Improving Trustworthiness of Deep Learning via Inspectable and Robust Representations

Max Losch

A dissertation submitted towards the degree
Doctor of Engineering Science (Dr.-Ing.)
of the Faculty of Mathematics and Computer Science
of Saarland University

Saarbrücken, 2024.

Date of Colloquium:	11.04.2025
Dean of the Faculty:	Prof. Dr. Roland Speicher
Chair of the Committee:	Prof. Dr. Peter Ochs
Reviewers:	Prof. Dr. Bernt Schiele
	Prof. Dr. Mario Fritz
	Prof. Dr. Ullrich Köthe
Academic Assistant:	Dr. Ruta Binkyte

Dedicated to Siegfried – the past – and Emil – the future.

Abstract

An increasing number of Deep Learning systems are applied in the real world and have potential impact on people lives: autonomous cars, assisted medical diagnosis or social scoring. Attributable to training increasingly complex models on increasingly large datasets, these applications have become useful since they are trained to be accurate prediction machines. Typically, this is achieved by optimizing for accuracy only while disregarding two critical weak points of deep learning persisting since its invention. Firstly, the complexity of used models render it difficult to explain and understand causes for incorrect predictions – coined as *black box* property. Secondly, models are susceptible to *adversarial examples* – slight input perturbations otherwise imperceptible to humans – that can result in dramatic changes in predictions. While mitigation approaches exist, these are often expensive to train and hence are rarely deployed in practice. Both issues reduce the trustworthiness of deep learning and could dampen further adoption for real world problems. This thesis addresses both issues in two parts. In the first part, we discuss two ways to mitigate the risk to adversarial examples with a focus on reducing the computational overhead of conventionally used *adversarial training*: (i) training on data subsets and (ii) utilize *Lipschitz bounds* to enable certification against adv. examples. In the second part, we investigate Semantic Bottlenecks that explicitly align intermediate representations to human meaningful concepts like *feet*, *leg*, *wood*, etc. while reducing dimensionality to address the black-box issue and show that these bottlenecks can be useful for error analysis.

In detail, in part I of this thesis, we propose *subset adversarial training (SAT)* and a *calibrated Lipschitz-margin loss (CLL)*. Both, to mitigate the vulnerability to adversarial examples while reducing the computational overhead of conventional adversarial training. With SAT, we specifically investigate the use of training data subsets and their impact on robust accuracy. Here, we find that a small fraction of data can already achieve comparable robustness in comparison to full adversarial training. Given that adversarial training requires the construction of adversarial examples during training, entailing non-negligible computational overhead, we look at Lipschitz regularization in our final study. As an alternative method to increase robustness to adversarial examples, we observe that existing methods imply a reduction of complexity – impairing clean and robust accuracy. As a counter-measure we propose a new loss, that links this observation to slack, returns control over the complexity and consistently improves performance.

In part II, we propose Semantic Bottlenecks to improve the interpretability of intermediate representations. In this bottleneck, each dimension is explicitly aligned with a human concept. That is, we show quantitatively that representations of deep models have poor visual concept alignment, which renders inspection of failure modes difficult. In contrast, we desire models to ground their decision on semantically meaningful attributes, e.g. wheels and headlights to detect cars. We address this objective with a supervised and unsupervised bottleneck type, provide quantitative alignment improvements over baselines and show in a second study, how they can be used to analyze prediction errors and test predictions during evidence interventions. All conducted studies w.r.t. Semantic Bottlenecks utilize pixel-level concept annotations to train and quantify inspectability. In a third study, we investigate the dependency of models on spatial information and show that on image classification tasks, the last third of layers can be spatially reduced – allowing for easy integration of global-level concepts.

In summary, this thesis addresses the trustworthiness issue of Deep Learning, specifically the lack of interpretability and vulnerability to adversarial examples. We propose Semantic Bottlenecks to improve inspectability of intermediate representations and provide two methods to increase robustness while reducing the computational overhead of conventional methodology.

Zusammenfassung

Eine zunehmende Anzahl von Deep-Learning-Systemen wird in der realen Welt eingesetzt und hat potenzielle Auswirkungen auf das Leben der Menschen: autonome Autos, unterstützte medizinische Diagnose oder soziale Bewertung. Aufgrund des Trainings von zunehmend komplexen Modellen auf immer größeren Datensätzen sind diese Anwendungen nützlich geworden, da sie darauf trainiert sind, genaue Vorhersagemaschinen zu sein. Typischerweise wird dies erreicht, indem nur für Genauigkeit optimiert wird, während zwei kritische Schwachstellen des Deep Learning seit seiner Erfindung außer Acht gelassen werden. Erstens macht die Komplexität der verwendeten Modelle es schwierig, die Ursachen für falsche Vorhersagen zu erklären und zu verstehen - in der Literatur als Black-Box-Eigenschaft bezeichnet. Zweitens sind Modelle anfällig für adversarial Examples - geringfügige Eingabeveränderungen, die für den Menschen sonst nicht wahrnehmbar sind - die zu dramatischen Änderungen in den Vorhersagen führen können. Obwohl es Ansätze zur Vermeidung gibt, sind diese oft teuer zu trainieren und werden daher selten in der Praxis eingesetzt. Beide Probleme reduzieren die Vertrauenswürdigkeit in Deep Learning und könnten eine weitere Adaption für reale Probleme dämpfen. Diese Arbeit befasst sich in zwei Teilen mit beiden Problemen. Im ersten Teil diskutieren wir zwei Möglichkeiten, das Risiko für adversarial examples zu mildern, wobei der Schwerpunkt auf der Reduzierung des Rechenaufwands des herkömmlich verwendeten adversarial Trainings liegt: (i) Training auf Daten-untermengen und (ii) Nutzung von Lipschitz-Grenzen zur Zertifizierung gegen adversarial Examples. Im zweiten Teil untersuchen wir semantische Bottlenecks, die explizit Zwischendarstellungen mit für den Menschen bedeutungsvollen Konzepten wie Füßen, Beinen, Holz usw. abgleichen und gleichzeitig die Dimensionalität reduzieren, um das Black-Box-Problem anzugehen, und zeigen, dass diese Bottlenecks für die Fehleranalyse nützlich sein können.

Im Detail schlagen wir in Teil I dieser Arbeit das Subset Adversarial Training (SAT) und den kalibrierten Lipschitz-Margin-Loss (CLL) vor. Beide dienen dazu, die Anfälligkeit für adversarial Examples zu mildern und gleichzeitig den Rechenaufwand des herkömmlichen adversarial Trainings zu reduzieren. Mit SAT untersuchen wir speziell den Einsatz von Trainings-untermengen und deren Auswirkungen auf die robust Accuracy. Hier stellen wir fest, dass bereits ein kleiner Bruchteil der Daten eine vergleichbare Robustheit im Vergleich zum vollständigen adversarial Training erreichen kann. Da das adversarial Training den Aufbau von adversarial Examples während des Trainings erfordert, was einen nicht zu vernachlässigenden Rechenaufwand mit sich bringt, betrachten wir in unserer abschließenden Studie die Lipschitz-Regularisierung. Als alternative Methode zur Erhöhung der Robustheit gegenüber adversarial Examples stellen wir fest, dass bestehende Methoden eine Reduzierung der Klassifikator-Komplexität implizieren - was die saubere und robuste Genauigkeit beeinträchtigt. Als Gegenmaßnahme schlagen wir einen neuen Loss vor, der diese Beobachtung mit Slack verknüpft, die Kontrolle über die Komplexität zurückgibt und die Leistung konsequent verbessert.

In Teil II schlagen wir semantische Bottlenecks vor, um die Interpretierbarkeit von Repräsentationen in Zwischenschichten zu verbessern. In diesem Bottleneck ist jede Dimension explizit zu einem menschlichen Konzept ausgerichtet. Das heißt, wir zeigen quantitativ, dass Repräsentationen in tiefen Modellen eine schlechte visuelle Konzeptausrichtung haben, was die Inspektion von Fehlermodi erschwert. Im Gegensatz dazu fordern wir, dass Modelle ihre

Entscheidung auf semantische Attribute stützen, z.B. Räder und Scheinwerfer zur Erkennung von Autos. Wir addressieren dieses Ziel mit einem überwachten und einem unüberwachten Bottleneck an, bieten quantitative Verbesserungen der Ausrichtung gegenüber den Baselines und zeigen in einer zweiten Studie, wie sie zur Analyse von Fehlermodi und zum Testen von Vorhersagen während Eingriffen in Beweise verwendet werden können. Alle durchgeführten Studien in Bezug auf semantische Engpässe nutzen pixelgenaue Konzeptannotationen, um die Inspizierbarkeit zu trainieren und zu quantifizieren. In einer dritten Studie untersuchen wir die Abhängigkeit von Modellen von räumlichen Informationen und zeigen, dass bei Bildklassifikationsaufgaben das letzte Drittel der Schichten räumlich reduziert werden kann - was die einfache Integration von globalen Konzepten ermöglicht.

Zusammenfassend befasst sich diese Arbeit mit dem Vertrauenswürdigkeitsproblem des Deep Learning, insbesondere dem Mangel an Interpretierbarkeit und der Anfälligkeit für adversarial Examples. Wir schlagen semantische Bottlenecks vor, um die Inspizierbarkeit von Zwischenrepräsentationen zu verbessern und bieten zwei Methoden zur Erhöhung der Robustheit an, während wir den Rechenaufwand der herkömmlichen Methodik reduzieren.

Acknowledgements

Pursuing *my* PhD has been the most exhausting, punishing and least gratifying endeavour in my life. However, in the process, I learned an immense skill set on how to do – and not to do – research, on how to – and not to – communicate, on how to structure and organize my work and own thoughts. While I gained a lot, in retrospect, I wished my younger self had achieved reward and encouragement earlier on. To, nonetheless, endure and stay motivated is not my accomplishment alone. My sincerest gratitude goes to many people that I shared my life with and that I had the pleasure to work alongside.

First and foremost, I want to thank my supervisors Bernt and Mario for their perseverance and support throughout our long journey and for guiding me to become a scientist. I also want to express my gratitude to Mario for finding encouraging words in difficult times.

Secondly, I want to thank my wife Maria and my son Emil who acted as a necessary counterbalance, therefore playing a central role in keeping me grounded.

I want to thank my colleagues at the Computer Vision and Machine Learning group and my collaborators at Bosch, Oliver and Dimitrios. Particularly, I want to thank David and Mohamed, who provided the much-needed energy and encouragement in the final years. I sincerely mean it: I wouldn't have finished it without you.

Big thanks go to the many people I had the pleasure to share time, space, thoughts or food with: Moritz, Bharat, Rakshith, Klaara, Jan, Paul, Yang, Philipp, Eldar, 2× Anna, Yongqin, Yue, Tribhu, Apratim, Stephan, Aymen, Wenjia, Evgeny, Hosna, Farzaneh, Alina, Nina, Sukrut, Andrea, Yaoyao, 2× Julian, Verica, Ahmed, Paul, Gerard, Ning, Anurag, Christopher, Garvita, Steffen, Margret, Di, Saurabh, Jan Eric, Mattia, Zhi, Qianru, Joon, Hossein, Jovita.

Finally, I want to thank my family – Manuela, Marlen and Norbert –, who believed in me unconditionally and listened.

Contents

Introduction 1

Todays deep learning methods deliver impressive performances on a wide range of vision tasks like classification, segmentation or object detection. Among other applications, it is used increasingly often for high stake applications like autonomous driving and medical diagnosis. Typically, this is done by providing large training datasets with annotated groundtruths and by minimizing prediction error of large models containing millions of parameters. However, while they attain very high accuracies, they entail safety risks that reduce the trust in these systems and may limit adoption. First, deep networks are easily fooled by adversarially crafted inputs: human imperceptible perturbations in the input image that flip the prediction with high confidence [BCM+13, SZS+14]. This is highly undesirable, particularly for autonomous driving, where malicious actors could intentionally influence systems in the real world, just by applying stickers to traffic signs or wearing prints on shirts [SEE+18, CCMC18, WLS+23]. While approaches like adversarial training [MMS+18] can mitigate this vulnerabilty, they require non-negligible computational overhead during training and only provide empirical robustness. That is, given enough compute and time, it might still be possible for an adversary to attack such systems successfully. Secondly, imagine a scenario like the following: a pedestrian is hit by a fully autonomous vehicle. Did the system miss the pedestrian, was the pedestrians appearance confusing, or was the system distracted by other signals in the input? Trained end-to-end by only minimizing the output error, these models do not lend themselves for trivial inspection. Specific failure reasons are hidden within millions of parameters and their non-linear relationships. This is referred to as the "black-box" problem in deep learning.

As a consequence, the basis for trust is still lacking in currently employed models. Despite their high prediction performance, automated predictions in high stake applications can have severe consequences for the patient, pedestrians or passengers, potentially resulting in injury or death. Therefore, the European Union demands a "right for explanation" [EC, Eur21] for safety critical systems, ensuring that deployers give reasonable guarantees about their systems and that consumers can make informed decisions. Thereby prompting more trustworthy systems that do not rely on plain prediction performance, but satisfy multiple properties like robustness to adversarial examples or interpretability [DVK17, Lip18].

In this dissertation, we focus on improving trust in deep learning models by addressing two of the discussed limitations: (I) we consider mitigation approaches against adversarial examples while reducing their computational overhead and investigate robustness *certification* and (II) we endow models with inspectable *Semantic Bottleneck* layers, containing human interpretable representations, that all subsequent model decisions are based upon. In particular, in part I, we focus on performing adversarial training on a subset of data only and investigate Lipschitz regularization as an alternative to adversarial training. The latter providing additional deterministic robustness certificates. In part II, we tackle inspectability of deep learning representations by means of reducing number of channels and aligning them to human interpretable concepts. The resulting Semantic Bottleneck layers are subsequently utilized to analyze prediction errors. In the following, we give a more detailed overview over the contents in each part and their contributions.

We have emphasized that trust in deep learning systems is a necessary condition for widespread adoption. However, the concept of trust is ambiguous and can be constituted

of various facets. It could be based on performance, whether the model can be understood mechanistically or whether it can provide explanations for its predictions [Lip18]. We review possible avenues in chapter 2. However, ultimately, it strongly depends on the observer (e.g. a human). Since trust as definition might be difficult to satisfy, in this dissertation, we follow the vision of trustworthy AI shaped by the European Union in the AI Act [Eur19, Eur21], that demands among other aspects: human agency and oversight, technical robustness and safety and transparency including explainable decisions. This is especially demanded of AI systems deployed in critical domains, like autonomous driving or assisted medical diagnosis. In this dissertation, we address technical robustness to input perturbations and inspectability of deep representations to enable human oversight and transparency. We outline problems and our contributions for the former in section 1.1 and for the latter in section 1.2 and list our publications in section 1.4.

