant to a particular transformation, it is necessary to probe it with inputs that only differ by this transformation.

Using real-world datasets for this task is very challenging for two reasons. First, information about how inputs are transformed relative to each other, for example whether objects in images differ by certain translations or rotations, is typically not available in most real-world datasets, beyond coarse-grained information like the class that an input belongs to. Second, even if such annotations were available for real-world data, they would come with the risk of confounders between transformations and other data factors, such as the objects present in the data. For example, images of huskies might all have been taken on snowy background [Ribeiro et al., 2016]. To carefully control for such confounders, data would have to be sampled in a randomized manner in a lab setting, thus diminishing realism benefits.

Therefore, in order to properly study invariance in representations, we introduce a family of synthetic image datasets that allows us to precisely control which objects are present in images, as well as the transformations acting on them. We call it the family of *Transforms-2D* datasets.

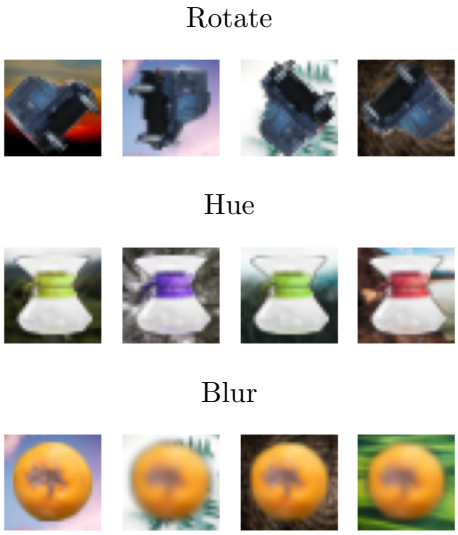


Figure 4.1: **[Transforms-2D dataset examples.]** Example images sampled from the Transforms-2D dataset. Each row shows a different transformation being applied to one of the object prototypes.

A Transforms-2D dataset $\mathcal{D}(O, T)$ is defined by a set of foreground objects O and a set of transformations T, with associated distributions P_O and P_T over O and T, respectively. In addition, there is a set B of background images with uniform distribution P_B, which is the same across all datasets in the family. To sample an image from $\mathcal{D}(O, T)$, we sample an object $o \sim P_O$, a transformation $t \sim P_T$, and a background $b \sim B$, and then create an image as $b(t(o))$, where $t(o)$ is the transformed object and $b(.)$ denotes pasting onto the background b. Each object $o \in O$ defines a class in the dataset, with each sample having o as its class. That means that each class is based on a single object prototype $o \in O$, and different instances of the same class are created by applying transformations from T to the prototype. Sample images for different transformations are shown in Figure 4.1.

Foreground objects and background images. Each sample from a $\mathcal{D}(O, T)$ dataset consists of a transformed foreground object pasted onto a background image. Foreground objects O are a subset of 61 photographs of real-world objects, such as food and household items, vehicles, animals, etc. Each object image has a transparency masks, such that it only contains the object and no background. P_O samples objects uniformly at random from O. We use the same set B of background images for each $\mathcal{D}(O, T)$, which are photographs of nature scenes, chosen uniformly at random from a set of 867 candidates. The foreground and background images are based on the SI-score dataset [Djolonga et al., 2021]. We subsample images to have a resolution of 32 x 32 pixels, since this size provides enough detail for models to be able to distinguish between different objects, even with transformations applied to them, while allowing for faster iteration times in terms of training and evaluation.

Transformations. A transformation t is typically a combination of multiple transformations of different types, *e.g.* a translation followed by a rotation. Therefore, the set of transformations T is the Cartesian product of sets of different transformation types, *i.e.* $T = T^{(1)} \times \ldots \times T^{(k)}$, and transformations $t \in T$, $t = t^{(1)} \circ \ldots \circ t^{(k)}$ are concatenations of the transformations of each type $t^{(i)} \in T^{(i)}$. We denote the cardinality of T by the number of transformation types that it uses, *i.e.* if $T = T^{(1)}, \ldots, T^{(k)}$, then $|T| = k$. To sample a transformation $t \sim P_T$, we sample transformations $t^{(i)} \sim P_{T^{(i)}}$ for each type i and concatenate them to form t. We use

three categories of transformation types that span a comprehensive set of image manipulations: i) *geometric* (translate, rotate, scale, vertical flip, horizontal flip, shear), ii) *photometric* (hue, brightness, grayscale, posterize, invert, sharpen), and iii) *corruption* (blur, noise, pixelate, elastic, erasing, contrast). Additional information on and examples of the transformations, including information about their distributions $P_{T^{(i)}}$ can be found in Appendix B.1.1.

In our experiments we use 50,000 training samples to train models with specific invariances, as well as 10,000 validation and 10,000 test samples, if not stated differently. These numbers mimic the size of the CIFAR datasets [Krizhevsky et al., 2009].

4.1.4 Measuring Invariance in Representations

In order to measure how invariant representations are to specific transformations T, we measure how much they change in response to applying transformations from T to the input, *i.e.* how *sensitive* representations are to transformations from T. Given a model g, a set of transformations T and objects O, we measure the sensitivity of g to T based on the L2-distance as

$$sens(g|T,O) = \frac{1}{C} \mathop{\mathbb{E}}_{x_1,x_2\sim\mathcal{D}(T,O)} [\, \|g(x_1) - g(x_2)\|_2 \,] \tag{4.1}$$

where $x_1, x_2 \sim \mathcal{D}(T,O)$ are pairs sampled from $\mathcal{D}(T,O)$ as $x_1 = b(t_1(o))$ and $x_2 = b(t_2(o))$, for $o \sim P_O$, $t_1, t_2 \sim P_T$ and $b \sim P_B$, *i.e.* x_1 and x_2 only differ in the transformation applied to them. C is a normalization constant, which measures the average distance between any two random samples from $\mathcal{D}(T,O)$, *i.e.* $C = \mathop{\mathbb{E}}_{x,x'\sim\mathcal{D}(T,O)} [\, \|g(x) - g(x')\|_2 \,]$, without any sampling constraints on x, x'. Intuitively, $sens(g|T,O)$ measures the ratio of the distance between two samples that only differ in their transformation, relative to the average distance between any two samples from $\mathcal{D}(T,O)$. The lower $sens(g|T,O)$, the more invariant the representations are to the transformations in T. In our experiments we approximate each of the expectations as an average over 10,000 sample pairs.

4.2 How Important is Representational Invariance for Transfer Learning?

We want to understand the factors that determine whether a representation trained on one dataset transfers, *i.e.* performs well, on another dataset. In particular, we are interested in understanding how important invariance is for transfer performance, compared to other factors, and whether the wrong invariances can harm transfer performance.

4.2.1 How Important is Invariance Compared to Other Factors?

To better understand how important representational invariance is compared to other factors, we leverage the Transforms-2D dataset to create training and test tasks with known required invariances. In particular, we compare invariance against the following factors: *dataset size*, *model architecture*, and the *number and identity of classes*.

We set up experiments that mimic the typical transfer learning setting. In each experiment, we sample disjoint sets of training and evaluation objects O_t and O_e for the source and target task, respectively, as well as disjoint sets of training and evaluation transformations T_t and T_e. Using these sets we create datasets as described below and train models on a training task, freeze their weights and transfer their penultimate layer representations to a target task, where we train a new linear output layer. Both the training and target task are classification problems

based on the Transforms-2D dataset, which differ in the set of objects that need to be classified. For our experiments, we set $|O_t| = |O_e| = 30$ (roughly half the set of 61 available objects) and $|T_t| = |T_e| = 3$, with transformation types sampled uniformly among the 18 available types of transformations in Transforms-2D.

Effect of invariance: To measure the effect of invariance on transfer performance, we train pairs of models on two versions of the training dataset, $\mathcal{D}_s = \mathcal{D}(O_t, T_e)$ and $\mathcal{D}_d = \mathcal{D}(O_t, T_t)$, respectively, whose only difference is that $\mathcal{D}_s$ uses the *same* transformations T_e as the target dataset, while $\mathcal{D}_d$ uses the *disjoint* set of training transformations T_t. This setup provides us with a "same-transformations" and a "different-transformations" model g_s and g_d, respectively, which only differ in the transformations in their training dataset and thus the invariances in their representations. Comparing the performance of these two models on the target dataset $\mathcal{D}_e = \mathcal{D}(O_e, T_e)$ after fine-tuning allows us to quantify how important having the right invariances is for the target task.

For example, the target task might transform objects by rotating, blurring and posterizing them (T_e). In order to perform well on this task ($\mathcal{D}_e$), a model needs to learn representations that are invariant to these transformations. Models g_s that are pretrained on source datasets $\mathcal{D}_s$ with the same transformations T_e acquire these invariances, whereas models g_d that are trained on datasets $\mathcal{D}_d$ with disjoint transformations T_t, *e.g.* on a dataset where objects are translated, scaled and color inverted, would not acquire the invariances required for the target task.

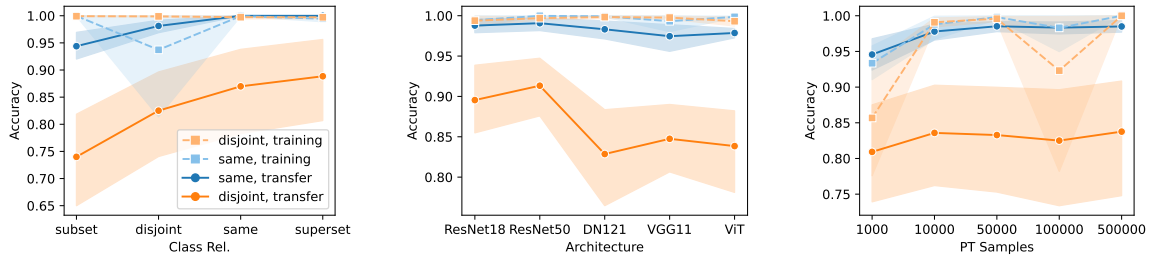
Effect of other factors: To compare the effect of invariance to that of the other factors, (*e.g.* the number of training samples) we train multiple such pairs of models g_s and g_d. Each pair is trained on datasets D_s and D_d that both use a different value of the factor that we want to compare to. Comparing the within-pair with the between-pair performance differences allows us to quantify how important invariance is compared to these other factors. Details on the training and evaluation procedure can be found in Appendix B.2.

- **Dataset size**: The size of the training dataset is typically considered an important factor in determining how useful representations are for downstream tasks. We compare its effect to invariance by training each pair of models with a different number n of training samples that are drawn from $\mathcal{D}_s$ and $\mathcal{D}_d$. We use $n \in \{1000, 10000, 50000, 100000, 500000\}$.

- **Model architecture**: The architecture and capacity of models is important for their performance, with larger models typically performing better. To compare the effect of architecture and model size, we train multiple pairs of models g_s, g_d, each with a different architecture. Here, we use ResNet-18, ResNet-50, Densenet-121, VGG-11 and Vision Transformer (ViT). For more details, see Appendix B.2.1.

- **Number and identity of classes**: In transfer learning, the objects in the training and target domain typically differ. To understand how much impact the difference of training and target objects has compared to invariance, we construct four pairs of training datasets whose objects O_t are related in different ways to the target objects O_e: a subset $O_{sub} \subset O_t$, $|O_{sub}| = \frac{1}{3}|O_t| = 10$, a disjoint set with the same cardinality $O_{disj} \cap O_t = \emptyset$ and $|O_{disj}| = |O_t| = 30$, the same $O_{same} = O_t$ and a superset $O_{sup} \supset O_t$ of O_t with $|O_{sup}| = 2 * |O_t| = 60$.

Validation on real-world data: The experiments on Transforms-2D data allow us to cleanly disentangle the effect of invariance from the other factors, but come with the risk of being limited to synthetic data. To test whether our observations hold up under conditions closer to real-world settings, we perform similar experiments as those on Transforms-2D on the CIFAR-10 and CIFAR-100 datasets. We cannot control transformations directly in these datasets, but we can approximate the setup described above by applying the transformations from Transforms-2D as data augmentations. As before, we pretrain models on two versions of the CIFAR datasets,



(a) Invariance vs class relationship.

(b) Invariance vs architecture.

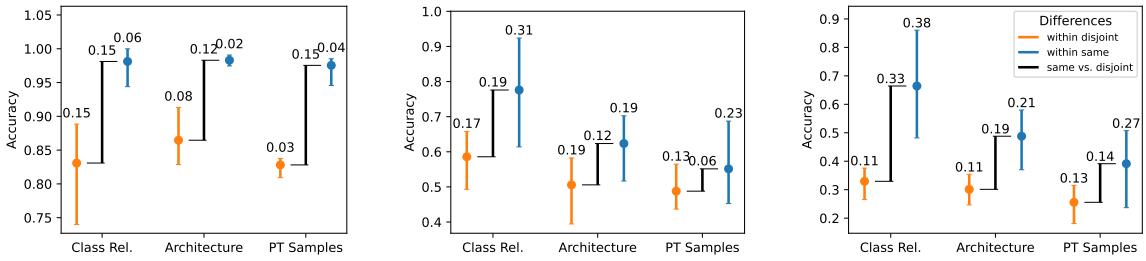
(c) Invariance vs number of training samples.

Figure 4.2: **[Impact of invariance vs other factors on transfer performance in Transforms-2D.]** Training (dotted lines) and transfer performance (solid lines) for models trained with different factors of variation and different invariances on the Transforms-2D dataset. Models trained to be invariant to the same transformations as the target tasks (blue) transfer significantly better than models trained to be invariant to different transformations (orange). This effect is very strong compared to the effect of other factors, such as the number of training samples, the model architecture or the relationship between the training and target classes. The reported numbers are aggregated over 10 runs.

one with the same transformations/data augmentations as the target task and the other with a disjoint set of transformations. We use a subset of 5 classes for CIFAR-10 and 50 classes for CIFAR-100 for pretraining and study transfer performance on the other half of the classes. Again, we compare the effect of using the same vs a disjoint set of invariances as the target task to the effect of varying the number of training samples, the model architecture and the class relationship.
Results: Figure 4.2 compares the effect of invariance on transfer accuracy with that of the other factors (number of training samples, architecture, class relationship), on the Transforms-2D dataset. The accuracy of all models on their training tasks is very similar. However, when transferred to the target task, the models that were trained with the same transformations as the target task (*i.e.* to have the invariances required by the target task) outperform the models that were trained with a disjoint set of transformations by a significant margin in all cases. We show analogous results for the CIFAR-10 and CIFAR-100 datasets in Appendix B.3.1 in Figures B.3 and B.4.

Figure 4.3 quantifies the difference in transfer performance caused by invariance and compares it to the differences caused by the other factors for the Transforms-2D, as well as the CIFAR-10 and CIFAR-100 datasets. For Transforms-2D, the difference attributable to invariance is larger than that attributable to all other factors, with the exception of the different-transformation model g_d for class-relationship, where it is comparable. For CIFAR-10 and CIFAR-100, the difference in transfer performance due to invariance is comparable to the difference due to class relationship and architecture. It is similar or lower than the difference due to the number of training samples, but still substantial.

In the latter case of sample counts, it is worth noting that in the CIFAR datasets, classes consist of a diverse set of images that can be seen as different transformations of the same object (*e.g.* cars or birds that differ in their model or species, color, camera angle, background, etc.). With more samples per class, the models likely see a more diverse set of transformations of the same object and thus might be able to become more invariant to irrelevant differences in object appearance. Therefore, comparing invariance due to data augmentations with the number of training samples, might implicitly compare explicit and implicit representational invariance. This



(a) Transforms-2D (b) CIFAR-10 (c) CIFAR-100

Figure 4.3: **[Difference in transfer performance due to invariance vs other factors.]** We compare the differences in transfer performance caused by representational invariance with the differences caused by changes to other factors on the Transforms-2D and the CIFAR-10 and CIFAR-100 datasets (with data augmentations). Orange (blue) bars show the span of transfer performance for different-transformation models g_d (same-transformation models g_s), for each comparison factor (class relationship, architecture, number of samples). Black bars show the difference between transfer performance means across factor values for g_d and g_s, *i.e.* the difference in performance caused by having the same vs different invariances as the target task. Across all datasets, the difference in transfer performance due to representational invariance is comparable and often larger than the difference due to varying the other factors.

relationship highlights the difficulties in studying the effect of invariance in uncontrolled real-world settings.

In Appendix B.3.2 we investigate how full-model fine-tuning affects the invariance of representations, and whether fine-tuning the whole model can change the invariances learned during pretraining. We find that with a small amount of fine-tuning data, representations that were pretrained with the right invariances still outperform ones that were pretrained with different invariances, but that the difference decreases the more data models are fine-tuned on.

Taken together, when transferring representations between tasks, our results show that invariance is as important or more important than other commonly studied factors in most cases. Our findings have important implications for model pretraining and pretraining dataset selection, domains that are especially relevant in an era where the usage of pretrained foundation models is becoming very prevalent [Bommasani et al., 2021]. Practitioners should pay special attention to the variations present in pretraining data, as well as to data augmentations, since both can strongly influence representational invariance and thus downstream performance. For instance, the high importance of invariance compared to the number and type of classes in the pretraining data shown in Figure 4.2a and Figure 4.3 means that when creating pretraining datasets, it may be beneficial to allocate more effort towards obtaining a larger diversity of object transformations compared to sampling more objects.

4.2.2 Can Invariance be Exploited to Harm Transfer Performance?

The previous results highlight that having the right representational invariances can significantly benefit transfer performance. On the flip-side, they also show that the wrong invariances harm transfer performance. Prior work has demonstrated similar detrimental effects for certain types of excessive invariance [Jacobsen et al., 2018]. An interesting question therefore is: Could an adversary exploit invariance to harm transfer performance on specific downstream tasks?

To answer this question, we combine our Transforms-2D data with the CIFAR-10 dataset [Krizhevsky et al., 2009], such that either the information in the Transforms-2D data or in CIFAR-10 is irrele-

vant for the target task. Concretely, we augment the CIFAR-10 dataset by pasting small versions of the Transforms-2D objects described in section 4.1.3 onto the images in a completely random manner, *i.e.* uncorrelated with the CIFAR-10 labels.

Notation. We denote by X the features available in the input and by Y the category of labels that the model is trained to predict. C stands for CIFAR-10 and O for objects from Transforms-2D. $X = C$ means that the input data is CIFAR backgrounds, and $Y = C$ means that the task is to classify images based on the CIFAR classes. $X = C + O, Y = C$ means that the inputs are CIFAR backgrounds with objects pasted on them, with the task of classifying the inputs based on their CIFAR classes.

In total, we use four datasets which differ in their combinations of features X and labels Y: the standard CIFAR-10 dataset ($X = C, Y = C$) the CIFAR-10 dataset with random objects pasted on it with the task of predicting CIFAR-10 classes ($X = C + O, Y = C$), the same augmented CIFAR-10 dataset with the task of predicting the category of the pasted objects ($X = C + O, Y = O$) and a dataset with only objects pasted on a black background, with the task of predicting the object category ($X = O, Y = O$). We use 10 object prototypes that are scaled down and pasted at a random position onto the CIFAR-10 images. Example images from these datasets can be found in Appendix B.1.2. We train models on each of the datasets, freeze their representations, and then fine-tune and evaluate each model's last layer on each of the other datasets. We use X_p, Y_p to refer to a model's pretraining dataset and objective and X_t, Y_t to refer to the dataset that its representations are transferred to.

			Accuracy				Sensitivity	
	X		C	C + O	C + O	O	C + O	C + O
		Y	C	C	O	O	C	O
Pre-training Dataset	C	C	$0.98_{\pm 0.00}$	$0.89_{\pm 0.01}$	$0.56_{\pm 0.06}$	$1.00_{\pm 0.00}$	$0.99_{\pm 0.00}$	$0.21_{\pm 0.02}$
	C + O	C	$0.98_{\pm 0.00}$	$0.97_{\pm 0.00}$	$0.15_{\pm 0.01}$	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$	$0.08_{\pm 0.01}$
	C + O	O	$0.26_{\pm 0.02}$	$0.13_{\pm 0.02}$	$1.00_{\pm 0.00}$	$0.98_{\pm 0.02}$	$0.13_{\pm 0.02}$	$0.99_{\pm 0.01}$
	O	O	$0.29_{\pm 0.03}$	$0.28_{\pm 0.03}$	$0.25_{\pm 0.04}$	$1.00_{\pm 0.00}$	$1.00_{\pm 0.01}$	$0.11_{\pm 0.04}$

Table 4.1: **[The impact of irrelevant features on downstream performance and invariance.]** Rows show pre-training tasks, with X (*i.e.* X_p) denoting the features available in the training dataset and Y (*i.e.* Y_p) the category of labels that the model was trained to predict. The accuracy columns denote the transfer accuracy after fine-tuning on the respective dataset (X_t) at predicting the respective label (Y_t). The sensitivity values show how sensitive resp. invariant the models' representations are according to the *sens*-metric defined in Section 4.1.4 on the $C + O$ data when changing the CIFAR background images ($Y_t = C$) while keeping the pasted foreground objects fixed, and while changing the pasted object ($Y_t = O$) while keeping the CIFAR background fixed. Lower values mean higher invariance. **Observations:** The representations of models pretrained with access to features that are irrelevant for their target task (objects O for $X_p = C + O, Y_p = C$ and CIFAR backgrounds C for $X_p = C + O, Y_p = O$) transfer worse to tasks where those features are important than their counterparts that did not have access to those features, *i.e.* $X_p = C, Y_p = C$ and $X_p = O, Y_p = O$. The sensitivity scores show that the difference in performance is due to representations becoming invariant to features that are not relevant to the pre-training objective, *i.e.* low sensitivity resp. high invariance for $X_p = C + O$ models towards the category $Y_t \neq Y_p$ they were not trained on, compared to high sensitivity towards the category $Y_t = Y_p$ they were trained on. Note that all models achieve $\sim 100\%$ accuracy on the $X_t = O, Y_t = O$ task, since they just have to separate 10 distinct object images.

Results: Table 4.1 shows the transfer accuracies of each model on each of the datasets. The main takeaway is that the transfer performance of the models differs significantly, depending on

Model Type	Transformation Relationship	Synthetic Datasets			Real-World Datasets	
		In-Distribution	Mild OOD	Strong OOD	CIFAR-10	CIFAR-100
Image	Same	$\mathbf{0.14_{\pm 0.07}}$	$\mathbf{0.15_{\pm 0.07}}$	$\mathbf{0.57_{\pm 0.16}}$	$\mathbf{0.37_{\pm 0.21}}$	$\mathbf{0.35_{\pm 0.18}}$
	Other	$0.40_{\pm 0.15}$	$0.39_{\pm 0.14}$	$0.69_{\pm 0.15}$	$0.50_{\pm 0.20}$	$0.49_{\pm 0.18}$
	None	$0.39_{\pm 0.15}$	$0.42_{\pm 0.15}$	$0.68_{\pm 0.16}$	$0.44_{\pm 0.21}$	$0.45_{\pm 0.19}$
Random	Same	$\mathbf{0.12_{\pm 0.08}}$	$\mathbf{0.44_{\pm 0.16}}$	$\mathbf{0.24_{\pm 0.10}}$	$\mathbf{0.39_{\pm 0.22}}$	$\mathbf{0.36_{\pm 0.20}}$
	Other	$0.64_{\pm 0.16}$	$0.68_{\pm 0.16}$	$0.33_{\pm 0.11}$	$0.46_{\pm 0.21}$	$0.45_{\pm 0.20}$
	None	$0.65_{\pm 0.18}$	$0.69_{\pm 0.17}$	$0.35_{\pm 0.12}$	$0.50_{\pm 0.20}$	$0.49_{\pm 0.19}$

Table 4.2: **[Invariance transfer under distribution shift.]** Each cell shows the average *sens*-score (lower is more invariant), for models trained on image and random data, under distribution shift. "Transformation Relationship" refers to the relationship between the training and test transformations of the models. Rows labeled as "Same" show how invariant models are to their training transformations on each of the datasets (*e.g.* a translation-trained model evaluated on translated data), whereas rows labeled as "Other" show how invariant they are on average to transformations other than the ones they were trained on (*e.g.* a translation-trained model on rotations). "None" rows show the baseline invariance of models trained without transformations, *i.e.* simply to classify untransformed objects. The most invariant models in each category are shown in bold. Models are significantly more invariant to the transformations they were trained on than to other transformations, and this relationship persists on mild and strong OOD, as well as real-world data, indicating that a medium to high degree of representational invariance is preserved under distribution shifts.

what information they have access to during pretraining, *i.e.* what features were *available*, and how *relevant* they are for the pretraining task. The impact that the relevance of input information has on transfer performance can be seen by comparing the two models trained on the augmented CIFAR images ($X_p = C + O$). The model pretrained to predict CIFAR labels ($Y_p = C$) performs well on this task ($Y_t = Y_p = C$), but poorly at predicting objects ($Y_t = O$), whereas the inverse relationship holds for its counterpart pretrained to predict objects ($Y_p = O$). The sensitivity scores (based on the *sens*-metric) show that that the representations of both models during pretraining become invariant to the irrelevant information in their inputs (objects or CIFAR backgrounds, respectively) and thus their representations cannot be used anymore to predict the corresponding classes ($Y_t \neq Y_p$). Additionally, models that had access to irrelevant features during pretraining ($X_p = C + O$) achieve lower transfer performance at predicting the category of classes they were not pretrained for ($Y_p \neq Y_t$) than models pretrained to predict the same category Y_p, but without access to the irrelevant features ($X_p = C$ and $X_p = O$).

The results show that during pretraining, the representations of models become invariant to information that is available in the input but irrelevant for the pretraining task. This effect can be exploited to harm transfer performance by introducing invariances to features that are relevant for downstream tasks into the representations. We further examine the relationship of relevance and availability with transfer performance in Appendix B.3.3. We find that more relevance gradually leads to less invariant representations, but that even if irrelevant objects are only available in a few inputs, models already become significantly more invariant to them.

In summary, our results show that invariance significantly affects how well representations transfer to downstream tasks. Its effect on transfer performance can be both positive, when representations share the right invariances with the target task, and negative, when they lack the right invariances or — even worse — possess invariance towards important information in the inputs.

4.3 How Transferable is Invariance?

So far, we have shown that sharing invariances between training and target tasks is quite important for the transfer performance of representations. But why does invariance have such an outsized influence? Our hypothesis is that invariances transfer well under distribution shift. If invariances learned on a pretraining task are largely preserved in different domains, that would make them more robust than specific features that might change from domain to domain, and it might help to more robustly detect features that are present across domains.

4.3.1 Invariance Transfer Under Distribution Shift

To test how well invariance in pretrained representations transfers under distribution shift, we train models to be invariant to specific transformations using the Transforms-2D dataset and evaluate their invariance on out of distribution tasks. We create two categories of out of distribution (OOD) datasets: a mild OOD category with only small differences from the training dataset and a strong OOD category that is very different. Concretely, we create the mild OOD datasets by sampling a different set of image objects than the ones used for training. For the strong OOD datasets we use structured random data, by sampling a specific random pattern with pixel values distributed uniformly at random for each class and then pasting it on random background patterns, also sampled uniformly at random. Note that we can apply transformations to the random objects in the same way as to the image objects in Transforms-2D. Using this setup we train a different model for each of the 18 transformations in Transforms-2D, *i.e.* such that each of the models is invariant to a different transformation, and evaluate its invariance on each of the transformations, for each dataset category. We use ResNet-18 models here but report similar results for other architectures in Appendix B.4.2. To measure the invariance resp. sensitivity of a model's representation to a particular transformation, we use the *sens*-metric defined in Section 4.1.4.

We also study the reverse direction of invariance transfer, by training models on the random strong OOD data described above, and evaluating their performance on OOD data analogously but in reverse, *i.e.* mild OOD data uses a different set of random objects and strong OOD data is the Transforms-2D data for these models. In addition to the synthetic datasets, we also evaluate how well invariance transfers to real-world datasets, *i.e.* to CIFAR-10 and CIFAR-100, by using the Transforms-2D transformations as data augmentations. Additional details can be found in Appendix B.4.1.

Results: Table 4.2 shows the invariance of the two types of models on each of the dataset categories. We see that the invariance of models to the transformations that they were trained on is consistently higher than to other transformations on all datasets. Models trained to be invariant to the target transformations are also consistently more invariant than the baseline models trained without transformations across all the datasets. This shows that models do, in fact, acquire a significant degree of invariance to their training transformations, and are subsequently able to transfer it to other distributions. However, it seems to be more difficult to transfer invariance to random data (strong OOD for the image model and mild OOD for the random model) than to image data. It is also interesting to note that the image and random models achieve very similar degrees of invariance on the real-world datasets. This suggests that even training on structured random data can be a good prior for learning invariant representations, provided that the necessary transformations can be expressed on this type of data. Additional results and breakdowns of invariance over individual transformations can be found in Appendix B.4.2.

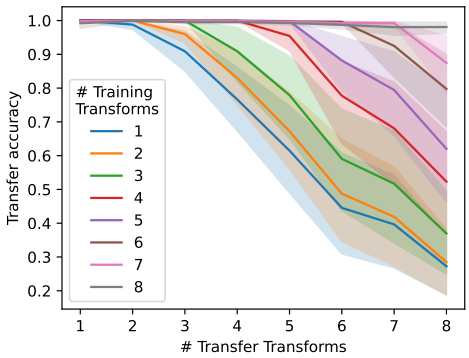


Figure 4.4: **[ResNet-18 models trained on nested sets of transformations and evaluated on datasets with super- and subsets of those transformations.]** Models trained on data with the same set or a superset of transformations as the target dataset consistently achieve almost 100% accuracy. However, models trained with only a subset of the transformations show considerably lower performance that decreases the smaller the subset of training transformations is compared to the target task. The results show that learning a superset of required invariances does not harm transfer performance but that missing required invariances degrades transfer performance.

4.3.2 Invariance Mismatch between Training and Target Tasks

A different type of distribution shift happens when there is a mismatch in the transformations present in training compared to the target task, which we suspect often happens in real-world settings. To better understand these cases, we investigate the effect of training models on sub- and supersets of the transformations required by the target task. We create nested sets of transformations $T_i, i \in \{1, \ldots, 8\}$, such that $T_i \subset T_j$ for $i < j$ and $|T_i| = i$. For each T_i, we train a model (as described in Section 4.2.1) and then evaluate its transfer performance on data for each of the T_i's.

Results in Figure 4.4 show that being invariant to more than the necessary transformations does not negatively impact performance (as long as those invariances do not conflict with the target task as in Section 4.2.2). However, if any required invariances are missing in the representations, a models' performance quickly decreases. This can help to explain why pretraining is often successful in practice: the invariances learned during pretraining do not need to perfectly match those required by the target task, as long as they cover a superset of required invariances. We hypothesize that commonly used pretraining datasets (such as ImageNet), together with data augmentations, induce a large set of broadly useful invariances. We show results for ResNet-18 models here and report very similar results for other architectures in Appendix B.4.3.

4.4 Related Work

Transfer learning is a well studied topic in the machine learning community. Prior work has identified a number of factors that contribute to transfer performance, such as model size [Abnar et al., 2021, Kolesnikov et al., 2020], size and characteristics of the pretraining dataset [Azizpour et al., 2015, Entezari et al., 2023, Huh et al., 2016, Kornblith et al., 2019, Neyshabur et al., 2020] and adversarial robustness [Salman et al., 2020]. In this work, we investigate the impact of invariance on transfer performance more broadly as another dimension and compare its effect to the aforementioned factors. In this context, prior work has found that DNNs can have difficulties

generalizing certain invariances [Azulay and Weiss, 2018, Zhou et al., 2022]. Our work also aims to understand this phenomenon better by investigating conditions under which invariance transfers more closely.

Invariance and equivariance have been studied in the context of representation learning with the goals of better understanding their properties [Bloem-Reddy and Teh, 2020, Cohen et al., 2019, Kondor and Trivedi, 2018, Lyle et al., 2020, Von Kügelgen et al., 2021], measuring them [Fawzi and Frossard, 2015, Goodfellow et al., 2009, Gopinath et al., 2019, Kvinge et al., 2022, Nanda et al., 2022], leveraging them for contrastive learning [Ericsson et al., 2021a, Wang and Isola, 2020] and building in- and equivariant models [Benton et al., 2020, Cohen and Welling, 2016, Weiler and Cesa, 2019, Zhang, 2019]. Our work is also attempting to understand the implications and benefits of invariant models better. However, most prior work on understanding invariance analyzes how invariance benefits a particular task or is focussed on a specific domain [Ai et al., 2023], whereas we are interested in understanding the relationship of invariance between different tasks more broadly. Additionally, we complement the theoretical perspective that many prior works are offering with an empirical analysis.

Data augmentations have been studied as a tool to improve model performance [Cubuk et al., 2018, Perez and Wang, 2017, Shorten and Khoshgoftaar, 2019], and to imbue models with specific invariances [Ratner et al., 2017]. Their effects have also been thoroughly investigated [Balestriero et al., 2022, Chen et al., 2020a, Geiping et al., 2022, Huang et al., 2022, Perez and Wang, 2017]. In our work we leverage data augmentations to both train models with specific invariances as well as to evaluate the degree and effect of invariance in their representations.

In **self-supervised learning (SSL)**, models are trained to be invariant to certain data augmentations in order to obtain pseudo labels [Doersch et al., 2015, Zhang et al., 2016] or to introduce contrast between similar and dissimilar inputs [Chen et al., 2020b, Ericsson et al., 2021b, Grill et al., 2020, Kim et al., 2020]. However, invariance is often treated in an ad hoc and opportunistic manner, *i.e.* data transformations are selected based on how much they boost the validation performance of models. Our findings complement work that uses invariance as a building block, by assessing the importance of invariance to data transformations, relative to other factors such as the model architecture.

The **robustness** literature has also studied invariance extensively in order to safeguard model performance against adversarial perturbations [Papernot et al., 2016, Szegedy et al., 2013], natural image corruptions [Geirhos et al., 2018b, Hendrycks and Dietterich, 2019, Taori et al., 2020] or distribution shift [Recht et al., 2019]. This line of work is similar in spirit to ours, as it also shows that invariance, or a lack thereof, can have a significant impact on model performance. However, robustness research is primarily interested in avoiding performance degradations when specific transformations are introduced to a dataset, rather than understanding how the ability to transfer representations between datasets depends on their invariance to transformations. Some prior works have found that too much invariance can be detrimental to the robustness of models [Jacobsen et al., 2018, Kamath et al., 2019, Singla et al., 2021]. We also investigate this phenomenon and expose conditions under which representational invariance can harm transfer performance. Additionally, the field of invariant risk minimization has investigated robustness to changes in spurious correlations between different domains [Arjovsky et al., 2019, Muandet et al., 2013].

Synthetic datasets. There have been proposals for fully synthetic datasets that allow for a more careful study of the properties of representations [Hermann and Lampinen, 2020, Matthey et al., 2017]. Our work leverages previous work by Djolonga et al. [2021] and uses it to construct a dataset that allows for precise control over variations in the data.

4.5 Conclusion

We study the importance of invariance in transfer learning by using a family of synthetic datasets, Transforms-2D, that allows us to precisely control differences between input points. By leveraging this method, we are able to show that invariance is a crucial factor in transfer learning and often as or more important than other factors such as the number of training samples, the model architecture and the class relationship of the training and target task. Sharing the right invariances with the target task positively impacts transfer performance, while a lack of the right invariance or even invariance towards important input features harms transfer performance. We further investigate the transferability of invariance under distribution shift and find that in most cases, models can transfer a high degree of invariance to new settings. Overall, our findings show that for transfer learning to be successful, the training and target tasks need to share important invariances.