1.1 Towards more Efficient Adversarial Robust Training

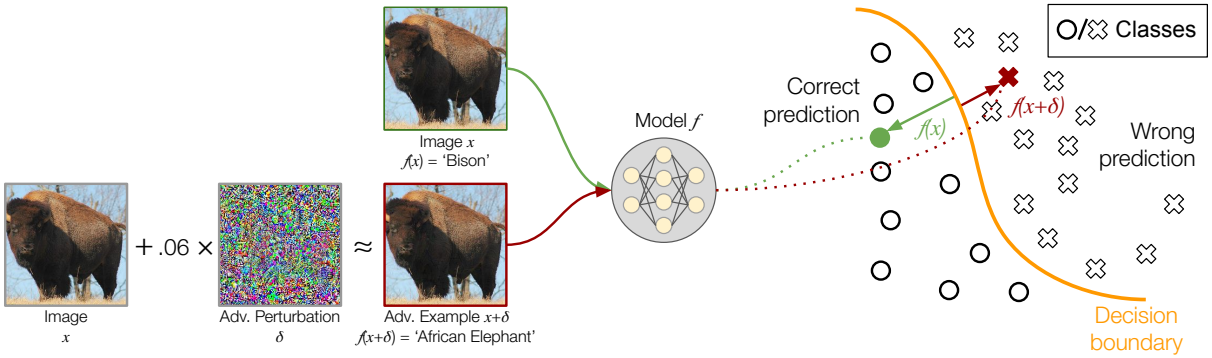


Figure 1.1: The vulnerability to adversarial examples: imperceptiple perturbations added to an image fool well performing models to predict incorrectly. An input image (green) is classified correctly by f, yet a corresponding adversarial example (red) lead to an incorrect prediction, indicated by the distance from the decision boundary (right).

Deep learning models are vulnerably to adversarial examples. Consider the toy example in figure 1.1 for a model f. A picture of a bison is fed into f and is correctly classified. Its input lies far away from the decision boundary, resulting in high confidence (see green arrow, right). Consider an adversarial example to the same picture, below (red). While the change δ is visually imperceptible (in figure 1.1, δ is scaled to make it visible), its classification is substantially altered, its placement w.r.t. the decision boundary far from the original example and on the wrong side (see red arrow, right), implying misclassification. Since its discovery [BCM+13, SZS+14], the predominant approach to improve robustness to adversarial examples is *adversarial training (AT)*. Adversarial Training (AT) improves robustness by generating adversarial perturbations (δ) during training, thereby augmenting the data. This approach has a main disadvantage though: perturbations are generated for all training examples for each training step, thus significantly increasing training time. As a means to reduce this dependency on considering all examples, in this dissertation, we consider two approaches. We consider generating perturbations for a subset of the training data only and we consider improving robustness with an alternative optimization approach: regularizing the Lipschitz constant of the model.

1.1.1 Subset Adversarial Training

Adversarial Training involves a multi-step adversarial attack applied on each example during training. Reducing its computational overhead, so far, has been investigated in two directions. Either by reducing the number of attack steps during training, or by perturbing only a fraction of training examples. While state-of-the-art robust performance is typically achieved with 7 to 10 steps, a single step has been shown to be sufficient [WRK20, dJABV+22] to reach competitive performances. While training instabilities have been shown to be an issue [AF20], single step attacks have successfully been used to achieve good robustness in multiple vision domains [], where large image sizes aggravate the computational overhead. Beyond reducing attack steps, [HZG+21] have investigated reductions in attacked examples. [HZG+21] use the examples closest to the decision boundary. Importantly, all existing methods may attack all examples during training. This is despite related work showing robustness is achieved easier for some classes than others [BZKK20, NDS+21, XCDX21, TKJ+21].

Contributions In chapter 3, we explore how groups of classes and examples contribute to the overall robustness. To this end, we construct Adversarial Examples (AEs) on predetermined subsets of the training examples. That is, we split the training set in two subsets A and B, train models on both ($A \cup B$) but construct AEs only for examples in A. Starting with A containing only a single class, we systematically increase the size of A and consider splitting by class and by examples. This enables a systematic analysis of robustness transfer between classes, examples and also tasks. The latter being of great interest for foundational models, where adversarial training is expensive due to model and dataset size.

We find surprisingly strong robustness transfer between classes and examples. Particularly, attacking only a single class provides non-trivial robustness on all other classes. This is in strong contrast to studies on the generalization of additive noise to images, for which transfer to out-of-distribution examples is generally poor [GTR+18, GJM+20, SVKT+21, OS22]. Moreover, we find that only 50% of examples are required to reach baseline robustness on all classes. In a task transfer setting, we observe this threshold to be even lower. Only 30% of training examples achieve near full baseline robustness on new tasks.

1.1.2 Lipschitz regularization

Alternatively to adversarial training, a wide range of methods have been proposed to improve robustness during training. Often justified by adversarial training only providing empirical robustness, many of these methods provide certified robustness [LXL23]. That is, for any given input, they give a guarantee whether an added perturbation within a radius will change the prediction. While methods like "randomized smoothing" [LAG+19, CRK19] must sample the input space exhaustively to only give probabilistic guarantees and hence are expensive during test time, alternatives like *interval bound propagation (IBP)* [GDS+19, ZCX+20] and *Lipschitz regularization* [TSS18] give deterministic guarantees which are as fast to evaluate as their non-robust counterpart models. Hereby, Lipschitz regularization in particular can achieve non-trivial robust certification for very deep models ($\geq$ 10 layers), which is difficult for IBP approaches [LXL23]. To certify a model f is robust for input x, the Lipschitz constant of f is utilized: a property all functions defined on an interval with bounded first derivative have. Any output translation (e.g. $f(x) - f(x+\delta)$) can be bounded relative to the input translation (δ with norm $p \geq 1$) by Lipschitz constant K:

$$||f(x) - f(x+\delta)||_p \leq K||\delta||_p.$$

While K is typically non-trivial to determine exactly for non-linear deep networks [VS18], an upper bound $\hat{K}$ is easy to compute. Its tightness can be substantially improved via regularizing the norm of layer weights [SZS+14, TSS18, LWF21] or by using orthogonal weights which have $K = 1$ by design [LHA+19, TK20]. The latter has established itself in recent years as state-of-the-art in terms of clean and certified robust accuracy [AHD+23], since very deep models (≥ 100) can be trained. Nevertheless, improvements on large datasets like ImageNet [DDS+09] have only recently provided significant gains [HZW+23, HLWF24].

Contributions In chapter 4, we take a look at how the value of $\hat{K}$ influences the smoothness of decision functions, which is theoretically linked to the Gaussian complexity – a measure of expressiveness – of a function [BM02]. We propose a new loss that can explicitly set a value for $\hat{K}$ and show that existing methods produce overly smooth classifiers – thereby impairing performance. Without requiring $\hat{K}$ to be small, we are able to train models with large Lipschitz constants that improve upon state-of-the-art results for CIFAR-100 [KH+09] and Tiny-ImageNet [LY14]. Moreover, in contrast to current trends, we unlock potential of much smaller models without hard $K = 1$ constraints.

1.2 Semantic Bottlenecks for Inspectable Representations

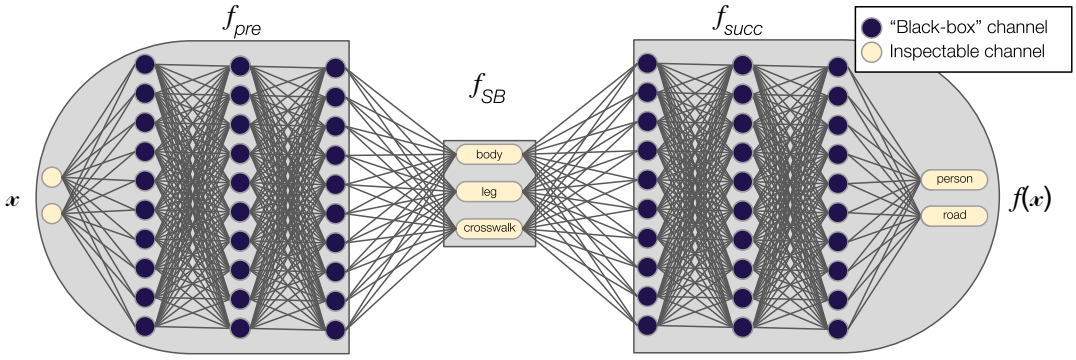


Figure 1.2: Thousands of channels in a deep model are not aligned with human meaningful concepts (dark circles), rendering the channels non-inspectable and the model a "black-box". Our Semantic Bottlenecks (SBs) introduce a bottleneck with few, explicitly concept-aligned channels (f_{SB}) that all final predictions are based on. This renders intermediate representations inspectable (cf. chapter 5) and enables error-analysis and evidence-intervention (cf. chapter 6).

In the second part of this dissertation, we have a look at another important issue with deep learning, reducing trust: its "black-boxness". In contrast to traditional computer vision involving hand-crafting feature detectors like SIFT or SURF [Low99, BTVG06], deep learning learns a hierarchy of feature representations purely from data in an end-to-end fashion. This has proven to be highly effective for training classifiers in the vision domain [LBH15, GBCB16], providing only a training set, a single score to optimize and a model with an appropriate inductive bias: convolutions [LBBH98] or vision transformers [DBK+20]. Alas, this power comes at a cost. The learned representations have no incentive to be interpretable and, as a consequence, are poorly aligned with human understandings of visual concepts. Initial work – to answer what representations are learned – studied the visualization of individual channels by means of inversion [ZF14], showing high pattern specificity for some channels. Indicating some alignment with human meaningful concepts, a later study proposed a quantification via measuring similarity to pixel-level image annotations corresponding to various visual concepts

like objects, parts, materials and colors [BZK+17], collected from various datasets. However, many channels remained unidentifiable and many of the matched channels were later shown to be less specific than initial results suggested [FV18]. In part II, we consequently press for explicit improvement of channel alignment and desire the number of channels to be reduced and each to be aligned via additional regularization or supervision signal. We call channels that satisfy these goals *inspectable*.

1.2.1 Quantifying and Improving Inspectability with Semantic Bottlenecks

To this end, we propose Semantic Bottlenecks (SBs). By constructing a model f as a composition of three functions, as depicted in figure 1.2 and formally via

$$f(x) = \left(f_{\text{succ}} \circ f_{\text{SB}} \circ f_{\text{pre}}\right)(x), \tag{1.1}$$

i.e., a preceding (deep) model f_{pre}, a succeeding (deep) model f_{succ} and a SB f_{SB}, we align channels of f_{SB} with concepts thus providing two properties: (i) the SB provides evidences (or lack there of) for known visual concepts, which (ii) all final decisions are based upon. To increase inspectability, we desire the number of channels in f_{SB} to be as low as possible while maintaining performance of the baseline.

Contributions In chapter 5, we show that the composition in equation 1.1 has negligible performance impact on the task of street scene segmentation, even though the dimensionality is reduced from thousands to only dozens of channels. We propose Supervised Semantic Bottlenecks (SSBs) and Unsupervised Semantic Bottlenecks (USBs) to improve alignment with visual concepts proposed in [BZK+17, XLZ+18]. The first, SSB, by explicitly minimizing alignment and the second, USB, by enforcing channel outputs to be one-hot encoded such that each channel becomes maximally specific and provides a complementary output. In contrast to aligning SSB-channels with additional annotations, USBs require no annotations. Moreover, we propose a *hyperparameter-free* metric to score the alignment between channel output and visual concepts and show that, vanilla models are poorly aligned, but that introducing SSBs or USBs improve this significantly up to a factor of six.

1.2.2 Error Analysis as a Use-Case of Semantic Bottlenecks

Large corpora of literature on explainable AI for computer vision have focused on providing explanations for *individual* predictions [SVZ14, BBM+15, RSG16, SCD+17, CLT+19, PHA+18, BFS22], yet lack a more global analysis of failure modes. In particular, while understanding errors on individual examples are important, we argue that high stake applications require a framework that can highlight error modes before they are deployed – spanning multiple examples of similar properties. Such a framework can pave the road to designing fail-safe systems, e.g. examples too similar to known error modes should be predicted with caution. While such a framework could theoretically be implemented on any model by clustering the receptive field on the input for each prediction and analyzing them, the input patch doesn't lend itself for easy inspection (which concepts are shown?). Moreover, the receptive field of a deep network generally encompasses the whole image, rendering clustering difficult (what distance metric is sensible?). SBs provide an entry point to this problem: for each prediction, the SB provides high level concept evidences and the receptive field on the bottleneck can be controlled to be small.

Contributions In chapter 6, we utilize Supervised Semantic Bottlenecks (SSBs) to conduct a use-case analysis on street scene segmentation – important for autonomous driving, as high stake application. We investigate the utility of gained concept evidences to analyze prediction errors and intervene evidences in the bottleneck layer to test whether final class predictions are based on the expected concepts. While we find that many concepts provide evidences for many classes, some concept groups related to class *person* or vehicles are particularly well delineated. This makes these classes ideal candidates for error inspection. Here, we show that via hierarchical clustering – agnostic to the number of clusters – SSBs can find useful error modes consisting of multiple input examples. Given the concept evidences of the bottleneck, we identify error modes that have competing or missing evidences w.r.t. true positive examples.

1.2.3 Analyzing Spatial Dependency in ConvNets

In a last study, originally conducted independently of the SB topic, we investigate the dependency of convolutional networks on spatial information. In contrast to the previous two chapters, we consider the task of image classification, reducing the output to a single instead of 2d prediction. Our findings have direct implications for potential adaptations of SBs for classification tasks: associating SB channels to concepts, can be done with global-level, instead of pixel-level, concept information. This is a consequence of our findings in chapter 7, as detailed in the following.

While we desire deep models to utilize global context information in image, as in: a face consists of two eyes, a nose, a mouth and all of them spatially arranged appropriately, the latter spatial integration has been shown to be particular brittle [BB19]. That is, training deep models on very small, disconnected patches of images (e.g. of size 17×17 instead of 224×224) results in comparable classification performance to full fledged baselines. Exhaustively applied to all disjoint patches, the information only needs to be averaged across all positions. This suggests that deep convolutional networks do not rely on global context information, but instead, learn strong local feature representations . This is supported by the observation, that convolutional models tend to rely on texture features instead of shape information [GRM+19]. The latter would require spatial integration over larger image regions.