Limitations. Since achieving precise control over data transformations is not possible with real-world data, our experiments heavily rely on synthetic data (see the discussion in Section 4.1.3). However, results derived from synthetic data might not generalize exactly to practical settings. To address this limitation we include validation experiments on real-world datasets in Sections 4.2.1 and 4.3.1 that show that the observations made using synthetic data can be largely extrapolated to more realistic settings as well.

Chapter 5

Understanding Memorization in Large Language Models with Random Strings

Understanding whether and to what extent large language models (LLMs) have memorized training data has important implications for the reliability of their output and the privacy of their training data. However, understand memorization is challenging with typically used real-world datasets, since disentangling whether a model is predicting a string correctly because it has memorized it, or because it is generalizing from other similar strings is often not possible.

In order to cleanly measure and disentangle memorization from other phenomena (*e.g.* in-context learning), we create an experimental framework that is based on *repeatedly exposing LLMs to random strings*. Our framework allows us to better understand the *dynamics*, *i.e.*, the behavior of the model, when repeatedly exposing it to random strings. Using our framework, we make several striking observations: a) we find consistent phases of the dynamics across families of models (Pythia, Phi and Llama2), b) we identify factors that make some strings easier to memorize than others, and c) we identify the role of local prefixes and global context in memorization. We also d) show that sequential exposition to different random strings has a significant effect on memorization.

Publication status: The work in this chapter is complete and under submission, as of the time of submitting this thesis.

5.1 Preliminaries and Experimental Setup

Approach: Throughout, we use random strings in order to train and test LLMs. To create a random string, we first choose an *alphabet* A that consists of $|A| = \ell$ unique tokens; we call ℓ the *size of the alphabet*. The alphabet we use for string generation is a subset of a much larger *vocabulary of all tokens* V, $A \subset V$. Tokens in an LLM's vocabulary can range from single characters to entire words and its size spans from tens of thousands to a few hundred thousand tokens. In most experiments, we use tokens corresponding to lowercase characters in the Latin alphabet.

We use P_A to denote a probability distribution over the unique tokens of the alphabet A. We can compute the *entropy* of P_A using the standard definition of entropy: $H(P_A) = -\sum_{a \in A} P_A(a) \log P_A(a)$. For any given alphabet of size ℓ we use H_ℓ to denote the entropy of the uniform probability distribution over the alphabet's tokens. We generate a random string $s = (s_1, \ldots, s_n)$ of length n by sampling every token s_i independently from P_A. Unless otherwise noted, we assume that P_A is the uniform probability distribution over the tokens in A. Given a

string s of length n we use $s_{[i,j]}$, with $i \leq j$, to denote the substring of s that consists of positions $(s_i, s_{i+1}, \ldots, s_{j-1}, s_j)$.

Given a string s, we train a causal, *i.e. autoregressive*, language model $\mathcal{M}$ to memorize s. We do so, by minimizing the cross-entropy loss over all positions of s. We denote by $P_{\mathcal{M}}(s_i = t|s_{[1,i-1]})$ the probability that $\mathcal{M}$ assigns to token t at the i-th position of string s. We define the prediction of $\mathcal{M}$ for position i as the token with the largest $P_{\mathcal{M}}(s_i = t \mid s_{[1,i-1]})$). Then, we define the *recollection accuracy* of $\mathcal{M}$ with respect to s based on greedy decoding as:

$$\text{Accuracy}(\mathcal{M}, s) = \frac{1}{n} \sum_{i=1}^{n} \delta(s_i = \arg\max_{t \in V} P_{\mathcal{M}}(s_i = t|s_{[1,i-1]})). \tag{5.1}$$

δ(condition) is an indicator variable; it takes value 1 (resp. 0) when the condition is true (resp. false). We report accuracy values in the main chapter, and loss values in the appendix.

Data generation process: In our experiments, we focus on alphabets A with $\ell \in \{2, 4, 7, 13, 26\}$. We primarily use tokens corresponding to the first ℓ lowercase characters from the Latin alphabet, *i.e.* $A \subseteq \{a, \ldots, z\}$, but also report results for non-Latin alphabets in Appendix C.2.3. We therefore often refer to the elements of the alphabet as characters, even though they are technically tokens. We generate random strings of lengths $n \in \{16, 32, 64, 128, \ldots, 1024\}$ by sampling tokens uniformly at random from A. All our results are aggregates over 5 runs with different random string samples; we highlight one standard deviation in the plots. We show examples of random strings in Appendix C.1.2. We also show that our observations on random string memorization are robust when random strings appear in the context of natural language data from the wikitext Merity et al. [2016] dataset in Section5.3 and Appendix C.4.

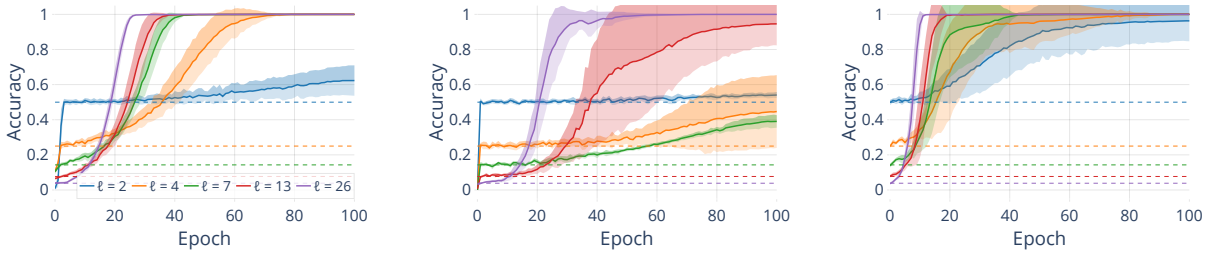
LLM models: We make use of the Pythia [Biderman et al., 2023b], Phi [Li et al., 2023b] and Llama-2 model [Touvron et al., 2023] families. For the Pythia family, we use variants with 70M, 1B and 12B parameters, for the Phi family we use 1.3B and 2.7B parameter variants, and for Llama-2 we use 7B and 13B parameter variants. We refer to each model by its parameter count, *e.g.*, Pythia-1B or Llama2-13B. We choose these models, since they represent popular, modern architectures, and span a wide spectrum of parameter counts (more than two orders of magnitude). We also report results for older GPT-2 and OPT models in Appendix C.5.4.

Training and fine-tuning: We experimented with both untrained and pretrained models to capture the memorization dynamics early during training and later during continual training or fine-tuning of a pretrained model. All our findings largely hold in both cases, with the primary difference being pretrained models memorize faster. We report the results for both pretrained models (in the main chapter) and untrained models (in the appendix). A detailed description of the training setup can be found in Appendix C.1.1.

5.2 The Dynamics of Repeated Exposure to Random Strings

We start by providing some basic observations we make as we repeatedly expose models to the same random string for 100 epochs. Figure 5.1 shows the recollection accuracy of three models for random strings with $\ell = 2, 4, 7, 13, 26$. The dotted lines in the two plots show the accuracy of a model that performs random guessing over alphabet A.

Two phases: At a high level, Figure 5.1 shows that for all models $\mathcal{M}$ the accuracy converges towards 1 as training progresses. The convergence is not uniform, however, since after an initial rise, the accuracy reaches a plateau at the random guessing baseline (*e.g.*for Pythia-1B and Phi-2.7B after about 8 epochs), before increasing towards towards 1 more slowly afterwards. We observe this pattern consistently across all ℓ and $\mathcal{M}$. As models are repeatedly exposed to the same random string, they go through two phases: During the first phase, the accuracy of the



(a) Pythia-1B (b) Phi-2.7B (c) Llama2-13B

Figure 5.1: **[Recollection accuracy for different alphabet sizes ℓ and models $\mathcal{M}$. ($n = 1024$)]** For all models, the accuracy initially increases quickly before stagnating at the random guess level during the *Guessing-Phase*. Afterwards, the accuracy converges more slowly towards 1 during the *Memorisation-Phase*. The accuracy of randomly guessing tokens from A is shown with dashed lines.

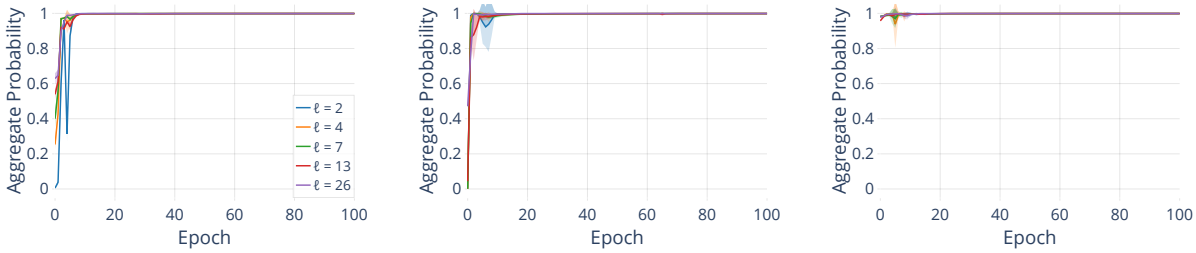
model reaches that of a random guess and then plateaus; thus, we call this the *Guessing-Phase*. During the second phase, the accuracy of the model exceeds the guessing plateau and converges to 1. We call this phase the *Memorisation-Phase*.

Understanding the transition into the *Memorisation-Phase*: We now take a closer look at the token-level probability distributions produced by the models. Figure 5.2, upper row, shows the aggregate probability mass that models assign to the tokens inside the alphabet A, i.e., $\sum_{t \in A} P_{\mathcal{M}}(s_i = t \mid s_{[1,i-1]})$, averaged over all positions i. Analogously to the quick initial increase in accuracy (Figure 5.1), we see that models quickly learn to assign all the probability mass to tokens within the alphabet A and separate those from the whole vocabulary of tokens V.

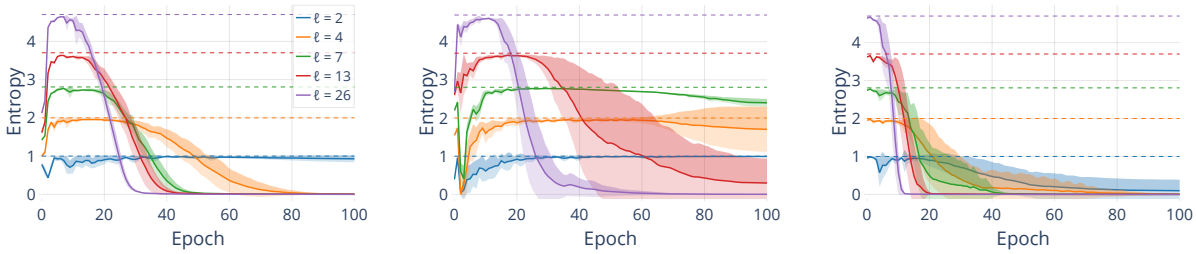
We also compute the token-level entropy over the course of training; i.e., the entropy of $P_{\mathcal{M}}(s_i = t \mid s_{[1,i-1]})$ for $t \in A$, averaged over i. The results in Figure 5.2, lower row, show a sharp increase in entropy to the maximum possible value in the initial stages of training, that coincides with the rise in aggregate probability mass and the initial rise of the accuracy curves. After the entropy peaks (at around epoch 8 for Pythia-1B and Phi-2.7B), it drops to 0, matching the second increase of the accuracy values.

Thus, in the initial *Guessing-Phase*, the models are learning which tokens are in A and separate those from V (rise in aggregate probability). In that phase, they do not know *which specific tokens* to predict at each position in the string and thus *guess* tokens from A randomly (high entropy). In the subsequent *Memorisation-Phase*, the models actually start to memorize the specific tokens at each position (decrease in entropy) and become more accurate. In Appendix C.2.1 (Figures C.1, C.3, C.5, and C.7) we show that other models exhibit the same two-phase dynamics, and in Figures C.9 and C.10 that during the *Guessing-Phase*, $\mathcal{M}$ actually approximates the distribution over A (KL-Divergence).

It is worth noting that Llama2-13B assigns a total probability mass of 1 to the tokens in A almost immediately (Figure 5.2c) and that its accuracy exhibits only a single ascend phase (Figure 5.1c). Furthermore, its initial accuracy values before any training match the random guess baseline that the other models only reach after completing the *Guessing-Phase*. We investigate this phenomenon further in Appendix C.2.6 and show in Figure C.20 that — in contrast to the other models — Llama2 models exhibit strong in-context learning abilities that enable them to infer the alphabet distribution P_A after about only 100 tokens of context. These models effectively shorten the *Guessing-Phase* to zero. However, when using non-Latin alphabets (see



(a) Pythia-1B, Aggregate Prob. (b) Phi-2.7B, Aggregate Prob. (c) Llama2-13B, Aggregate Prob.



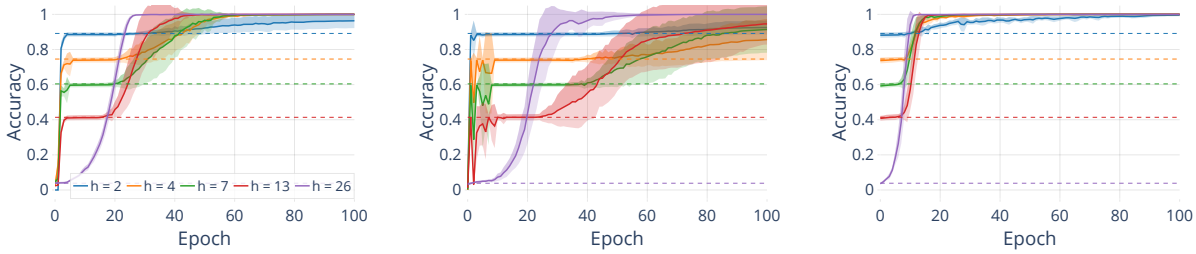
(d) Pythia-1B, Entropy (e) Phi-2.7B, Entropy (f) Llama2-13B, Entropy

Figure 5.2: **[Aggregate probability mass and entropy for different ℓ. ($n = 1024$)]** i) Plots on the top show the probability mass that $\mathcal{M}$ assigns to tokens in A. In all cases, models quickly learn to allocate the maximum possible probability mass to the tokens within the alphabet A, *i.e.* they only predict tokens from A after a few training epochs. ii) We show the average entropy of the probability distribution of model $\mathcal{M}$ over A. The entropy initially rises to its maximum value, before decreasing to 0. The maximum attainable entropy (for different ℓ) is shown with dashed lines.

Appendix C.2.2, Figure C.11), Llama2-13B also exhibits a *Guessing-Phase*, indicating that its ability to learn the distribution from the context is limited.

Memorization order: We also analyse the order in which models memorize the tokens in the string. In Appendix C.3, we show that tokens are memorized in random order, *i.e.* that there is no connection between the position of a token in the string, and the epoch at which it is memorized. This observation is based on Spearman rank correlation values between the position of a token in the string and the epoch at which it is memorized, which are between -0.1 and 0.1 in most cases, with moderate position correlation only for some untrained models on higher entropy strings.

Implications for studying and quantifying memorization: Our discovery of the two phases in memorization dynamics underscores the importance of our experimental setup, which enables us to distinguish between token recollection due to in-context learning (*Guessing-Phase*) and memorization (*Memorisation-Phase*). Our findings also call for fundamentally rethinking existing approaches to quantify memorization [Carlini et al., 2022]. On one hand, current measures risk *overestimating* the degree of memorization by not discounting for token recollection due to in-context learning (guessing) – see Figures C.17 and C.18 in Appendix C.2.4. On the other hand, the measures risk *underestimating* the degree of memorization by focusing on the recollection of contiguous token sequences, while tokens are memorized in random order – see Figure C.40 in Appendix C.7. To be robust, future measures of memorization need to account for these dynamics.



(a) Pythia-1B (b) Phi-2.7B (c) Llama2-13B

Figure 5.3: [**Recollection accuracy for different entropy levels** h**.** ($n = 1024$)] Analogously to strings with different ℓ, strings with lower h are easier to guess, but harder to memorize. Dashed lines indicate the performance of a random guess, equivalent to always guessing "a".

5.3 Q1: Are Some Strings Easier to Memorize than Others?

In the previous section, we saw that during repeated exposure to the same random string, models undergo two phases. Figure 5.1 shows that for the same model $\mathcal{M}$, the *Guessing-Phase* has the same length for different ℓ, but that the length of the *Memorisation-Phase* varies significantly, depending on ℓ. Even though strings with smaller ℓ end the *Guessing-Phase* with higher accuracy values, they take longer to memorize in the subsequent *Memorisation-Phase*, and are eventually "overtaken" by strings with larger ℓ. For all models, the strings with $\ell = 26$ and $\ell = 13$ are memorized first, *i.e.* reach accuracy 1, whereas the $\ell = 2$ and $\ell = 4$ strings are memorized last (for Llama2-13b) or not at all within 100 epochs (Pythia-1B, Phi-2.7B). The question therefore is: What property makes the random strings harder or easier to memorize?

Effect of entropy on memorization: The strings we use in Section 5.2 have different alphabet sizes $\ell \in \{2, 4, 7, 13, 26\}$, but they also have different levels of entropy H_ℓ, with $H_\ell < H_{\ell'}$ for $\ell < \ell'$. To test whether ℓ or H_ℓ is responsible for the differences in memorization dynamics, we create strings with the same ℓ, but with different entropy levels. Specifically, we create random strings with an $\ell = 26$-letter alphabet A with uniform P_A, except for the first letter ("a"), which is oversampled to match the entropy of the previous strings (*i.e.*, $H_2, H_4, H_7, H_{13}, H_{26}$)[1].

Accuracy curves for repeated exposure to such strings for different models are shown in Figure 5.3. We see strikingly similar patterns to those shown in Figure 5.1, with lower entropy strings achieving higher accuracy during the *Guessing-Phase*, but subsequently being memorized more slowly than higher entropy strings. These results indicate that it is the entropy of the probability distribution of the tokens that affects the memorability of random strings. In Appendix C.2.1 (Figures C.2, C.4, C.6 and C.8) we show that the same observations hold for additional models. For the rest of the chapter we will only consider strings with different ℓ and uniform P_A, keeping in mind that the H_ℓ of the strings is the important part.

Other factors affecting memorability

Model size: In addition to entropy, we find that the *size of the model* affects memorization, and show in Appendix C.2.1, Figures C.3 and C.4 that larger models within the same family tend to memorize strings faster. This observation is congruent with findings by other work on memorization Biderman et al. [2023a], Carlini et al. [2021, 2022].

String length and structure: Further, we find in Appendix C.2.4 that the length of the random string plays a role (Figures C.13, C.14), with longer strings being harder to memorize than shorter

[1]The more we oversample a letter in A, relative to the other letters, the lower the entropy of the string.

ones, but only if all tokens are sampled independently. If we create longer strings by repeating shorter random strings (Figures C.17, C.18), only the length of the unique base string matters, since the models can predict the remaining tokens without memorization from the context. Additionally, memorising a random string as one long piece or in multiple shorter partitions in the same batch does not affect memorization speed (Figures C.15, C.16), which indicates that a main factor for memorization difficulty — other than entropy — is the total number of independently sampled tokens in the string.

Conditional entropy: In Appendix C.2.5 we change the n-gram conditional entropy of strings, while keeping their conditional entropy fixed, and show that unconditional entropy affects the *Guessing-Phase*, but that it does not significantly impact the length of the *Memorisation-Phase*.

Alphabet type: In Appendix C.2.2 we conduct ablations with non-Latin alphabets (Figure C.11) and non-pretrained versions of the models shown earlier (Figure C.12), and observe the same memorization dynamics as before, although non-pretrained models memorize more slowly.

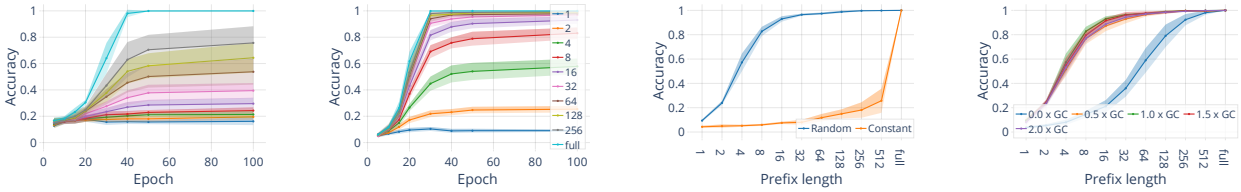
Real-world training setups: In practice, memorized strings appear embedded into a context of other data and not in isolation, *e.g.* an email address or phone number embedded inside other text. To validate whether memorization follows the same patterns when it happens in the context of other data, we train models to memorize random strings under conditions that closely resemble real-world settings. We present random strings to the model in the context of natural language data from wikitext Merity et al. [2016], in two different ways: 1) by presenting random strings to the model as elements inside larger batches of natural language strings, and 2) by embedding random strings inside longer natural language strings as substrings. We show in Appendix C.4 that – while random strings are memorized more slowly, the more natural language context there is – the same dynamics as for memorization in isolation hold, and that we thus can expect our findings to generalize to practical training scenarios.

Implications for risks and ways of memorising training data: Our findings demonstrate that *not all strings are equally memorable* – in fact, we establish that memorability of random strings is intrinsically related to their entropy. But, generalising our findings to natural language strings remains an open challenge as it is far from clear how one would estimate entropy of such strings. Intuitively, however, our findings imply that the less guessable a string is (higher entropy), the easier it is for an LLM to memorize it. Put differently, rather ironically, the strings (e.g., cryptographic secret keys) that are harder to guess and offer more security are the ones that are at greater risk of being memorized by LLMs. Finally, as the more guessable strings (lower entropy) are also more compressible, our findings rule out compression as a potential (latent) model for how strings are stored by LLMs during memorization. A detailed exploration of memorization techniques used by LLMs is beyond the scope of this work, but we take the first step towards exploring prefix associations used by LLMs to recollect tokens in the next Section.

5.4 Q2: What Information do Models Need to Recall Memorized Tokens?

We setup the following experiment: Given a token at position i in random string s sampled from P_A as described in Section 5.1, we split i's full prefix $s_{[1,i-1]}$ into two parts: a *local prefix* of size k, *i.e.* $s_{[i-k,i-1]}$, and the *global context*, given by $s_{[1,i-k-1]}$. In our analysis, we keep the local prefix fixed while modifying the global context. Our goal is to study the role that the prefix and the global context play in the recollection accuracy of the model.

The role of local prefixes: To determine how much information the model needs to recall tokens, we test the accuracy of the model for different prefix lengths. For this, we keep the local prefix fixed, while changing the global context using a `Random` replacement policy. `Random` replaces



(a) Prefix performance $\ell = 7$ (b) Prefix performance $\ell = 26$ (c) GC distribution (d) Amount of GC

Figure 5.4: **[Recollection accuracy for different prefix lengths and for changes in the global context (GC) during training. ($n = 1024, \mathcal{M} =$ Pythia-1B)]** (a) and (b) show what fraction of tokens can be recollected correctly with different prefix lengths, at different points during training. In many cases, prefixes much shorter than the full string are sufficient to predict most of the tokens accurately. (c) shows the performance of a randomly re-sampled vs a constant global context with only one repeated token, and (d) shows the impact of changing the size of the global context, where the numbers indicate multiples of the GC size.

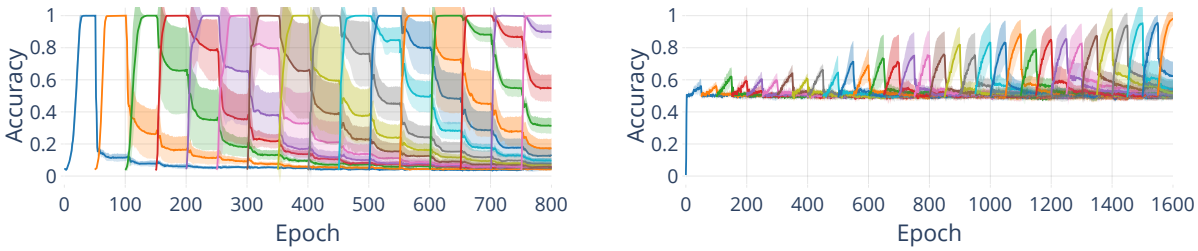
every token of the global context with one sampled randomly from P_A. For each position i, we sample 10 global context replacements and count i as predicted correctly if s_i is predicted most frequently by the model (using greedy decoding) among the samples.

We show the recollection accuracy for different prefix lengths $k = 1, 2, 4 \ldots, 1024$ over the first 100 epochs of training for $\ell = 7$ in Figure 5.4a and for $\ell = 26$ in Figure 5.4b. We observe that in the initial *Guessing-Phase* (at epoch 5 and 10), all prefix lengths achieve very low accuracy. Starting from epoch 15 (after transitioning into the *Memorisation-Phase*), the accuracy starts to increase substantially, even for short prefixes. Especially at epoch 30 and later, for the larger ℓ, the accuracy of short prefixes (less than $5-10\%$ of the total string length) is close 100%. As ℓ gets smaller, shorter prefixes perform worse, relative to the full prefix. Overall, small local prefixes — much shorter than the entire string — are very effective at correctly recalling tokens. Moreover, as memorization progresses, the same level of accuracy can be achieved with shorter prefixes. In Appendix C.5.2 we show results for additional models and values of ℓ. There, we observe that for untrained models, short prefixes are not sufficient for recalling tokens, which suggests that during pretraining models might acquire a recency bias.

The role of global context: To further investigate the role of global context, we also consider – `Constant` replacement policy, which replaces every token in the global context with a random token from A. We also change the size of the global context by sampling more or less tokens than the original string.

Importance of the replacement policy: Figure 5.4c shows the accuracy of prefixes of different lengths with global contexts being generated using `Random` and `Constant` policies. Our results show that when trying to recollect tokens from memorized strings, the global context is important. Maintaining the original distribution P_A of the tokens in the global context (`Random`), even though the substring itself changes, leads to high recollection accuracy. Biasing (`Constant`) the global context leads the model to assume a wrong distribution, lowering its recollection accuracy.

Importance of the length of global context: Is it enough to maintain P_A when generating the global context, or does the length of the context also matter? To determine this, we increase (by 50% or 100%) or decrease (by 50% or 100%) the number of tokens in the global context while applying the `Random` replacement strategy. We show in Figure 5.4d that adding or removing tokens from the global context does not impact the recollection accuracy for fixed local prefix length, as long as some global context is preserved. When fully eliminating the global context



(a) $\ell = 26, K = 16$ (b) $\ell = 2, K = 32$

Figure 5.5: **[Accuracy on different strings during sequential memorization.** ($n = 1024, \mathcal{M} =$ **Pythia-1B)**] Each curve denotes a new string. As the model memorizes new strings, it forgets old ones, shown by the drop in accuracy after the first 50 epochs per string.

(0 x GC), recollection accuracy drops drastically. Thus, the amount of global context, and the position of i within s is not important, as long as some global context information, and the local prefix, are present.

In summary, while only keeping the tokens in the local prefix constant is mostly sufficient to recollect the target token, this only holds when the token distribution in the global context is preserved. We provide additional details in Appendix C.5.1 and show in Appendix C.5.3 results for additional models in Figure C.32 for replacement strategy, and in Figure C.33 for changes in global context size, with the same takeaways.

Models with absolute position encodings: We repeat the above experiments for models with older architectures that use absolute position encoding, such as GPT-2 and OPT. The results are shown in Figures C.32 and C.33 in Appendix C.5.4. Those results show that these models are relying much more on position-based memorization and therefore the global context does not matter as long as it does not affect the position of the tokens in the string.

Implications for how LLMs memorize strings: Our investigation here reveals the subtle but important differences in the role played by the prefix tokens close to (local prefix) and far away (global context) from the token being recollected from memory. Our finding that the greater the degree of memorization, the smaller the length of local prefix needed for recollection, suggests a potential (proxy) measure to estimate and quantify memorization of a string. More importantly, our findings offer a starting point for a potential explanation for why higher entropy strings are more easily memorized – in higher entropy strings, the same length of local prefix is more predictive of the next token. However, we are far from having a comprehensive theory explaining all our experiments. For example, we observed that creating strings where small prefixes are more predictive alone does not make them more memorable (Appendix C.2.5), and introducing regularities into strings can also lead to interference with in-context learning (Appendix C.2.4).

5.5 Q3: How do Models Behave when Sequentially Memorising Random Strings?

In order to address this question we perform the following experiment: given an alphabet A of size ℓ we generate $K \in \{16, 32\}$ random strings from A: $\{s_1, s_2, \ldots, s_K\}$. Then, we train model $\mathcal{M}$ to sequentially memorize each of the s_i's, for 50 epochs per string.

Figure 5.5 shows the accuracy of a Pythia-1B model $\mathcal{M}$ during sequential memorization. Accuracy for different strings is indicated by different colours, and we show the accuracy for each

s_i during the initial 50 epochs when $\mathcal{M}$ trains on it, as well as during the following epochs, when $\mathcal{M}$ trains on other strings. We make two key observations: (i) As models memorize subsequent strings, they forget the strings they previously memorized, as seen by the decreasing accuracy curves after the first 50 epochs in Figure 5.5. However, forgetting happens more slowly for strings memorized later in the sequence. (ii) With sequential memorization, models become better at memorising new strings, since strings memorized later in the sequence are memorized faster. For instance, without previous memorization, Pythia-1B takes 29 epochs (iterations) to achieve 99% accuracy in memorising strings with $\ell = 26$, but after memorising 15 other strings, it can achieve the same accuracy within 8 epochs, i.e., nearly four times fewer iterations. This speedup happens during the *Memorisation-Phase*. We show similar results for more models in Appendix C.6.

Implications for conditioning models to memorize or forget: Our findings here show that we can both trigger forgetting of previously-memorized information, and make models better at memorising new information. Specifically, it suggests that one way of ensuring that a trained model forgets cryptographic keys of a certain format might be to memorize new randomly generated keys with a similar format. On the other hand, it is worrisome that models can be primed to better memorize specific types of strings, which raises the spectre of new types of memorization risks and attacks. We are not aware of any prior works that identified such risks.

Additionally, we considered how memorization affects the model's performance on other tasks. For this, we experimented with how memorising random strings impacts the model's loss on the wikitext testset Merity et al. [2016]. Our results in Appendix C.4.3 show that memorising a single random string in isolation can negatively affect the model's performance. However, when memorizing random strings in the context of natural language data, models can both improve their natural language modelling abilities, while also memorizing the random strings.

5.6 Related Work

The topic of memorization has received great attention in the context of LLMs that are trained on large "internet-scale" data [Biderman et al., 2023a, Carlini et al., 2019, Lukas et al., 2023, Mattern et al., 2023, McCoy et al., 2023, Song and Shmatikov, 2019, Zhang et al., 2021]. Most of these works propose a definition of memorization to test whether the model can generate a given string (present in the training data) using particular prompts or prefixes. While they subtly differ in how exactly they operationalize a measure of memorization, at a higher level, all these works are concerned with answering the "why" question around memorization, *e.g.* why should memorization be a practical concern? To this end, these works show compelling examples of cases where memorization can hurt (*e.g.* privacy leaks via reconstruction [Carlini et al., 2021] or membership inference [Mattern et al., 2023]). Similarly, there is also a case to be made for memorization being desirable in cases where the goal is to generate facts and reduce LLM hallucinations. Grounding the generation by LLMs in some verified training data sources can be an effective way to generate trustworthy information [AlKhamissi et al., 2022, Borgeaud et al., 2022, Guu et al., 2020, Haviv et al., 2022, Khandelwal et al., 2019, Li et al., 2023a, Petroni et al., 2019, Tay et al., 2022].

We differ from existing works in two key aspects. Firstly, our key goal is to build a foundational understanding of *how* these models memorize. Thus, we do not engage with the question of memorization being desirable or undesirable and rather provide observations on *how* memorization happens at an input-output level. Secondly, prior works are motivated by applications and thus simulate scenarios where memorization happens *unintentionally*, *i.e.*, these works typically do not repeat token sequences during training or finetuning [Carlini et al., 2022, Tirumala et al., 2022]. We instead force the model to memorize random strings by training on the same tokens multiple times, until the model can generate these random strings verbatim. Our work adds to

the nascent literature focused on building a better scientific understanding of memorization in LLMs (*e.g.* [Carlini et al., 2022, Jagielski et al., 2022, Kharitonov et al., 2021, Tirumala et al., 2022]).

5.7 Conclusions and Limitations

Conclusions: In this chapter, we study the phenomenon of memorization at a foundational level. We do so using random strings, which provide us with controlled "laboratory" conditions that ensure that our observations meet three important validity criteria: 1) *Isolation from other memorized data.* 2) *Isolation from non-memorization phenomena* such as in-context learning. 3) *Targeted intervention on string properties without confounders.* We make a number of intriguing observations, including that models exhibit a *Guessing-Phase* and a *Memorisation-Phase*, that strings with higher entropy are easier to memorize, that models can often recall memorized tokens using small subsets of the entire context, and that sequentially memorising strings changes the memorization dynamics.

Our findings have significant implications for studies focusing on quantifying memorization, understanding how memorization works, and estimating privacy risks with memorization. Furthermore, many of our empirical findings cannot be easily explained and the quest for a comprehensive explanatory theory of all our findings raises many open and challenging questions.

Limitations: Our insights on memorization heavily rely on random data, and it is possible that some of the observations might change for real-world data. We conduct extensive validation experiments to ensure that our findings are robust, but there might be additional factors that impact memorization behaviour in more complex scenarios. We also focus on observing *what* happens during memorization, and leave it up to future work to develop a deeper understanding for *why* models behave this way during memorization, *e.g.* why memorising multiple strings in sequence makes memorization faster.

Chapter 6

Conclusion

This thesis makes contributions both towards understanding unfairness in algorithmic decision systems, and towards understanding the mechanisms that underpin the behavior of deep learning algorithms. Through four comprehensive studies in the domains of fair machine learning, targeted advertising, transfer learning, and memorization in large language models, the research develops new methods and provides new insights into the issues in each domain.

Measuring Unfairness in Algorithmic Decisions via Inequality Indices The first part of the thesis presents a novel approach to measuring algorithmic unfairness using inequality indices from economics. The work introduces a justified and general framework for comparing the unfairness of different algorithms. By quantifying unfairness at both the individual and group levels, the study reveals critical trade-offs between various fairness notions. It shows that minimizing between-group unfairness can increase within-group unfairness, thus increasing overall unfairness. This finding underscores the importance of considering both components in fairness assessments and contributes a new perspective to the discourse on algorithmic fairness.

Fairness Case Study: Discrimination in Facebook Advertising The second part of the thesis addresses the pervasive issue of discrimination in online targeted advertising, specifically on the Facebook ad platform. It demonstrates that current measures to prevent discrimination, such as prohibiting the use of sensitive attributes, are insufficient. The study systematically examines Facebook's targeting methods, revealing that there are several different ways in which discriminatory ads can be created without explicit use of sensitive attributes. The findings advocate for the development of fundamentally new methods to mitigate discrimination, emphasizing the need for a shift in how fairness in targeted advertising is evaluated and enforced.

The Role of Representational Invariance in Transfer Learning In the third part of the thesis, we explore the importance of representational invariance in transfer learning. We introduce a family of synthetic datasets that allows us to precisely control variation factors in the data. By systematically investigating how different invariances affect transfer performance, our study finds that in many cases, invariance is more important than other factors like the number of training samples or model architecture for achieving high transfer performance. Additionally, our results highlight conditions where invariance can hinder the transferability of representations, and we investigate how well invariance transfers between domains. Taken together, our findings provide insights into the role of invariance in transfer learning and suggest that considering invariance carefully is critical for creating models whose representations transfer well to downstream tasks.