Contributions In chapter 7, we complement these studies on spatial dependency by systematically investigating at what layer spatial information becomes irrelevant to the decision. To this end, we propose two different architectural modifications: (i) randomly shuffling convolutional layer outputs and (ii) applying global average pooling at an intermediate layer to collapse the spatial information for all succeeding layers. For the task of image classification, we find about 30% of the last convolutional layers can be ablated while retaining baseline performance.

1.3 Outline

This dissertation is organized in 8 chapters, divided in two parts.

Chapter 2, Related Work: We discuss previous and recent work on adversarial machine learning, corresponding to part I and discuss previous and recent work on explainable AI, corresponding to part II.

Part I, Towards more Efficient Adversarial Robust Training

Chapter 3, On Adversarial Training Without Perturbing all Examples: In contrast to existing work on adversarial training, we analyze its facets when perturbing fixed subsets of the training

data and observe that robust features generalize surprisingly well across classes and tasks.
This chapter is based on the ICLR publication [LOS+24]: Max Losch is the first author of this paper.

Chapter 4, Certified Robust Models with Slack Control and Large Lipschitz Constants: We discuss an alternative optimization approach to achieve adversarial robustness: regularization of a models Lipschitz constant. We show that existing methods may result in overly smooth classifiers and propose a loss that alleviates this issue.
This chapter corresponds to the GCPR publication [LSSF23]. Max Losch is the first author of this paper.

Part II, Inspectable Representations with Semantic Bottlenecks

Chapter 5, Semantic Bottlenecks: Quantifying and Improving Inspectability of Deep Representations: In order to lift the black-box property of deep models, we propose the integration of Semantic Bottlenecks, which improve alignment with visual concepts like objects, parts or materials. We show quantitatively, that Semantic Bottlenecks improve alignment over vanilla models substantially, while retaining prediction performance.
This chapter is based on the IJCV publication [LFS21]: Max Losch is the first author of this paper. This paper is a significant extension of a prior publication by the same authors [LFS20], which was nominated for best paper award at the DAGM GCPR 2020 and was presented at the Compositionality in Computer Vision workshop (CICV) held in conjunction with CVPR 2020.

Chapter 6, Semantic Bottleneck Clustering for Prediction-Error Analysis: A Use-case: In this chapter, we investigate the utility of Semantic Bottlenecks for analyzing prediction errors on the task of street scene segmentation. We perform sanity checks on the concept-to-class relation by means of intervening bottleneck evidences and find three typical error categories: (i) image misinterpretation, (ii) conflicting evidences and (iii) missing evidences.
This chapter is based on the pre-print [LFS19]: Max Losch is the first author of this paper. This paper was presented at the international computer vision summer school (ICVSS) in 2019 and won the best presentation prize. It was also presented at the Computer Science in Cars Symposium (CSCS) 2019. Since its pre-print publication, large parts of the manuscript were rewritten and additional experiments were conducted to improve comprehension and fit for this dissertation.

Chapter 7, Analyzing the Dependency of ConvNets on Spatial Information: In a parallel study to our Semantic Bottlenecks, we investigate how much models ground their decision on spatial information. We find that roughly 30% of the last convolutional layers can be spatially collapsed without impairing performance. This has direct consequences for Semantic Bottlenecks, which could be constructed with image-level- instead of pixel-level annotations to reduce the dependency on costly, dense annotations.
This chapter is based on the GCPR publication [FXLS20]: Max Losch was co-author, with *Yue Fan* being the first author. Max Losch was in a supervising role, contributing to the conceptualization, writing of the paper and contributed a piece of source code to improve run-time efficiency of the spatial shuffling operation.

Chapter 8, Conclusion and Future Work: In the final chapter, we summarize the key insights of this thesis and discuss promising future work directions.

1.4 Publications

[LOS+24] Max Losch, Mohamed Omran, David Stutz, Mario Fritz and Bernt Schiele. On Adversarial Training without Perturbing all Examples. *In Proc. of the International Conference on Learning Representations (ICLR)*, 2024.

[LSSF23] Max Losch, David Stutz, Bernt Schiele and Mario Fritz. Certified Robust Models with Slack Control and Large Lipschitz Constants. *In Proc. of the German Conference on Pattern Recognition (GCPR)*, 2023.

[LFS21] Max Losch, Mario Fritz, and Bernt Schiele. Semantic bottlenecks: Quantifying and Improving Inspectability of Deep Representations. *International Journal of Computer Vision (IJCV)*, 2021.

[LFS20] Max Losch, Mario Fritz, and Bernt Schiele. Semantic bottlenecks: Quantifying and Improving Inspectability of Deep Representations. *In Proc. of the German Conference on Pattern Recognition (GCPR)*, 2020.

[FXLS20] Yue Fan, Yongqin Xian, Max Losch and Bernt Schiele. Analyzing the Dependency of ConvNets on Spatial Information. *In Proc. of the German Conference on Pattern Recognition (GCPR)*, 2020.

[LFS19] Max Losch, Mario Fritz and Bernt Schiele. Interpretability Beyond Classification Output: Semantic bottleneck networks. *arXiv.org*, abs/1907.10882, 2019.

2 Related Work

Contents

Deep learning is pushing the boundaries of artificial intelligence and machine learning further and further at a break neck speed – thereby providing tremendous opportunities for human society. Importantly, however, deep learning models also come with diverse risks in practice: predictions are typically not explainable; they don't come with a measure of uncertainty; they can be biased w.r.t. sensitive inputs like gender or income and are generally vulnerable to slight input perturbation attacks. Depending on the use case, aforementioned aspects are only a subset of risks [Var19, KR19, HKR+20, MKN+23] reducing the trust in systems incorporating deep learning.

Many of these issues are out of scope for this thesis, illustrating the complexity of utilizing deep learning in practice. In this chapter, we review literature on two of the key challenges for trustworthiness: adversarial robust deep learning (section 2.1) and explainable deep learning (section 2.2), thereby integrating our work in part I and II to relevant previous and current research.

2.1 Adversarial Robust Deep Learning

Despite strong performances on vision tasks, some aspects of generalization of deep learning models remain elusive. One of the most peculiar issues are Adversarial Examples (AEs) – slight, visually imperceptible input perturbations, that drastically change a models prediction. This phenomenon is somewhat unexpected given the otherwise strong generalization performances on inputs drawn from similar distributions. In this section, we summarize the history and background on adversarial examples and defenses that have been proposed to mitigate this vulnerability. We organize this section into three subsections. First, we discuss ways to construct adversarial examples, called adversarial attacks. Then, we discuss methods to defend against these attacks, in which we focus on adversarial training – providing empirical robustness – and Lipschitz regularization – providing certified robustness. At last, we discuss intriguing properties and limitations that follow from applying these defense mechanisms.

2.1.1 Adversarial Attacks

Initially discovered as an intriguing property of deep networks, adversarial examples have initially been thought to be "extremely" rare negatives [SZS+13]. However, the ease of generation and their generalizability across many different architectures sparked an interest in the research community. Contrary to expectation, deep models appeared to learn brittle feature representations [SZS+13, GSS15]. Significant performance impairments, down to 0% accuracy, were the consequence. In an attempt to improve generalization of features and robustness, various metrics to measure adversarial robustness were investigated [MDFF16, MMS+18, CH20b].

White-box Attacks The most direct AE generation requires access to the models gradient. This is termed as white-box attacks. Initially, perturbations were constructed to be as small as possible [SZS+13, MDFF16], later they were constrained to remain within some small L_p norm [CW17, MMS+18]. This constrained generation remains state-of-the-art, with attack success rate improvements via including momentum terms [DLP+18] or adaptive step-sizes [CH20b]. While attacks for a wide range of L_p norms have been proposed [PMJ+16, BRK+19, MMDF19, PRBB21], L_2 and L_∞ remain most intensively studied. Since access to gradients is possible, most of the proposed white-box attacks can be targeted, with explicit target label, or untargeted. As a curious side-effect, white-box attacks have been shown to enable reprogramming of models [EGSD18].

In chapter 3 we utilize the white-box attack framework of [MMS+18] and its derivative [CH20b] to compare adversarial robustness across models.

Black-box Attacks In addition to white-box attacks, *black-box* attacks have been shown to successfully impair performance without access to gradients – only by querying predictions. Hereby, input perturbations are either sampled from an L_p constrained space [BRB18, GGY+19, ACFH20, LJL+20, CH20a] or AEs are transferred from a known, accessible model. This generalization of AEs is a well known phenomenon that has been intensively studied [LCLS17, TPG+17, CW17, DMP+19, WWX+20]. The transfer attack vulnerability is aggravated by the observation, that a target model can be stolen, simply by probing predictions with – potentially unrelated images [OSF19]. In practice, black-box attacks pose likely the greatest threat to deployed deep models. To standardize comparisons, various toolboxes have been developed [EIS+19, CH20b] and are used to benchmark model robustness [CAS+20, DFY+20].

Similar to the white-box attacks, we quantify robustness of our models in chapter 3 with existing black-box attack toolboxes [CH20b].

Other attacks While typical attacks consider perturbing each input individually, it is also possible to craft universal adversarial perturbations [MDFFF17, MGB18], that can be added to all inputs independent of their label. Furthermore, while above described attacks manipulate the whole image within some L_p constraint, various alternative methods attack models by perturbing small image patches [BMR+17, KZG18, SEE+18, ZZRP19, Hay18, RSS20]. These types of attacks can more readily be transferred to the physical space, e.g. by printing them on stickers or clothing. Models acting in the real world, e.g. autonomous vehicles, have been shown to be vulnerable to such attacks [SEE+18, CCMC18, ZZL+19, WLS+23].

2.1.2 Defenses

To defend against adversarial attacks, it has been proven useful to extend the empirical risk minimization framework by either crafting adversarial examples in an inner optimization loop during training (adversarial training), by augmenting inputs with Gaussian noise (randomized

smoothing) or by regularizing the spectral norm of a models weights (Lipschitz regularization). In the following, we discuss adversarial training as a means to improve empirical adversarial robustness and Lipschitz regularization as a means to improve certified robustness.

Adversarial Training Shortly after discovery of Adversarial Examples (AEs) in deep models [SZS+13], on-the-fly generation during training was proposed as a simple defense mechanism [GSS15]. Initially involving a single attack step and mixing clean and AEs, adversarial training gained traction with the multi-step attack proposed in [MMS+18] using AEs *only*. Various generalizations of this adversarial training framework have been proposed, pushing the robustness performance of models further. [ZYJ+19] add a Kullback-Leibler divergence between clean and AEs to the training loss, [WXW20] add adversarial perturbations to the weights of a model, [BGH19, DSLH20, WZY+20, KTSH21, ZZN+21a] adapt adversarial training to be instance aware and [AUH+19, CRS+19, GHvdO+21, XLHL20] incorporate self-supervised training. Alternative training strategies against universal adversarial perturbations have been proposed as well [SNX+20]. Despite improving robustness performances, three major challenges remain largely unresolved. First, adversarial training does not generalize well beyond the L_p constraint(s) used during training [LSF21, TB19, MWK20, PTSS21]. Second, robustness improvements usually involves a decrease in clean accuracy, coined the robustness-accuracy trade-off [SZC+18, ZYJ+19, TSE+19, YRZ+20]. Third, adversarial training requires on-the-fly generation of adversarial examples, thereby incurring a substantial increase in computational complexity. For the latter, multiple variants have been proposed that reduce the number of attacked examples [HZG+21, DEL22, KZS+22] or attack-steps [SNG+19, WRK20]. However, single-step attacks can lead to catastrophic overfitting [WRK20, AF20]. We continue the discussion of these challenges in section 2.1.3.

In chapter 3, we propose a generalization of the state-of-the-art adversarial training framework from [MMS+18] that enables perturbation on a subset of training examples only. Thereby enabling investigation on robustness transfer among classes, examples and tasks while reducing the computational complexity. We find that just $30 - 50\%$ of examples is sufficient to recover baseline adversarial robustness, revealing encouraging prospects for improving robustness of foundational models, where AT on the full dataset can be prohibitively expensive.

Lipschitz Bounds While AT is generally considered the standard method to improve robustness w.r.t. AE, it can only provide empirical robustness. I.e., AT does not provide robustness guarantees for all possible AE within a perturbation sphere of radius ϵ. Consequently, with enough time and compute at hand, there remains a chance to find AEs for any given input. To eliminate this risk entirely, it is possible to train models that are *certifiably* robust within the same ϵ-sphere. One of these methods is utilizing *Lipschitz bounds*. Instead of perturbing inputs, as with AT, this method regularizes or constrains the weights of the model such that the Lipschitz constant of the whole model is known [HA17]. Thereby bounding the maximum output change for a given input perturbation. First conjectured to be useful in [SZS+14], initially, local (input dependent) Lipschitz bounds where used to obtain certified robustness in [HA17] at the expense of additional calculations at inference. The use of global Lipschitz bounds has a key advantage over most other methods: certification at inference is deterministic and cheap to compute. Alas, it was originally considered intractable due to very loose bounds [HA17, VS18]. More recently, however, a number of methods allow robustness certification based on global Lipschitz bounds, including [LHA+19, ALG19, TSS18, SSF21, MDAA22, PL22, AHD+23, WM23, TK20, LWF21, LLP20, HZS+21, ZCL+21, ZJHW22]. These improvements can fundamentally be attributed to two independent developments: (i) preventing gradient norm attenuation by specialized non-linearities to preserve classifier complexity [LHA+19, ALG19] and (ii) architectural constraints that guarantee a Lipschitz constant

of 1 [LHA+19, TK20, SSF21, MDAA22, XLL22, PL22, AHD+23, WM23]. Additional work investigated the use of tighter bound estimation [VS18, LHA+19, FRH+19, JD20, HZS+21, DBAA23]. Recently, certification on large scale datasets like ImageNet have been addressed with some success [HZW+23, HLWF24]. Nevertheless, training deep Lipschitz networks with appropriate expressive power remains an open problem [HCC18, RWGM20].

In chapter 4, we approach this issue using a novel loss that provides more control over the global Lipschitz constant which bounds expressiveness [BM02] and provides control over the classifiers slack. We show that such a loss provides consistent performance improvement in both clean as well as robust accuracy over existing state-of-the-art.