Understanding Memorization in Large Language Models with Random Strings: The final part of the thesis examines memorization in large language models (LLMs) using a framework based on exposing models to random strings. We conduct a detailed analysis of the dynamics of memorization, identifying two consistent phases across different model families and other factors that models undergo during the memorization process. We further find that entropy plays a key role in influencing the memorabilitly of random strings, that the memorization order of tokens is random, that that sequential exposure to random strings affects memorization signif-

icantly, and we characterize the role that local prefixes and global context play in memorization. These insights have important implications quantifying memorization and privacy risks in LLMs, and provide important clues for better understanding how LLMs acquire linguistic knowledge.

Appendix A

Additional Material on Measuring Unfairness in Algorithmic Decisions via Inequality Indices

A.1 Appendix: Technical Material

Proof of Proposition 2.2.1 If there exists a classifier θ' with $L_{\mathcal{D}}(\theta') = 0$, then that classifier minimizes I: For all $i = 1, \cdots, n$, the benefit i receives under θ', denoted by $b_i^{\theta'}$, is equal to $1 + \theta'(\mathbf{x}_i) - y_i = 1$. That is everyone gets the same benefit under θ', and as the result, $I(\mathbf{b}^{\theta'}) = I(\mathbf{1}) = 0$.

If there exists a classifier θ with $I(\mathbf{b}^{\theta}) = 0$, θ must assign the same benefit to everyone: there exists $b \in \{0, 1, 2\}$ such that for all $i = 1, \cdots, n$, $b_i^{\theta} = 1+\theta(\mathbf{x}_i)-y_i = b$. Now let $\theta'(\mathbf{x}) = \theta(\mathbf{x})+1-b$. It is easy to verify that θ' has zero error (i.e. $L_{\mathcal{D}}(\theta') = 0$). □

Proof of Proposition 2.2.2 Let $p = \mathbb{P}_{(\mathbf{x},y)\sim\mathcal{P}}[y = 1|\mathbf{x} = \tilde{\mathbf{x}}]$ so that $0 < p < 0.5$. We know that the Bayes/accuracy optimal classifier assigns label 0 to every instance i with $\mathbf{x}_i = \tilde{\mathbf{x}}$. Suppose $\mathcal{D}$ consists of n instances all with $\mathbf{x}_i = \tilde{\mathbf{x}}$. Given the population invariance property of I, n can be arbitrarily large without affecting the value of $I(.)$. Therefore we instead reason about the limiting case where $n = \infty$. In this case, we expect exactly p fraction of the instances to have $y = 1$ and the other $(1 - p)$ fraction to have $y = 0$. The benefit distribution $\mathbf{b}^A$ for θ^A, therefore, consists of p fraction of the population receiving benefit 0 and the other $(1 - p)$ fraction receiving 1.

Now consider a probabilistic classifier θ^q that randomly assigns label 1 to each instance in $\mathcal{D}$ with probability $q \in (0, 1)$. The resulting benefit distribution $\mathbf{b}^q$ of θ^q is as follows: $p(1 - q)$ fraction of the population receive benefit 0; $pq + (1 - p)(1 - q)$ faction receive benefit 1; and $(1 - p)q$ faction receive benefit 2.

We claim that for $q = (1 - p)$, $I_{\mathcal{D}}(\theta^A) > I_{\mathcal{D}}(\theta^q)$. To see this, note that $\mathbf{b}^q$ can be constructed from $\mathbf{b}^A$ via a series of inequality-reducing operations:

1. $\mathbf{b}' = 2 \times \mathbf{b}^A$, so that $\mathbf{b}'$ consists of p fraction of the population receiving benefit 0 and the other $(1 - p)$ fraction receiving 2. Due to the scale invariance property of I, we know $I(\mathbf{b}') = I(\mathbf{b}^A)$.

2. Perform the following progressive transfer on $\mathbf{b}'$ to obtain $\mathbf{b}''$: Take one unit of benefit from $p(1 - p)$ fraction of the population whose benefit is 2, and give it to $p(1 - p)$ fraction with benefit 0. The resulting distribution, $\mathbf{b}''$, consists of p^2 faction with benefit 0; $2p(1 - p)$ faction with benefit 1; and $(1 - p)^2$ faction with benefit 2. Because I satisfies the Dalton principle and $p(1 - p) > 0$, we have that $I(\mathbf{b}'') < I(\mathbf{b}')$.

Combining the above two, we have $I(\mathbf{b}'') < I(\mathbf{b}^A)$. It only remains to note that $\mathbf{b}''$ is precisely $\mathbf{b}^q$ for $q = (1-p)$. Therefore, we conclude $I(\mathbf{b}'') = I(\mathbf{b}^{(1-p)}) < I(\mathbf{b}^A)$. This finishes the proof. □

The following example shows that the accuracy of the fairness optimal classifier can be arbitrarily bad compared to that of the accuracy optimal classifier.

Example 1. *Consider the example in the proof of Proposition 2.2.2. Let I be the generalized entropy with $\alpha = 2$. We claim that the fairness optimal classifier is one that assigns label 1 to every instance. To see this, recall that under θ^q, $p(1-q)$ fraction of the population receive benefit 0; $pq + (1-p)(1-q)$ faction receive benefit 1; and $(1-p)q$ faction receive benefit 2. So the mean benefit μ is equal to $1-p+q$. Taking derivative with respect to q, we have*

$$\frac{d}{dq}\left((pq+(1-p)(1-q))\left(\frac{1}{1-p+q}\right)^2 + (1-p)q\left(\frac{2}{1-p+q}\right)^2\right) = 0$$

$$\Rightarrow q^* = \frac{1-3p+2p^2}{3-2p}$$

The derivative is positive for $q < q^*$ and negative for $q > q^*$. The minimum therefore happens at either $q = 0$ or $q = 1$. Given that for $0 < p < 1$, $\frac{1}{1-p} > \frac{4-3p}{(2-p)^2}$, we obtain that $q = 1$ minimizes I.

The fairness optimal classifier assigns label 1 to every instance resulting in accuracy p, whereas the accuracy optimal classifier can achieve accuracy $(1-p)$. The ratio $\frac{1-p}{p}$ can be arbitrarily large if p is taken to be sufficiently small.

Proof of Proposition 2.2.3 Note that because of the optimality of θ^*_β for (2.4), if $I_\beta(\theta^*_\beta) \neq I_\beta(\theta^*)$, it must be the case that $I_\beta(\theta^*_\beta) < I_\beta(\theta^*)$. If $I(\theta^*_\beta) \leq I(\theta^*)$, then θ^*_β is an optimal solution to (2.3), and $I_\beta(\theta^*_\beta) < I_\beta(\theta^*)$. This is a contradiction with the choice of θ^*. If $I_\omega(\theta^*_\beta) \leq I_\omega(\theta^*)$, then combined with the fact that $I_\beta(\theta^*_\beta) < I_\beta(\theta^*)$, we have that $I(\theta^*_\beta) < I(\theta^*)$, which is a contradiction with the optimality of θ^* for (2.3). □

Proof of Proposition 2.2.4 Suppose $|G| = m$ and $|G'| = m'$. Let $\mathbf{b} = \left(\mathbf{b}^{(g_1,g'_1)}, \cdots, \mathbf{b}^{(g_m,g'_{m'})}\right)$ where $\mathbf{b}^{(g_i,g'_j)}$ specifies the benefit distribution for individuals in group (g_i, g'_j) and $\boldsymbol{\mu}^{(g_i,g'_j)}$ specifies the distribution in which each individual in (g_i, g'_j) receives the group's mean benefit. Note that $I_\beta^{G\times G'}(\mathbf{b})$ can be written as

$$\begin{aligned}
& I\left(\boldsymbol{\mu}^{(g_1,g'_1)}, \cdots, \boldsymbol{\mu}^{(g_m,g'_{m'})}\right) \\
= \;& I_\beta^G\left(\boldsymbol{\mu}^{(g_1,g'_1)}, \cdots, \boldsymbol{\mu}^{(g_m,g'_{m'})}\right) + I_\omega^G\left(\boldsymbol{\mu}^{(g_1,g'_1)}, \cdots, \boldsymbol{\mu}^{(g_m,g'_{m'})}\right) \\
= \;& I\left(\boldsymbol{\mu}^{g_1}, \cdots, \boldsymbol{\mu}^{g_m}\right) + I_\omega^G\left(\boldsymbol{\mu}^{(g_1,g'_1)}, \cdots, \boldsymbol{\mu}^{(g_m,g'_{m'})}\right) \\
\geq \;& I\left(\boldsymbol{\mu}^{g_1}, \cdots, \boldsymbol{\mu}^{g_m}\right) \\
= \;& I_\beta^G(\mathbf{b})
\end{aligned}$$

where to obtain the second line, we used the additive decomposability property of I; for the third line we used the definition of the between-group component, and finally to obtain the conclusion, we used the zero-normalization property of I. □

Proof of Proposition 2.2.5 Recall that $I(\mathbf{b}) = I_\beta^G(\mathbf{b}) + I_\omega^G(\mathbf{b})$ and $I_\beta^G(\mathbf{b}), I_\omega^G(\mathbf{b}) \geq 0$. Therefore, we have $0 \leq \frac{I_\beta^G(\mathbf{b})}{I(\mathbf{b})} \leq 1$.

Consider a benefit distribution $\mathbf{b}$ in which members of group $g_1 \in G$ receive benefit 1, and everyone else receives benefit 0. It is easy to see that for this distribution $\frac{I_\beta^G(\mathbf{b})}{I(\mathbf{b})} = 1$. Similarly,

consider a benefit distribution $\mathbf{b}'$ that assigns a benefit of 1 to half of the population in each group, and 0 to everyone else. It is easy to that $\frac{I_\beta^G(\mathbf{b}')}{I(\mathbf{b}')} = 0$. □

The following example shows that an added feature may in fact worsen the unfairness of the accuracy optimal classifier.

Example 2. *Let $I(.)$ be the generalized entropy with $\alpha = 2$. Suppose for all $\mathbf{x}_i = \tilde{\mathbf{x}}$, $p = \mathbb{P}_{(\mathbf{x},y)\sim D}[y = 1|\mathbf{x} = \tilde{\mathbf{x}}] > 0.5$, so θ^A assigns label 1 to every instance with $\mathbf{x}_i = \tilde{\mathbf{x}}$. So in the resulting benefit distribution, p fraction of the population receives benefit 1 and the other $(1-p)$ fraction receives 2. The mean benefit is, therefore, $(2-p)$ and GE is equal to*[1]

$$p\left(\frac{1}{2-p}\right)^2 + (1-p)\left(\frac{2}{2-p}\right)^2 = \frac{4-3p}{(2-p)^2}.$$

Suppose with the addition of a new binary feature, the population breaks down into two sub-populations, one corresponding to $\mathbf{x} = (\tilde{\mathbf{x}}, 0)$ and the other corresponding to $\mathbf{x} = (\tilde{\mathbf{x}}, 1)$. Let $\frac{r}{1-r}$ be the relative size of the former sub-population to the latter ($0 \leq r \leq 1$). We would like the accuracy optimal classifier to be different for each subpopulation, so for now let's assume:

- $\mathbb{P}_{(\mathbf{x},y)\sim D}[y = 1 \wedge \mathbf{x} = (\tilde{\mathbf{x}}, 0)] = \frac{r}{2} - \epsilon$, *so θ^A assigns label 0 to every instance with $\mathbf{x} = (\tilde{\mathbf{x}}, 0)$—this is because $\frac{r}{2} - \epsilon < \frac{1}{2}r$.*
- $\mathbb{P}_{(\mathbf{x},y)\sim D}[y = 1 \wedge \mathbf{x} = (\tilde{\mathbf{x}}, 1)] = p - \frac{r}{2} + \epsilon$, *so θ^A assigns label 1 to every instance with $\mathbf{x} = (\tilde{\mathbf{x}}, 1)$—this is because $p - \frac{r}{2} + \epsilon > \frac{1}{2}(1-r)$.*

In the resulting benefit distribution, $\frac{r}{2} - \epsilon$ fraction of the total population receives benefit 0, $p + 2\epsilon$ fraction of the total population receives benefit 1, and the other $(1 - p - \frac{r}{2} - \epsilon)$ fraction receives benefit 2. The mean benefit is, therefore, $(2 - p - r)$ and GE is equal to

$$\begin{aligned}&(p+2\epsilon)\left(\frac{1}{2-p-r}\right)^2 + (1-p-\frac{r}{2}-\epsilon)\left(\frac{2}{2-p-r}\right)^2 \\ &= \frac{4-3p-2r-2\epsilon}{(2-p-r)^2}.\end{aligned}$$

Take $p = 0.9$, $r = 0.2$ and $\epsilon = 0.001$, and we have:

$$\frac{4-3p}{(2-p)^2} = 1.075$$

$$\frac{4-3p-2r-2\epsilon}{(2-p-r)^2} = 1.10$$

The above shows the addition of a new feature can worsen the fairness of the accuracy optimal classifier.

[1] We are dropping the constants from the definition of the inequality index, as they don't affect the comparison.

Appendix B

Additional Material on the Role of Representational Invariance in Transfer Learning

B.1 Dataset Details

B.1.1 Transforms-2D Dataset Details

Transforms-2D datasets $\mathcal{D}(O, T)$ are parameterized by a set of foreground objects O and a set of transformations of those objects T. An image is sampled by choosing a foreground object o uniformly from a set of objects O, sampling a transformation t from T and pasting the transformed object $t(o)$ onto a background image chosen uniformly at random.

Foreground objects and background images. Foreground and background images are based on the SI-Score dataset Djolonga et al. [2021], which was introduced to study the connection between out-of-distribution and transfer performance. The creators of the SI-Score dataset have curated 61 foreground categories [1] (such as banana, jeans, sock, etc.) with a total of 614 object images, and 867 background images of nature scenes. The dataset is available under the following URL: `https://github.com/google-research/si-score`. It uses the Apache 2.0 license.

There are several images for each foreground category, *e.g.* several images of bananas, and each image has a transparency mask such that images only contain the object and no background. To be able to precisely control the variation between different images and to ensure that any changes in object appearance are only due to the transformations that are applied to them, we only use one image per foreground category throughout the analysis. In practice, we simply choose the image whose file name has the lowest lexicographical rank. For each parameterization $\mathcal{D}(O, T)$ of the Transforms-2D dataset, we always use the same set of 867 background images (hence they do not appear as a parameter), but choose a subset O out of all available 61 objects categories.

Transformations. To cover a reasonably large variety of transformations and to ensure that our results are not overly dependent on the specific choice of transformations, we use three categories of 2D transformations: *geometric*, *photometric* and *corruption* transformations. Geometric transformations are affine transformations of the object geometry, photometric transformations change the color of the object and corruption transformations, inspired by Hendrycks and Dietterich [2019] degrade the quality the object. Each category consists of six transformation types, *e.g.* rotation is part of the geometric transformations, and for each transformation type, there can be potentially a large or infinite number of actual transformations (*e.g.* there are infinitely many rotations, one for each possible angle). Transformations are mainly implemented via standard

[1] The original paper Djolonga et al. [2021] mentions 62 categories, but publicly available dataset only contains 61 categories.

PyTorch [Paszke et al., 2019] data augmentations. An overview over the transformations, their parameters and samples for each of them is shown in Figure B.1.

We scale all images to a resolution of 32×32, which mimics the resolution of the CIFAR-10 and CIFAR-100 datasets Krizhevsky et al. [2009]. Resolution is configurable though, and it is also possible to sample images with considerably higher resolution. We do not use additional data augmentations to train models, since that would interfere with the transformations present in the datasets. For most experiments, we use 50,000 training, 10,000 validation and 10,000 test samples, again mimicking the CIFAR datasets.

B.1.2 Additional Information on the CIFAR-10 + Transforms-2D Dataset

Figure B.2 shows example images for the augmented CIFAR-10 images that we use in the analysis in section 4.2.2 to investigate the effects of irrelevant information on learned invariances.

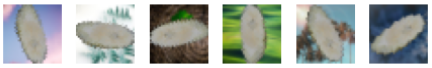
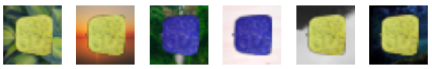
Category	Transformation	Parameters used	Samples
Geometric	Translate	x and y position by $[0\%, 50\%]$ image size	
	Rotate	by $[0, 360]$ degrees	
	Scale	to $[40\%, 100\%]$ size	
	Shear	x and y by $[-50, 50]$ degrees	
	Vertical flip	with 50% probability	
	Horizontal flip	with 50% probability	
Photometric	Hue	deviation in $[-0.5, 0.5]$	
	Brightness	change in $[-1, 1]$	
	Grayscale	with 50% probability	
	Posterize	with 50% probability to 1 bit	
	Invert	with 50% probability	
	Sharpen	with probabiilty 50%, sharpness factor 7	
Corruption	Gaussian blur	kernel size 7 pixels, $\sigma \in [0.1, 1.5]$	
	Gaussian noise	with $\mu = 0, \sigma = 1$, probability 50%	
	Pixelate	to half resolution, probability 50%	
	Elastic distortion	$\alpha = 150$, probability 50%	
	Erasing	square with 14 x 14 pixels size at random position, probability 50%	
	Contrast	change in $[-1, 1]$	

Figure B.1: **[Transforms-2D transformations.]** Categories, transformation types, transformation parameters and samples generated using the transformations for the Transforms-2D dataset.

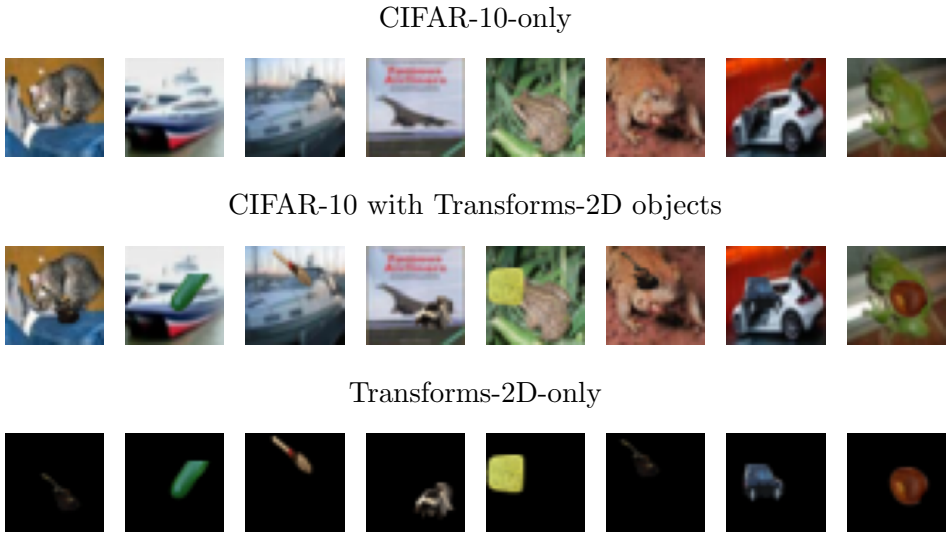


Figure B.2: **[Examples of test images in the irrelevant feature analysis]**. From top to bottom: CIFAR-10 only ($X = C$), CIFAR-10 with pasted objects ($X = C + O$), objects only ($X = O$)

B.2 Additional details on the training and evaluation setup

B.2.1 Architectures

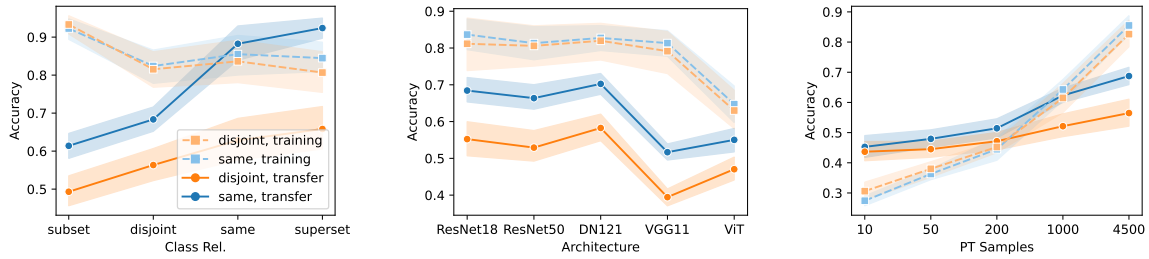
For most of the experiments in the chapter we use ResNet-18 models [He et al., 2016]. They are generally sufficient for the 32 x 32 input size that we use for the Transforms-2D dataset and achieve close to 100% accuracy on their training distributions.

We also compare the importance of invariance with other pretraining factors, including architectures in Section 4.2.1 and in Appendix B.3.1. For this, we additionally use ResNet-50 [He et al., 2016], DenseNet-121 Huang et al. [2017], VGG-11 Simonyan and Zisserman [2014] and Vision Transformer (ViT) Dosovitskiy et al. [2020] models. All models are implemented in PyTorch [Paszke et al., 2019]. For all CNN models we use implementations adapted for CIFAR datasets available here `https://github.com/kuangliu/pytorch-cifar`. For ViTs, we use an implementation from this repository `https://github.com/omihub777/ViT-CIFAR` (achieving more than 80% accuracy on CIFAR-10) with a patch size of 8, 7 layers, 384 hidden and MLP units, 8 heads and no Dropout.

B.2.2 Training

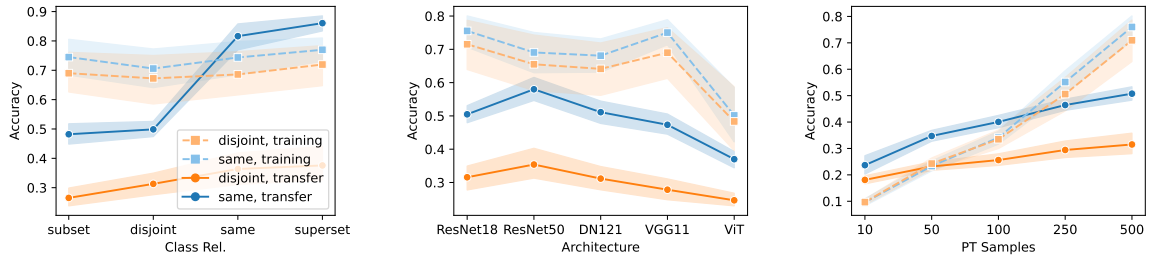
We train models using Pytorch Lightning [Falcon and The PyTorch Lightning team, 2019] on the Transforms-2D dataset for 50 epochs and fine-tune their output layer (while keeping the rest of the network frozen) for 200 epochs. We find that 50 training epochs are sufficient for models to reach close to 100% accuracy. For fine-tuning we choose a larger number of 200 epochs, because models that have been pretrained with invariances different from those required by the target task typically converge only slowly. Models that have been trained with the same invariances on the other hand converge much faster.

All CNN models are trained and fine-tuned using the Adam optimizer [Kingma and Ba, 2014] with a learning rate of 0.001. For ViTs, we use cosine learning rate scheduler [Loshchilov and



(a) Invariance vs class relationship. (b) Invariance vs architecture. (c) Invariance vs number of training samples per class.

Figure B.3: [**Invariance to data augmentations vs the importance of other factors on CIFAR-10.**] Training and transfer performance for models trained with different factors of variation and different transformations on the CIFAR-10 dataset.



(a) Invariance vs class relationship. (b) Invariance vs architecture. (c) Invariance vs number of training samples per class.

Figure B.4: [**Invariance to data augmentations vs the importance of other factors on CIFAR-100.**] Training and transfer performance for models trained with different factors of variation and different transformations on the CIFAR-100 dataset.

Hutter, 2016] with a learning rate that is decayed from 0.001 to 0.00001 over the duration of training, and a weight decay of 0.00001, with 5 warmup epochs. We keep the checkpoint that achieves the highest validation accuracy during training and fine-tuning.

B.2.3 Experiments

We repeat each experiment 10 times and report mean and variance values. Shaded regions in the plots indicate 95% confidence intervals, and annotations in the tables indicate one standard deviation. During each run, we randomize the configuration of the Transforms-2D dataset and the sampling process. For the configuration, this means that we sample different sets of objects O and different sets of transformation types in T. For the sampling process, we randomize the order of sampling foreground objects, transformations of each transformation type and background images.

B.3 Additional Results on the Importance of Invariance for Transfer Performance

B.3.1 Real-world Experiments on Augmented CIFAR data

We conduct experiments analogous to those described in Section 4.2.1 on real-world CIFAR-10 and CIFAR-100 data. We observe results similar to those reported in Figure 4.2 for CIFAR-10 in Figure B.3 and for CIFAR-100 in Figure B.4. In all cases, the model trained with the same transformations as the target task significantly outperforms its counterpart pretrained with a disjoint set of transformations (in most cases between 10% and 20%). One difference to the results reported in Section 4.2.1 is that in Figure 4.2, even the worst performing model using the same transformations as the target task outperforms the best-performing model using a disjoint set of transformations. This is no longer the case with the CIFAR models, for example the model with disjoint transformations trained with 4500 samples per class outperforms the model with the same transformations trained with 200 samples per class (but not the one with 1000 samples per class). This is somewhat expected though, as more samples presumably help the models to better learn invariances to the dataset inherent (as opposed to transformation-induced) invariances.

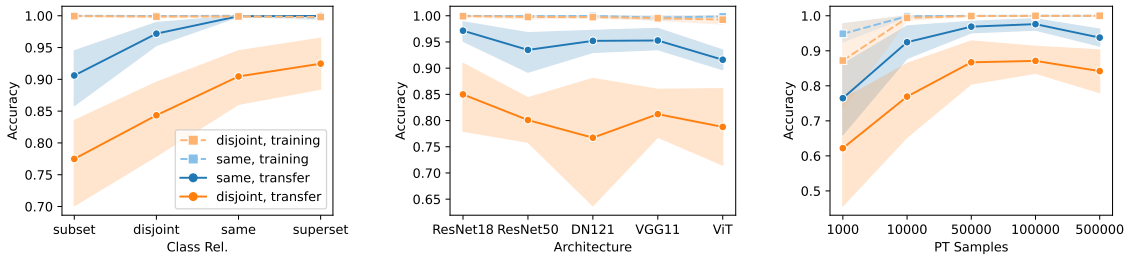
Takeaways: The results on real-world CIFAR data largely confirm our findings made using the synthetic Transforms-2D dataset. Invariance to the right transformations significantly improves the transfer performance of models and is a very important factor compared to other pretraining dimensions such as the number of samples, the architecture and the class relationship. E.g. on CIFAR-10, you need roughly 20 times more samples to compensate for a mismatch in invariance.

B.3.2 How Does Fine-tuning the Whole Model Impact Invariance?

We want to better contextualize our findings, and understand how difficult it is to change the invariance properties of pretrained representations. Therefore, we apply an additional fine-tuning pass to the models in Section 4.2.1 and Appendix B.3.1 on the target dataset after their last layer has been tuned on top of a frozen feature extractor, mimicking the linear probing, then full fine-tuning approach discussed in Kumar et al. [2022].

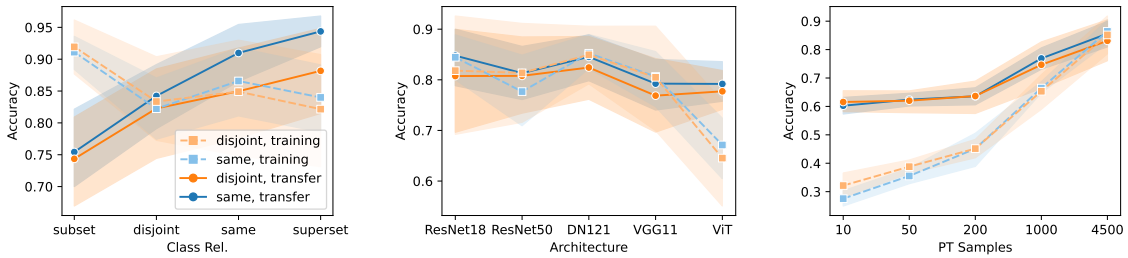
We fine-tune models after they have been trained with linear probing on the target task for and additional 50 epochs with a learning rate of 10^{-3} (chosen based on a grid search over the values $10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$) and a linearly decreasing learning rate schedule. For Transforms-2D, we explore two fine-tuning dataset sizes: a low-data setting with 200 and a high-data setting with 2000 fine-tuning samples. For the augmented CIFAR-10 and CIFAR-100 datasets, we use 1% and 10% of the original training data for the low- and high-data settings, respectively. Results are averages over 5 runs.

Figure B.5 shows the results for Transforms-2D with 200 samples, Figure B.6 for CIFAR-10 with 1% of the full training samples and Figure B.7 for CIFAR-100 with 1% of the full training samples. Figure B.8 and Figure B.9 show the comparison of the differences in transfer performance due to invariance vs other factors, for the low- and high-data fine-tuning scenarios, respectively. The results show that full fine-tuning can change the invariance properties of representations such that even the representations not trained with the right set of invariances can adapt to the target task. However, the degree to which this is possible depends on the amount of available fine-tuning data. Especially for Transforms-2D in the low-data setting, the representations trained with the right invariances still perform significantly better than the ones trained with disjoint transformations. In the high-data setting the difference diminishes, since the training starts to approximate full retraining on the target task. However, the representations trained with the same



(a) Invariance vs class relationship. (b) Invariance vs architecture. (c) Invariance vs number of pre-training samples.

Figure B.5: **[Invariance to data augmentations vs the importance of other factors on Transforms-2D with full fine-tuning on 200 samples.]** Training and transfer performance for models trained with different factors of variation and different transformations on the Transforms-2D dataset, fine-tuned on 200 samples after linear probing.



(a) Invariance vs class relationship. (b) Invariance vs architecture. (c) Invariance vs number of pre-training samples.

Figure B.6: **[Invariance to data augmentations vs the importance of other factors on augmented CIFAR-10 with full fine-tuning on 1% of the full training samples.]** Training and transfer performance for models trained with different factors of variation and different transformations on the CIFAR-10 dataset, fine-tuned on 1% of the full training set samples after linear probing.

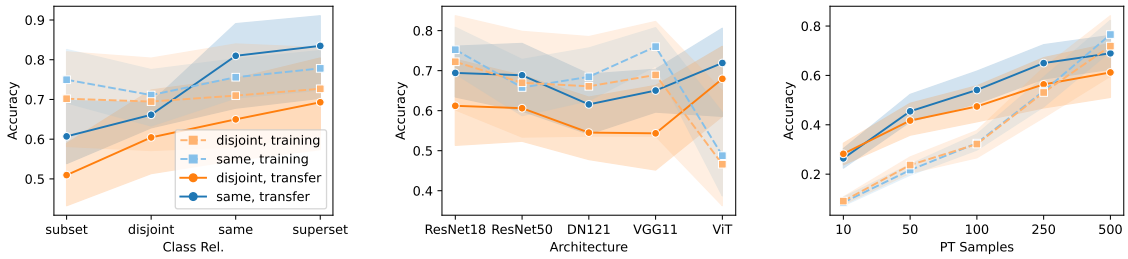
transformations as the target task still perform at least as well as those trained with different transformations, in all cases.

B.3.3 How Does Relevance and Availability of Input Information Impact Invariance?

In Section 4.2.2 we have shown that the *relevance* and *availability* of input information impacts the invariances learned by a model. In this Section, we make two observations.

First, we show that the two models trained on CIFAR-10 images with objects pasted on them ($X_p = C + O$) perform very differently after fine-tuning at predicting CIFAR-10 classes ($Y_t = C$) and object categories ($Y_t = O$) respectively, depending on whether they were pre-trained to predict the CIFAR-10 classes ($Y_p = C$) or object categories ($Y_p = O$), even though they saw exactly the same input data. This result shows that the relevance of a feature for a target task is a key driver in determining which input features a model becomes invariant to.

Second, the models trained on only CIFAR-10 images to predict CIFAR-10 classes ($X_p = C, Y_p = C$) and on only object images to predict object categories ($X_p = O, Y_p = O$) show better transfer performance on the respective other task ($Y_t \neq Y_p$) than their counterparts that



(a) Invariance vs class relationship. (b) Invariance vs architecture. (c) Invariance vs number of pretraining samples.

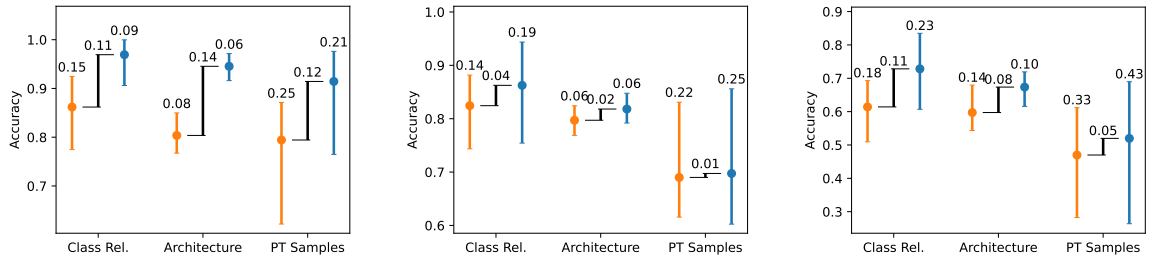
Figure B.7: **[Invariance to data augmentations vs the importance of other factors on augmented CIFAR-100 with full fine-tuning on 1% of the full training samples.]** Training and transfer performance for models trained with different factors of variation and different transformations on the CIFAR-100 dataset, fine-tuned on 1% of the full training set samples after linear probing.

were trained on CIFAR-10 images with pasted objects ($X_p = C + O$). These two models did not have access to the object- and CIFAR-features respectively and could therefore not develop invariances towards them, as did the models that saw them during training. This means that another important property in determining the invariances learned by a model is the availability of features during training. Next, we investigate the effects of relevance and availability in more detail.

Relevance. To understand how the invariance to features changes with their relevance for the target task, we train models on the dataset described in Section 4.2.2 and Appendix B.1.2. However, we now introduce correlation of different strengths between the objects and the CIFAR labels. In particular, we train models g_α on datasets $\mathcal{D}_\alpha$, where $\alpha \in \{0, 0.2, 0.4, 0.6, 0.8, 0.85, 0.9, 0.95, 1.0\}$ is the correlation strength between CIFAR labels and object categories. Input images in $\mathcal{D}_\alpha$ are constructed by pasting one specific object per CIFAR-10 class (out of a set of 10 total objects) on the CIFAR-10 training images α-fraction of the time and pasting a random object otherwise. Then, we fine-tune the g_αs' last layers and evaluate them on the augmented CIFAR-10 images, both for predicting the CIFAR-10 class ($X_t = C + O, Y_t = C$) as well as the object category ($X_t = C + O, Y_t = O$).

Availability. To investigate the effect of availability, we train models on datasets $\mathcal{D}_\beta, \beta \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ that are equivalent to the $X = C + O, Y = C$ dataset, *i.e.*objects are pasted randomly onto CIFAR backgrounds, but with the difference that objects are only pasted β-fraction of the time for dataset $\mathcal{D}_\beta$. We again fine-tune and evaluate the last layer of the models trained on $\mathcal{D}_\beta$ on the same datasets as for the relevance analysis. In both cases we paste the object in the upper right corner of the image.