Other defenses Beyond adversarial training and Lipschitz regularization, many alternative approaches for improving robustness exist [LHL15, ZNR17, JIDD17, JG18, RDV18, KKG18, PMG+18, WWZ+18, GRCvdM18, GDS+19, ZCX+20, MWYA19, JSZ+19, ZL19, YRZ+20, RNR20, CTOF20, SRPR20, AMRK20, AF21, MAKR21]. These approaches, however, are often difficult to scale to large datasets or deep networks [LXL23], have been broken by adaptive attacks [CW17, MAT+18, CH20b] or are outperformed by adversarial training or alternative certification methods. One major alternative, however, is *randomized smoothing*, providing probabilistic guarantees [LAG+19, CRK19, SSY+20, JPK+21, CTD+23]. While this certification approach scales well to large datasets like ImageNet [CTD+23] – leading the certification leaderboards [Li24] to date – it only provides probabilistic guarantees. I.e. typically a false positive certification rate of 0.1% is accepted. Additionally, the certification process at inference requires substantial computational overhead, by densely sampling noise around the input. This overhead is significant enough that randomized smoothing is typically evaluated on only $1\% - 10\%$ of the ImageNet validation data [CTD+23]. For a much more inclusive review of certification methods, we refer the interested reader to an exhaustive review on methods [LXL23].

2.1.3 Properties of Adversarial Robust Learning

While adversarial attack defenses endow models with increased input robustness, their application requires consideration of additional properties and implications. In this section, we outline the most important properties. Hereby, the trade-off observed between clean and robust accuracy [TSE+19, SZC+18] applies to all forms of defenses that require training or fine-tuning. Robust overfitting [RWK20] and robustness transfer [SIE+20, SSZ+20], however, generally applies to adversarial training only. Computational complexity provides an overview over every incurred costs and applies to all forms of defenses. And finally, we discuss the existence of a vulnerability of certified robust models.

Robustness-Accuracy Trade-Off Initially, it was conjectured, that training with adversarial examples could be understood as an effective data augmentation scheme [GSS15, MMKI18] – resulting in improved accuracies. Unfortunately, however, the opposite was observed. Adversarial training improves robustness, yet trades-off some clean accuracy [TSE+19, ZYJ+19, SZC+18]. [ZYJ+19] thus propose an extended loss to facilitate control over this trade-off. This robustness-accuracy trade-off has also been observed for increasing certified robustness [GRF+20]. Interestingly, while accuracy drops on the source task, multiple studies observed an opposite trend during task transfer [SSZ+20, SIE+20, YO22]. That is, finetuning robust models provides better clean accuracy on the target task than their non-robust counterpart.

In chapter 3, we investigate the robustness-accuracy trade-off during subset adversarial training. That is, perturbing only a fixed subset of examples. We complement existing literature in showing that clean accuracy improves on the target task proportional to the subset size,

while accuracy on the source task drops, as expected. Additionally, in chapter 4, we propose a reformulation of the cross-entropy loss that provides explicit control over the robustness-accuracy trade-off while achieving certified robustness.

Robust Overfitting Adversarial training has been observed to lead to robust overfitting [RWK20]: robust training accuracy decreases continuously, while robust accuracy on the test set increases. The simplest form of mitigation has been to stop training early [RWK20]. However, continuing research is trying to alleviate this generalization defect completely. One line of research links the flatness of the loss landscape to this issue and addresses adversarial weight perturbations [WXW20], weight averaging [CZL+20] with data augmentation [RGC+21]. Follow-up investigations provided support for this link between flatness and reduced robust overfitting [SHS21, AF22]. A second line of research links robust overfitting to memorization [DXY+21], observing that adversarial examples generated in a small ϵ ball do not lead to overfitting and that models can memorize adversarial examples with random labels.

In chapter 3, we utilize early stopping during adversarial training to mitigate robust overfitting as proposed in [RWK20].

Robustness Transfer Despite its drawbacks, AT also provides an opportunity: trained models transfer robustness to downstream tasks [SSZ+20, YO22]. That is, adversarially training on one task provides improved robustness on an auxiliary task, when finetuned. This has been shown to be the case, even without conducting additional adversarial training on the auxiliary task [SSZ+20]. Additionally, these models have been shown to provide better clean accuracy generalization on new tasks [SIE+20, JSK+23]. This indicates, that the learned robust features are overall more useful for a broad set of tasks.

In chapter 3, we complement these works by proposing subset adversarial training, thereby considering only constructing adversarial examples on a pre-defined subset of the training set. This setup enables an analysis on how robustness transfers across examples and tasks. We observe hard examples and classes to generally provide best robustness transfer and find 30-50% of attacked examples to be sufficient to reach baseline robustness performances.

Computational complexity While it is paramount for trustworthiness to maximize robustness, the efficiency of the defense method during training and inference is important for deployment. Adversarial training constructs adversarial examples for the whole training set, each requiring an additional inner optimization loop. Typically, this increases training time by a factor of up to 10, making AT a significant expense when training large foundational models with billions of parameters [BHA+21]. Research on reducing training time is highly active, proposing decreased number of inner optimization steps [SNG+19, WRK20, AF20, DSLH20, WZY+20, KTSH21] or reducing the number of attacked examples [HZG+21, DEL22]. However extra care needs to be taken during training, as training with few optimization steps can be brittle [WRK20, AF20].

Furthermore, randomized smoothing requires costly sampling of noise around the input during inference, which is significant enough, that inference takes seconds per image, instead of milliseconds on ImageNet data [CTD+23].

The added cost of these methods can limit the utilization of such defenses in practice. We consequently focus our investigation in this dissertation on Lipschitz bounds in chapter 4 which provides cheap to compute deterministic guarantees and consider subset adversarial training to analyze robustness gains on reduced data access in chapter 3.

Availability Attacks for Certifiers A fairly recent and under-investigated area of research is the evasion issue of certifiers like Lipschitz regularization or randomized smoothing. That is, while these defenses provide certificates w.r.t. a given input, it remains possible to craft input perturbations that force a model to abstain from classification [LKF23]. While this result is arguably better than providing a wrong decision, it remains a vulnerability of any system

deploying such certifiers. This can partially be mitigated via hierarchical certification [ALSF24]. I.e., if the predicted class cannot be certified, attempt to certify the superseding class (e.g. vehicle supersedes car).

2.2 Explainable and Interpretable Deep Learning

Deep learning on vision tasks has shown to be able to reach human level performances [MHG+14, HZRS15], yet its performance is often brittle during domain generalization [BMN+19, SvKT+22] and especially for adversarial examples [MMS+18]. Understanding the underlying principles of deep learning and improve upon existing methods motivated the advent of explainable deep learning. In literature this is often referred to in a more general term: *explainable AI* (XAI). Loosely, XAI attempts to understand how the model infers a prediction, what features this decision is based on and what features are prototypical (e.g. the bird *robin* has a red breast). Hereby exist two large areas of research focusing on either *explainability* or *interpretability*. Here, the term explainability is assigned an active trait, i.e. to act upon the model or data to clarify the decision logic [Gui22]. Interpretability, in contrast, is assigned a passive trait, i.e. a trait built into the model that makes sense to humans. The first two subsections considers related work on explainability, the remaining two consider interpretability.

For simple linear classifiers it is argued that the change in output is proportional to the change in input evidences, thus enabling straight forward explanations [Mol20]. However, for deep learning, the interaction between input and output is highly non-linear, rendering existing methodologies for linear classifiers to transfer poorly. In this section, we give an overview over existing methods that attempt to achieve explainable deep learning and highlight current trends. Hereby, we focus on XAI for the vision domain. Beyond, we refer the interested reader to a number of more complete surveys on the topic [DVK17, Lip18, Var19, JGB+20, HKR+20, LPK20]. We organize this section into 4 subsections: feature attribution, example based explanations, mechanistic interpretability and intrinsic interpretability.

2.2.1 Feature Attribution

One of the earliest trends to help explain predictions has been feature attribution methods. These methods attempt to highlight how much each input feature (pixel) contributes to the final decision. Feature attribution provides an intuitive explanation – the human brain can parse such visual information efficiently and make sense of it intuitively. Since models typically do not provide feature attribution by design, multiple methods have been proposed in recent years to generate attributions.

Backpropagation based In one of the first studies, it was observed that the image-gradients w.r.t. a class highlight salient pixels [SVZ14]. That is, changing pixels with large gradients, provide the largest change to the class score – a notion of feature importance that is considered desirable [SF19]. This idea was used for *GradCAM*, which calculates the gradients not for the image, but the last convolutional layer, producing a coarse attribution map [SCD+17]. However, Springenberg et al., found the gradients alone to be not representative enough to provide contributions [SDBR15]. The argument: rectified linear units (ReLUs), commonly used in neural networks, zero out negative contributions during the backward pass. This issue was found to be partially alleviated by calculating the product of gradient and input [BBM+15, SGK17, KSMD16] and completely alleviated by integrating gradients over a range of input values [STY17]. The latter, however, adding additional computational overhead.

To simplify, Shrikumar et al., proposed to calculate activations of neurons with respect to a *reference* [SGK17]. This reference, however, was found to be particularly critical in acquiring faithful contributions [ZLB+23] (more on that in the next subsection on *faithfulness*). Another issue with gradient based methods had been identified to be a neglect of the bias term [SF19] – which is dropped during calculating the gradients w.r.t. the image. Incorrect attribution to null features – features not contributing to the output score – has also been found to be an issue of most methods [RBS22, KKNN22].

Perturbation based Most of the previous methods attempt to reverse the accumulation of contributions during the forward process, thereby requiring detailed knowledge of the models internal processes. This can be avoided with model-agnostic methods, which perturb or permute features in the input, measure the change in output and assign these features an importance. *LIME* and *SHAP* implements this via masking out super-pixels [RSG16, LL17], *RISE* by masking random input features [PDS18] and *Occlusion* by masking rectangular patches [ZF14]. A drawback of all, however, is that feature masking results in unusual artifacts that themselves may confuse the model. *LIME* and *SHAP* additionally, calculate explanations on a linear *surrogate model*, which approximates the model in the local neighborhood for an input. This assumption may be dissatisfactory, given that the underlying model is highly non-linear and output scores can change with small input changes – e.g. adversarial examples. More involved, yet attempting to perturb inputs such that they remain within distribution, are prediction difference analysis [ZCAW17], Shapley-values [ŠK14], feature permutation analysis [FRD19] and extremal perturbations [FPV19]. Prediction difference analysis hereby sample input perturbations from a prior distribution derived from training data statistics, [ŠK14, FRD19] sample features from the training set and [FPV19] apply blurring. Overall, disregarding the computational overhead resulting from sampling, the issue with all permutation based approaches is the assumption that features are independent [AJL21]. This is aggravated for vision models, where pixels are strongly dependent on proximal as well as distant pixels. This issue can be partially addressed by incorporating additional heuristics for finding image regions that maximally change a prediction, e.g. extremal perturbations [FPV19]. More recent work on *information bottleneck attribution (IBA)* calculates input attributions by perturbing latent features. Along these lines, similar heuristics have been used to improve models alignment with feature attribution explanations by guiding attributions with bounding boxes [RBPAS23].

Faithfulness Importantly, while all listed methods attempt to explain the same input and model, they may provide different attributions. Moreover, [AGM+18] and follow-up work [KSA+18, KKNN22] showed that many methods are not *faithful* to the model they attempt to explain. They propose a set of sanity checks that every attribution method should satisfy. For example, when randomizing model weights, all methods are expected to provide random attributions. However, some methods have been found to largely retain their original explanation, indicating that they are much more dependent on salient regions in the input image than contributions of model weights. Quantitatively, faithfulness has been largely explored in two tangential directions. Either, by measuring whether attributions fall into expected input regions [ZBL+18, RBS22, HSMR23] (pointing game) or by measuring the performance impact when removing or perturbing input features that are attributed high relevance [SBM+16, ACÖG18, HEKK19]. The latter approaches come with various drawbacks: they either require retraining of models [HEKK19] – potentially changing the model to interpret significantly – or require expensive sampling procedures [SBM+16, ACÖG18]. This is aggravated by both methods not having controlled groundtruths, specifying what image regions *should* be important to the model. To establish more controlled settings for evaluation and provide attribution groundtruths, [KKNN22] and [HSMR23] proposed synthetic datasets

tailored to analyze attribution methods. To maximize control, it was recently proposed to also hand design the model itself [ZLB+23]. Overall, given the ambiguity of attributions and the lack of attribution-groundtruth in real image data, feature attribution in computer vision remains an active area of research with – so far – no unique solution. While feature attribution could prove to be a useful tool for analyzing model predictions, its utility needs to be proven in practice [MKR21]. However, for any method that is model unfaithful, trust in the model itself is not improved [CRA20].

2.2.2 Example Based Explanations

In contrast to providing additional attributions to a given image, example based explanations provide a set of images (one or many) that can help to understand a models prediction. Either by providing examples from the training set (influential examples) that had the largest contribution to minimize the loss for this decision, or by providing a set of similar images, but with a different target (counterfactuals).

Influential Examples Given that deep learning attempts to model the data distribution, an alternative research direction has been to look closer at the training data to elucidate the model's properties. In particular, it has been shown, that via first order Taylor expansion, the influence of a single training sample on the loss can be measured accurately under appropriate conditions [KL17]. This is based on *influence functions* [Lam81]. In a follow up study, this has been shown to generalize to groups of training samples [KATL19]. This line of work enables an appealing analysis of model behavior. E.g., it can highlight errors in the training set, debug models for discriminative biases [KSH22] and investigate training samples that degrade the models test performance [WZD+20]. However, concurrent work notes that such an analysis can be fragile for deep models, and needs to be used with caution [BPF20].

Counterfactual examples Consider the "right for an explanation" demand in the General Data Protection Regulation of the EU [EC]. For the human observer, a useful explanation is often a counterfactual one, which provides a minimal input change required to change the outcome – determined by minimizing a target loss [WMR17] –, or by learning a surrogate model [DCY+20, YACH21]. Thereby revealing information on missing evidences. It is to note, that the counterfactual description is similar to adversarial examples. However, while adversarial examples may leave the training data manifold [SHS19], counterfactual examples are intended to provide examples directly on it [WMR17]. While counterfactuals are gaining popularity in machine learning literature for tabular data [DMBB20, DCY+20, JKV+19, MST20, PBK20, WMR17, YACH21], in computer vision they are not straight forward to be utilized, due to the complexity of image data. Promising first directions combine counterfactuals with prototype clustering [VLK21], i.e. the mean representations for each class, or by utilizing invertible neural networks [MAKR21, HIA21]. However, the goal of finding on-manifold counterfactuals in the image space is challenging and highly ambiguous. In chapter 6, we propose to use clustering of Semantic Bottlenecks – a high level concept space –, which provides simple access to counterfactuals.

2.2.3 Mechanistic Interpretability

An alternative recent trend in deep learning – and especially large language models around the topic of *circuit discovery* [CCG+20, PBE+22, CMPL+23] – is to understand the mechanistic principles of information processing in deep networks: a concept termed *mechanistic interpretability*.

Mechanistic interpretability attempts to explain how individual neurons or representations work together to build more complex ones.

Individual neurons As first building block, [SVZ14, YCN+15, OSJ+18] visualized what individual neurons of an image classifier responded highly to. Thereby attempting to address whether semantic concepts like parts and objects emerge automatically in deep models without explicit training [ZKL+15]. And indeed, many neurons were found to only respond to highly specific input patterns like *wheels*, *eyes* or faces of *dogs*. However, concept assignment had only been based on visual appearance and were shown to be unreliable [GZB+23]. In an attempt to automate such assignments and find concepts for all neurons, [GGMF17] and [BZK+17] used hundreds of annotated visual concepts in hold-out datasets to determine which concepts the neurons respond most highly to and [MA20] investigated matching concept-combinations. While [GGMF17] found less than 20% neurons to systematically respond to concepts, [BZK+17] could improve assignments by utilizing a larger dataset with more concept annotations. Hereby, [BZK+17] and [KMB+21] observed that more concepts emerged in fully supervised models than partially- or self-supervised models, indicating that additional supervision could improve concept assignments. In chapter 5, we realize this observation by supervising individual neurons to align with visual concepts – thereby explicitly optimizing for interpretability.

Ensemble of neurons Results on matching individual neurons, however, revealed a large fraction of unidentifiable neurons [GGMF17, BZK+17]. Subsequently, [FV18] put forth the hypothesis that representations are not encoded by the individual neuron, but by a composition of many thus proposing to match linear combinations. Their contribution improved on the assignability of concepts, yet also highlighted two remaining issues. First, neuronal responses are assumed to be linear w.r.t. inputs, even though they are non-linear. That is, neurons are often not concept specific, but help encode multiple concepts [FV18]. Second, it is assumed that the human decomposition of the visual world in parts constitutes a meaningful representation. Whether this is shared with neuronal networks is not fully evident. Models have been shown to learn unexpected, yet valid solutions to the task of image classification, indicated by work around *short-cut-learning* [GRM+19, GJM+20, IST+19]. However, work by Kim et al., also showed statistically that many visual concepts are indeed important for a model [KWG+18]. Findings from [BZK+17, FV18] and [KWG+18] motivated a range of successful applications for generative models that can remove concepts to alter outputs predictably [BZS+19, BLW+20, GMFKB23]. Thus providing utility to the framework of concept identification. However, the underlying approaches require an expensive, curated dataset with concept annotations.

To alleviate the need for concept annotations, [GWZK19, ZMM+21, FPB+23, WLNN23] proposed methods for automatic concept discovery, based on clustering outputs [GWZK19, WLNN23] or by singular value decomposition [FPB+23]. However, these methods do not provide intuitive labels as text, only as visual image. This can be hard to parse for a human observer if the visual concept is ambiguous. With the onset of foundational large language models, recent literature consequently focused on describing representations utilizing natural language decoder, mapping outputs to text [MRSF23, OW23, DRTAA23, LWW+23].

Lastly, the current trend in interpreting models mechanistically moves away from labeling individual units to a more holistic point of view: by discovering circuits of representations fulfilling subtasks [OCS+20, CCG+20, PBE+22, CMPL+23]. In particular, given the steadily increasing size of foundational models, research on circuit discovery is often focused on investigating small, down-scaled models. Herein often lies the crux of mechanistic interpretability though: interpretation and analysis can be challenging if the model is too complex [MCB19]. This asks for new types of models that reduce this complexity, ideally by constructing models

that are intrinsically interpretable and (partially) alleviate the need for mechanistic interpretability.

2.2.4 Intrinsic Interpretability

As a natural extension to understanding the inner workings of complex deep models is the move towards constructing models that are interpretable by design. Typically, this requires changes to the architecture to improve *alignment* of information processing with intuitions from human understanding. Proposed approaches range from introducing *linearities* to *embedding concepts*, thereby integrating additional priors during training.

Linearized models Often, it is argued, that the non-linear nature of deep learning is undesirable for interpretability. While non-linearities introduce expressiveness and can solve non-linear problems (vision is considered one), they are more difficult to align with human intuitions. More intuitive are linear models, which combine inputs linearly to arrive at their output $f(a; W) = W^T a,\ a \in \mathbb{R}^N, W \in \mathbb{R}^{N \times M}$. That is, increasing the evidence for a single feature a_i, also increases the output linearly by its weight $a_i \cdot W_i$ (in log-space for logistic regression). Consequently, a range of work considers to infer the outputs *linearly* but the weights *non-linearly*. That is, [MJ18, BFS22, BFS23] compute variants of $g(a) = w(a)a$, where $w(a)$ is a non-linear function producing the weight matrix $w(a) : \mathbb{R}^N \to \mathbb{R}^{N \times M}$. For fixed input a, this computation provides an appealing advantage over the regular setup: the final model output is a linear product of input a and $w(a)$, which is retained even for deep models. In contrast to regular deep models, this lends itself for trivial input feature attribution, rendering previously discussed feature attribution methods redundant. However, as the weight matrix changes with the input, the weights themselves are more difficult to analyze and interpret mechanistically.

Concept embeddings To mitigate the need for post-training mechanistic interpretation, a broad range of work consequently proposed to embed individual layers of known concept representations. Either, by training intermediate concept detectors with additional supervision [ASDX20, BHJ18, KNT+20, CBR20, MPT22], by disentangling and whitening of representations [ALK+19, CBR20, ERO20, PPZ+20, WLNN23] or by clustering late stage outputs to find concept-prototypes for fine-grained image classification [CLT+19, LLCR18, HCLR19, DBC22]. Along these lines, ideas and implementations have been present even before the deep learning era, with *object bank* [LSFFX10] – classifying scenes on the outputs of a bank of pretrained object detectors. All of these concept embeddings have the benefit of providing presentations that are better aligned with human meaningful concepts and, in the generative case, offer direct access to probing representations to determine concepts post-hoc [ALK+19, ERO20, PPZ+20]. Knowing these representations provides a simple measure for the *presence* or *absence* of concept evidence, thereby enabling interpretation for individual images as well as globally, by clustering the concept space and by distilling the dependencies between concepts and classes.

In chapters 5 and 6, we propose a supervised and an unsupervised variant of such a concept embedding and show its usefulness for analyzing error modes for street scene segmentation. Thereby, our first manuscript on supervised Semantic Bottlenecks [LFS19] pioneered the use of concept embeddings in deep learning and was cited by highly influential follow-up work, in which one of our major contributions – map high dimensional representations to a few human meaningful concepts – has remained true [KNT+20, STE+21, WLNN23, MPT22, SN22, MCL+21, BH20, MFL+20, MLT19].

I

Part 1

In the first part of this dissertation, we address a fundamental flaw of deep learning, that severely reduces the trustworthiness in trained models: they are vulnerable to imperceptible input changes, resulting in drastic prediction changes with high certainty. In literature, this is referred to as adversarial robustness – the robustness to adversarially crafted input examples. To mitigate, adversarial training is the de-facto standard procedure which involves training the model additionally on adversarial examples. However, this comes at substantial cost to the training time, since such adversarial examples are crafted for every training example. To remedy, in this part, we investigate methods to reduce this dependency.

Specifically, in chapter 3, we find that crafting adversarial examples for a training subset only can recover baseline adversarial robustness with as low as 50% of data. When utilizing such models for task transfer, we find that just 30% of data is sufficient.

In chapter 4, we delve into the theory of Lipschitz constants of bounded, differentiable functions as a promising direction to improve adversarial robustness without the requirement to craft adversarial examples during training. Here, we observe that related work has focused on regularizing the Lipschitz constant K to be small, while theory predicts small constants to lead to overly smooth decision functions. We propose a reformulation of the widely used cross-entropy loss, which relaxes the dependency to minimize K and achieve improved clean and robust accuracies with larger K.

3 Adversarial Training without Perturbing all Examples

Contents

In this first it is the third chapter... you can write: in this first technical chapter - but I would suggest to simply write" 'In this chapter of the...' chapter of this dissertation, we have a look at adversarial training in order to improve robustness w.r.t. Adversarial Examples (AEs). In general, deep learning has been observed to be vulnerable to AEs to such an extent, that the accuracy on image classification tasks can be reduced to near 0% [SZS+13]. Such dramatic performance degradation can be mitigated with Adversarial Training (AT) [MMS+18], which is the de-facto standard for improving robustness against adversarial examples. This usually involves a multi-step adversarial attack applied on each example during training. In this chapter, we explore only constructing AEs on a subset of the training examples, thereby reducing the dependency to craft AEs exhaustively. That is, we split the training set in two subsets A and B, train models on both ($A \cup B$) but construct AEs only for examples in A. Starting with A containing only a single class, we systematically increase the size of A and consider splitting by class and by examples. We observe that: (i) adv. robustness transfers by difficulty and to classes in B that have never been adv. attacked during training, (ii) we observe a tendency for hard examples to provide better robustness transfer than easy examples, yet find this tendency to diminish with increasing complexity of datasets (iii) generating AEs on only 50% of training data is sufficient to recover most of the baseline AT performance even on ImageNet. We observe similar transfer properties across tasks, where generating AEs on only 30% of data can recover baseline robustness on the target task. We evaluate our subset analysis on a wide variety of image datasets like CIFAR-10, CIFAR-100, ImageNet-200 and show transfer to SVHN, Oxford-Flowers-102 and Caltech-256. In contrast to conventional practice, our experiments indicate that the utility of computing AEs varies by class and examples and that weighting examples from A higher than B provides high transfer performance.

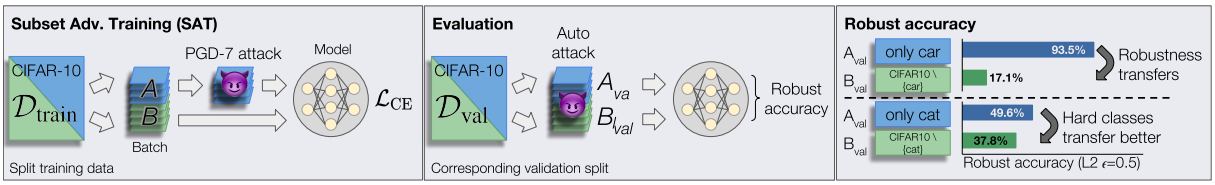


Figure 3.1: Adversarial robustness transfers among classes. Using Subset Adversarial Training (SAT), during which only a subset of all training examples (A) are attacked, we show that robust training even on a single class provides robustness transfer to all other, non adv. trained, classes (B). E.g., SAT for A=cat, we observe an robust accuracy of 37.8% on B. Noteworthy is the difference of transfer utility between classes. I.e. A=car provides very little transfer to B (17.1%). We investigate this transfer among classes and provide new insights for robustness transfer to downstream tasks.

3.1 Introduction

Imperceptible changes to an image can change the output of a well-performing classification model dramatically. These so-called Adversarial Examples (AEs) have been the focus of a large body on deep learning vulnerabilities of works since its discovery [SZS+14]. To date, Adversarial Training (AT) [MMS+18, CAP+19] and its variants [ZYJ+19, CRS+19, WXW20] is the de-facto state-of-the-art in improving the robustness against AEs. Essentially, AT generates adversarial perturbations for all examples seen during training. While adversarial training is known to transfer robustness to downstream tasks [SSZ+20, SIE+20, YO22] and that robustness is distributed unevenly across classes [TKJ+21, XCDX21], common practice dictates that AT "sees" adversarial examples corresponding to the whole training data, including all classes and concepts therein. This is independent of whether only adversarial robustness is optimized or a trade-off between robustness and clean performance is desired [SHS19]. This also holds for variants that treat individual examples differently [DSLH20, WZY+20, KTSH21] or adaptively select subsets to attack during training to reduce computational overhead [HZG+21, DEL22]. In this common setup, it is thus difficult to ascertain if a generic, transferable robustness is learned by the model, or if adversarial robustness is learned separately for each class in the training set. In the worst case, a model that can only learn to be robust on a class-specific basis would have to be presented with adversarially-perturbed images for every class of interest. A more preferable outcome, on the other hand, would be a model with *systematic adversarial robustness* [BMN+19, OS22]: i.e. a model that systematically generalizes to novel combinations of perturbations and classes that were separately encountered during training.

To shed light on this issue, we consider an analysis setup depicted in figure 3.1, we coin Subset Adversarial Training (SAT), where we split the training data into two subsets A and B, train the model conventionally on the union ($A \cup B$), but generate AEs only on examples from A (indicated by the emoji). For example, we can split training data by class, with $\mathrm{A} = \{car\}$ or $\mathrm{A} = \{cat\}$ and $\mathrm{B} = A^c$, and investigate how adversarial robustness transfers. Surprisingly, we observe significant adversarial robustness on B_{val} at test time, the degree of which depends on the class(es) in A. Of course, A and B can be arbitrary partitions of the training data. For example, we could put only "difficult" examples in A during training. At test time, we evaluate overall adversarial robustness (since there is no natural split into A_{val} or B_{val}). These experiments reveal a rather complex interaction of adversarial robustness between classes and examples.

Our analysis provides a set of **contributions** revealing a surprising generalizability of

robustness towards non-adv. trained classes and examples even under scarce training data setups. **First**, selecting subsets of whole classes, we find that SAT provides transfer of adversarial robustness to classes which have never been attacked during training. E.g. only generating adversarial examples for class *car* on CIFAR-10, achieves a non-trivial robust accuracy of 17.1% on all remaining CIFAR-10 classes (see figure 3.1, right). **Secondly**, we observe classes and examples that are hard to classify do generally provide better robustness transfer than easier ones. I.e. class *cat* achieves more than twice the robust accuracy on the remaining classes (37.8%) over class *car* (17.1%). **Thirdly**, SAT with 50% of training data is sufficient to recover the baseline performance with vanilla AT even on hard datasets like ImageNet. **Fourthly**, we observe similar transfer properties of SATed models to downstream tasks. In this setting, exposing the model to only 30% of AEs during training, can recover baseline AT performance on the target task. **Lastly**, while we observe promising amounts of transfer, there is room for improvement. Our SAT approach can be viewed as an analytical tool that can foster the development of models that exhibit stronger systematic generalization to adversarial perturbations.