Figure B.10a and Figure B.10b show the transfer accuracies of models trained to predict CIFAR-10 classes and object categories, respectively. The blue curve correspond to the models pretrained on $X_p = C + O$ datasets where the objects were pasted with different correlations with the CIFAR labels. Dashed lines show the performance of reference models. As the correlation of the pasted object categories with the CIFAR-10 classes increases, CIFAR-10 transfer accuracy mostly remains constant, whereas object category accuracy increases steadily. Only when pasted objects and CIFAR labels become perfectly correlated, CIFAR-10 accuracy drops and object detection accuracy reaches 100%. This trend shows that as the pasted objects become more relevant for the target task, the models' representations gradually become less invariant to them and allow for increasingly better prediction of the object category. CIFAR-10 accuracy on the



(a) Transforms-2D (b) CIFAR-10 (c) CIFAR-100

Figure B.8: **[Difference in transfer performance due to invariance vs other factors, for full fine-tuning, for 200 samples on Transforms-2D and 1% of samples on the CIFAR datasets.]** We compare the differences in transfer performance caused by representational invariance with the differences caused by changes to other factors on the Transforms-2D and the CIFAR-10 and CIFAR-100 datasets (with data augmentations). Orange (blue) bars show the span of transfer performance for different-transformation models g_d (same-transformation models g_s), for each comparison factor (class relationship, architecture, number of samples). Black bars show the difference between transfer performance means across factor values for g_d and g_s, *i.e.* the difference in performance caused by having the same vs different invariances as the target task.

other hand remains constant, up to the point where the pasted object can reliably replace the CIFAR background.

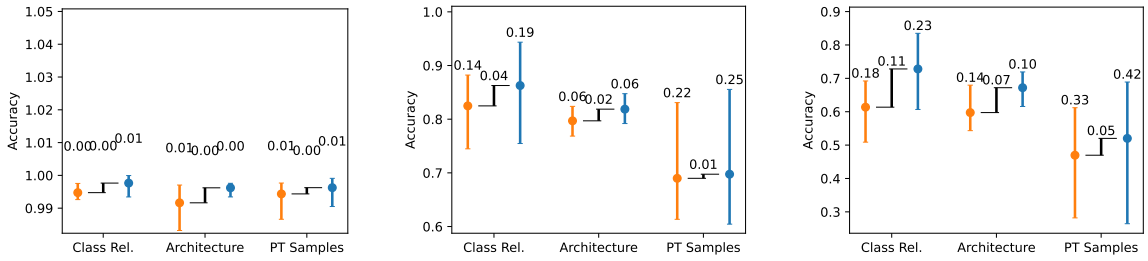
The CIFAR performance does not depend on the availability of the pasted objects, *i.e.*whether or not an irrelevant object is present in the training data has no impact on it. Object category prediction performance on the other hand quickly drops as soon as the model has access to the irrelevant object features, showing that it immediately develops an invariance towards the objects. In summary, relevance and availability both play a role in determining which invariances a model learns, but relevance seems to be the more important property.

B.4 Additional Details and Results on Invariance Transfer

B.4.1 Details on the invariance transfer experiments

Data. For the synthetic image datasets we use the Transforms-2D dataset and generate samples as described in Appendix B.1.1. We create out of distribution variants by using a new set of image objects, that the models have not seen during training. We create the synthetic random datasets by sampling pixel-values for foregrounds and backgrounds uniformly at random. Classes are created by using a different fixed random foreground pattern for each class, *i.e.* all samples for a class are transformed versions of the same random pattern on different random backgrounds. Random patterns are square-shaped, and have a length of 70% of the image size along each dimension. Examples of random images can be seen in Figure B.11a. For both synthetic image and random datasets, we use 30 classes and a single type of transformation per dataset. Images have a size of 32x32 pixels.

For the real world datasets (CIFAR-10 and CIFAR-100 Krizhevsky et al. [2009]), we apply the transformations from the Transforms-2D dataset as data augmentations. We use the same transformations as in the synthetic datasets, with the exception of the translation transformation, which here translates images by up to 30% of the image size along each axis, instead of positioning objects at a random position in the image. This change is necessary since with the CIFAR images



(a) Transforms-2D (b) CIFAR-10 (c) CIFAR-100

Figure B.9: **[Difference in transfer performance due to invariance vs other factors, for full fine-tuning, for 2000 samples on Transforms-2D and 10% of samples on the CIFAR datasets.]** We compare the differences in transfer performance caused by representational invariance with the differences caused by changes to other factors on the Transforms-2D and the CIFAR-10 and CIFAR-100 datasets (with data augmentations). Orange (blue) bars show the span of transfer performance for different-transformation models g_d (same-transformation models g_s), for each comparison factor (class relationship, architecture, number of samples). Black bars show the difference between transfer performance means across factor values for g_d and g_s, *i.e.* the difference in performance caused by having the same vs different invariances as the target task.

Model Type	Transformation Relationship	Synthetic Datasets			Real-World Datasets	
		In-Distribution	Mild OOD	Strong OOD	CIFAR-10	CIFAR-100
Image	Same	$\mathbf{0.12}_{\pm \mathbf{0.06}}$	$\mathbf{0.19}_{\pm \mathbf{0.09}}$	$\mathbf{0.66}_{\pm \mathbf{0.16}}$	$\mathbf{0.34}_{\pm \mathbf{0.17}}$	$\mathbf{0.33}_{\pm \mathbf{0.16}}$
	Other	$0.39_{\pm 0.12}$	$0.39_{\pm 0.12}$	$0.75_{\pm 0.15}$	$0.50_{\pm 0.17}$	$0.49_{\pm 0.16}$
	None	$0.38_{\pm 0.12}$	$0.39_{\pm 0.12}$	$0.78_{\pm 0.16}$	$0.51_{\pm 0.18}$	$0.49_{\pm 0.17}$
Random	Same	$\mathbf{0.12}_{\pm \mathbf{0.08}}$	$\mathbf{0.41}_{\pm \mathbf{0.17}}$	$\mathbf{0.20}_{\pm \mathbf{0.12}}$	$\mathbf{0.33}_{\pm \mathbf{0.20}}$	$\mathbf{0.30}_{\pm \mathbf{0.19}}$
	Other	$0.64_{\pm 0.13}$	$0.71_{\pm 0.14}$	$0.29_{\pm 0.11}$	$0.44_{\pm 0.16}$	$0.43_{\pm 0.16}$
	None	$0.65_{\pm 0.13}$	$0.73_{\pm 0.15}$	$0.26_{\pm 0.13}$	$0.41_{\pm 0.19}$	$0.40_{\pm 0.18}$

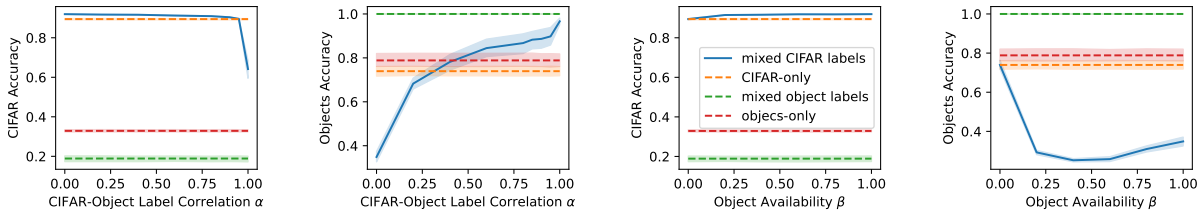
Table B.1: **[Invariance transfer under distribution shift for VGG-11 models.]**

occupying the entire image, translations would otherwise have no effect. Examples of augmented CIFAR-10 images can be seen in Figure B.11b.

Measuring invariance. For the experiments here, we report the average *sens*-score as defined in Section 4.1.4, *i.e.* the ratio of L2-distance between representations at the penultimate layer for inputs that only differ in their transformations, to the L2-distance between any two inputs from the respective dataset (lower values mean more invariance). For example, when computing the *sens*-score for the translation transformation, we compute the average L2-distances between instances of the same object on the same background in different positions, divided by the average L2-distances of different objects on different backgrounds, in different positions. We use the penultimate layer representations here, since those are the representations most relevant for transfer learning.

B.4.2 Additional results on invariance transfer

Invariance transfer results for additional architectures. Tables B.1, B.2 and B.3 show the results for the experiments on invariance transfer under distribution shift for VGG-11, DenseNet-121 and ViT models, respectively. Each cell shows the average *sens*-score (lower values mean more invariance), for models trained on image and random data. "Transformation Relationship" refers to the relationship between the training and test transformations of the models. Rows



(a) Correlation-dependent transfer performance to CIFAR-10 labels

(b) Correlation-dependent transfer performance to object labels

(c) Availability-dependent transfer performance to CIFAR-10 labels

(d) Availability-dependent transfer performance to object labels

Figure B.10: **[The effect of feature relevance and availability on learned invariances]** Left plots: Transfer accuracy on the $X_t = C + O$ dataset for models trained with different correlation strengths of pasted object categories with CIFAR labels. As the correlation resp. relevance of the pasted objects for the target task increases, models become increasingly better at predicting them and their representations become more sensitive to their presence.
Right plots: Transfer accuracy on the $X_t = C + O$ dataset for models trained with different availability of pasted object categories. As soon as a feature becomes available in the input but is irrelevant for the target task, models start to become invariant to it.

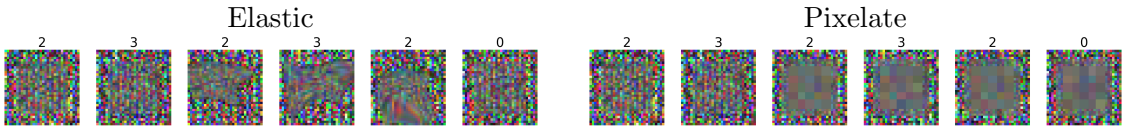
Model Type	Transformation Relationship	Synthetic Datasets			Real-World Datasets	
		In-Distribution	Mild OOD	Strong OOD	CIFAR-10	CIFAR-100
Image	Same	$\mathbf{0.16}_{\pm \mathbf{0.09}}$	$\mathbf{0.24}_{\pm \mathbf{0.10}}$	$\mathbf{0.67}_{\pm \mathbf{0.14}}$	$\mathbf{0.42}_{\pm \mathbf{0.20}}$	$\mathbf{0.40}_{\pm \mathbf{0.18}}$
	Other	$0.39_{\pm 0.12}$	$0.40_{\pm 0.12}$	$0.73_{\pm 0.14}$	$0.50_{\pm 0.18}$	$0.48_{\pm 0.17}$
	None	$0.39_{\pm 0.13}$	$0.41_{\pm 0.13}$	$0.74_{\pm 0.15}$	$0.50_{\pm 0.19}$	$0.48_{\pm 0.17}$
Random	Same	$\mathbf{0.17}_{\pm \mathbf{0.10}}$	$\mathbf{0.50}_{\pm \mathbf{0.12}}$	$\mathbf{0.24}_{\pm \mathbf{0.09}}$	$\mathbf{0.44}_{\pm \mathbf{0.22}}$	$\mathbf{0.42}_{\pm \mathbf{0.21}}$
	Other	$0.66_{\pm 0.15}$	$0.72_{\pm 0.14}$	$0.28_{\pm 0.10}$	$0.46_{\pm 0.22}$	$0.44_{\pm 0.20}$
	None	$0.69_{\pm 0.17}$	$0.75_{\pm 0.14}$	$0.30_{\pm 0.10}$	$0.46_{\pm 0.22}$	$0.46_{\pm 0.21}$

Table B.2: **[Invariance transfer under distribution shift for DenseNet-121 models.]**

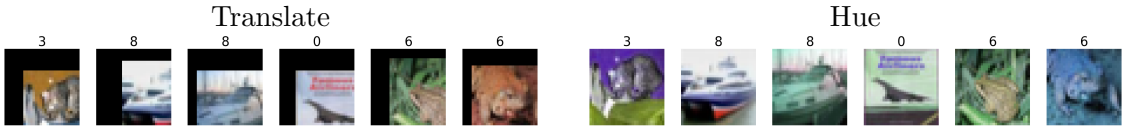
labeled as "Same" show how invariant models are to their training transformations on each of the datasets (*e.g.* a translation-trained model evaluated on translated data), whereas rows labeled as "Other" show how invariant they are to transformations other than the ones they were trained on (*e.g.* a translation-trained model on rotations). "None" rows show the baseline invariance of models trained without transformations, *i.e.* simply to classify untransformed objects. The most invariant models in each category are shown in bold. Results are computed over 10 runs that randomize the image and random objects, backgrounds, the way transformation values are sampled and the weight initialization of the models. The subscript numbers indicate the standard deviation of the *sens*-score over the 10 runs.

For all architectures, we observe very similar trends, which indicates that our observations about how models transfer invariance are robust. First, training models to be invariant to specific transformations indeed makes them more invariant to primarily those transformations. The invariance on the training transformations ("Same" rows) is significantly higher than for other transformations ("Other" rows), and also higher than the invariance of the baseline models trained without transformations ("None") rows.

Second, models can transfer a medium to high degree of invariance to out-of-distribution data. The invariance of models trained on image data is similar to that on other image data with different foreground objects ("Mild OOD"), and the same is roughly true for models trained on random data when transferring to image data ("Strong OOD"). Both types of models are less invariant on



(a) Examples of images from the synthetic random dataset.



(b) Examples of augmented CIFAR-10 images.

Figure B.11: [**Examples of images used for the OOD analysis.**]. The top part shows examples of images from the synthetic random dataset, while the bottom part shows examples of augmented CIFAR-10 images. Each row shows the examples for a single transformation.

Model Type	Transformation Relationship	Synthetic Datasets			Real-World Datasets	
		In-Distribution	Mild OOD	Strong OOD	CIFAR-10	CIFAR-100
Image	Same	$\mathbf{0.22_{\pm 0.09}}$	$\mathbf{0.26_{\pm 0.11}}$	$\mathbf{0.52_{\pm 0.18}}$	$\mathbf{0.36_{\pm 0.14}}$	$\mathbf{0.33_{\pm 0.13}}$
	Other	$0.41_{\pm 0.17}$	$0.41_{\pm 0.16}$	$0.64_{\pm 0.19}$	$0.48_{\pm 0.17}$	$0.46_{\pm 0.17}$
	None	$0.43_{\pm 0.19}$	$0.42_{\pm 0.17}$	$0.65_{\pm 0.19}$	$0.48_{\pm 0.18}$	$0.47_{\pm 0.18}$
Random	Same	$\mathbf{0.28_{\pm 0.13}}$	$\mathbf{0.43_{\pm 0.15}}$	$\mathbf{0.23_{\pm 0.11}}$	$\mathbf{0.31_{\pm 0.14}}$	$\mathbf{0.30_{\pm 0.14}}$
	Other	$0.61_{\pm 0.20}$	$0.64_{\pm 0.19}$	$0.34_{\pm 0.14}$	$0.45_{\pm 0.18}$	$0.43_{\pm 0.18}$
	None	$0.63_{\pm 0.20}$	$0.67_{\pm 0.19}$	$0.34_{\pm 0.15}$	$0.46_{\pm 0.19}$	$0.45_{\pm 0.19}$

Table B.3: [**Invariance transfer under distribution shift for ViT models.**]

(unseen) random data ("Strong OOD" for the image models, "Mild OOD" for the random models), but they are still more invariant to their training transformations than to other transformations, and also more invariant than the baseline models trained without transformations. Both types of models achieve high invariance on the real-world CIFAR-10 and CIFAR-100 datasets. These results indicate that models can retain a high degree of invariance, even when the data that they are evaluated on is substantially different from the data they were trained on. This helps to explain why invariance is such an important property for transfer learning: it can be transferred well to out-of-distribution data.

Third, the models trained on image and random data achieve similar invariance on the real-world CIFAR datasets. This indicates that for learning representations that are invariant to certain transformations, the specific features of the data do not matter as much. The caveat is that this approach only works if the features in the dataset are suitable to express the desired transformations. The pixel-level transformations in the Transforms-2D datatset can be applied to random data, but for higher-level transformations, *e.g.* transformations of the pose of an animal, the dataset needs to contain features that can express the transformation, *e.g.* animals with certain poses.

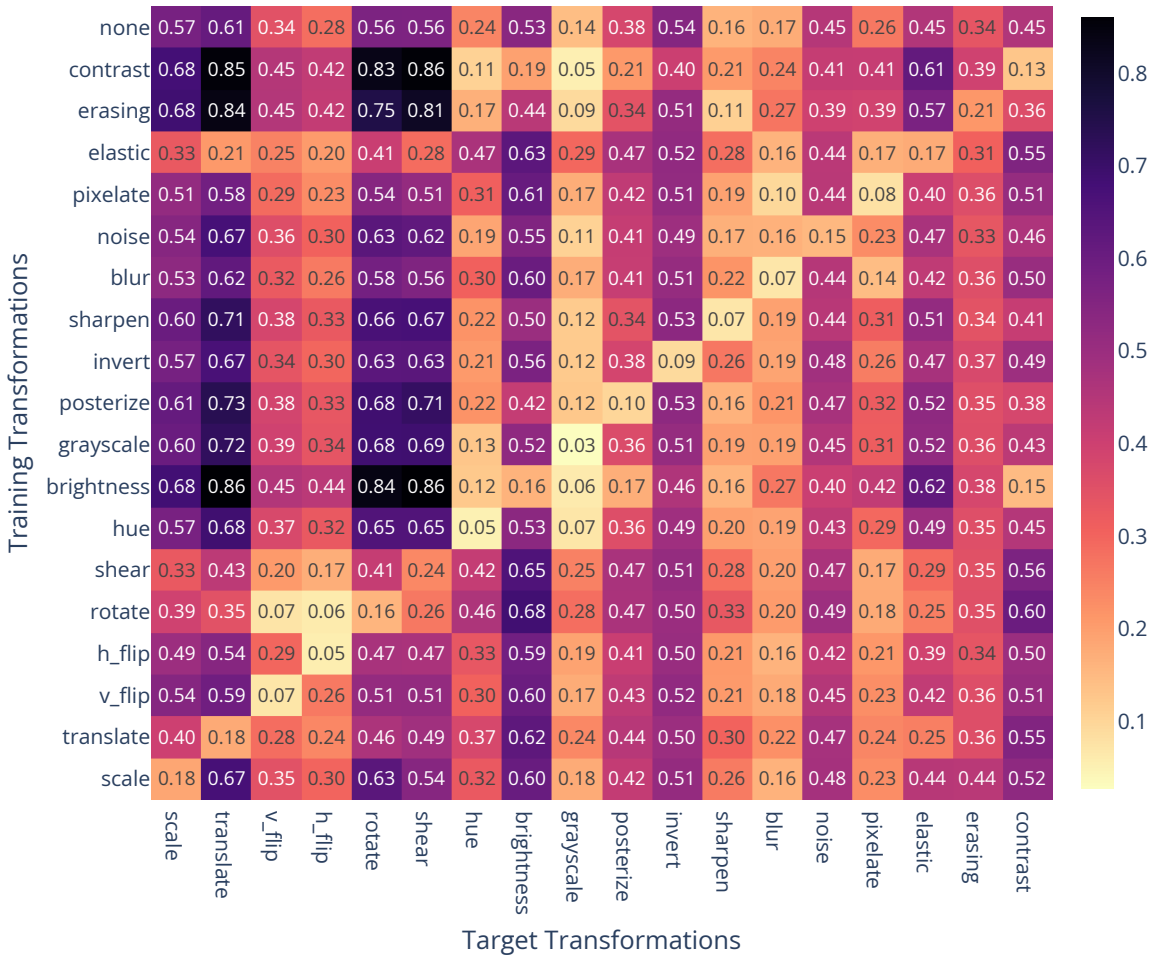


Figure B.12: **[Invariance transfer between transformations, for models trained on image data.]** Rows show how invariant models trained to be invariant to a specific transformation are to each of the transformations. Columns show how invariant each model is to the specific target transformation. Values are computed using the *sens*-metric (see Section 4.1.4). Lower numbers indicate higher invariance. Models trained with a specific transformation in their training data become more invariant to this transformation. We show the invariance of each model on the test set of its training dataset type ("In-Distribution").

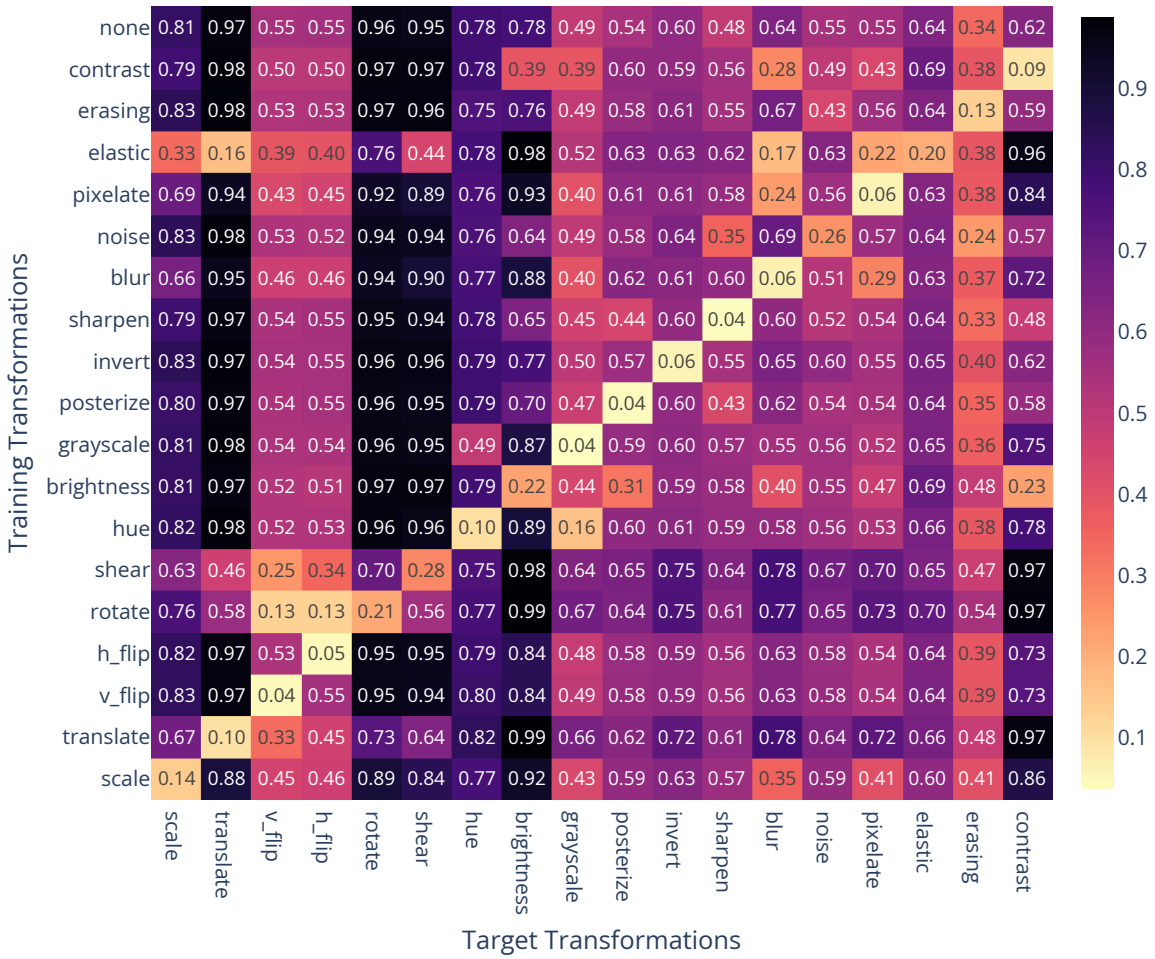
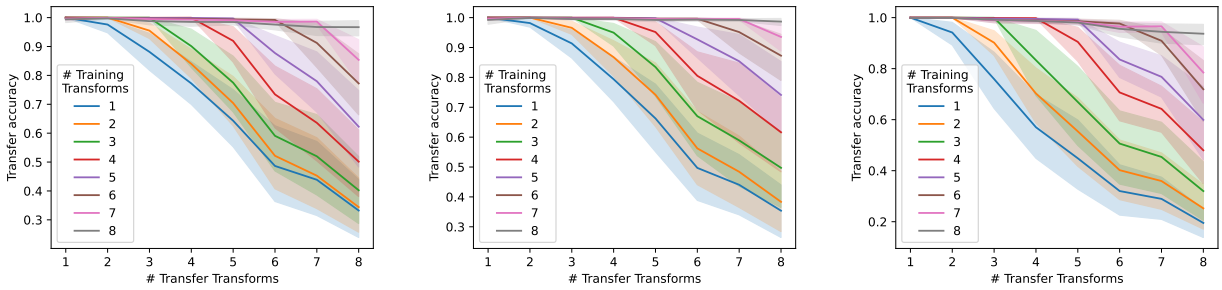


Figure B.13: [**Invariance transfer between transformations, for models trained on random data.**] Rows show how invariant models trained to be invariant to a specific transformation are to each of the transformations. Columns show how invariant each model is to the specific target transformation. Values are computed using the *sens*-metric (see Section 4.1.4). Lower numbers indicate higher invariance. Models trained with a specific transformation in their training data become more invariant to this transformation. We show the invariance of each model on the test set of its training dataset type ("In-Distribution").

Invariance Transfer Results Between Different Transformations. We show a breakdown of the invariance transfer on a per-transformation basis, on the training distributions (objects and backgrounds) of the image and random models for ResNet-18 architectures in Figures B.12 and B.13, respectively. For each transformation, we show the invariance of models trained to be invariant to that transformation, *i.e.* trained to classify objects modified by that transformation, for each of the other transformations. The results show that models indeed become primarily invariant to their training transformations (low values on the diagonal). However, we can also observe some "spillovers", *e.g.* training models to be rotation- and shear-invariant also increases the invariance to horizontal and vertical flips (but not vice versa), and training for hue- and grayscale-invariance increases the invariance to the respective other transformation.

B.4.3 Additional Results on Invariance Mismatch



(a) VGG-11 (b) DenseNet-121 (c) ViT

Figure B.14: [**Models trained on nested sets of transformations and evaluated on datasets with super- and subsets of those transformations.**]. Models trained on data with the same set or a superset of transformations as the target dataset consistently achieve almost 100% accuracy. However, models trained with only a subset of the transformations show considerably lower performance that decreases the smaller the subset of training transformations is compared to the target task. The results show that learning a superset of required invariances does not harm transfer performance but that missing required invariances degrades transfer performance.

Figure B.14 shows the results analogous to those in Section 4.3.2 for invariance mismatch between training and transfer tasks, for addtional model families: VGG-11, DenseNet-121 and ViT models. We observe the same pattern as in Section 4.3.2, *i.e.* models trained to be invariant to the same set or a superset of transformations as those in the target dataset consistently achieve almost 100% accuracy, but models that are missing invariance to transformations in the target dataset achieve significantly lower performance.

Appendix C

Additional Material on Understanding Memorization in Large Language Models with Random Strings

C.1 Additional details on the experimental setup

C.1.1 Technical details on the training setup

Models: In this chapter, we use pretrained models of the Pythia [Biderman et al., 2023b], Phi [Li et al., 2023b] and Llama2 [Touvron et al., 2023] families. For the Pythia family, we use variants with 70M, 1B and 12B parameters, for the Phi family we use 1.3B and 2.7B parameter variants, and for Llama-2 we use 7B and 13B parameter variants. We choose these models, since they represent popular, modern architectures, and span a wide spectrum of parameter counts (more than two orders of magnitude).

In addition to the above, we also use GPT-2 [Radford et al., 2019] and OPT Zhang et al. [2022] models for some experiments, to study the effect of absolute position encodings. In particular, we use GPT2-140M (GPT-2) and GPT2-1.5B (GPT-2-XL) parameter variants of GPT-2 and the OPT-350M model. Pretrained versions for all models are publicly available on the Huggingface Model Hub.

Training: We train models to minimise the cross-entropy loss over string s. We define the cross-entropy loss of a model $\mathcal{M}$ on string s as follows:

$$\text{Loss}(\mathcal{M}, s) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{t\in V}\delta(s_i = t)\log P_{\mathcal{M}}(s_i = t|s_{[1,i-1]}). \tag{C.1}$$

In most experiments on pretrained models we train models on random strings for 100 epochs (for single strings, each step is an epoch), with a linearly decaying learning rate schedule. Untrained models memorize more slowly, so we train them for 300 epochs in most cases. For Pythia-70M, Phi-1.3B and Phi-2.7B we use an initial learning rate of $5 * 10^{-5}$, for OPT-350M 10^{-4}, for GPT2-124M $5 * 10^{-4}$ and for all other models 10^{-5}. These learning rate values resulted in the fastest convergence during a grid search over values from 10^{-3} to 10^{-6}.

C.1.2 Examples of random strings used in this chapter

Table C.1 shows examples of random token strings used in this chapter. Each character is tokenized individually.

Alphabet and distribution	Tokens
2 characters, uniform	bbabbabbababbabaaabbababaaaababb
4 characters, uniform	cdbccbddbcaddbcabaccbcbcabaacadd
7 characters, uniform	efceecffdeaggdebbbffddbdabaafaff
13 characters, uniform	hleijdjkfibllfhcdcjjghdgbdaajakk
26 characters, uniform	pwjqshtulrcxxlpegessmognchaatauv
26 characters, H2	aaaaaaaaaagaaaaaaaaaaaaaaaaaaaa
26 characters, H4	alaaaabfaaaroaaaaaaaaaaaaaaaadj
26 characters, H7	bqadhakmagausabaaaiiaaaaaaaajalp
26 characters, H13	taknapqbmawvbjaaaoodgafaaaaoaqsa

Table C.1: **[Examples of random strings used in this chapter.]** We show the first 32 tokens/characters.

We also use non-Latin alphabets in Appendix C.2.2. An example of such a random chosen alphabet for $\ell = 26$, from the Pythia tokenizer, is the following: “Ġecosystem”, “281”, “Ġredistribute”, “ĠEurope”, “eni”, “ricted”, “Meanwhile”, “Ġpropensity”, “.”””, “Du”, “ĠAlice”, “ortical”, “Ġultrasonic”, “Ġinclud”, “Blocks”, “thur”, “Ġyears”, “ramento”, “ashion”, “)}$$”, “onical”, “Beck”, “].)”, “Ġpendant”, “uma”, “ynote”

C.1.3 Computational resources

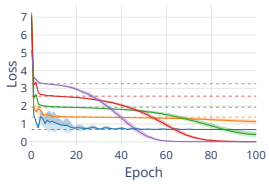
All experiments were conducted on machines in an in-house cluster with 2 x NVIDIA A40 GPUs with 48GB of memory, and with NVIDIA 2 x or 8 x A100 GPUs with 80GB of memory. We use the A40 machines for training smaller models, such as Pythia-1B, and also Phi-2.7B with main memory offloading. We use the A100 machines for training larger models, such as Pythia-12B, and Llama2-7B and Llama2-13B, and for some Phi-2.7B training runs. It is possible to run most experiments, except the ones using larger batch sizes in Appendix C.4.1, on a single GPU, possibly using main memory offloading for larger models.

The experiments on the memorization dynamics in Sections 5.2 and 5.3, including the corresponding results reported in the appendix, used around 1100 GPU hours. The experiments on the role of local prefixes and global context in Section 5.4 used around 900 GPU hours. The experiments on sequential memorization in Section 5.5 used around 500 GPU hours. In total, the experiments in the chapter used around 2500 GPU hours, distributed over the different GPU types.

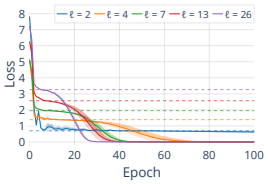
C.2 Additional results for the memorization dynamics

C.2.1 Additional models and metrics

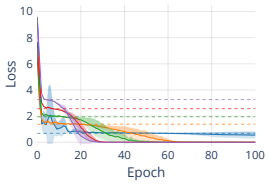
We show results for additional models for the experiments in Sections 5.2 and 5.3. Memorization dynamics are shown for different alphabet sizes ℓ and entropy levels h. In Figures C.1 and C.2 we show training loss, in Figures C.3 and C.4 we show accuracy, in Figures C.5 and C.6 we show aggregate probabilities over A, in Figures C.7 and C.8 we show entropy over A, and in Figures C.9 and C.10 we show the KLD of $\mathcal{M}$'s distribution $P_{\mathcal{M}}$ over A from the true distribution P_A.