3.2 Background and Method

3.2.1 Adversarial Training (AT)

It is a well known fact that conventional deep networks are vulnerable to small, often imperceptible, changes in the input. As mitigation, AT is a common approach to extend the empirical risk minimization framework [MMS+18]. Let $(x, y) \in \mathcal{D}_{\text{train}}$ be a training set of example and label pairs and θ be trainable parameters, then AT is defined as:

$$\min_{\theta} \mathbb{E}_{(x,y)\in\mathcal{D}_{\text{train}}} \left[\max_{||\delta||_2 \leq \epsilon} \mathcal{L}(x+\delta, y; \theta) \right], \tag{3.1}$$

where δ is a perturbation that maximizes the training loss $\mathcal{L}$ and thus training error. The idea being that, simultaneously to minimizing the training loss, the loss is also optimized to be stable within a small space ϵ around each training example $||\delta||_p \leq \epsilon$, $p \geq 1$. We consider the L_2 norm in our main paper and provide additional results for L_∞. This additional inner maximization is solved by an iterative loop; conventionally consisting of 7 or more steps. In some settings [GSS15, SHS19, ZYJ+19], the robust loss is combined with the corresponding loss on clean examples in a weighted fashion to control the trade-off between adversarial robustness and clean performance.

3.2.2 AT without Perturbing all Training Examples

Most proposed AT methodologies generate AEs on the whole training set. This being also valid for methods which adaptively select subsets [HZG+21, DEL22] during training or more traditional AT in which only a subset per batch is adversarially attacked. These methods do not guarantee the exclusion of examples, that is, the model is likely to see an AE for every example in the training set. From a broader perspective, the necessity to generate AEs exhaustively for all classes appears unfortunate though. Ideally, we desire robust models to be scalable, i.e. transfer flexibly from few examples and across classes to unseen ones [OS22]. We propose SAT to investigate to what extent AT provides this utility. To formalize, let A be a training subset and B contain the complement: $A \subset \mathcal{D}_{\text{train}}, B = \mathcal{D}_{\text{train}} \setminus A$. Then SAT applies

the inner maximization loop of AT on the subset A only; on B the conventional empirical risk is minimized:

$$\min_{\theta} \mathbb{E}_{(x,y)\in\mathcal{D}_{\text{train}}} \left[w_A \mathbb{1}_{(x,y)\in A} \max_{||\delta||_2\leq\epsilon} \mathcal{L}(x+\delta, y; \theta) + w_B \mathbb{1}_{(x,y)\in B} \mathcal{L}(x, y; \theta) \right], \tag{3.2}$$

where $\mathbb{1}_{(x,y)\in A}$ is 1 when the training example is in A and 0 otherwise. w_A and w_B define optional weights, which are by default both set to 1. Note that this is different from balancing robust and clean loss as discussed in [GSS15, SHS19, ZYJ+19], where the model still encounters adversarial examples on the whole training set.

Loss balancing. The formulation in equation 3.2 implies an imbalance between left and right loss when the training split is uneven ($|A| \neq |B|$). To counteract, we assign different values to w_A and w_B based on their subset size. E.g., to equalize, we assign $w_B = 1$ and $w_A = {}^{|B|}/_{|A|}$. We will utilize this loss balancing to improve robustness for transfer learning in section 3.3.4.

3.2.3 Training and evaluation recipes

Consider the depiction of SAT in figure 3.1. Prior to training, the training set is split into A and B (left). For evaluation (middle), we split the validation set into a corresponding split of A_{val} and B_{val}, if possible. For **Class-subset Adversarial Training (CSAT)**, this split aligns with the classes on the dataset: A and B are all training examples corresponding to two disjoint sets of classes while A_{val} and B_{val} are the corresponding test examples of these classes. As experimenting with all possible splits of classes is infeasible, we motivate splits by class difficulty where we measure difficulty by the average entropy of predictions per class – introduced as $\mathcal{H}_C$ in the next paragraph. In contrast, we can also split based on individual example difficulty. We provide empirical support for this approach in the experimental section 3.3. Additionally, example difficulty has been frequently linked to proximity between decision boundary and example [BMN21, DSLH20, KTSH21, HZG+21, ADH22]. The closer the example is to the boundary, the harder it is likely to classify. The hypothesis: hard examples provide a larger contribution to training robust models, since they optimize for large margins [DSLH20, WZY+20]. We refer to this experiment as **Example-subset Adversarial Training (ESAT)**. In contrast to CSAT, however, there is no natural split of the test examples into A_{val} and B_{val} such that we evaluate robustness on the whole test set (i.e., $\mathcal{D}_{\text{val}}$).

As difficulty metric, we utilize entropy over softmax, which we empirically find to be as suitable as alternative metrics (discussed in the next subsection 3.2.4). Consider a training set example $x \in \mathcal{D}_{train}$ and a classifier f mapping from input space to logit space with N logits. Then the entropy of example x is determined by $\mathcal{H}(f(x))$ and of a whole class $C \subset \mathcal{D}_{train}$ is determined by $\mathcal{H}_C(f)$ – the average over all examples in C:

$$\mathcal{H}(f(x)) = -\sum_{i=1}^{N} \sigma_i(f(x)) \cdot \log \sigma_i(f(x)), \quad \mathcal{H}_C(f) = \frac{1}{|C|} \sum_{x\in C} \mathcal{H}(f(x)),$$

where σ denotes the softmax function. For our SAT setting, we rank examples prior to adversarial training. This requires a classifier pretrained on $\mathcal{D}_{train}$ enabling the calculation of the entropy. To strictly separate the effects between entropy and AT, we determine the entropy using a non-robust classifier trained without AT. Similar to [ADH22], we aggregate the classifier states at multiple epochs during training and average the entropies. Let $f_1, f_2, ...f_M$ be snapshots of the classifier from multiple epochs during training, where M denotes the number

of training epochs. Then the average entropy for an example is given by $\overline{\mathcal{H}}(x)$ and for a class by $\overline{\mathcal{H}}_C(f)$:

$$\overline{\mathcal{H}}(x) = \frac{1}{M}\sum_{e=1}^{M}\mathcal{H}(f_e(x)), \quad \overline{\mathcal{H}}_C = \frac{1}{M}\sum_{e=1}^{M}\mathcal{H}_C(f_e). \tag{3.3}$$

3.2.4 Alternative rankings

While we focused our experiments on using entropy as a proxy to measure example and class difficulty (c.f. equation 3.3), alternative difficulty metrics have been discussed in literature [CLMM17, HZG+21, BMN21, ADH22]. We select the following from recent literature to compare to: signed variance (SVar) [HZG+21] and variance of gradients (VoG) [ADH22]. Figure 3.2 compares these two metrics with our used entropy metric using ESAT on CIFAR-100. Overall, *VoG* has a slight edge over *SVar* and *Entropy*, yet the differences remain small. On 5k attacked examples, *Entropy* (yellow line) achieves 21.0%, *VoG* (red line) 21.9% and *SVar* (purple line) 22.3% robust accuracy. On 25k attacked examples, *Entropy* achieves 38.0%, *VoG* 38.8% and *SVar* 38.1%. While some improvements over our simple *Entropy* metric appear possible, no proposed metric has a clear edge over the other.

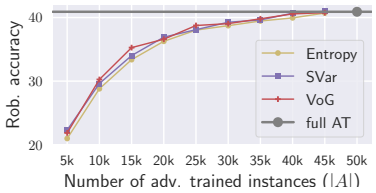


Figure 3.2: Various hardness metrics result in similar rob. accs. for ESAT on CIFAR-100.

3.3 Experiments

As aforementioned, common practice performs AT for the whole training set. In the following, we explore CSAT and ESAT, which splits the training set in two subsets A and B and only constructs AEs for A such that the model never sees AEs for B. We start with single-class CSAT – A contains only examples of a single class – and increase the size of A (section 3.3.2) by utilizing the entropy ranking of classes $\mathcal{H}_C$ (equation 3.3). ESAT, which splits into example subsets is discussed in section 3.3.3. Both SAT variants reveal complex interactions between classes and examples while indicating that few AEs can provide high transfer performance to downstream tasks when weighted appropriately (section 3.3.4).

3.3.1 Training and evaluation details.

General training For all training setups listed in table 3.2, we train our models from scratch using SGD with a momentum of 0.9. Dataset sizes are listed in table 3.1. All are data augmented based on the definitions in [EIS+19]. The sequence of transformations are listed in figure 3.3. Left, for CIFAR-10, CIFAR-100 and SVHN. Right, for ImageNet-200, Caltech-256 and Flowers-102.

Robust overfitting Since AT is prone to overfitting [RWK20], it is common practice to stop training when robust accuracy on a hold-out set is at its peak. This typically happens after a learning rate decay. We adopt this "early stopping" for all our experiments by following the methodology in [RWK20] but utilize Auto Attack (AA) to evaluate robust accuracy. Throughout the course of the training, we evaluate AA on 10% of the validation data $\mathcal{D}_{\text{val}}$ after each learning rate decay and perform final evaluation with the model providing the highest robust accuracy.

This final evaluation is performed on the remaining 90% of validation data. This AA split is fixed throughout experiments to provide consistency. If not specified otherwise, we generate adversarial examples during training with PGD-7 within an L_2 epsilon ball of $\epsilon = 0.5$ (all CIFAR variants) or $\epsilon = 3.0$ (all ImageNet variants) – typical configurations found in related work. We train all models from scratch and use ResNet-18 [HZRS16] for all CIFAR-10 and CIFAR-100 [KH$^+$09] experiments and ResNet-50 for all ImageNet-200 experiments. Here, ImageNet-200 corresponds to the ImageNet-A subset [HZB$^+$21] to render random baseline experiments tractable (to reduce training time). This ImageNet-200 dataset, contains 200 classes that retain the class variety and breadth of regular ImageNet, but remove classes that are similar to each other (e.g. fine-grained dog types). We use all training and validation examples from ImageNet [DDS$^+$09] that correspond to this subset classes.

Adversarial training AT for the L_2 norm is performed with 7 steps of projected gradient descent (PGD-7) within an $\epsilon_2 = 0.5$ for CIFAR and SVHN and $\epsilon = 3.0$ for ImageNet-200, Caltech-256 and Flowers-102. For each step, we use a step size of 0.1 and 0.5 respectively. For the L_∞ norm, we use 10 steps of PGD during ESAT and a step size of $1/255$ to avoid catastrophic forgetting when $|A|$ is small (see section A.5). For S-ESAT, we find it to be sufficient to perform PGD-7 with a step-size of $2/255$. This is likely mitigated by the weighted loss. For all datasets, we constrain the maximum perturbation norm to $\epsilon_\infty = 8/255$. For all experiments, including L_2 and L_∞, we maximize the default cross-entropy loss. For vanilla AT, all examples in a training batch are attacked. For all SAT variants, we randomly sample training examples to construct a training batch and attack only examples that are contained in A.

3.3.2 Class subset splits

We start by investigating the interactions between individual classes in A using CSAT on CIFAR-10, followed by an investigation on increasing the number of classes.

Single-class subsets (CSAT). We train all possible, single class CSAT runs (10) and evaluate robust accuracies on the adv. trained class (A) and the non-adv. trained classes (B). The results are shown in figure 3.4, left. Each rows represents a different training run. Note that the baseline robust accuracy, trained without AT achieves practically 0% (indicated by orange line). Most importantly, we observe non-trivial robustness gains for all classes that have never been attacked during training (*B*-sets). That is, irrespective of the chosen class, we gain at least 17.1% robust accuracy (A=*car*) on the remaining classes and can gain up to 37.8% robust accuracy when A=*cat*. These robustness gains are unexpectedly good, given many features of the non-adv. trained classes can be assumed to not be trained robustly.

- pad 4 pixels	
- random crop to 32x32	- random crop to 224x224
- random horizontal flip	- random horizontal flip
- color jitter [0.25, 0.25, 0.25]	- color jitter [0.1, 0.1, 0.1]
- random rotation within +/- 2 deg.	- random rotation within +/- 2 deg.

Figure 3.3: Input transformation for CIFAR and SVHN datasets (left) and ImageNet-200, Caltech-256 and Flowers-102 (right) during training. During testing, no transformations are applied to CIFAR and SVHN. The remaining datasets are resized such that the shortest side equals 256, after which they are center cropped to 224.

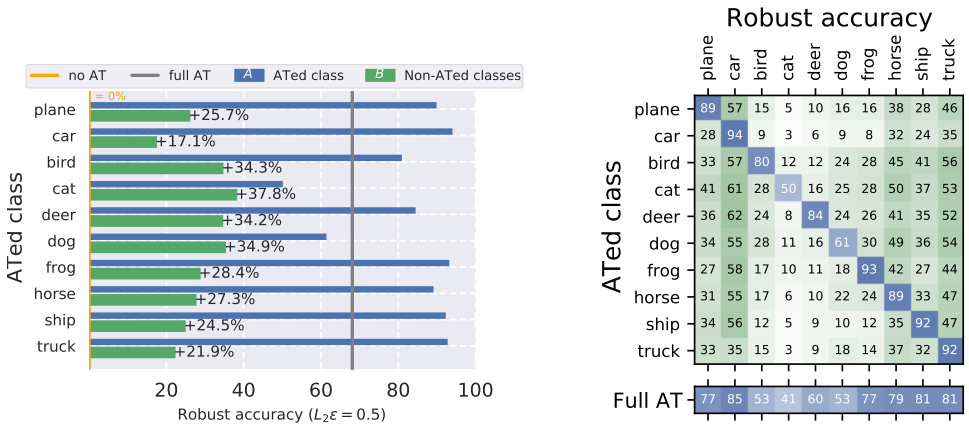


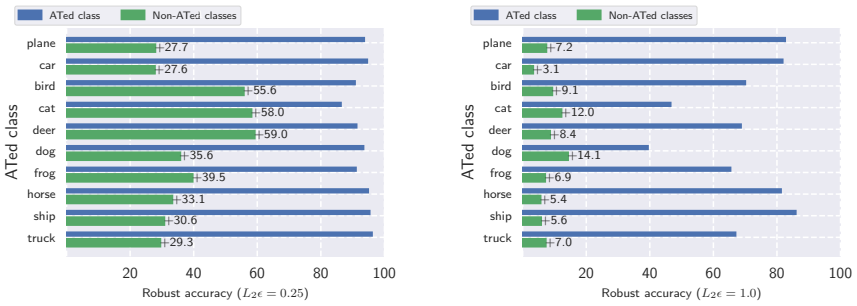
Figure 3.4: CSAT on a single CIFAR-10 class A (blue), we observe non-trivial transfer to the non-adv. trained classes B (green). Classes considered hard in CIFAR-10 (cat) offer best generalization (+37.8% gain on non-adv. trained), while easy classes offer the worst (car, +17.1% gained). Note that without AT, robust accuracy is close to 0% for all classes (orange). Right: same as left, but robust accuracy is evaluated per class (along columns). Here, we observe an unexpected transfer property: hard classes provide better transfer to seemingly unrelated classes (cat $\rightarrow$ truck: 53%) than related classes (car $\rightarrow$ truck: 35%). Additional results for $\epsilon = 0.25$ and $\epsilon = 1.0$ in figure 3.5.

This is consistent across different values for $\epsilon \in \{0.25, 1.0\}$, as shown in the additional figure 3.5. Here, we continue to observe non-trivial robustness transfer to B, irrespective of the value of ϵ. Interestingly though, the robustness transfer notably increases for smaller $\epsilon = 0.25$ (left), but also notably decreases for larger $\epsilon = 1.0$ (right). We conjecture, that with increasing ϵ, each feature representation becomes more class-specific and is thus less reusable for other classes. This could be aggravated by robust overfitting [DXY+21, RWK20] In contrast, with decreasing ϵ, the generalizability is improved.

To investigate this phenomenon further, we analyze robust gains for each individual class and present robust accuracies in the matrix in figure 3.4, right, where training runs are listed in rows and robust accuracies per class are listed in columns. Blue cells denote the adv. trained class and green cells denote non-adv. trained classes. While we see some expected transfer properties, e.g. CSAT on *car* provides greater robust accuracy on the related class *truck* (35%) than unrelated animal classes *bird*, *cat*, *deer*, *dog* (between 5% and 16%), the reverse is not

Dataset	Classes	Size (Train/Test)
CIFAR-10 [KH+09]	10	50 000 / 10 000
CIFAR-100 [KH+09]	100	50 000 / 10 000
ImageNet-200 [DDS+09, HZB+21]	200	259 906 / 10 000
Caltech-256 [GHP07]	257	24 485 / 6122
Flowers-102 [NZ08]	102	1020 / 1020
SVHN [NWC+11]	10	73 257 / 26 032

Table 3.1: Number of training and validation examples per dataset used. ImageNet-200 uses examples from [DDS+09] only for classes defined in [HZB+21]



(a) $\epsilon = 0.25$ (b) $\epsilon = 1.0$

Figure 3.5: Our main observations in figure 3.8 remain valid for alternative ϵ-values: robustness transfers above non trivial levels to B. Beyond, we observe transfer to be reduced with increasing ϵ. For smaller ϵ though, the transfer is notably stronger. For $\epsilon = 1.0$ we use a step-size of 0.2 and 0.1 otherwise.

straight-forward. CSAT on *bird* provides 56% robust accuracy on the seemingly unrelated class *truck*, 20%-points more than CSAT on *car*. More generally, animal classes provide stronger robustness throughout all classes than inanimate classes. We observe, that these classes are also harder to classify and have a higher entropy $\overline{\mathcal{H}}_C$ as shown in figure 3.6. This influence of class-entropy might also be utilized to augment the dataset, by adding high entropy samples. We provide a simple proof-of-concept in the following, adding an 11th class to CIFAR-10.

Dataset augmentation – CIFAR-10+1 As observed, we can utilize the class-entropy to estimate the robustness transfer. Here, we investigate whether it is possible to augment a dataset with a high entropy class and perform CSAT only on this new class. We anticipate to see strongest robustness transfer to the original dataset from hard classes and weakest robustness transfer from easy classes. As a proof-of-concept, we synthesized a set of 11th classes from CIFAR-100s super-classes (see figure A.3 in the appendix) – of which there are 20 – and perform SAT on this 11th class to evaluate the robust accuracy gains on the original CIFAR-10 classes. Results are reported in figure 3.7. For each added class along the x-axis, we report the entropy $\overline{\mathcal{H}}_c$ for the new class (green) as well as the distribution over all CIFAR-10 classes (blue). The orange line shows the robust accuracy on B: CIFAR-10. We continue to observe a correspondence between average entropy $\overline{\mathcal{H}}_c$ (determined on non-adv. trained models) and the robustness transfer of a class. Note that, while the best performing setup ($A = \{rodent\}$) with rob. acc of 33.3% does not improve upon the best above ($A = \{cat\}$, with a rob. acc $> 37.8\%$), the number of examples in A is only $|A| = 2500$, thus less than 5% of training data. This provides an indication that such a dataset augmentation is possible.

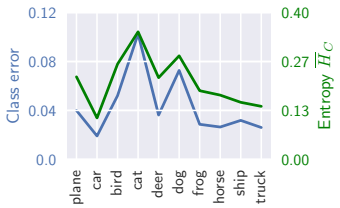


Figure 3.6: The hardest classes (blue) have the highest entropy (green).

Many-class subsets (CSAT). Going back to our original setup without augmentation, we increase the number of classes in A while maintaining a minimal computational complexity, we utilize the average class entropy $\overline{\mathcal{H}}_C$ proposed in equation 3.3 to inform us which ranking to select from. To improve clarity, we begin with a reduced set of experiments on CIFAR-10 before transitioning to larger datasets. We utilize the observed correlation between class difficulty, average class entropy and robustness transfer $\overline{H}_C$ to rank classes and construct 4 adv. trained

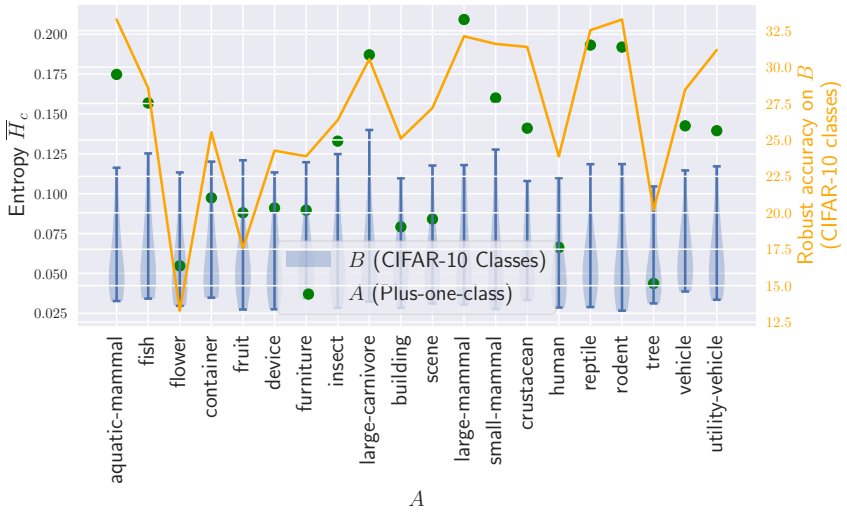


Figure 3.7: CSAT on CIFAR10 plus an additional class, synthesized from CIFAR-100. A consists only of this additional 11th class, to test how much robust accuracy can be gained on the original CIFAR-10 classes (orange). Green displays the single entropy on A, blue displays the entropy distribution over classes in B. We observe a consistent link with the entropy of this 11th class (green) with respect to the entropy of the CIFAR-10 classes (blue). The higher $\mathcal{H}_c$, the higher the robust accuracy.

subsets. Ranked by class entropy $\overline{\mathcal{H}}_C$, we select 4 subsets showing in figure 3.8, left. As observed before, *cat* and *dog* are hardest and thus first chosen to be in subset A. *Truck* and *car* on the other hand are easiest and thus last. To gauge the utility of this ranking, we provide a robust and clean accuracy comparison with a random baseline in figure 3.8, center and right. I.e., for each subset A we select 10 random subsets and report mean and std. deviation (red line and shaded area). Similar to the single-class setup, we observe subsets of the hardest classes to consistently outperform the random baseline (upper middle plot), up until a subset size of $|A| = 8$, when it draws even. Also note that the robust accuracy on B_{val} is improved across all splits, thus providing support that harder classes – as initially observed on animate vs inanimate classes – offer greater robustness transfer.

For our experiments on larger datasets like CIFAR-100 and ImageNet-200, we additionally evaluate a third ranking strategy. Beside selecting at random and selecting the hardest first, we additionally compare with selecting the easiest (inverting the entropy ranking). We construct 9 subsets per type of ranking (instead of 4) and report robust accuracies for selecting the easiest classes as well. Results are presented in three columns in figure 3.9; one dataset per column. As before, we show robust accuracies on the tested dataset (upper row) and robust accuracies on B_{val} (lower row). For CIFAR-10, we calculate mean and std. dev. over 10 runs, for CIFAR-100 over 5 runs and for ImageNet-200 over 3 runs. Selecting hardest first (highest entropy) is marked as a solid line and easiest first (lowest entropy) as a dashed line. First and foremost, we observe that irrespective of the dataset and the size of A, we see robustness transfer to B_{val}. This transfer remains greatest with classes we consider hard, while easy classes provide the least. Nonetheless, we see diminishing returns of such an informed ranking when dataset complexity is increased. E.g. the gap between dashed and solid line on ImageNet-200 is small and random class selection is on-par with the best. The results are similar on CIFAR-100, as

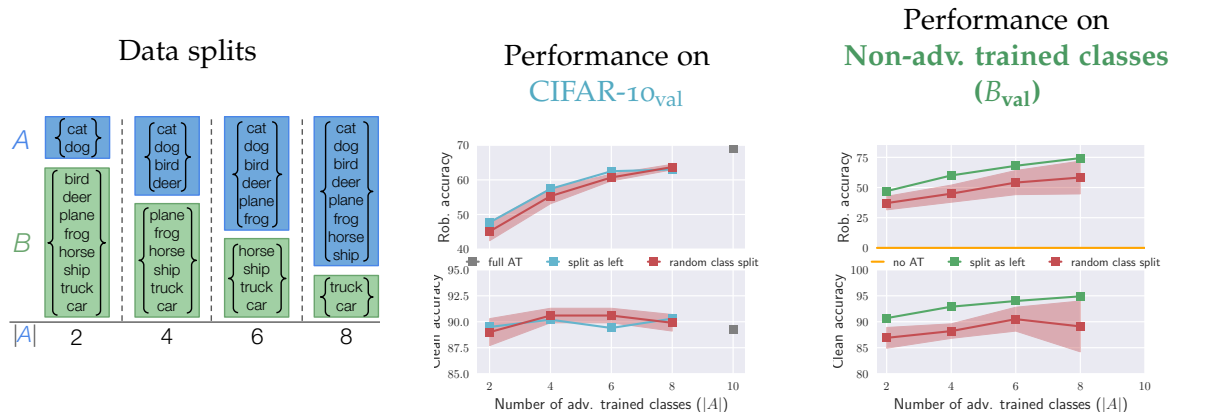


Figure 3.8: Ranking CIFAR10 classes by difficulty (using entropy as proxy), we perform CSAT with an increasing size of adv. trained classes in A. Class splits used for training (A and B) are stated on the left. The resulting robust and clean accuracies on the validation set is shown on the right, separated into performance on B_{val} and *all*. Compared with a random baseline of random class ranking (red), we find the ranking by difficulty to have consistently better transfer to non-adv. trained classes (B). Overall, this results in an improved robust accuracy on average over all classes.

shown in figure 3.9, middle). Based on these results, entropy ranking and selecting classes provides only slight improvements in general. Importantly though, we continue to see the tendency of increased robustness transfer to B_{val}, which we will come back to in section 3.3.4.

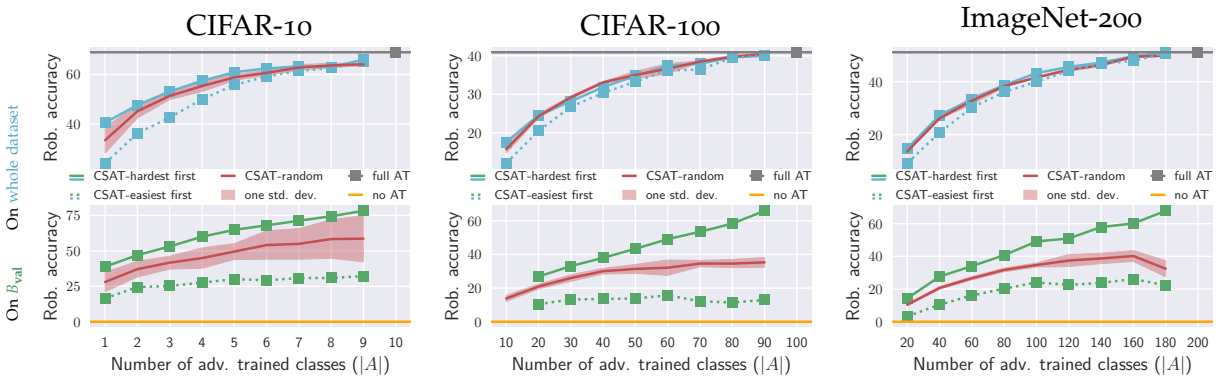


Figure 3.9: Class-subset Adversarial Training (CSAT) produces non-trivial robustness on classes that have never been attacked during training (B_{val}). Along the x-axes we increase the class subset size of A on which AEs are constructed and compare three different class-selection strategies: select hardest first (solid lines), select easiest first (dashed line) and select at random (red). On average, random selection performs as well as informed ranking (upper row), while the robustness transfer to B_{val} is best for the hardest classes (lower row). AT on a single class provides already much greater robust accuracies than without AT (orange).

3.3.3 Example subset splits (ESAT)

Considering that splits along classes are inefficient in terms of reaching the full potential of adversarial robustness, we investigate ranking examples across the whole dataset (ESAT). We

follow with the same setup as before but rank examples – and not classes – by entropy $\overline{\mathcal{H}}$. Since it is not feasible to construct corresponding rankings on the validation set, we cannot gauge robustness transfer to B_{val}. Instead, we will test transfer performance to downstream tasks in section 3.3.4. We consequently report robust accuracy and clean accuracy on the whole validation set in figure 3.10. Similar results for L_∞ are provided in the appendix, section A.5.