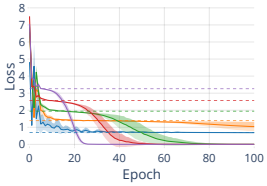
(a) Pythia-70M



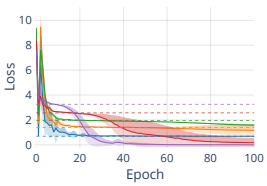
(b) Pythia-1B



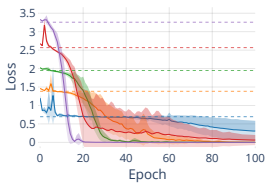
(c) Pythia-12B



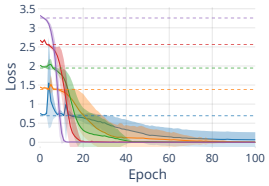
(d) Phi-1.3B



(e) Phi-2.7B

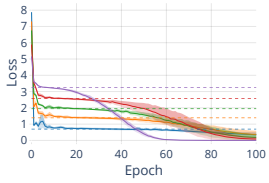


(f) Llama2-7B

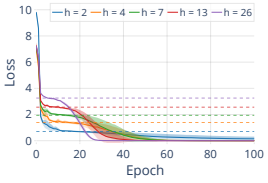


(g) Llama2-13B

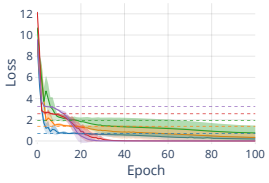
Figure C.1: **[Loss for all models for different ℓ. ($n = 1024$)]**



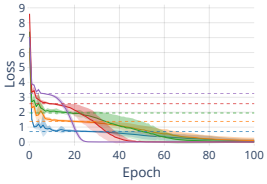
(a) Pythia-70M



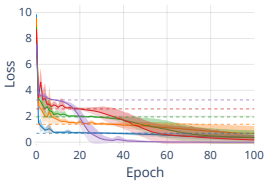
(b) Pythia-1B



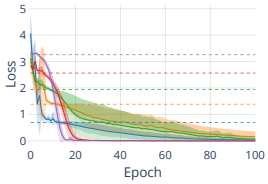
(c) Pythia-12B



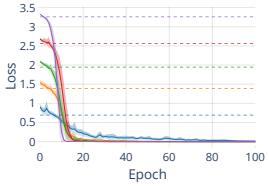
(d) Phi-1.3B



(e) Phi-2.7B

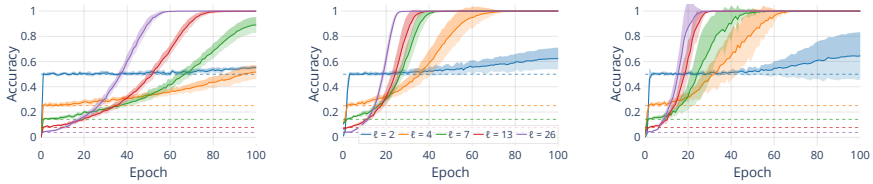


(f) Llama2-7B

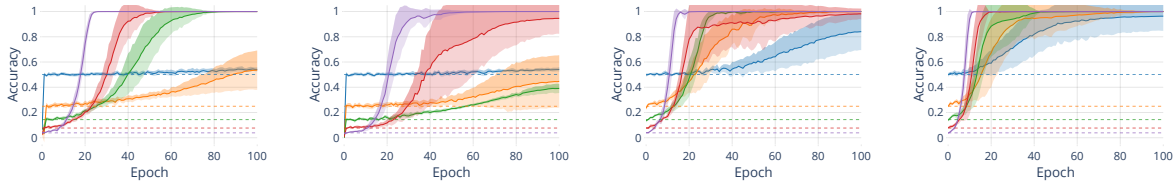


(g) Llama2-13B

Figure C.2: **[Loss for all models for different h. ($n = 1024, \ell = 26$)]**

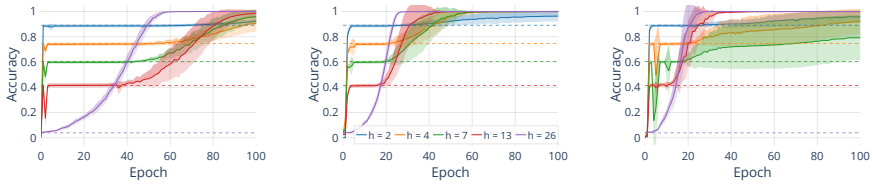


(a) Pythia-70M (b) Pythia-1B (c) Pythia-12B

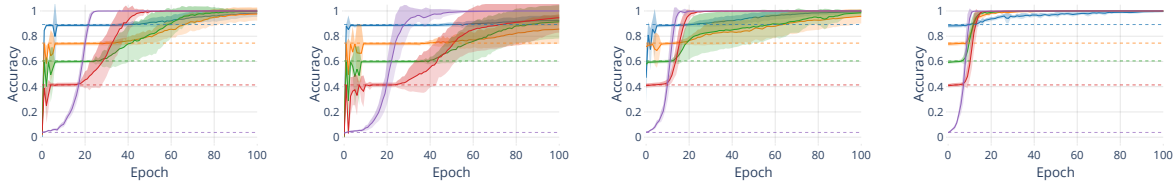


(d) Phi-1.3B (e) Phi-2.7B (f) Llama2-7B (g) Llama2-13B

Figure C.3: [**Accuracy for all models for different ℓ.** ($n = 1024$)]

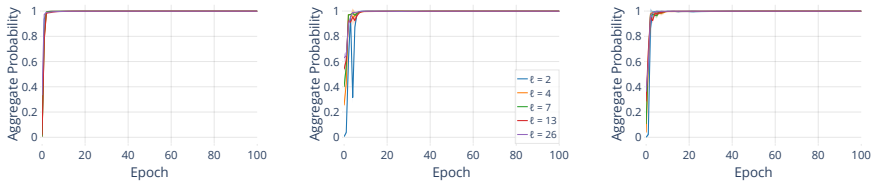


(a) Pythia-70M (b) Pythia-1B (c) Pythia-12B

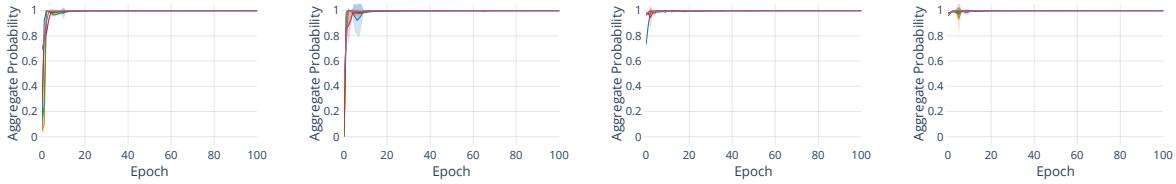


(d) Phi-1.3B (e) Phi-2.7B (f) Llama2-7B (g) Llama2-13B

Figure C.4: [**Accuracy for all models for different h.** ($n = 1024, \ell = 26$)]

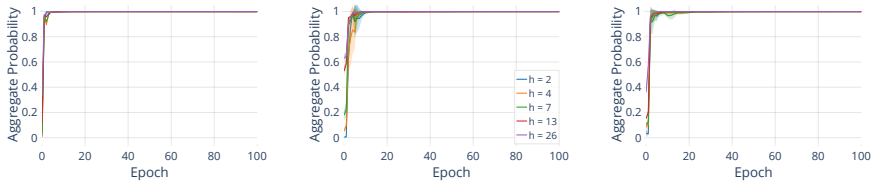


(a) Pythia-70M (b) Pythia-1B (c) Pythia-12B

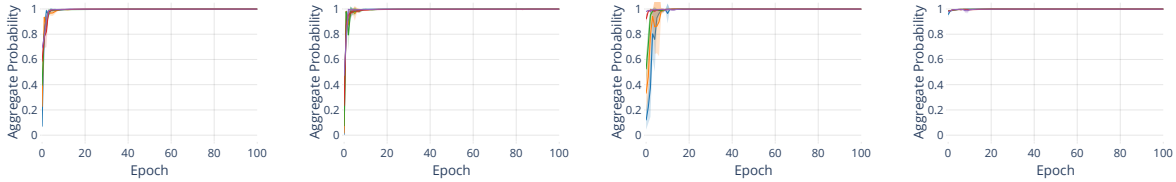


(d) Phi-1.3B (e) Phi-2.7B (f) Llama2-7B (g) Llama2-13B

Figure C.5: [**Aggregate Probability over A for all models for different ℓ. ($n = 1024$)**]

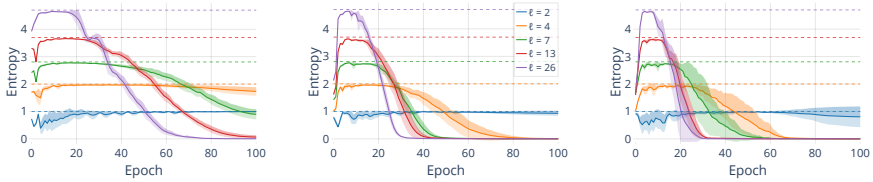


(a) Pythia-70M (b) Pythia-1B (c) Pythia-12B

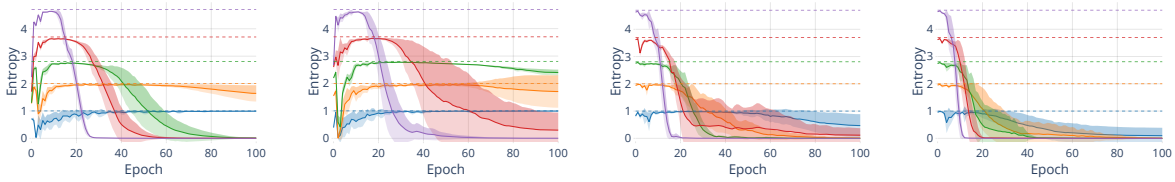


(d) Phi-1.3B (e) Phi-2.7B (f) Llama2-7B (g) Llama2-13B

Figure C.6: [**Aggregate Probability over A for all models for different h. ($n = 1024, \ell = 26$)**]

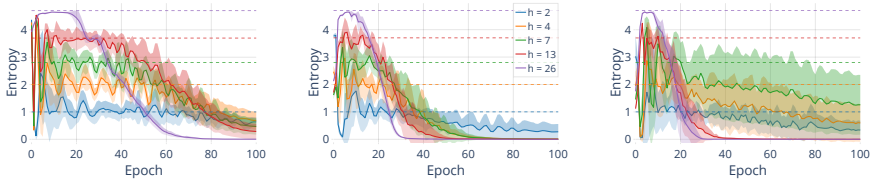


(a) Pythia-70M (b) Pythia-1B (c) Pythia-12B

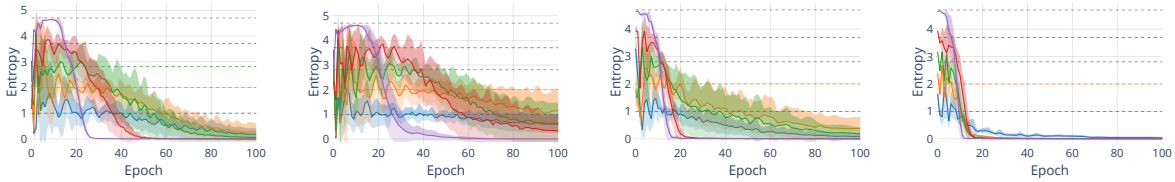


(d) Phi-1.3B (e) Phi-2.7B (f) Llama2-7B (g) Llama2-13B

Figure C.7: [**Entropy for all models for different ℓ. ($n = 1024$)**]

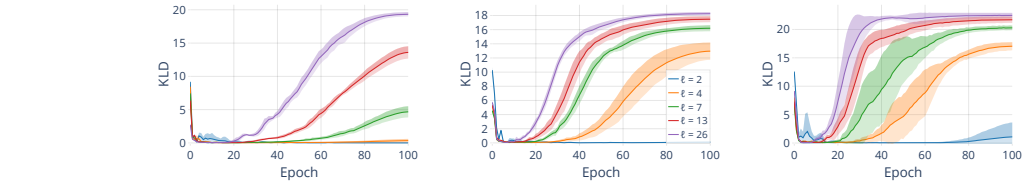


(a) Pythia-70M (b) Pythia-1B (c) Pythia-12B

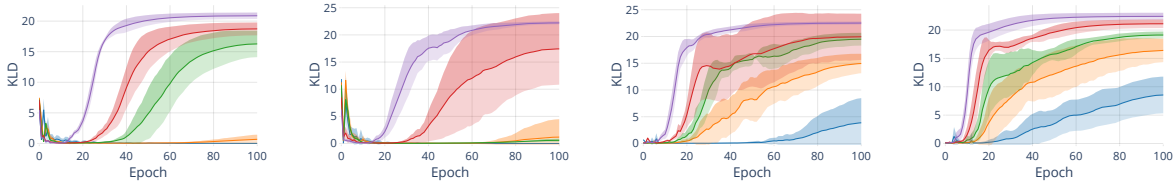


(d) Phi-1.3B (e) Phi-2.7B (f) Llama2-7B (g) Llama2-13B

Figure C.8: [**Entropy for all models for different h. ($n = 1024, \ell = 26$)**]

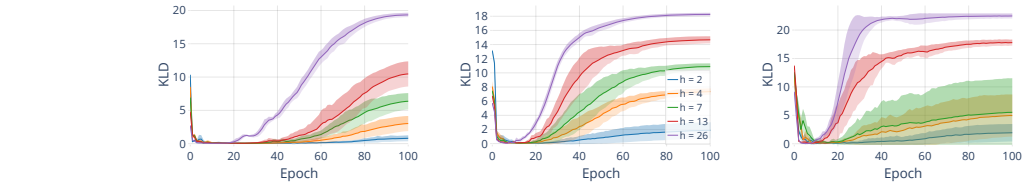


(a) Pythia-70M (b) Pythia-1B (c) Pythia-12B

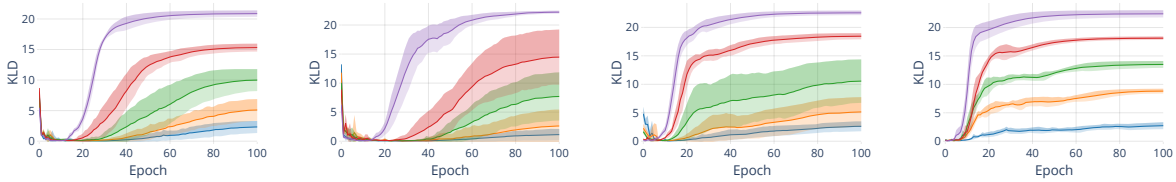


(d) Phi-1.3B (e) Phi-2.7B (f) Llama2-7B (g) Llama2-13B

Figure C.9: [**KL-Divergence from the true distribution for all models for different ℓ. ($n = 1024$)**] We compute $D_{KL}(P_A||P_{\mathcal{M}})$. The dip during the *Guessing-Phase* to 0 shows that the models, in fact, approximate the string's true distribution P_A.



(a) Pythia-70M (b) Pythia-1B (c) Pythia-12B

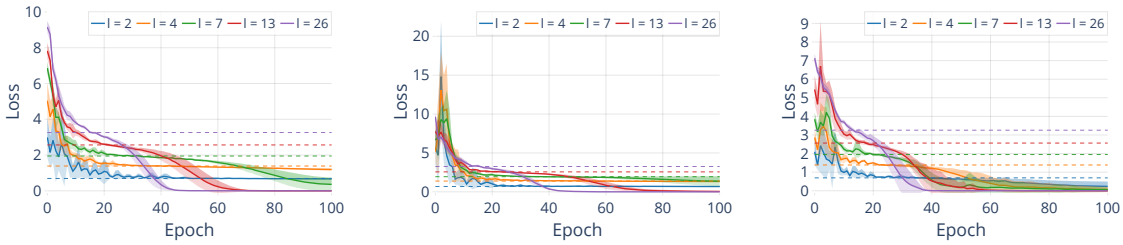


(d) Phi-1.3B (e) Phi-2.7B (f) Llama2-7B (g) Llama2-13B

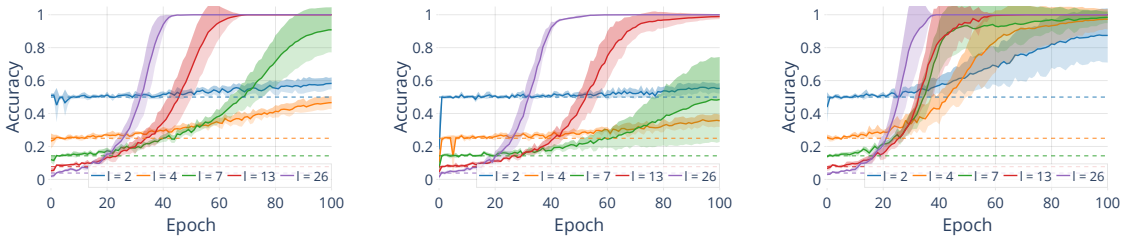
Figure C.10: [**KL-Divergence from the true distribution for all models for different h. ($n = 1024, \ell = 26$)**] We compute $D_{KL}(P_A||P_{\mathcal{M}})$. The dip during the *Guessing-Phase* to 0 shows that the models, in fact, approximate the string's true distribution P_A.

C.2.2 Results for non-Latin alphabets

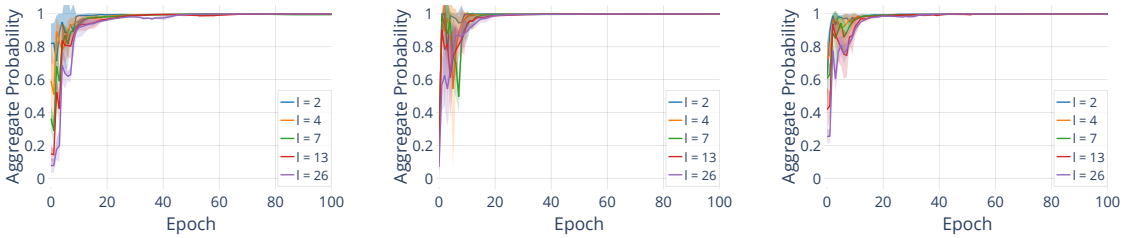
In Figure C.11 we show ablation results for different models and ℓ for non-Latin alphabets, i.e. with ℓ tokens chosen randomly from entire token vocabulary V. We observe the same patterns as for the Latin alphabets shown in Sections 5.2 and 5.3 and in Appendix C.2.1. The only difference is that Llama2-13B also exhibits a *Guessing-Phase* in this context.



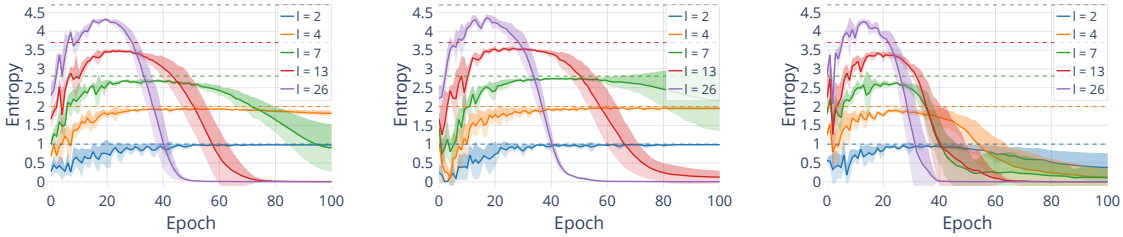
(a) Pythia-1B, Loss (b) Phi-2.7B, Loss (c) Llama2-13B, Loss



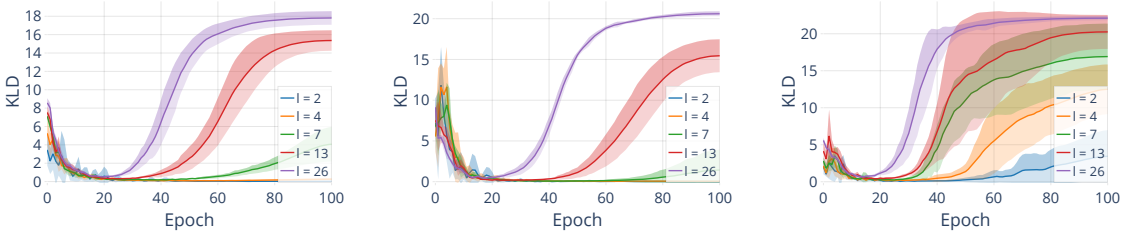
(d) Pythia-1B, Accuracy (e) Phi-2.7B, Accuracy (f) Llama2-13B, Accuracy



(g) Pythia-1B, Cumulative Probability (h) Phi-2.7B, Cumulative Probability (i) Llama2-13B, Cumulative Probability



(j) Pythia-1B, Entropy (k) Phi-2.7B, Entropy (l) Llama2-13B, Entropy

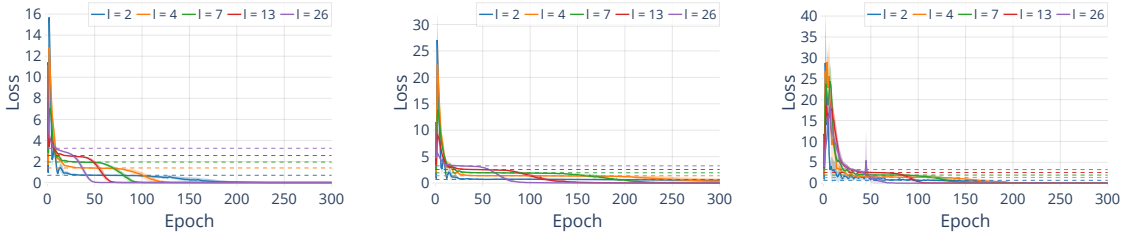


(m) Pythia-1B, KL-Divergence (n) Phi-2.7B, KL-Divergence (o) Llama2-13B, KL-Divergence

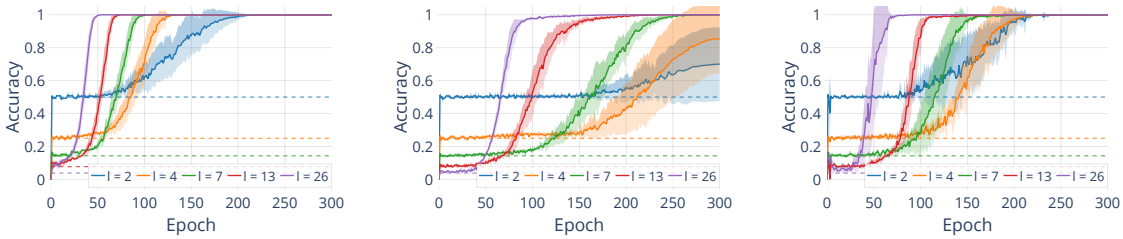
Figure C.11: **[Accuracy, loss, cumulative probability, entropy and KLD for different ℓ with non-latin alphabets. ($n = 1024$)]** We choose ℓ tokens randomly from entire token vocabulary V instead of using lowercase Latin letters. We observe the same patterns as for the Latin alphabets shown in Sections 5.2 and 5.3 and in Appendix C.2.1. However, Llama2-13B also exhibits a *Guessing-Phase* in this context.

C.2.3 Results for untrained models

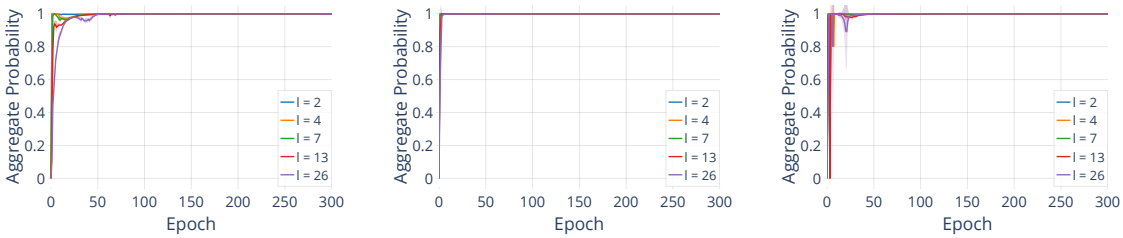
In Figure C.12 we show ablations for untrained, *i.e.* non-pretrained models. Again, we see the same patterns as for the pretrained models shown in Sections 5.2 and 5.3 and in Appendix C.2.1, although convergence is generally slower. Note that we show memorization dynamics over 300, instead of 100 epochs.



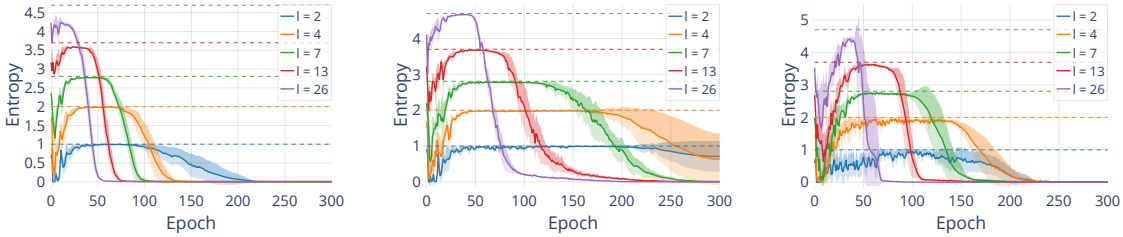
(a) Pythia-1B, Loss (b) Phi-2.7B, Loss (c) Llama2-13B, Loss



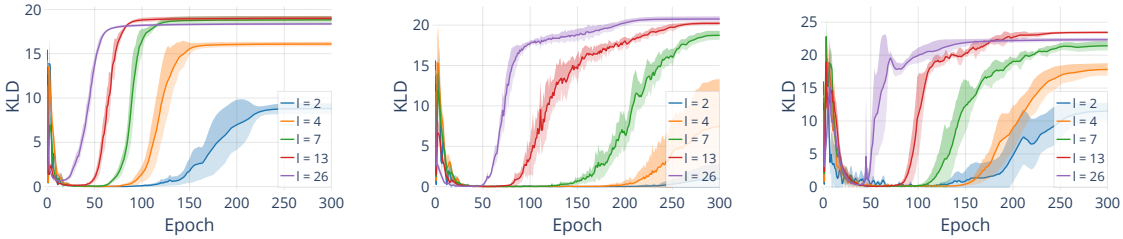
(d) Pythia-1B, Accuracy (e) Phi-2.7B, Accuracy (f) Llama2-13B, Accuracy



(g) Pythia-1B, Cumulative Probability (h) Phi-2.7B, Cumulative Probability (i) Llama2-13B, Cumulative Probability



(j) Pythia-1B, Entropy (k) Phi-2.7B, Entropy (l) Llama2-13B, Entropy



(m) Pythia-1B, KL-Divergence (n) Phi-2.7B, KL-Divergence (o) Llama2-13B, KL-Divergence

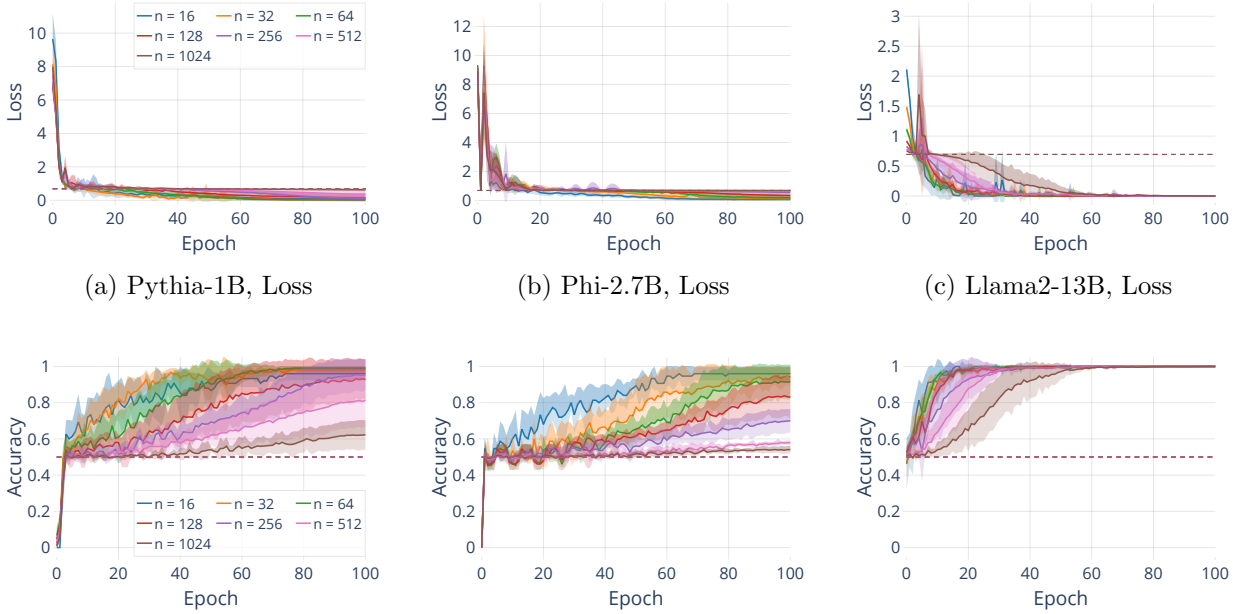
Figure C.12: [**Accuracy, loss, cumulative probability, entropy and KLD for different ℓ with non-pretrained models. ($n = 1024$)**] We observe the same patterns for untrained models as for the pretrained models shown in Sections 5.2 and 5.3 and in Appendix C.2.1, although convergence is generally slower. Note that we show memorization dynamics over 300, instead of 100 epochs here.

C.2.4 Additional results on string length, string partitions, and repeated substrings

We show results for $\ell = 2$ and $\ell = 26$. In Figures C.13 and C.14 we show the effect of string length on the memorization dynamics. Shorter strings are memorized faster than longer ones.

In Figures C.15 and C.16 we show results for partitioning the same $n = 1024$ token string into $k \in \{1, 2, 4, 8, 16, 32, 64\}$ pieces and memorising them in a batch. Whether the string is memorized in one long piece or as multiple shorter fragments does barely affect memorization speed.

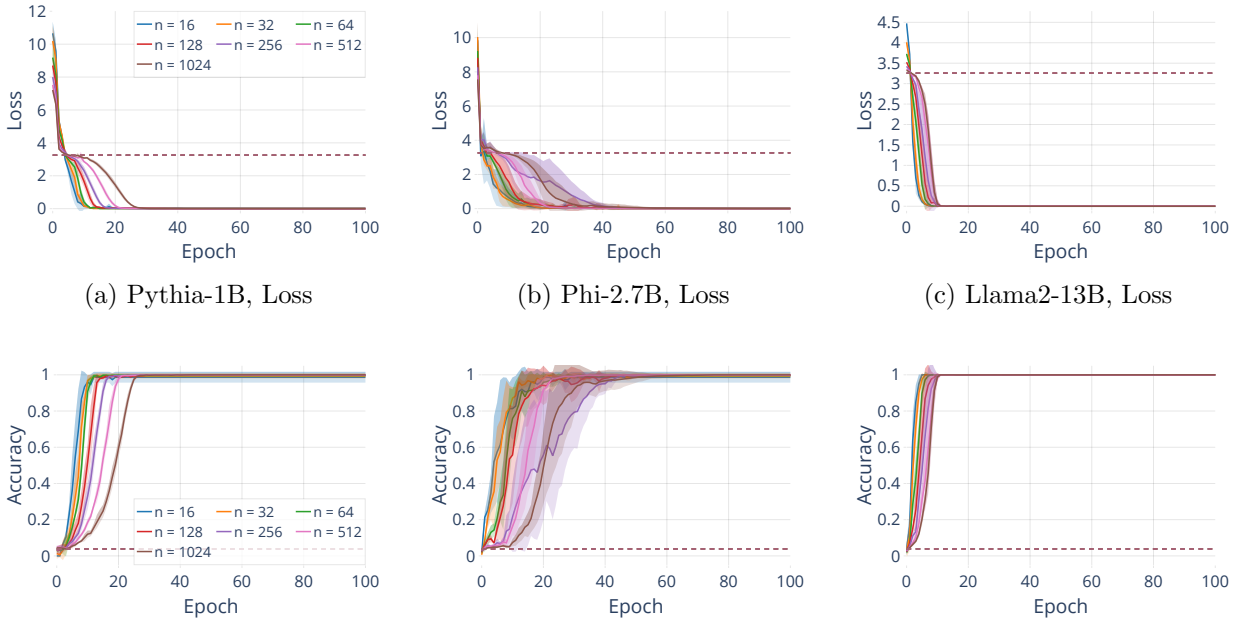
In Figures C.17 and C.18 we show results for sampling a shorter substring of length $u \in \{16, 32, 64, 128, 256, 512, 1024\}$ and then repeating it n/u times to create the full $n = 1024$ token string. The results show that the memorization speed strongly depends on u and not on n, indicating that what matters is the fraction of unique tokens resp. independently sampled tokens in the string. Repetitions of the same random string do not increase memorization speed. Furthermore, the accuracy plots show that the initial accuracy of the models at epoch 0, before they are trained, are at values $1 - (u/n)$, which means that for a small u, resp. many repetitions of the unique substring, the model can predict most of the string correctly. This can only happen because the models uses in-context learning to predict tokens in subsequent occurrences of the substrings.



(a) Pythia-1B, Loss (b) Phi-2.7B, Loss (c) Llama2-13B, Loss

(d) Pythia-1B, Accuracy (e) Phi-2.7B, Accuracy (f) Llama2-13B, Accuracy

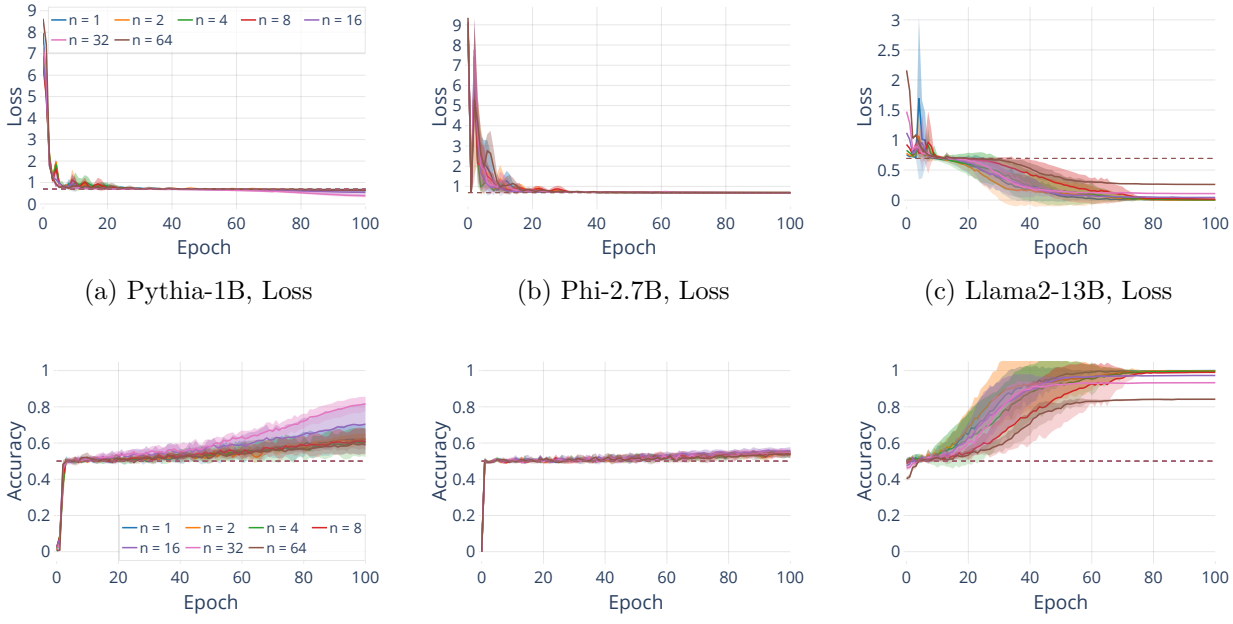
Figure C.13: [**Accuracy and loss for different string lengths** n**.** $(\ell = 2)$]



(a) Pythia-1B, Loss (b) Phi-2.7B, Loss (c) Llama2-13B, Loss

(d) Pythia-1B, Accuracy (e) Phi-2.7B, Accuracy (f) Llama2-13B, Accuracy

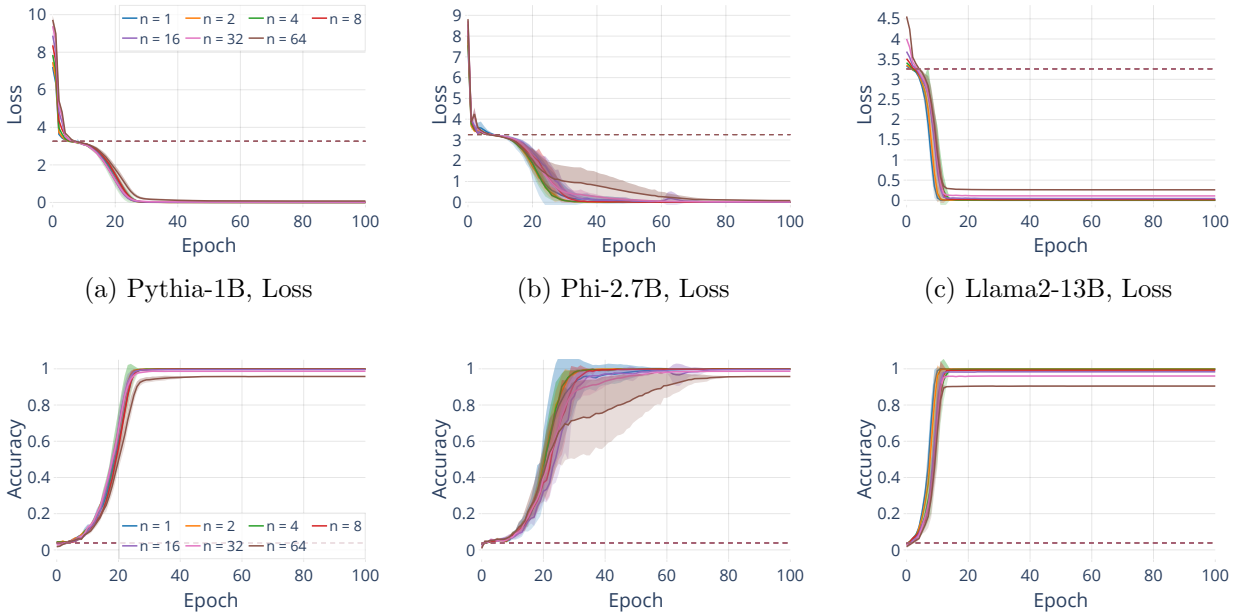
Figure C.14: [**Accuracy and loss for different string lengths** n**.** ($\ell = 26$)]



(a) Pythia-1B, Loss (b) Phi-2.7B, Loss (c) Llama2-13B, Loss

(d) Pythia-1B, Accuracy (e) Phi-2.7B, Accuracy (f) Llama2-13B, Accuracy

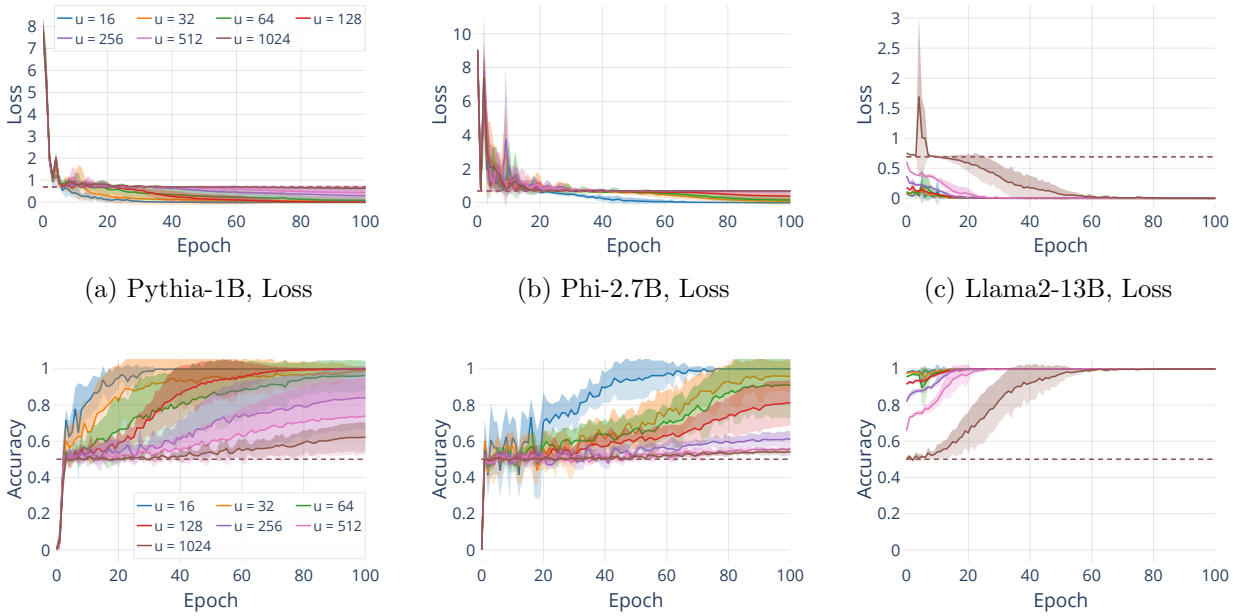
Figure C.15: [**Accuracy and loss for different partitions of the same string.** ($n = 1024, \ell = 2$)]



(a) Pythia-1B, Loss (b) Phi-2.7B, Loss (c) Llama2-13B, Loss

(d) Pythia-1B, Accuracy (e) Phi-2.7B, Accuracy (f) Llama2-13B, Accuracy

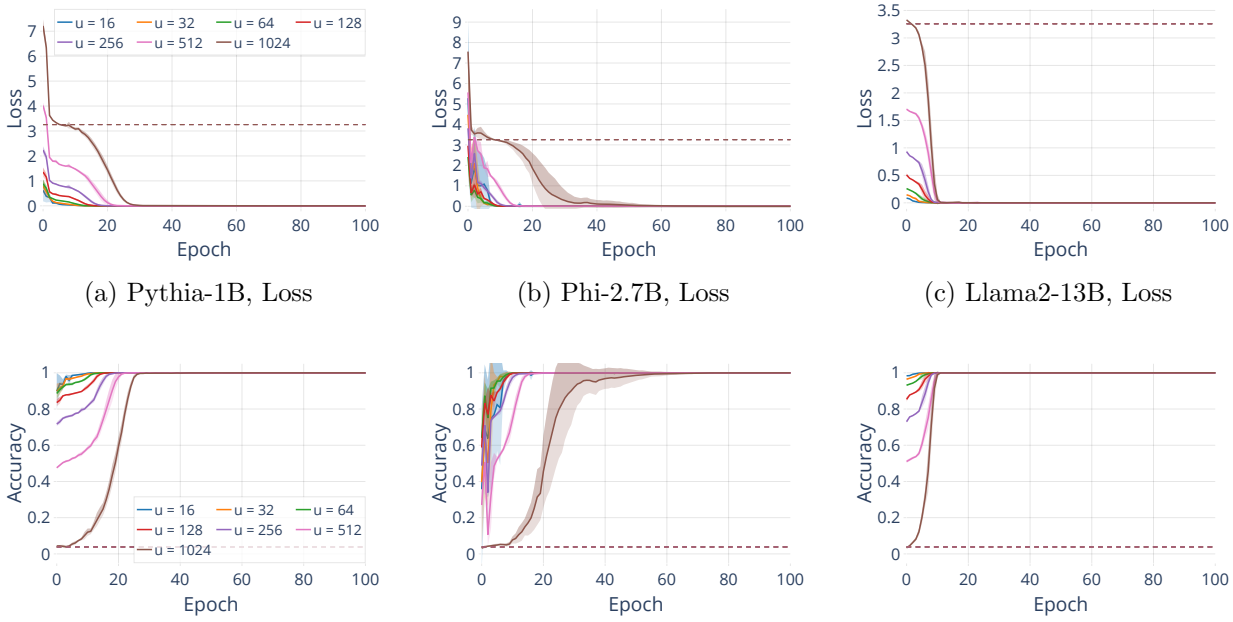
Figure C.16: **[Accuracy and loss for different partitions of the same string.** ($n = 1024, \ell = 26$)]



(a) Pythia-1B, Loss (b) Phi-2.7B, Loss (c) Llama2-13B, Loss

(d) Pythia-1B, Accuracy (e) Phi-2.7B, Accuracy (f) Llama2-13B, Accuracy

Figure C.17: [**Accuracy and loss for different sizes of unique substrings u. ($n = 1024, \ell = 2$)**] We sample u tokens independently and then repeat the resulting substring n/u times to create the $n = 1024$ token string. Repetitions of the same random string do not increase memorization speed. Additionally, the accuracy at epoch 0 before any training is higher, the smaller u, indicating that the models use in-context learning to predict subsequent occurrences of the substrings without having memorized them.



(a) Pythia-1B, Loss (b) Phi-2.7B, Loss (c) Llama2-13B, Loss

(d) Pythia-1B, Accuracy (e) Phi-2.7B, Accuracy (f) Llama2-13B, Accuracy

Figure C.18: **[Accuracy and loss for different sizes of unique substrings u. ($n = 1024, \ell = 26$)]** We sample u tokens independently and then repeat the resulting substring n/u times to create the $n = 1024$ token string. Repetitions of the same random string do not increase memorization speed. Additionally, the accuracy at epoch 0 before any training is higher, the smaller u, indicating that the models use in-context learning to predict subsequent occurrences of the substrings without having memorized them.

C.2.5 Results on conditional probability strings

We know that the entropy of a string affects how hard it is for models to memorize it. To obtain a better understanding about how entropy interacts with the memorability of a string, we test how conditional entropy affects memorability. In order to do so, we construct strings with the same unconditional entropy, but different levels of conditional entropy.

By the n-conditional entropy $H_n(s)$ of a string s, we refer to the entropy

$$H_n(s) = H(s_i | s_{i-n}, \ldots, s_{i-1}),$$

i.e. the entropy over tokens in s, that the preceding n tokens, i.e. the preceding n-gram is known. We are interested in knowing whether at the same level of unconditional entropy $H(s)$, i.e. 0-conditional entropy, strings with different levels of n-conditional entropy $H_n(s)$ differ in their memorability.

Data construction:

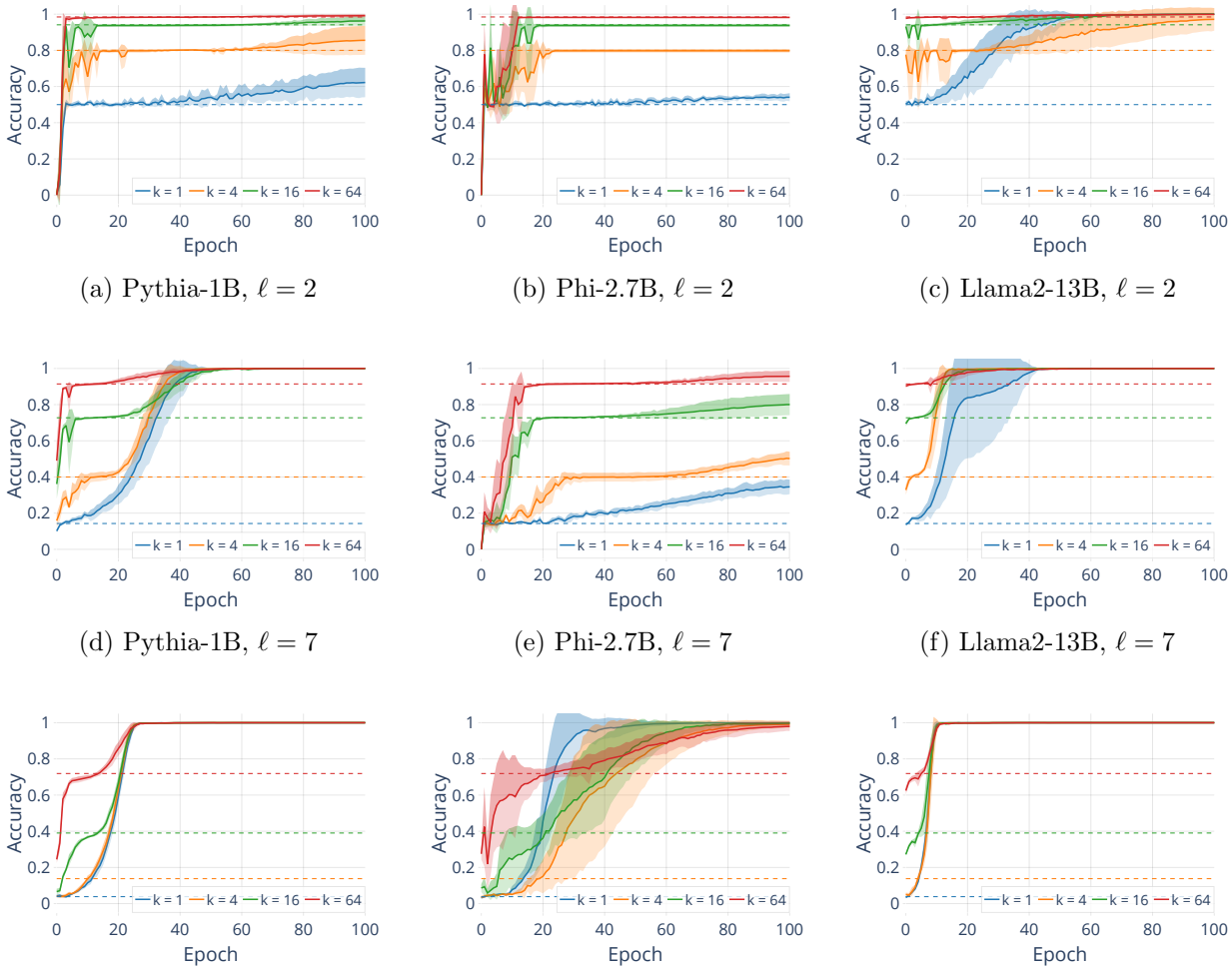
Privileged continuation tokens: We create string s with alphabet A with a certain level of n-conditional entropy by assigning each possible n-gram g over A a certain *privileged continuation token* t_g. E.g. for $A = \{a, b\}$, there are the 2-grams aa, ab, ba, bb, and each of them would have a privileged continuation, e.g. b for aa, a for ab, etc.

Constructing strings with different levels of conditional entropy: To sample string s, we first sample n tokens from A uniformly at random. To sample the next token s_i, we get its preceding n-gram $g = s_{i-n}, \ldots, s_{i-1}$, look up its privileged token t_g and then sample a token from A with $k\times$

relative probability $p_k = k * p_u$ for t_g, and uniform probability p_u for all other tokens $t \in A \setminus \{t_g\}$. I.e. we are k times more likely to sample the privileged token t_g as a continuation to g than the other tokens in A. We obtain p_k as $p_k = \frac{k}{|A|-1+k}$ and $p_u = \frac{1-p_k}{|A|-1}$. Increasing the relative probability p_k lowers the conditional entropy $H_n(s)$ of string s.

Ensuring the same level of unconditional entropy: To ensure that strings with different p_k have the same unconditional entropy $H(s)$, we ensure that each token $t \in A$ appears the same number of times as a privileged continuation token. I.e. for 1-grams, where there are $|A|$ combinations (single tokens from A), each $t \in A$ appears once as the privileged token of a 1-gram. For 2-grams, with $|A|^2$ possible combinations, each token appears A times as privileged token, etc. E.g. for 2-grams over $A = \{a, b\}$ a privileged token mapping $aa \to b, ab \to b, ba \to a, bb \to a$ would be valid, whereas the mapping $aa \to b, ab \to b, ba \to b, bb \to b$ would be not. Making each token appear the same number of times as privileged continuation ensures that the overall probability of each $t \in A$ is the same, and thus the unconditional entropy of the strings is the same.

As usual, we train models for 100 epochs to memorize strings with alphabets of different sizes (i.e. entropy levels) and record their memorization dynamics. We also compute the empirical unconditional and conditional entropy of the sampled strings.



(a) Pythia-1B, $\ell = 2$ (b) Phi-2.7B, $\ell = 2$ (c) Llama2-13B, $\ell = 2$

(d) Pythia-1B, $\ell = 7$ (e) Phi-2.7B, $\ell = 7$ (f) Llama2-13B, $\ell = 7$

(g) Pythia-1B, $\ell = 26$ (h) Phi-2.7B, $\ell = 26$ (i) Llama2-13B, $\ell = 26$

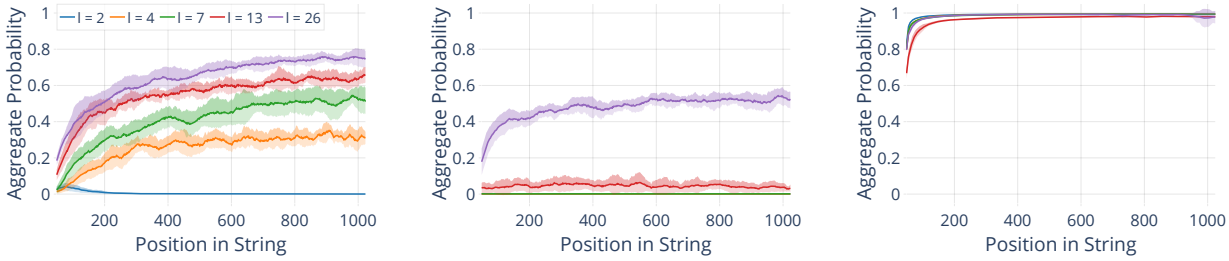
Figure C.19: **[Accuracy for conditional probability strings with different ℓ. ($n = 1024$)]** While the conditional entropy affects the accuracy models achieve during the *Guessing-Phase*, *i.e.* models learn the conditional probability distribution of the string, the length of the *Memorisation-Phase* is not affected by the conditional entropy.

We fix $n = 1$ and train and evaluate models on strings with different relative probabilities p_k, i.e. where the priviledged continuation tokens are k times as likely to appear after their n-grams than the ramining tokens from A. We use $k \in \{1, 4, 16, 64\}$. For $k = 1$ the unconditional entropy $H(s)$ is the same as the conditional entropy $H_1(s) = H(s)$.

Figure C.19 shows the memorization dynamics for different models and ℓ. Conditional entropy affects the accuracy models achieve during the *Guessing-Phase*, which means that models are able to learn the conditional probabilities of the strings. However, the length of the *Memorisation-Phase* is not affected by the conditional entropy, since all strings are fully memorized at roughly the same epoch, with no consistent relationship between relative probability and full memorization epoch. The effect of unconditional entropy, by comparison, is much stronger.

C.2.6 Additional results on in-context learning

We saw in Section 5.2 that all models exhibit two phases of memorization, except the Llama2 models, which skip the *Guessing-Phase* and start directly with the *Memorisation-Phase*. To better understand why this is happening, we test how well the different models are able to learn the distribution of the random strings P_A via in-context learning.



(a) Pythia-1B (b) Phi-2.7B (c) Llama2-13B

Figure C.20: [**Aggregate probability over A at different string positions, before training. ($n = 1024$)**] The aggregate probability over the tokens in the alphabet shows how well models are able to infer P_A from the prefix of the string at a given positions. Models differ in their ability to learn the distribution via in-context learning. Llama-2 models are particularly good, assigning almost all probability mass to tokens in the alphabet with just 100 tokens of context. Other models do not exhibit the same in-context learning abilities.

In Figure C.20 we show the aggregate probability over all tokens in the alphabet A that models assign at each position in the string. We use a sliding window of size 50 to smooth the curves. The figure shows models at epoch 0, *i.e.* before they have started to memorize the string. Without any training, models can only infer the string distribution, *i.e.* detect that the string only contains tokens from A and not other tokens from V, via in-context learning.

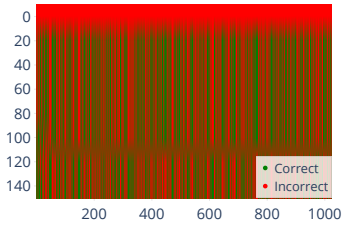
Indeed, we see that Llama2 models exhibit strong in-context learning abilities and quickly assign all probability mass to tokens within the alphabet. The other models are not able to infer the distribution nearly as well. Note that the differences in in-context learning ability observed in Figure C.20 correspond to the differences in the initial loss in Figure 5.1. Thus, the *Guessing-Phase* appears to be a stage that all models go through, but sufficiently strong in-context learning abilities allow models to effectively shorten it to zero.

C.3 Results on Memorization Order

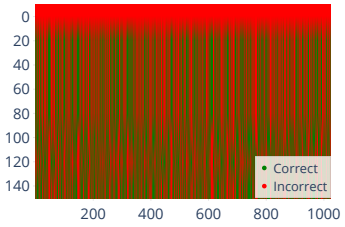
We aim to characterize the *Memorisation-Phase* more closely and ask: *Is there a specific ordering in which tokens are memorized or are the positions of the correctly recollected tokens random?*

C.3.1 Visualizing Memorization Order

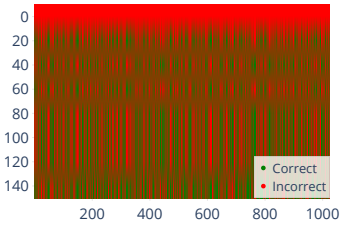
Figure C.21 shows which tokens in a string have been memorized correctly during the early epochs of training. There is no discernible order to the memorization. Some tokens in the middle or at the end of the string are memorized before earlier tokens and vice versa. Additionally, memorization is not stable in the initial parts of training, until the *Memorisation-Phase* has made some progress; previously memorized tokens are often forgotten (*i.e.*, predicted incorrectly) at later epochs, until memorization starts to converge, around epoch 20.



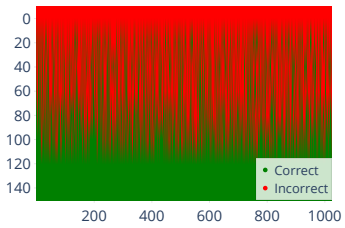
(a) Pythia-1B, $\ell = 2$



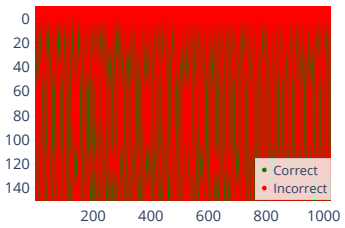
(b) Phi-2.7B, $\ell = 2$



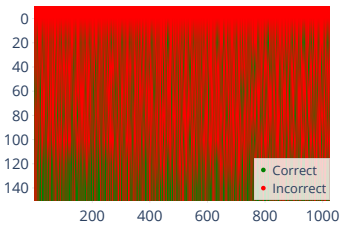
(c) Llama2-13B, $\ell = 2$



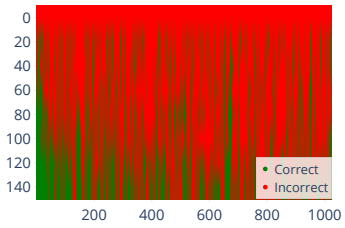
(d) Pythia-1B, $\ell = 4$



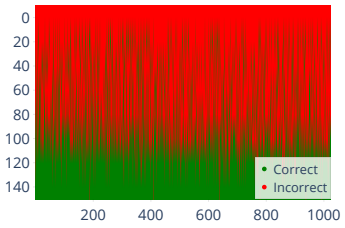
(e) Phi-2.7B, $\ell = 4$



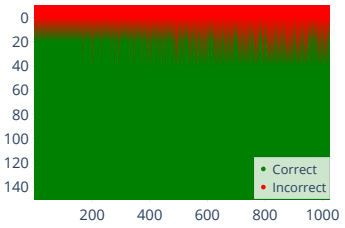
(f) Llama2-13B, $\ell = 4$



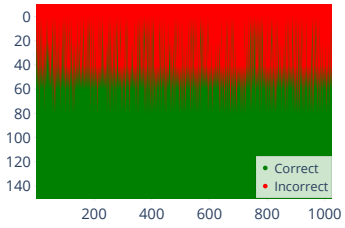
(g) Pythia-1B, $\ell = 7$



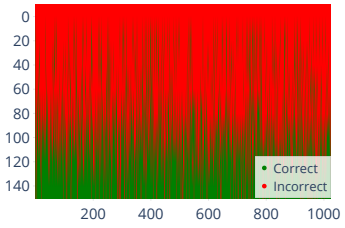
(h) Phi-2.7B, $\ell = 7$



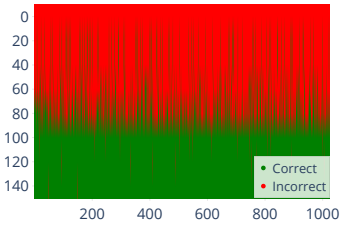
(i) Llama2-13B, $\ell = 7$



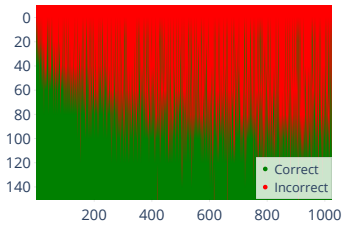
(j) Pythia-1B, $\ell = 13$



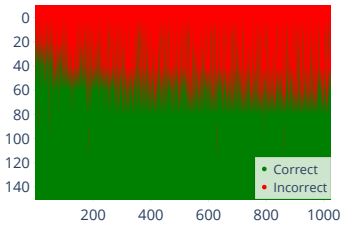
(k) Phi-2.7B, $\ell = 13$



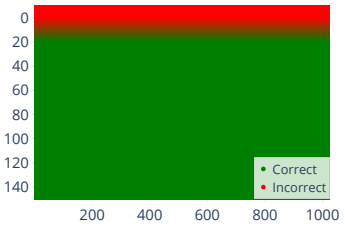
(l) Llama2-13B, $\ell = 13$



(m) Pythia-1B, $\ell = 26$



(n) Phi-2.7B, $\ell = 26$



(o) Llama2-13B, $\ell = 26$

Figure C.21: **[Memorization order for different pretrained models for different ℓ. ($n = 1024$)]** Memorization across strings happens in essentially random order, and is unstable until the *Memorisation-Phase* starts to converge. Llama2-13B models tend to memorize tokens at the beginning of the string slightly earlier, which may be because — in contrast to the other models — they use a beginning of string (BOS) token that could serve as a reference. Whether BOS tokens indeed affect memorization order needs more careful exploration, however.

	Initial Memorization					Stable Memorization				
Model	$l = 2$	$l = 4$	$l = 7$	$l = 13$	$l = 26$	$l = 2$	$l = 4$	$l = 7$	$l = 13$	$l = 26$
Pythia-1B	-0.052	-0.079	-0.058	-0.026	-0.002	0.081	0.105	-0.005	0.060	0.095
Phi-2.7B	-0.007	-0.000	-0.017	-0.019	-0.021	0.018	0.072	0.026	0.055	-0.020
Llama2-13B	-0.017	-0.029	0.031	0.045	0.095	0.109	0.138	0.253	0.166	0.171

Table C.2: [**Spearman rank correlation between token position in the string and the epoch at which tokens are memorized, for *pretrained models.* ($n = 1024$)**] Correlation is very low in all cases, except for Llama2-13B models, where the stable memorization correlation is slightly higher, presumably because they use a BOS token that could serve as a reference.

	Initial Memorization					Stable Memorization				
Model	$l = 2$	$l = 4$	$l = 7$	$l = 13$	$l = 26$	$l = 2$	$l = 4$	$l = 7$	$l = 13$	$l = 26$
Pythia-1B	0.031	0.009	0.026	0.028	0.243	0.230	0.206	0.434	0.336	0.657
Phi-2.7B	-0.001	0.018	0.014	0.003	0.108	0.187	0.132	0.268	0.236	0.572
Llama2-13B	-0.058	0.002	0.066	0.029	0.076	0.144	0.124	0.309	0.102	0.157

Table C.3: [**Spearman rank correlation between token position in the string and the epoch at which tokens are memorized, for *untrained models.* ($n = 1024$)**] Correlation is low for initial memorization and low to medium for stable memorization.

C.3.2 Quantifying randomness in memorization order using rank correlation

To quantify whether memorization order is indeed random, *i.e.* does not depend on the position of a token in the string, we compute the rank correlation between the tokens' positions and the epochs at which they are memorized As memorization epoch, we use both the initial memorization epoch (*i.e.* the epoch when the token is first predicted correctly), as well as the stable memorization epoch (*i.e.* the first epoch at and after which the token is not predicted incorrectly anymore).

We report results for Spearman rank correlation for pretrained Pythia-1B, Phi-2.7B and Llama2-13B models, for different alphabet sizes in Table C.2. Across the board we observe a very low correlation between a token's position in the string and the epoch at which it is memorized (both for initial and stable memorization). Correlation values in almost all cases are between -0.1 and 0.1.

We also report results for untrained models in Table C.3. Rank correlation between token position and initial memorization epoch is similarly low as for pretrained models. For stable memorization it is also low in most cases, but reaching medium correlation values for larger ℓ, for Pythia-1B and Phi-2.7B models. Overall, the results suggest that memorization order is mostly random, and does not depend on the position of a token in the string.

C.3.3 Quantifying randomness in memorization order using the discrepancy score

In order to quantify whether the memorized positions are uniformly distributed we compute the *discrepancy score*; this score is motivated by the notion of discrepancy in statistics [Niederreiter, 1992] and is defined as follows: for any epoch, we count the number of correct recollections of our model on the *input string* and then we pick uniformly at random the same number of positions on *random string* of the same length. Utilizing a fixed window of 20 tokens, we randomly sample 50 substrings from each of the two strings. For each of these sampled substrings, we calculate difference in the number of correct recollections between the tested and target substrings. The average of these differences provides the *discrepancy score*.

Formally, the discrepancy score is defined as follows:

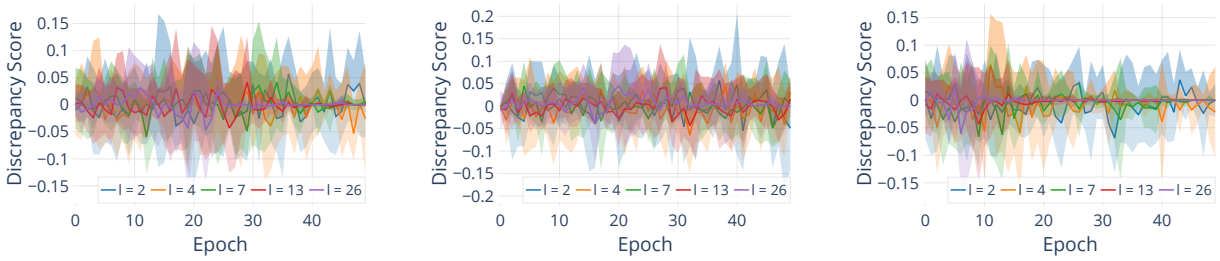
$$discrepancy(M, s) = \frac{1}{|PS|} \sum_{i \in PS} \Delta_{i,k}(M, s, c^{(rand)})$$

where

$$\Delta_{i,k}(M, s, c^{(rand)}) = \frac{1}{k} \sum_{j=1}^{k} \delta \left(pred(M|s_{[1,i+j-1]}) = s_{i+j}) - c_{i+j}^{(rand)} \right)$$

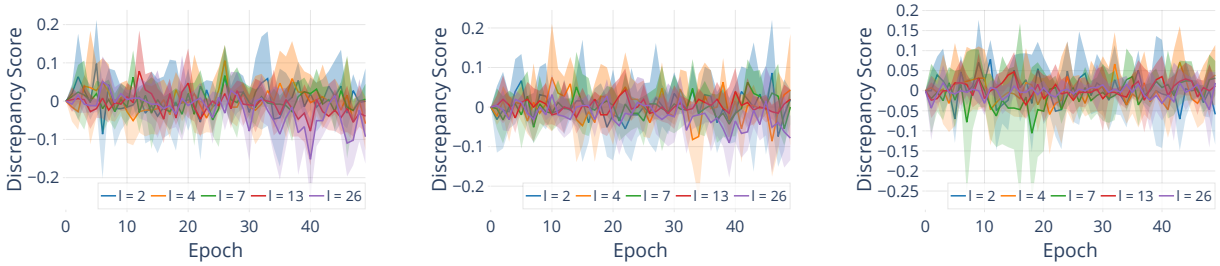
and PS is a set of randomly selected positions in the string s, $c^{(rand)}$ is a binary string where $N_{correct} = \sum_i^n \delta(pred(M|s_{[1,i-1]}) = s_i)$ (*i.e.* as many positions as there are correct predictions by M) randomly chosen positions are 1, and else 0. $pred(M|s_{[1,i-1]}) = \arg\max_{t \in V} P_M(t|s_{[1,i-1]})$ is the model's prediction at position i. In our analysis, we sample $|PS| = 50$ positions in the string s and use a window size of $k = 20$.

Intuition on discrepancy: The intuition behind the discrepancy score is to measure how different the set of positions correctly predicted by the model is from randomly picking the same number positions as there are correct predictions. If the difference is close to zero, then selecting the same number of string positions based on whether they are correctly predicted or randomly picked is equivalent.



(a) Pythia-1B (b) Phi-2.7B (c) Llama2-13B

Figure C.22: **[Discrepancy scores for different pretrained models. ($n = 1024$)]** Discrepancy scores for all models and strings are low, indicating random memorization order.



(a) Pythia-1B (b) Phi-2.7B (c) Llama2-13B

Figure C.23: **[Discrepancy scores for different untrained models. ($n = 1024$)]** Discrepancy scores for all models and strings are low, indicating random memorization order.

Results: The low discrepancy scores for pretrained models observed in Figure C.22 suggests that for all the models evaluated, the memorized positions are random. Thus, we conclude

that memorization happens at the granularity of individual tokens and not entire strings. The discrepancy scores for untrained models in Figure C.23 are also low, though slightly higher than for pretrained models, indicating that memorization order is largely random for untrained models as well.

C.4 Memorization dynamics under real-world conditions

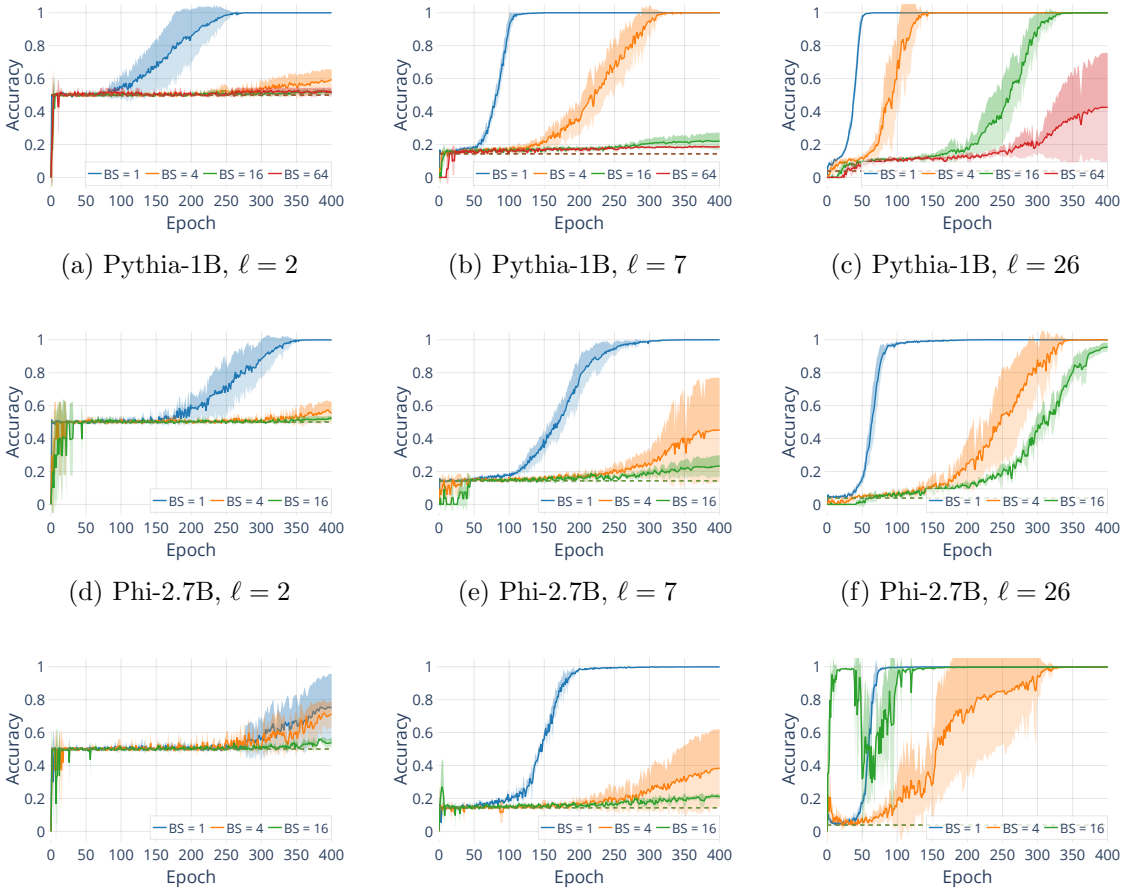
To validate our observations in practical settings, we train models to memorize random strings under conditions that closely resemble real-world settings. We present random strings to the model in the context of natural language data, in two different ways: 1) by adding additional natural language sequences to the training batches, and 2) by presenting single natural language sequences inside which random strings appear as substrings. We use the wikitext Merity et al. [2016] dataset as a source of natural language training data.

In both of the cases, the random string to natural data ratio varies. To minimise its loss, the model still has to memorize the random string, but it simultaneously also needs to become better at modeling the wikitext data. We study both pretrained and untrained (to mimic pretraining from scratch) models, over alphabet sizes $l = 2, 7, 26$.

Results summary: We investigate the memorization dynamics of random strings as part of larger natural language training batches in Appendix C.4.1, and of random strings as substrings of larger natural language strings in Appendix C.4.2. We make the same observations about memorization dynamics as with single strings. When memorising random strings in the context of other natural data, models still exhibit the two phases during the memorization process, and lower entropy strings are also harder to memorize. Memorization becomes slower, however, the more natural data there is, relative to the size of the random string. Overall, our results on the dynamics of memorization are robust under more practical training schemes, since the observations made on random strings in isolation match those made when random strings are embedded in natural language data.

C.4.1 Embedding random strings within batches of natural language

To mimic typical pretraining setups, we train models on batches of size up to $BS = 1, 4, 16, 64$, with sequence length $n = 1024$. The same $n = 1024$ token random string appears repeatedly, once in each batch, *i.e.* as one of the batch elements. Batch size $BS = 1$ is equivalent to our previous experiments. The other sequences in each batch come from wikitext and change at every step without repetition.

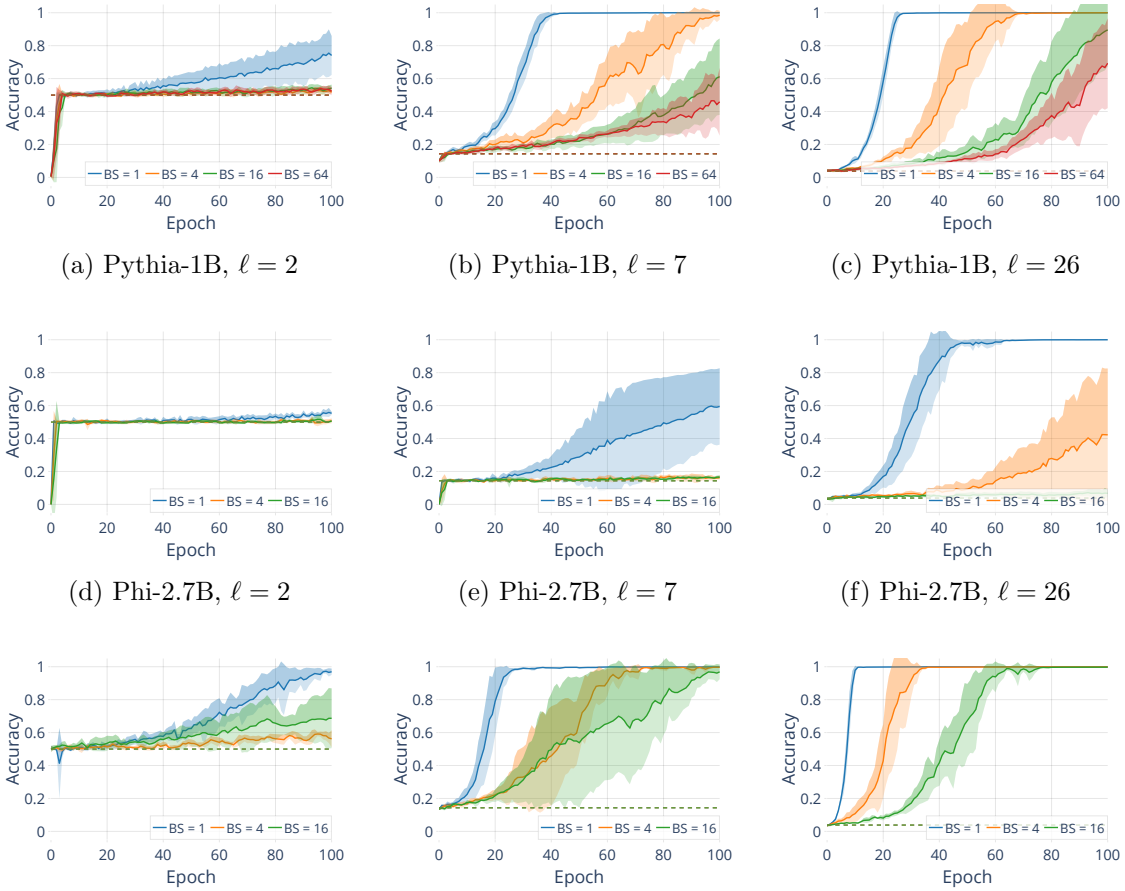


(a) Pythia-1B, $\ell = 2$ (b) Pythia-1B, $\ell = 7$ (c) Pythia-1B, $\ell = 26$

(d) Phi-2.7B, $\ell = 2$ (e) Phi-2.7B, $\ell = 7$ (f) Phi-2.7B, $\ell = 26$

(g) Llama2-13B, $\ell = 2$ (h) Llama2-13B, $\ell = 7$ (i) Llama2-13B, $\ell = 26$

Figure C.24: [**Accuracy for *untrained* models, when embedding random strings inside of batches of natural language data, for multiple ℓ and batch sizes b. ($n = 1024$)**]



(a) Pythia-1B, $\ell = 2$ (b) Pythia-1B, $\ell = 7$ (c) Pythia-1B, $\ell = 26$

(d) Phi-2.7B, $\ell = 2$ (e) Phi-2.7B, $\ell = 7$ (f) Phi-2.7B, $\ell = 26$

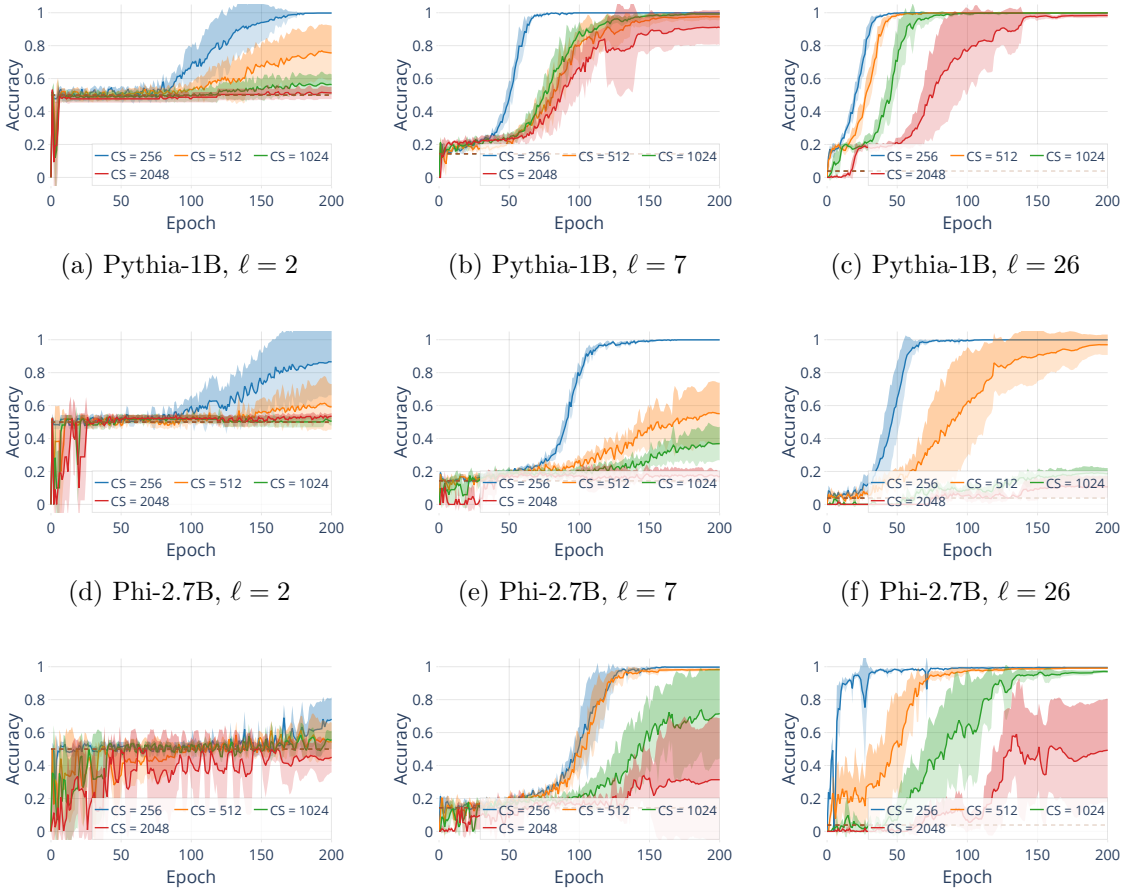
(g) Llama2-13B, $\ell = 2$ (h) Llama2-13B, $\ell = 7$ (i) Llama2-13B, $\ell = 26$

Figure C.25: [**Accuracy for *pretrained* models, when embedding random strings inside of batches of natural language data, for multiple ℓ and batch sizes b. ($n = 1024$)**]

Figure C.24 shows the accuracy of untrained models on batched training, and Figure C.25 shows the accuracy of pretrained models. In all cases, models initially converge to the random guess level during the *Guessing-Phase*, and then start memorising the random string during the *Memorisation-Phase*, at least for higher-entropy strings. Higher entropy strings are consistently easier to memorize. These observations match our previous results where we train on random strings in isolation.

C.4.2 Embedding random strings within longer natural language strings

To simulate cases where random strings appear as substrings inside other strings (e.g. as with typical sensitive data such as email addresses, phone numbers, SSH-keys, etc.), we train models on single natural language strings of lengths $CS = 256, 512, 1024, 2048$. We use a different string from wikitext at each step, but always insert the same 256 token random substring at a random position. For context size $CS = 256$ the entire string is the random string, as in our previous experiments.

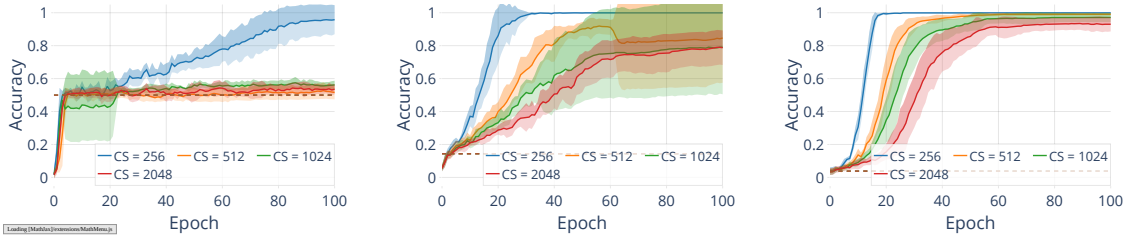


(a) Pythia-1B, $\ell = 2$ (b) Pythia-1B, $\ell = 7$ (c) Pythia-1B, $\ell = 26$

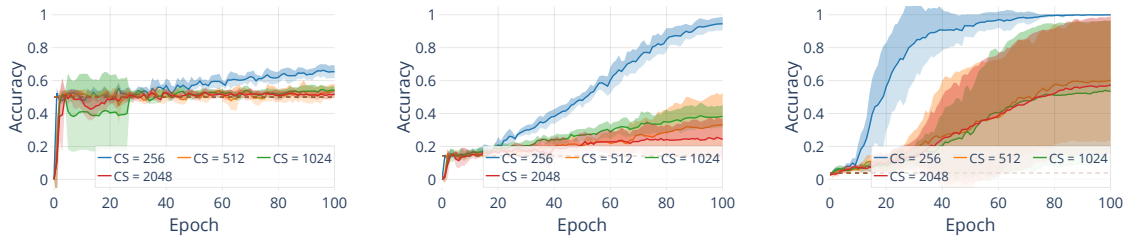
(d) Phi-2.7B, $\ell = 2$ (e) Phi-2.7B, $\ell = 7$ (f) Phi-2.7B, $\ell = 26$

(g) Llama2-13B, $\ell = 2$ (h) Llama2-13B, $\ell = 7$ (i) Llama2-13B, $\ell = 26$

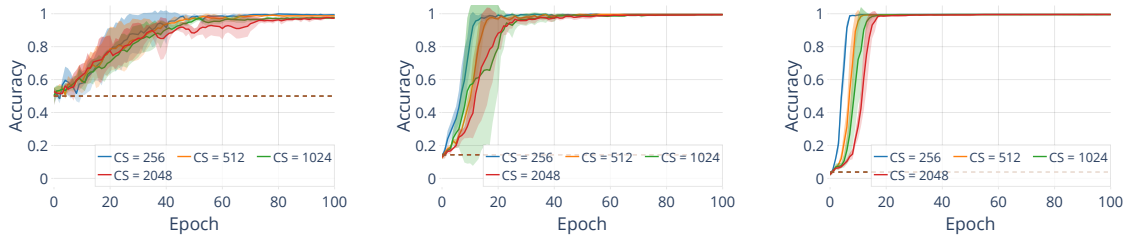
Figure C.26: [**Accuracy for *untrained* models, when embedding random strings inside of longer strings of natural language data, for multiple ℓ and context sizes c.** ($n = 256$)]



(a) Pythia-1B, $\ell = 2$ (b) Pythia-1B, $\ell = 7$ (c) Pythia-1B, $\ell = 26$



(d) Phi-2.7B, $\ell = 2$ (e) Phi-2.7B, $\ell = 7$ (f) Phi-2.7B, $\ell = 26$



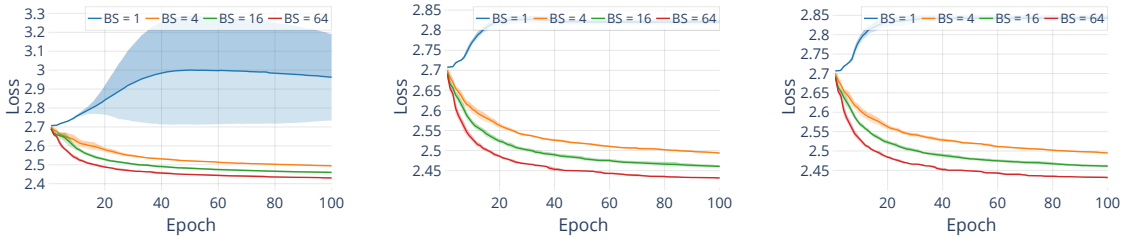
(g) Llama2-13B, $\ell = 2$ (h) Llama2-13B, $\ell = 7$ (i) Llama2-13B, $\ell = 26$

Figure C.27: [**Accuracy for *pretrained* models, when embedding random strings inside of longer strings of natural language data, for multiple ℓ and context sizes c. ($n = 256$)**]

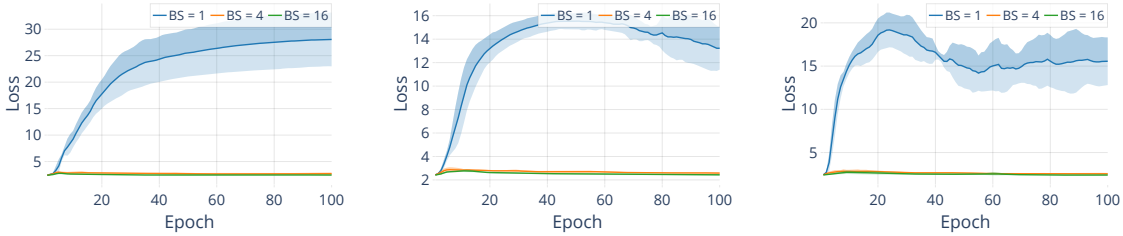
Figure C.26 shows the accuracy of untrained models on embedded random strings, and Figure C.27 shows the accuracy of pretrained models. As with batched training, the models exhibit both the *Guessing-Phase* and the *Memorisation-Phase*. Again, the difficulty of memorization depends on the entropy of the random string, with higher entropy strings being easier to memorize. These findings are are also consistent with our previous results where we train on random strings in isolation.

C.4.3 Impact of random string memorization on natural language performance

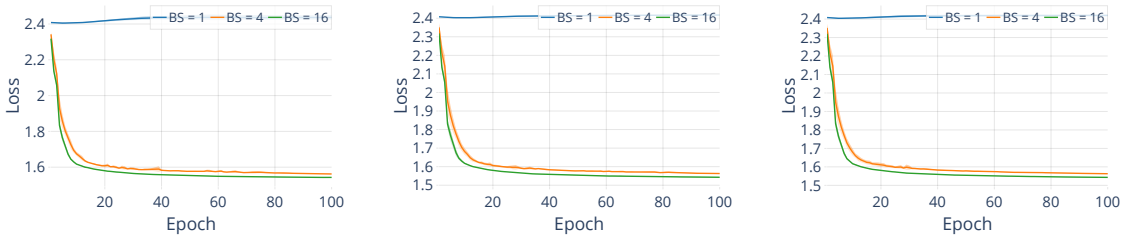
To better understand the implications of our experimental setup, we estimate the performance impact of memorising random strings on natural language modelling. In particular, we measure how memorising a single random string impacts the model's loss on the wikitext testset Merity et al. [2016]. We perform these measurements for the same models trained on batches of natural language data with single random sequences injected as described in Appendix C.4.1, and on strings of natural data with embedded random substrings as in Appendix C.4.2. We focus on pretrained models here to understand how their previously acquired language modelling capabilities are affected by memorising random strings.



(a) Pythia-1B, $\ell = 2$ (b) Pythia-1B, $\ell = 7$ (c) Pythia-1B, $\ell = 26$

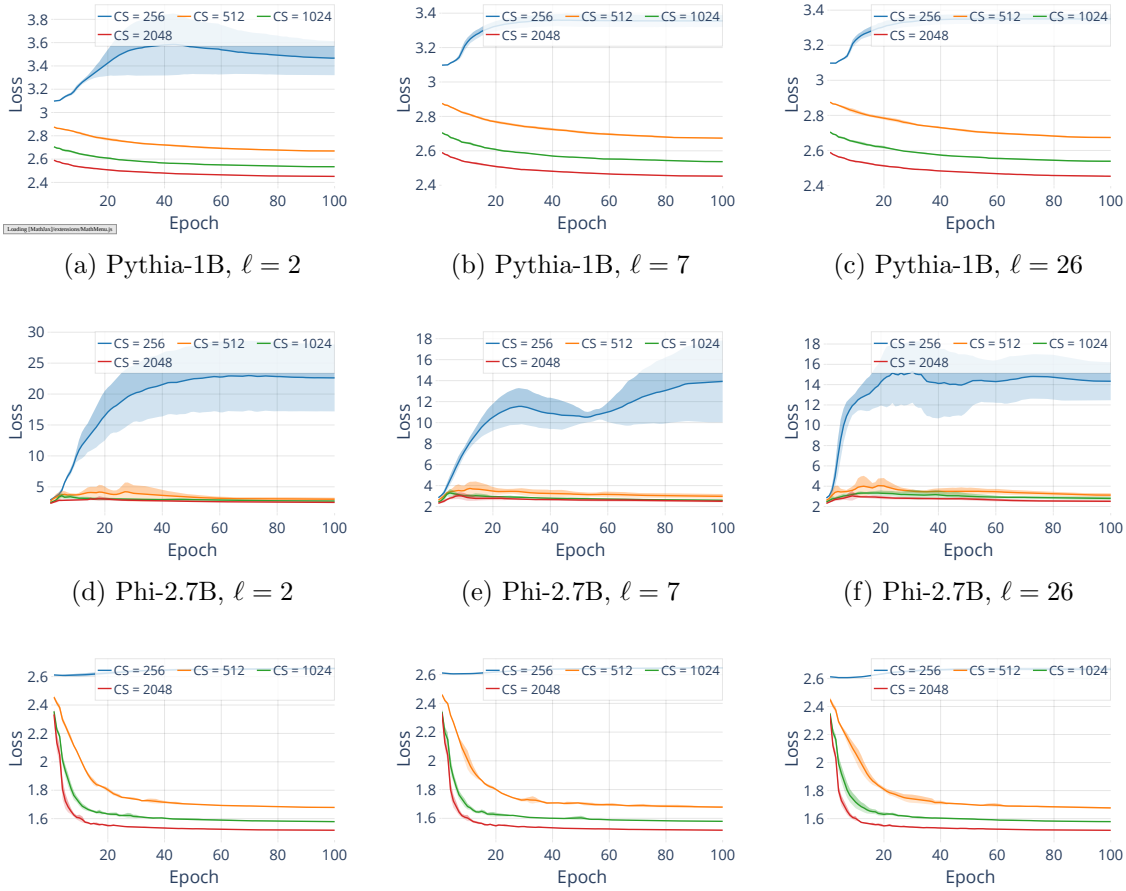


(d) Phi-2.7B, $\ell = 2$ (e) Phi-2.7B, $\ell = 7$ (f) Phi-2.7B, $\ell = 26$



(g) Llama2-13B, $\ell = 2$ (h) Llama2-13B, $\ell = 7$ (i) Llama2-13B, $\ell = 26$

Figure C.28: **[Loss on a natural-language testset during memorization, when embedding random strings inside of batches of natural language data, for multiple ℓ and batch sizes. ($n = 1024$)]**



(a) Pythia-1B, $\ell = 2$ (b) Pythia-1B, $\ell = 7$ (c) Pythia-1B, $\ell = 26$

(d) Phi-2.7B, $\ell = 2$ (e) Phi-2.7B, $\ell = 7$ (f) Phi-2.7B, $\ell = 26$

(g) Llama2-13B, $\ell = 2$ (h) Llama2-13B, $\ell = 7$ (i) Llama2-13B, $\ell = 26$

Figure C.29: **[Loss on a natural-language testset during memorization, when embedding random strings inside of longer strings of natural language data, for multiple ℓ and context sizes. ($n = 256$)]**

Memorising random strings in isolation: The blue curves in Figure C.28 ($BS = 1$) and Figure C.29 ($CS = 256$) show the loss of Pythia-1B, Phi-2.7B and Llama2-13B models on the wikitext testset while memorising strings of length $n = 1024$, resp. $n = 256$ with different alphabet sizes ($l = 2, 7, 26$) for 100 epochs. The results show that memorising a single random string does indeed increase the loss on the testset. For Pythia-1B and Llama2-13B models, the loss increase is quite moderate, from ~ 2.7 to ~ 2.9 for Pythia-1B, and from ~ 2.4 to less than ~ 2.5 for Llama2-13B. The loss increases quite significantly from ~ 2.4 to $15 - 30$ (depending on the alphabet size) for the Phi-2.7B model.

Memorising random strings in the context of natural language: The other curves in Figure C.28 ($BS > 1$) and Figure C.29 ($CS > 256$) show the loss on the wikitext testset when the model is trained on batches of (non-repeated) text of size BS from the wikitext dataset where one of the elements is the (repeated) random string, and on strings from wikitext with random substrings of length 256, respectively. When we show the models wikitext data in addition to the random string, their loss on the testset actually decreases. For Pythia-1B from ~ 2.7 to ~ 2.45, for Phi-2.7B from ~ 2.4 to ~ 2.35, and for Llama2-13B from ~ 2.4 to ~ 1.5. At the same time we still observe the same memorization dynamics, as discussed in Appendices C.4.1 and C.4.2.

In summary, our results show that memorising a single random string in isolation can increase the model's loss on natural language text. However, when embedding random strings into larger natural language contexts, models memorize random strings with the same dynamics as for strings in isolation, while increasing natural language performance. Therefore, it is likely that the memorization dynamics we observe also occur in real-world training settings.

C.5 Additional details on prefix mappings

C.5.1 Additional details on the experimental setup for testing local prefix accuracy

We test whether a token s_i in string s sampled from distribution P_A can be correctly recalled with a local prefix $s_{[i-k,i-1]}$ of length k for the different replacement strategies `Random` and `Constant`.

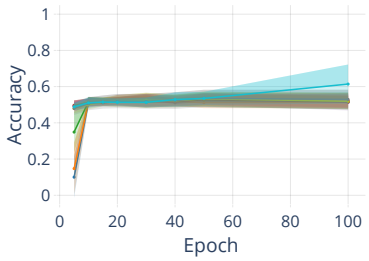
To test recall for the `Random` strategy, we sample 10 replacements r_j of length $i-k-1$ for the global context $s_{[1,i-k-1]}$ from P_A. Then we compute how many times the model correctly predicts token s_i as the token with the highest probability, given the input $r_j \circ s_{[i-k,i-1]}$, *i.e.* when we randomize all tokens in the input according to P_A, other than the local prefix. If a *plurality* of predictions among the 10 samples match s_i, *i.e.* if s_i is the most frequently predicted token, then we say that the local prefix of size k can correctly recall s_i.

For the `Constant` strategy we follow a similar process, but instead of sampling all tokens in each r_j randomly from P_A, we only sample a single token for each r_j, that we use at all positions in r_j.

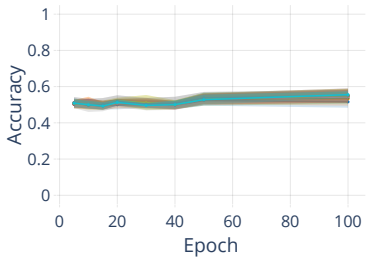
Since finding local prefixes with multiple samples for different lengths and positions in the string requires a large number of inference calls, for the 1024 tokens strings, we randomly subsample 256 positions and compute the performance of their local prefixes of different lengths.

C.5.2 Additional results on local prefixes

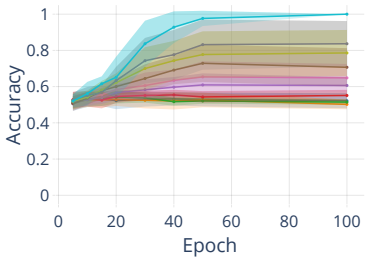
Figure C.30 shows the recollection accuracy of prefixes with different length for different models and ℓ, for pretrained models. In Figure C.31 we show the same for untrained models. For untrained models, prefixes shorter than the full prefix recollect significantly fewer tokens correctly than for pretrained models, in most cases not performing much better than random guessing. This could mean that untrained models either rely on more tokens from the context to make predictions, or that they use a similar number of tokens as pretrained models, but which are not necessarily immediately preceding the token to predict, but more spread out over the context. In the latter case this would mean that during pretraining, models acquire a bias that makes them pay more attention to tokens immediately preceding the token to predict.



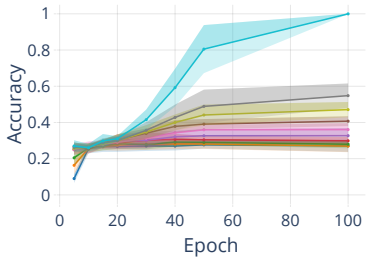
(a) Pythia-1B, $\ell = 2$



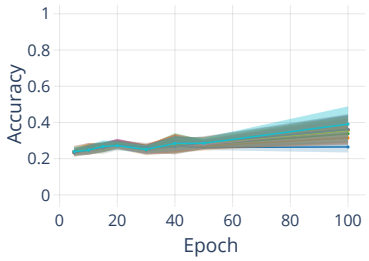
(b) Phi-2.7B, $\ell = 2$



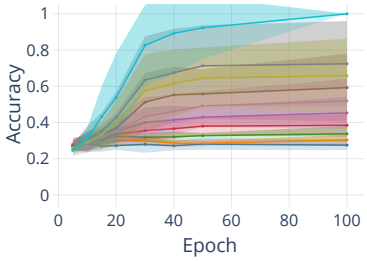
(c) Llama2-13B, $\ell = 2$



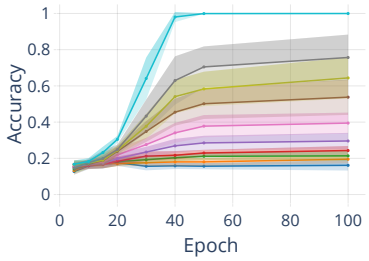
(d) Pythia-1B, $\ell = 4$



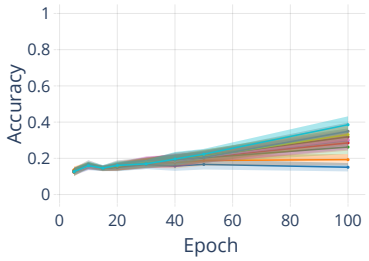
(e) Phi-2.7B, $\ell = 4$



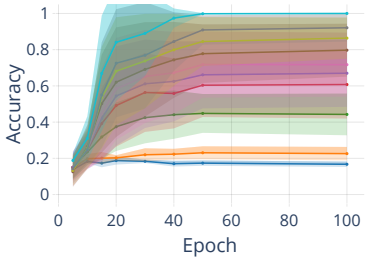
(f) Llama2-13B, $\ell = 4$



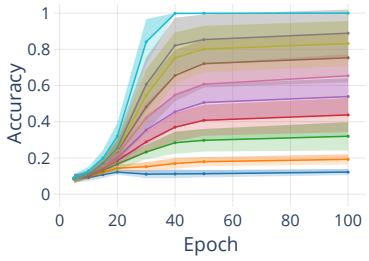
(g) Pythia-1B, $\ell = 7$



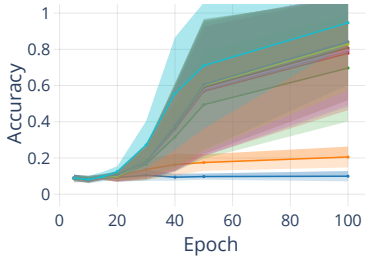
(h) Phi-2.7B, $\ell = 7$



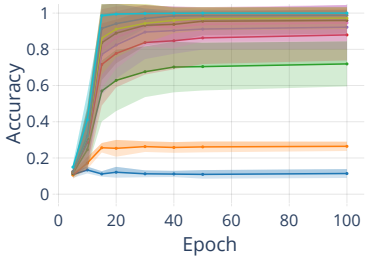
(i) Llama2-13B, $\ell = 7$



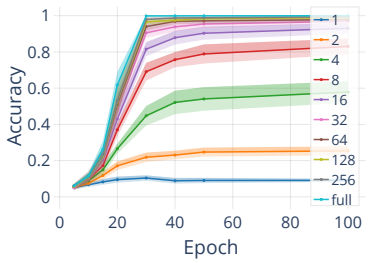
(j) Pythia-1B, $\ell = 13$



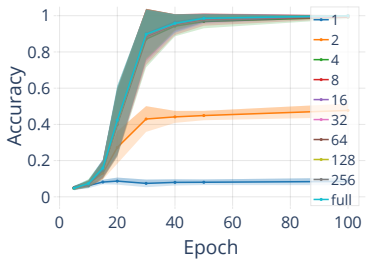
(k) Phi-2.7B, $\ell = 13$



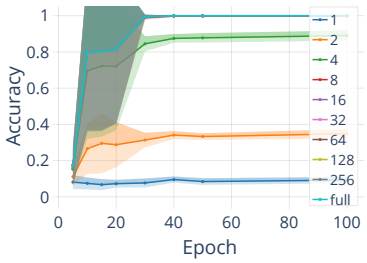
(l) Llama2-13B, $\ell = 13$



(m) Pythia-1B, $\ell = 26$

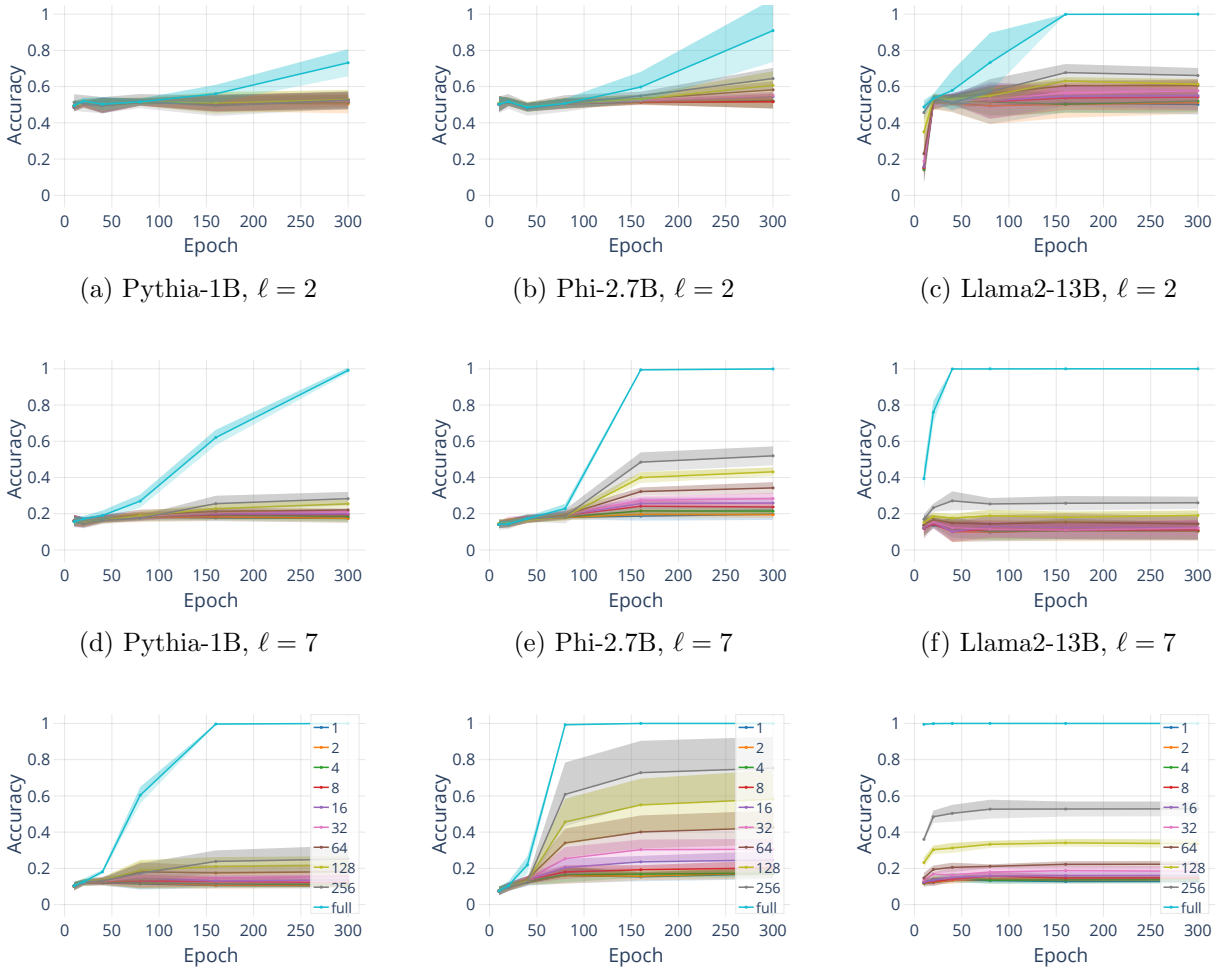


(n) Phi-2.7B, $\ell = 26$



(o) Llama2-13B, $\ell = 26$

Figure C.30: **[Recollection accuracy for *pretrained* models for different prefix lengths during training for different ℓ's ($n = 1024$)]** In most cases, local prefixes, *i.e.* a small number of tokens immediately preceding the token to predict, much fewer than the full string's length of $n = 1024$ perform well.



(a) Pythia-1B, $\ell = 2$ (b) Phi-2.7B, $\ell = 2$ (c) Llama2-13B, $\ell = 2$

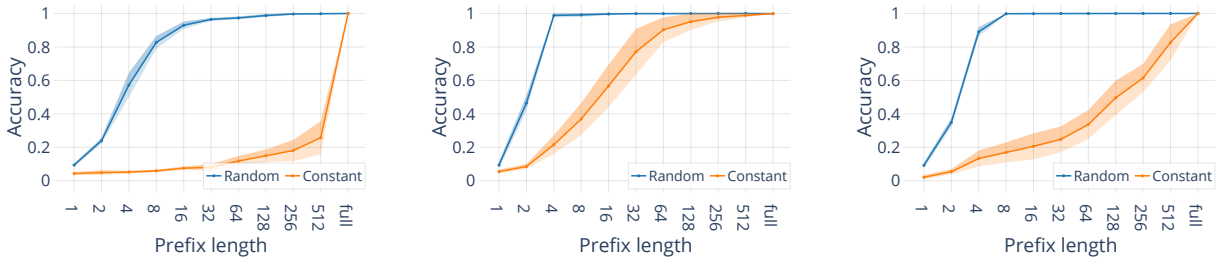
(d) Pythia-1B, $\ell = 7$ (e) Phi-2.7B, $\ell = 7$ (f) Llama2-13B, $\ell = 7$

(g) Pythia-1B, $\ell = 26$ (h) Phi-2.7B, $\ell = 26$ (i) Llama2-13B, $\ell = 26$

Figure C.31: **[Recollection accuracy for *untrained* models for different prefix lengths during training for different ℓ's ($n = 1024$)]** For untrained models, local prefixes perform significantly worse than for pretrained models, which might mean that untrained models either rely on more tokens from the context to make predictions, or that they use a similar number of tokens as pretrained models which are distributed throughout the context.

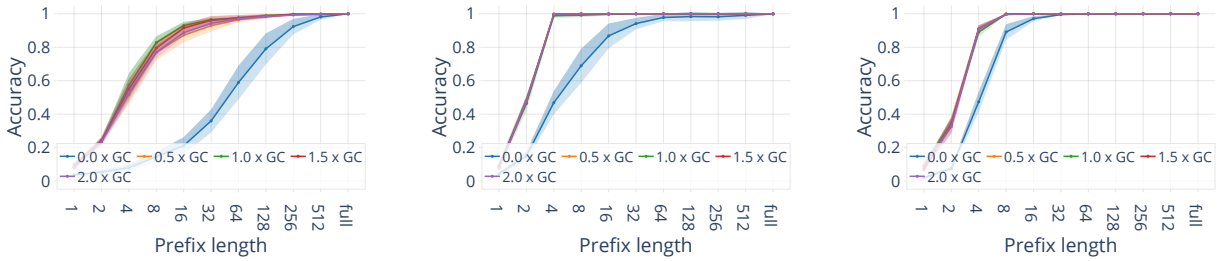
C.5.3 Additional results on global context

In Figure C.32 we show the effect of the replacement strategy (`Random` and `Constant`) on recollection accuracy. In Figure C.33 we show the effect of changing the size of the global context.



(a) Pythia-1B (b) Phi-2.7B (c) Llama2-13B

Figure C.32: [**Effect of replacement strategies for the global context on recollection accuracy.** ($n = 1024, \ell = 26$)]

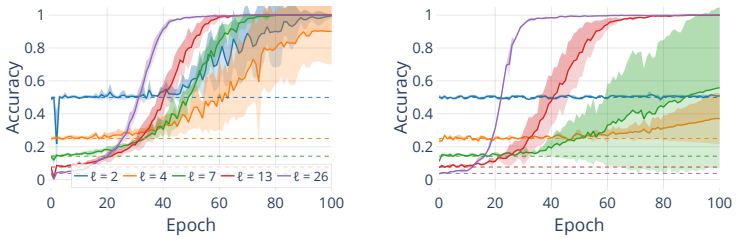


(a) Pythia-1B (b) Phi-2.7B (c) Llama2-13B

Figure C.33: [**Effect of changing the size of the global on recollection accuracy.** ($n = 1024, \ell = 26$)]

C.5.4 Results for models with absolute position embeddings

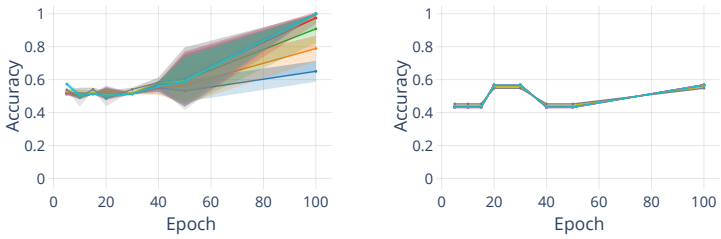
The main model families discussed in the chapter (Pythia, Phi, Llama2), as well as many other modern architectures, use *relative position encodings*, and in particular rotary embeddings [Su et al., 2024]. Some older architectures, however, use *absolute position encodings*, such as GPT-2 [Radford et al., 2019], GPT-3 [Brown et al., 2020] and OPT Zhang et al. [2022]. To account for this difference, we also study models from these families, in particular the 140M parameter GPT-2 and the OPT-350M model.



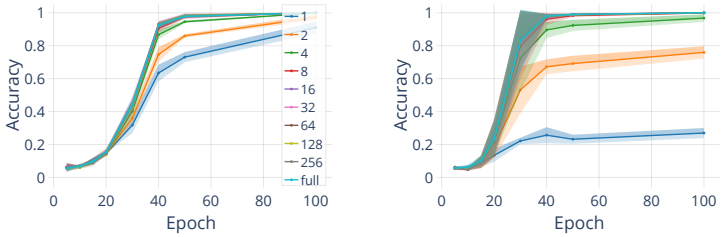
(a) GPT2-124M, Accuracy (b) OPT-350M, Accuracy

Figure C.34: [**Memorization dynamics for different ℓ, for models with absolute position encodings** ($n = 1024$)] We observe the same memorization dynamics for models with absolute position encodings as for models with relative position encodings.

Figure C.34 shows that these models behave similarly to their more modern counterparts in terms of memorization behaviour (phases of memorization, higher entropy strings being easier to memorize, etc.) However, similar to Sinha et al. [2022] we find in our analysis on the role of local prefixes and global context, that these models tend to over-rely on position information, which can impact their memorization behaviour.

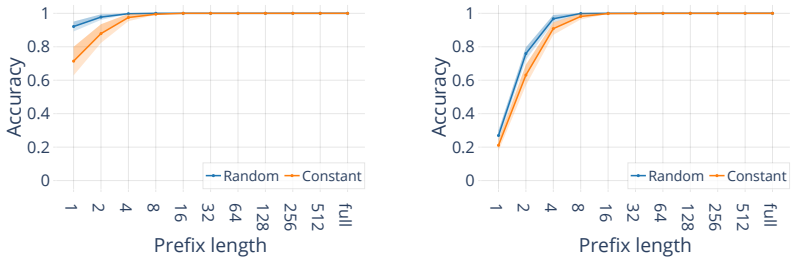


(a) GPT2-124M, $\ell = 2$ (b) OPT-350M, $\ell = 2$



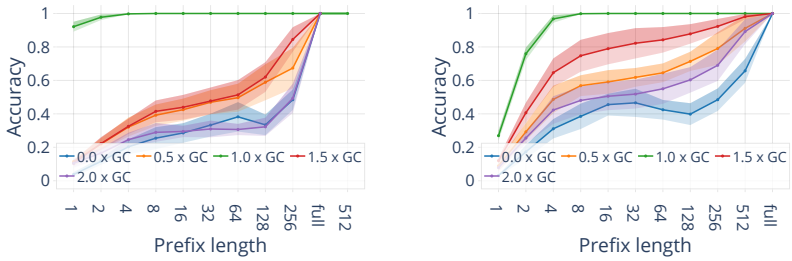
(c) GPT2-124M, $\ell = 26$ (d) OPT-350M, $\ell = 26$

Figure C.35: **[Recollection accuracy for different prefix lengths during training for different ℓ's]** Short prefixes perform considerably better for models with absolute position encodings, than for ones with relative position encodings.



(a) GPT2-124M (b) OPT-350M

Figure C.36: **[Effect of replacement strategies for the global context on recollection accuracy. ($n = 1024, \ell = 26$)]** Models that use global position encodings are barely affected by changes in the global context distribution.



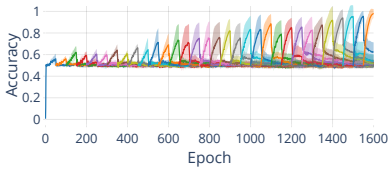
(a) GPT2-124M, $n = 512$ (b) OPT-350M, $n = 1024$

Figure C.37: [**Effect of changing the size of the global context on recollection accuracy. ($\ell = 26$)**] In contrast to models using relative position encodings, models with absolute position encodings are heavily affected by changes to the length of the global context, *i.e.* to the token position. We use 512 token strings for GPT2 here, since its context window is only 1024 tokens long, so we would not be able to double the context size with a 1024 base string.

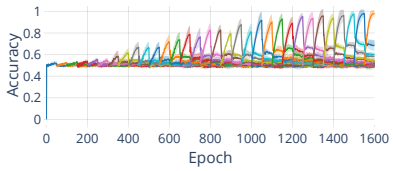
Figure C.35 shows that for GPT-2 and OPT, short prefixes perform considerably better than for models with relative position encodings. In Figure C.36 we show results for changing the replacement strategy for the global context, and in Figure C.37 we show results for changing the size of the global context. In contrast to models using relative position encodings, models with absolute position encodings are not affected by changes in the global context distribution, but heavily affected by changes in the size of the global context, *i.e.* in the position of the token in the string. These results indicate, that absolute position encoding models rely, at least partially, on position information for memorization. Exploring this connection further is an interesting avenue for future work.

C.6 Additional Results on Repeated Memorization

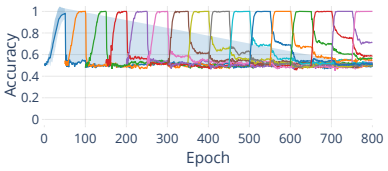
Figure C.38 shows results for different models for iteratively memorising a series of random strings. Figure C.39 shows the accuracy during the initial 50 epochs of memorization for the respective strings.



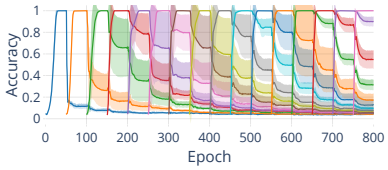
(a) Pythia-1B, $\ell = 2$, 32 strings



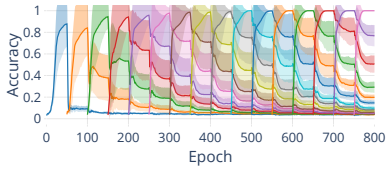
(b) Phi-2.7B, $\ell = 2$, 32 strings



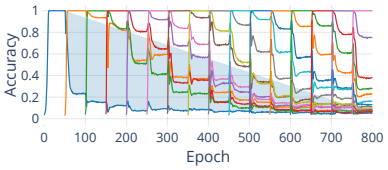
(c) Llama2-13B, $\ell = 2$, 16 strings



(d) Pythia-1B, $\ell = 26$, 16 strings

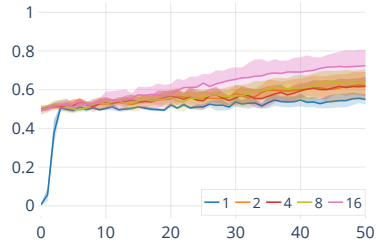


(e) Phi-2.7B, $\ell = 26$, 16 strings

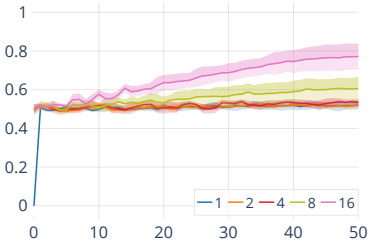


(f) Llama2-13B, $\ell = 26$, 16 strings

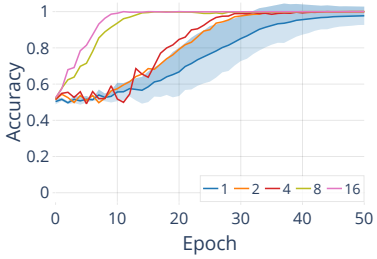
Figure C.38: [**Accuracy on repeated memorization for different models.** ($n = 1024$)] As models memorize additional strings, they forget previous ones. However, forgetting slows down as more strings are memorized.



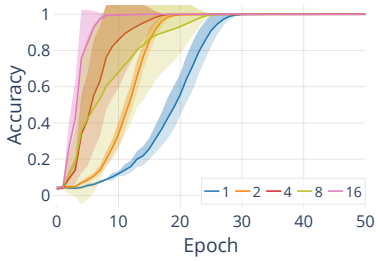
(a) Pythia-1B, $\ell = 2$



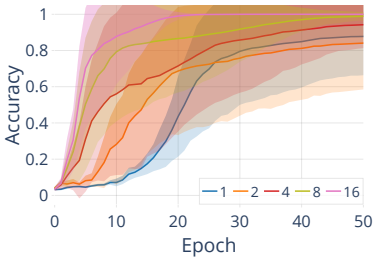
(b) Phi-2.7B, $\ell = 2$



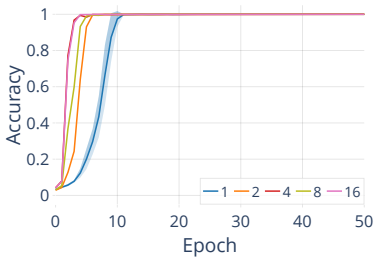
(c) Llama2-13B, $\ell = 2$



(d) Pythia-1B, $\ell = 26$



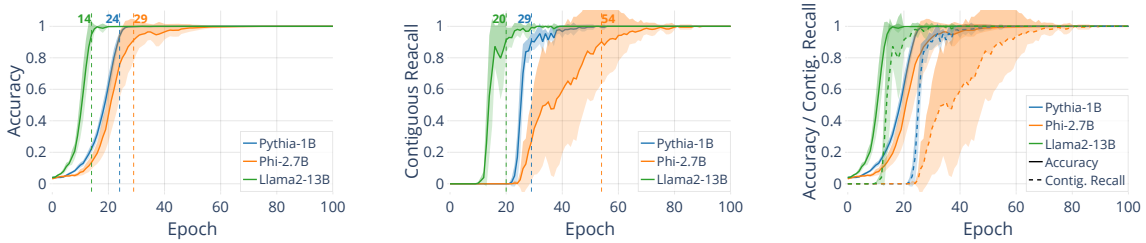
(e) Phi-2.7B, $\ell = 26$



(f) Llama2-13B, $\ell = 26$

Figure C.39: [**Accuracy comparison for the initial 50 epochs for different models.** ($n = 1024$)] During repeated memorization, models become faster at memorising new random strings.

C.7 Existing memorization measures can severely underestimate the degree of memorization



(a) Individual token accuracy (b) 50 contiguous tokens recall (c) Accuracy vs. contiguous recall

Figure C.40: **[Applying string-based memorization measures to random strings. ($n = 1024, \ell = 26$)]** String-based memorization metrics severely underestimate the degree of memorization in random strings. The number of substrings detect as memorized with a strict 50 correct adjacent token requirement in (b) is much lower than the prediction accuracy of the model over individual string positions in (a). Dashed vertical lines indicate the positions at which the respective metric first crosses the 90% mark.

We apply the popular memorization metric from Carlini et al. [2022] in our random token string setting. The measure detects a string s as memorized by model $\mathcal{M}$ if $\mathcal{M}$ produces s (with greedy decoding) when prompted with string prefix p, and $[p||s]$ is contained in $\mathcal{M}$'s training data.

We apply this measure to $n = 1024$ random token strings at each training epoch of models and measure the fraction of substrings that it detects as memorized. Analogously to Carlini et al. [2022], we set the string length $|s|$ to 50 tokens. Then, we detect for every contiguous position i in the 1024 token string whether, when prompted with the entire preceding string, $\mathcal{M}$ predicts all of the next 50 tokens correctly, i.e. with the highest probability. If it does then we consider tokens $[i : i + 50]$ to be memorized, otherwise not.

Figure C.40b shows what fraction of the positions in the 1024 token string the measure recollects as memorized at each epoch. For the Pythia-1B model, recall remains essentially zero up until epoch 25, when the model already accurately predicts more than 90% of the tokens (Figure C.40a). Recall is low, because even when the string is largely memorized, models still make a small number of mispredictions, which are randomly scattered. Any misprediction, however, results in many substrings not being detected as memorized. The requirement of 50 contiguous correctly predicted tokens is too strict in this case, and the contiguous string-based memorization metric largely fails to detect memorization. Therefore, we argue that memorization metrics should operate at the token- rather than at the (sub-)string level.

Bibliography

Hervé Abdi. Coefficient of variation. *Encyclopedia of research design*, 1:169–171, 2010.

Samira Abnar, Mostafa Dehghani, Behnam Neyshabur, and Hanie Sedghi. Exploring the limits of large scale pre-training. *arXiv preprint arXiv:2110.02095*, 2021.

Adult. `http://tinyurl.com/UCI-Adult`, 1996.

Bo Ai, Zhanxin Wu, and David Hsu. Invariance is key to generalization: Examining the role of representation in sim-to-real transfer for visual navigation. *arXiv preprint arXiv:2310.15020*, 2023.

Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. A review on language models as knowledge bases. *arXiv preprint arXiv:2204.06031*, 2022.

Athanasios Andreou, Giridhari Venkatadri, Oana Goga, Krishna P. Gummadi, Patrick Loiseau, and Alan Mislove. Investigating ad transparency mechanisms in social media: A case study of Facebook's explanations. In *NDSS*, 2018.

Julia Angwin and Terry Parris Jr. Facebook lets advertisers exclude users by race. `https://www.propublica.org/article/facebook-lets-advertisers-exclude-users-by-race`, 2016.

Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. `https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing`, 2016.

Julia Angwin, Ariana Tobin, and Madeleine Varner. Facebook (still) letting housing advertisers exclude users by race. `https://www.propublica.org/article/facebook-advertising-discrimination-housing-race-sex-national-origin`, 2017a.

Julia Angwin, Madeleine Varner, and Ariana Tobin. Facebook enabled advertisers to reach 'jew haters'. `https://www.propublica.org/article/facebook-enabled-advertisers-to-reach-jew-haters`, 2017b.

Stephen Ansolabehere and Eitan Hersh. Gender, race, age and voting: A research note. *Politics and Governance*, 2013.

Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Anthony B. Atkinson. On the measurement of inequality. *Journal of economic theory*, 2(3): 244–263, 1970.

Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. Factors of transferability for a generic convnet representation. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1790–1802, 2015.

Aharon Azulay and Yair Weiss. Why do deep convolutional networks generalize so poorly to small image transformations? *arXiv preprint arXiv:1805.12177*, 2018.

Randall Balestriero, Ishan Misra, and Yann LeCun. A data-augmentation is worth a thousand samples: Exact quantification from analytical augmented sample moments. *arXiv preprint arXiv:2202.08325*, 2022.

Solon Barocas and Andrew D Selbst. Big data's disparate impact. *California Law Review*, 104: 671, 2016a.

Solon Barocas and Andrew D. Selbst. Big data's disparate impact. *California Law Review*, 2016b.

Lorenzo Giovanni Bellù and Paolo Liberati. Inequality analysis: The gini index. *Food and Agriculture Organization of the United Nations, EASYPol Module*, 40, 2006.

Gregory Benton, Marc Finzi, Pavel Izmailov, and Andrew G Wilson. Learning invariances in neural networks from training data. *Advances in neural information processing systems*, 33: 17605–17616, 2020.

Richard Berk, Hoda Heidari, Shahin Jabbari, Michael Kearns, and Aaron Roth. Fairness in criminal justice risk assessments: The state of the art. *arXiv preprint arXiv:1703.09207*, 2017.

Dan Biddle. *Adverse Impact and Test Validation: A Practitioner's Guide to Valid and Defensible Employment Testing*. Gower, 2005.

Stella Biderman, USVSN Sai Prashanth, Lintang Sutawika, Hailey Schoelkopf, Quentin Anthony, Shivanshu Purohit, and Edward Raf. Emergent and predictable memorization in large language models. *arXiv preprint arXiv:2304.11158*, 2023a.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR, 2023b.

Benjamin Bloem-Reddy and Yee Whye Teh. Probabilistic symmetries and invariant neural networks. *The Journal of Machine Learning Research*, 21(1):3535–3595, 2020.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 267–284, 2019.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*, 2022.

Shuxiao Chen, Edgar Dobriban, and Jane H Lee. A group-theoretic framework for data augmentation. *The Journal of Machine Learning Research*, 21(1):9885–9955, 2020a.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020b.

Alexandra Chouldechova. Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments. *arXiv:1610.07524*, 2016.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

Taco S Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant cnns on homogeneous spaces. *Advances in neural information processing systems*, 32, 2019.

Pedro Conceição and Pedro Ferreira. The young person's guide to the theil index: Suggesting intuitive interpretations and exploring analytical applications. 2000.

Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. Algorithmic decision making and the cost of fairness. In *Proceedings of KDD*, 2017.

Frank A. Cowell and Kiyoshi Kuga. Additivity and the entropy concept: an axiomatic approach to inequality measurement. *Journal of Economic Theory*, 25(1):131–143, 1981.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

Hugh Dalton. The measurement of the inequality of incomes. *The Economic Journal*, 30(119): 348–361, 1920.

Josip Djolonga, Jessica Yung, Michael Tschannen, Rob Romijnders, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Matthias Minderer, Alexander D'Amour, Dan Moldovan, Sylvain Gelly, Neil Houlsby, Xiaohua Zhai, and Mario Lucic. On robustness and transferability of convolutional neural networks. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16453–16463, 2021. doi: 10.1109/CVPR46437.2021.01619.

Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS)*, pages 214–226, 2012.

Rahim Entezari, Mitchell Wortsman, Olga Saukh, M Moein Shariatnia, Hanie Sedghi, and Ludwig Schmidt. The role of pre-training data in transfer learning. *arXiv preprint arXiv:2302.13602*, 2023.

Linus Ericsson, Henry Gouk, and Timothy M Hospedales. Why do self-supervised models transfer? investigating the impact of invariance on downstream tasks. *arXiv preprint arXiv:2111.11398*, 2021a.

Linus Ericsson, Henry Gouk, and Timothy M Hospedales. How well do self-supervised models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5414–5423, 2021b.

Facebook. `https://newsroom.fb.com/news/2017/02/improving-enforcement-and-promoting-diversit` 2017.

William Falcon and The PyTorch Lightning team. PyTorch Lightning, 3 2019. URL `https://github.com/Lightning-AI/lightning`.

Alhussein Fawzi and Pascal Frossard. Manitest: Are classifiers really invariant? *CoRR*, abs/1507.06535, 2015. URL `http://arxiv.org/abs/1507.06535`.

Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *ACM KDD*, 2015.

Anthony W. Flores, Christopher T. Lowenkamp, and Kristin Bechtel. False Positives, False Negatives, and False Analyses: A Rejoinder to "Machine Bias: There's Software Used Across the Country to Predict Future Criminals. And it's Biased Against Blacks.". 2016.

Mark J. Furletti. An overview and history of credit reporting, 2002. `http://dx.doi.org/10.2139/ssrn.927487`.

Jonas Geiping, Micah Goldblum, Gowthami Somepalli, Ravid Shwartz-Ziv, Tom Goldstein, and Andrew Gordon Wilson. How much data are augmentations worth? an investigation into scaling laws, invariance, and implicit regularization. *arXiv preprint arXiv:2210.06441*, 2022.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018a.

Robert Geirhos, Carlos R. M. Temme, Jonas Rauber, Heiko H. Schütt, Matthias Bethge, and Felix A. Wichmann. Generalisation in humans and deep neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018b. URL https://proceedings.neurips.cc/paper/2018/file/0937fb5864ed06ffb59ae5f9b5ed67a9-Paper.pdf.

Corrado Gini. Variabilità e mutabilità. *Reprinted in Memorie di metodologica statistica (Ed. Pizetti E, Salvemini, T). Rome: Libreria Eredi Virgilio Veschi*, 1912.

Ian J. Goodfellow, Quoc V. Le, Andrew M. Saxe, Honglak Lee, and Andrew Y. Ng. Measuring invariances in deep networks. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, NIPS'09, page 646–654, Red Hook, NY, USA, 2009. Curran Associates Inc. ISBN 9781615679119.

Divya Gopinath, Hayes Converse, Corina S. Pasareanu, and Ankur Taly. Property inference for deep neural networks, 2019. URL https://arxiv.org/abs/1904.13215.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR, 2020.

Anikó Hannák, Gary Soeller, David Lazer, Alan Mislove, and Christo Wilson. Measuring price discrimination and steering on e-commerce web sites. In *ACM IMC*, 2014.

Anikó Hannák, Claudia Wagner, David Garcia, Alan Mislove, Markus Strohmaier, and Christo Wilson. Bias in online freelance marketplaces: Evidence from TaskRabbit and Fiverr. In *ACM CSCW*, 2017.

Moritz Hardt, Eric Price, and Nathan Srebro. Equality of Opportunity in Supervised Learning. In *NIPS*, 2016.

Adi Haviv, Ido Cohen, Jacob Gidron, Roei Schuster, Yoav Goldberg, and Mor Geva. Understanding transformer memorization recall through idioms. *arXiv preprint arXiv:2210.03588*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJz6tiCqYm.

Katherine Hermann and Andrew Lampinen. What shapes feature representations? exploring datasets, architectures, and training. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9995–10006. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/71e9c6620d381d60196ebe694840aaaa-Paper.pdf.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

Kevin H Huang, Peter Orbanz, and Morgane Austern. Quantifying the effects of data augmentation. *arXiv preprint arXiv:2202.09134*, 2022.

Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

Jörn-Henrik Jacobsen, Jens Behrmann, Richard Zemel, and Matthias Bethge. Excessive invariance causes adversarial vulnerability. *arXiv preprint arXiv:1811.00401*, 2018.

Matthew Jagielski, Om Thakkar, Florian Tramer, Daphne Ippolito, Katherine Lee, Nicholas Carlini, Eric Wallace, Shuang Song, Abhradeep Thakurta, Nicolas Papernot, et al. Measuring forgetting of memorized training examples. *arXiv preprint arXiv:2207.00099*, 2022.

Matthew Joseph, Michael Kearns, Jamie Morgenstern, and Aaron Roth. Fairness in learning: Classic and contextual bandits. In *Proceedings of NIPS*, pages 325–333, 2016.

Nanak Kakwani. On a class of poverty measures. *Econometrica: Journal of the Econometric Society*, pages 437–446, 1980.

Sandesh Kamath, Amit Deshpande, and KV Subrahmanyam. Invariance vs robustness of neural networks. 2020. In *URL https://openreview. net/forum*, 2019.

Faisal Kamiran and Toon Calders. Classifying without Discriminating. In *IC4*, 2009.

Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma. Efficiency Improvement of Neutrality-Enhanced Recommendation. In *RecSys*, 2013.

Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. *arXiv preprint arXiv:1711.05144*, 2017.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.

Eugene Kharitonov, Marco Baroni, and Dieuwke Hupkes. How bpe affects memorization in transformers. *arXiv preprint arXiv:2110.02782*, 2021.

Minseon Kim, Jihoon Tack, and Sung Ju Hwang. Adversarial self-supervised contrastive learning. *arXiv preprint arXiv:2006.07589*, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. 2017.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK , August 23–28, 2020, Proceedings, Part V 16*, pages 491–507. Springer, 2020.

Serge-Christophe Kolm. Unequal inequalities. i. *Journal of economic Theory*, 12(3):416–442, 1976a.

Serge-Christophe Kolm. Unequal inequalities. ii. *Journal of Economic Theory*, 13(1):82–111, 1976b.

Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *International Conference on Machine Learning*, pages 2747–2755. PMLR, 2018.

Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2661–2671, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*, 2022.

Henry Kvinge, Tegan Emerson, Grayson Jorgenson, Scott Vasquez, Timothy Doster, and Jesse Lew. In what ways are deep neural networks invariant and how should we measure this? In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=SCD0hn3kMHw`.

Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. Data and analysis for 'How we analyzed the COMPAS recidivism algorithm'. `https://github.com/propublica/compas-analysis`, 2016.

Jure Leskovec, Lars Backstrom, and Jon Kleinberg. Meme-tracking and the dynamics of the news cycle. In *ACM KDD*, 2009.

Daliang Li, Ankit Singh Rawat, Manzil Zaheer, Xin Wang, Michal Lukasik, Andreas Veit, Felix Yu, and Sanjiv Kumar. Large language models with controllable working memory. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1774–1793, Toronto, Canada, jul 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.112. URL `https://aclanthology.org/2023.findings-acl.112`.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: **phi-1.5** technical report. *arXiv preprint arXiv:2309.05463*, 2023b.

Julie A. Litchfield. Inequality: Methods and tools. *World Bank*, 4, 1999.

Larry Long and Alfred Nucci. The hoover index of population concentration: A correction and update. *The Professional Geographer*, 1997.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. Analyzing leakage of personally identifiable information in language models. *arXiv preprint arXiv:2302.00539*, 2023.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pages 4768–4777, 2017.

Clare Lyle, Mark van der Wilk, Marta Kwiatkowska, Yarin Gal, and Benjamin Bloem-Reddy. On the benefits of invariance in neural networks, 2020. URL `https://arxiv.org/abs/2005.00178`.

Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schölkopf, Mrinmaya Sachan, and Taylor Berg- Kirkpatrick. Membership inference attacks against language models via neighbourhood comparison. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 11330–11343. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-acl.719. URL `https://doi.org/10.18653/v1/2023.findings-acl.719`.

Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. `https://github.com/deepmind/dsprites-dataset/`, 2017.

R Thomas McCoy, Paul Smolensky, Tal Linzen, Jianfeng Gao, and Asli Celikyilmaz. How much do language models copy from their training data? evaluating linguistic novelty in text generation using raven. *Transactions of the Association for Computational Linguistics*, 11:652–670, 2023.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Jakub Mikians, László Gyarmati, Vijay Erramilli, and Nikolaos Laoutaris. Detecting price and search discrimination on the internet. In *ACM HotNets*, 2012.

Tehila Minkus, Yuan Ding, Ratan Dey, and Keith W. Ross. The city privacy attack: Combining social media and public records for detailed profiles of adults and children. In *ACM COSN*, 2015.

Alan Mislove, Sune Lehmann, Yong-Yeol Ahn, Jukka-Pekka Onnela, and J. Niels Rosenquist. Understanding the demographics of Twitter users. In *AAAI ICWSM*, 2011.

Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *International conference on machine learning*, pages 10–18. PMLR, 2013.

Vedant Nanda, Till Speicher, Camilla Kolling, John P. Dickerson, Krishna P. Gummadi, and Adrian Weller. Measuring representational robustness of neural networks through shared invariances. In *ICML*, 2022.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *Advances in neural information processing systems*, 33:512–523, 2020.

Harald Niederreiter. *Random number generation and quasi-Monte Carlo methods*. SIAM, 1992.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library . In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning, 2017. URL `https://arxiv.org/abs/1712.04621`.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.

Arthur Cecil Pigou. *Wealth and welfare*. Macmillan and Company, limited, 1912.

John Podesta, Penny Pritzker, Ernest Moniz, John Holdren, and Jeffrey Zients. Big Data: Seizing Opportunities, Preserving Values. *Executive Office of the President. The White House.*, 2014.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Alexander J Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. Learning to compose domain-specific transformations for data augmentation. *Advances in neural information processing systems*, 30, 2017.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400. PMLR, 2019.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

Andrea Romei and Salvatore Ruggieri. A Multidisciplinary Survey on Discrimination Analysis. *KER*, 2014.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? *Advances in Neural Information Processing Systems*, 33:3533–3545, 2020.

Amartya Sen. *On economic inequality*. Oxford University Press, 1973.

Anthony F Shorrocks. The class of additively decomposable inequality measures. *Econometrica: Journal of the Econometric Society*, pages 613–625, 1980.

Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019. doi: 10.1186/s40537-019-0197-0. URL `https://doi.org/10.1186/s40537-019-0197-0`.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Vasu Singla, Songwei Ge, Basri Ronen, and David Jacobs. Shift invariance can reduce adversarial robustness. *Advances in Neural Information Processing Systems*, 34:1858–1871, 2021.

Koustuv Sinha, Amirhossein Kazemnejad, Siva Reddy, Joelle Pineau, Dieuwke Hupkes, and Adina Williams. The curious case of absolute position embeddings. *arXiv preprint arXiv:2210.12574*, 2022.

Congzheng Song and Vitaly Shmatikov. Auditing data provenance in text-generation models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 196–206, 2019.

Till Speicher, Muhammad Ali, Giridhari Venkatadri, Filipe Nunes Ribeiro, George Arvanitakis, Fabrício Benevenuto, Krishna P. Gummadi, Patrick Loiseau, and Alan Mislove. Potential for discrimination in online targeted advertising. In Sorelle A. Friedler and Christo Wilson, editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 5–19. PMLR, 23–24 Feb 2018a. URL `https://proceedings.mlr.press/v81/speicher18a.html`.

Till Speicher, Hoda Heidari, Nina Grgic-Hlaca, Krishna P. Gummadi, Adish Singla, Adrian Weller, and Muhammad Bilal Zafar. A unified approach to quantifying algorithmic unfairness: Measuring individual & group unfairness via inequality indices. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 2239–2248, New York, NY, USA, 2018b. Association for Computing Machinery. ISBN 9781450355520. doi: 10.1145/3219819.3220046. URL `https://doi.org/10.1145/3219819.3220046`.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Subbu Subramanian. Inequality measurement with subgroup decomposability and level-sensitivity. 2011.

Charles Summers and Tim Willis. Pretrial risk assessment, 2010.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of ICML*, pages 3319–3328, 2017.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Cong Tang, Keith Ross, Nitesh Saxena, and Ruichuan Chen. What's in a name: A study of names, gender inference, and gender behavior in Facebook. In *DASFAA*, 2011.

Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. In *Advances in Neural Information Processing Systems*, 2020. URL `https://proceedings.neurips.cc/paper/2020/file/d8330f857a17c53d217014ee776bfd50-Paper.pdf`.

Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. Transformer memory as a differentiable search index. *Advances in Neural Information Processing Systems*, 35:21831–21843, 2022.

Henri Theil. *Economics and Information Theory*. North Holland, Amsterdam., 1967.

Kushal Tirumala, Aram Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. Memorization without overfitting: Analyzing the training dynamics of large language models. *Advances in Neural Information Processing Systems*, 35:38274–38290, 2022.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Julius Von Kügelgen, Yash Sharma, Luigi Gresele, Wieland Brendel, Bernhard Schölkopf, Michel Besserve, and Francesco Locatello. Self-supervised learning with data augmentations provably isolates content from style. *Advances in neural information processing systems*, 34:16451–16467, 2021.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939. PMLR, 2020.

Maurice Weiler and Gabriele Cesa. General e (2)-equivariant steerable cnns. *Advances in Neural Information Processing Systems*, 32, 2019.

Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5028–5037, 2017.

Muhammad Bilal Zafar, Isabel Valera Martinez, Manuel Gomez Rodriguez, and Krishna P. Gummadi. Fairness Beyond Disparate Treatment & Disparate Impact: Learning Classification without Disparate Mistreatment. 2017a.

Muhammad Bilal Zafar, Isabel Valera Martinez, Manuel Gomez Rodriguez, and Krishna P. Gummadi. Fairness Constraints: Mechanisms for Fair Classification. 2017b.

R. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork. Learning Fair Representations. In *ICML*, 2013.

Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, Lucas Beyer, Olivier Bachem, Michael Tschannen, Marcin Michalski, Olivier Bousquet, Sylvain Gelly, and Neil Houlsby. The visual task adaptation benchmark, 2020. URL `https://openreview.net/forum?id=BJena3VtwS`.

Chiyuan Zhang, Daphne Ippolito, Katherine Lee, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. Counterfactual memorization in neural language models. *ArXiv*, abs/2112.12938, 2021. URL `https://api.semanticscholar.org/CorpusID:245502053`.

Richard Zhang. Making convolutional networks shift-invariant again. In *International conference on machine learning*, pages 7324–7334. PMLR, 2019.

Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 649–666, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46487-9.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Allan Zhou, Fahim Tajwar, Alexander Robey, Tom Knowles, George J. Pappas, Hamed Hassani, and Chelsea Finn. Do deep networks transfer invariances across classes? In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=Fn7i_r5rR0q`.