Firstly, note that the increase in robust accuracy is more rapid than with CSAT w.r.t. the size of A. AT only on 50% of training data (25k examples on CIFAR and 112k on ImageNet-200) and the resulting average robust accuracy is very close to the baseline AT performance (gray line). Secondly, note that gap between hard (solid line) and easy example selection (dashed line) has substantially widened. In practice, it is therefore possible to accidentally select poor performing subsets, although the chance appears to be low given the narrow variance of random rankings (red). To some extent, this observation supports the hypothesis that examples far from the decision border (the easiest to classify) provide the least contribution to robustness gains. This is also supported by the reverse gap in clean accuracy (bottom row in figure 3.10). That is, easiest-first-selection results in higher clean accuracies than hardest-first, while robust accuracies are much lower. In contrast however, we observe random rankings (red) to achieve similar performances to hard rankings (solid lines) on all datasets and subset sizes. This is somewhat unexpected, especially on small sizes of A (e.g. 5k). Given the results, we conjecture that the proximity to the decision boundary plays a subordinate role to increasing robustness. Instead, it is plausible to assume that diversity in the training data has a large impact on learning robust features, also indicated by [GK22].

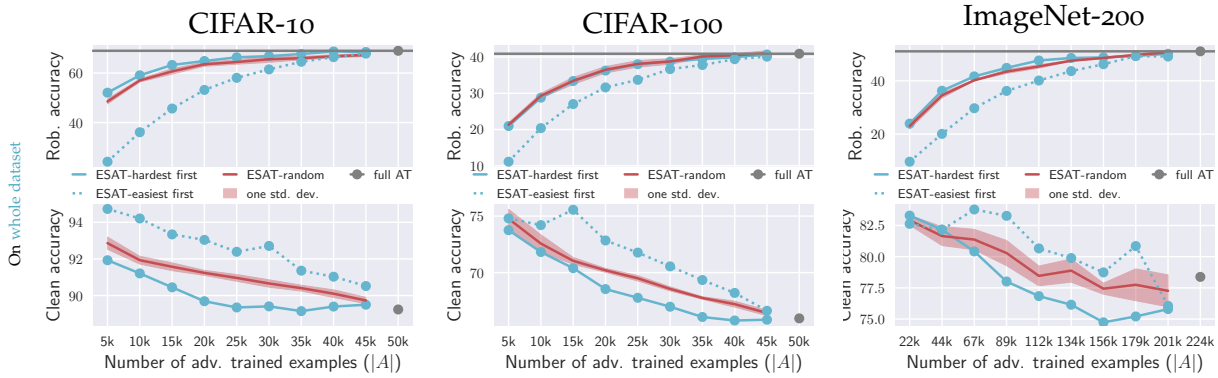


Figure 3.10: Example-subset Adversarial Training (ESAT) on CIFAR and ImageNet-200, provide quick convergence to a full AT baseline (gray line and dot) with increasing size of A. We report robust accuracy (upper row) and clean accuracy (lower row) and observe similar characteristics as with CSAT (figure 3.9). I.e., selecting the hardest examples first (solid line) provide higher rob. accuracy than easy ones (dashed line), although the gap substantially widens. Random example selection (red) provides competitive performance on average. Across all datasets, common clean accuracy decreases while robust accuracy increases [TSE+19]. L_∞ results in appendix, figure A.6.

3.3.4 Transfer to downstream tasks

Previous experiments on ESAT could not provide explicit robust accuracies on the non-adv. trained subset B_{val} since training and testing splits do not align naturally – recall the evaluation recipe outlined in section 3.2.3. In order to test transfer performance regardless, we make use of the fixed-feature task transfer setting proposed in [SSZ+20]. The recipe just slightly

changes: split the data into A and B as usual and perform SAT. Fix all features, replace the last classification layer with a 1-hidden layered classifier and finetune only the new classifier on the target task. Importantly, neither training nor validation set for the target task are split. We consider CIFAR-100 and ImageNet-200 and transfer to CIFAR-10, SVHN, Caltech-256 [GHP07] and Flowers102 [NZ08]. We call SAT trained for transfer Source-task Subset Adversarial Training (S-SAT), to emphasize that the subset training is performed on the source-task dataset.

In this section, we consider models that have "seen" only a fraction of AEs on the source task and investigate the robustness transfer capabilities to tasks on which they have not explicitly adversarially trained on. We find unexpectedly strong transfer performances for models that have both low clean and robust accuracy, only by putting more weight on the AEs.

Loss balancing improves robustness transfer. In contrast to the previously explored setting, we observe the transfer setting to benefit from loss balancing. Recall equation 3.2 in section 3.2.2 in which w_A and w_B can be assigned different values to balance the loss when $|A| \neq |B|$. We show that the vanilla configuration $w_A = w_B = 1$ transfers robustness to downstream tasks poorly, that balancing the loss with $w_B = 1, w_A = {}^{|B|}/_{|A|}$ lacks transfer performance for small $|B|$ and that weighting examples from A higher results in improved robustness transfer. We present results for all three weightings in figure 3.11. The figure is organized in three columns, all reporting robust accuracy. The first column reports the robust accuracy on subset A_{val}, the second on subset B_{val} and the third reports the robust accuracy on the downstream task. Here, we train on CIFAR-100 and transfer to CIFAR-10. The vanilla loss is indicated by circles and a solid line, the balanced loss $w_A = {}^{|B|}/_{|A|}$ by squares and a dotted line and the loss overemphasizing A by a plus and a dashed line.

First and foremost, note that the robustness transfer for the vanilla configuration is substantially worse than both alternatives (robust accuracy in top right). Transfer improves with use of loss balancing, e.g. for $|A| = 10$, robust accuracy improves from 8% to 30%, but does not converge to the baseline AT performance (gray line). This is an unwanted side effect of equalizing the weight between A and B. When A is much smaller than B, less weight is assigned to the AEs constructed for A and robustness reduces. Note, this effect can also be seen on A_{val} (top left in figure). Instead, we find it beneficial to overemphasize on the AEs (plus with dashed line). This configuration assigns $w_A = 2^{|B|}/_{|A|}$ for $|A| = 10$ and increases the weight to $w_A = 10^{|B|}/_{|A|}$ for $|A| = 90$. This results in improved robust accuray on A_{val}, but low robust and clean accuracy on B_{val}. Interestingly, while the generalization to B_{val} is low, robustness transfer to CIFAR-10 is very high. We use this loss weighting for all following task transfer experiments.

Robustness transfer from example subsets. Using the weighted loss, we focus in the following on S-ESAT on two source tasks: CIFAR-100 and ImageNet-200, and train on three downstream tasks. Similar results for S-CSAT and SVHN as additional downstream task can be found in the supplement, sections A.3 and A.6. Figure 3.12 presents results for three settings: CIFAR-100 $\rightarrow$ CIFAR-10 and ImageNet-200 $\rightarrow$ Caltech-256, Oxford-Flowers-102. The first and second row show robust and clean accuracy on the downstream task respectively. As before, we compare with a random (red) and a full AT baseline (gray line). Selecting A to contain the hardest examples first (highest entropy) is marked by a solid line; selecting easiest is marked by a dashed line. Similar results for L_∞ are provided in the appendix, section A.5.

In line with the improvements seen using appropriate loss weighting, we see similarly fast recovery of baseline AT performance across all dataset. In fact, $|A|$ containing only 30% of training data (15k and 70k) is sufficient to reach near baseline performance. On CIFAR-100 $\rightarrow$ CIFAR-10 and ImageNet-200 $\rightarrow$ Flowers-102 even slightly outperforming the same with a further increase in size. Similar to the non-transfer settings tested before, we

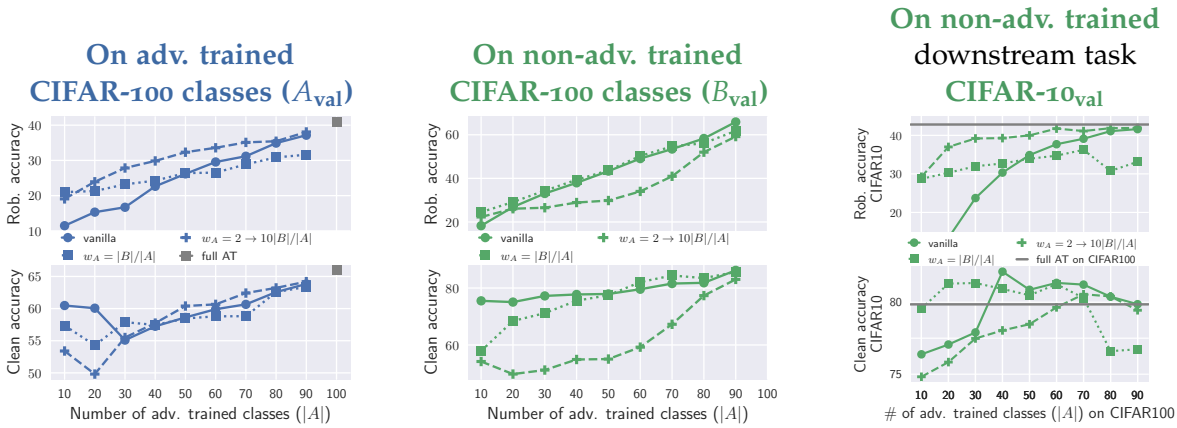


Figure 3.11: Impact of cross-entropy weighting on robustness transfer. For subset AT, we test different weighting strategies for sets A and B given they are of unequal size. We observe that vanilla cross-entropy (*circle*) offers the worst robustness transfer to CIFAR-10 (right). The best transfer (*plus*) is provided when loss weights are chosen such that training is overemphasized on A, indicated by dropping robust accuracies on B (compare left and center).

see similar interactions between subset selection strategies. I.e. hardest examples (solid line) provide greater robustness transfer than easiest (dashed line) while a random baseline (red) achieves competitive performances. The latter consistently outperforming entropy selection on ImageNet-200 $\to$ Flowers-102, supporting our observation in section 3.3.3: with increasing dataset complexity, informed subset selection provides diminishing returns. Note that all robust accuracy increases proportionally correlate to an increase in clean accuracy as well. This is in stark contrast to the inverse relationship in previous settings. C.f. figure 3.9 and 3.10, for which clean accuracy decreases. This interaction during transfer is similar to what is reported in [SIE+20]: increased robustness of the source model results in increased clean accuracy on the target task (over a non-robust model). Intriguingly though, with appropriate weighting, the biggest robustness gains on the downstream task happen under fairly small A. This is a promising outlook for introducing robustness in the foundational setting [BHA+21], where models are generally trained on very large datasets, for which AT is multiple factors more expensive to train. Note that our results generalize to single-step attacks like fast gradient sign method (FGSM) [GSS15, WRK20] as well. We provide evaluations in section 3.3.5. While we consider the fixed-feature transfer only, recent work has shown this to be a reliable indicator for utility on full-network transfer [SIE+20, KSL19].

3.3.5 Extended analysis

In the following, we complement our main results by combining them with current trends. First, we investigate the impact of applying *TRADES* [ZYJ+19] – adversarial training with tuneable trade-off between clean and robust accuracy. This is of particular interest given that clean accuracy for small $|A|$ is higher than with full AT (e.g. see figure 3.10, enabling improved robust accuracy. Second, we investigate whether our observations with SAT generalize to the current trend of fast AT [WRK20, AF20, dJABV+22], i.e. single step adv. attacks during training.

Robust accuracy trade-off with TRADES [ZYJ+19] proposed a loss with principled trade-off capability between clean and robust accuracy for AT. We observe that clean accuracy is

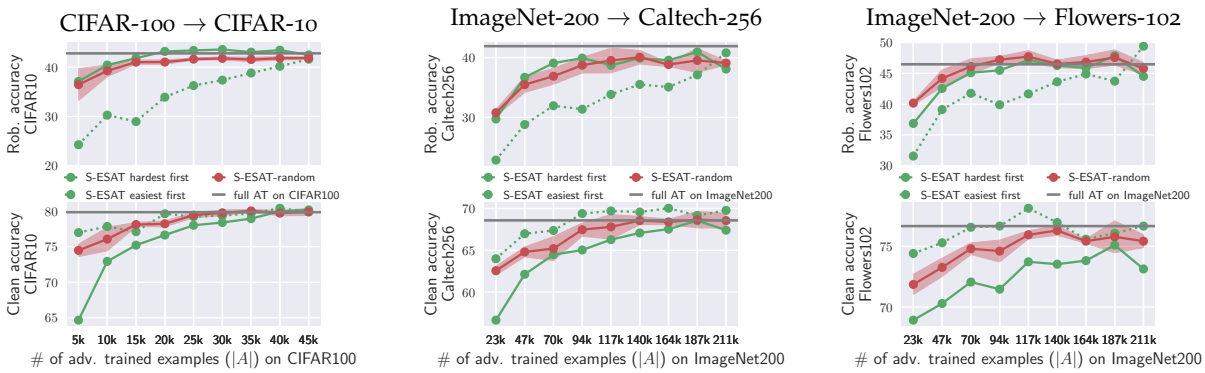


Figure 3.12: Transfer from S-ESAT to three different downstream tasks. S-ESAT is trained on source dataset CIFAR-100 (left) and ImageNet-200 (middle and right). We report robust (top row) and clean (bottom) accuracies for increasing size of A. Similar to our investigation on transfer from A to B, we find that hard examples provide better robustness transfer than easy ones, but random selections (red) achieve competitive performances. Most importantly, "seeing" only few AEs (here 30% of source data) recovers baseline AT performance (gray line). L_∞ results in appendix, figure A.7.

higher than with full AT when $|A|$ is small, i.e. less than 50% of examples. We investigate to what extent robust accuracy can be improved. First, we conduct a hyperparameter sweep for TRADES β parameter which controls the trade-off. As baseline we choose the best performing CSAT configuration on CIFAR-10 with $L_2\epsilon = 0.5$, that is $A = \{\text{cat}\}$. Results are presented in table 3.3. Note that – as expected – clean accuracy decreases with increasing β while robust accuracy increases. Although [ZYJ+19] recommends β to be less than 10.0, we find 12.0 to provide best robustness transfer from A to B. This configuration achieves 42.9%(+5.1) robust accuracy on B while also increasing rob. acc. on A to 78.2%(+28.6), with the expected decrease in overall clean accuracy (81.0%(−10.0)).

In a second experiment, we apply TRADES to ESAT on CIFAR-10. Given the $\beta = 6.0$ recommendation in [ZYJ+19] and our $\beta = 12.0$ observation for single class CSAT, we use linear interpolation between these two values for general ESAT. That is, when $|A|$ is small, we trade off some of the gained clean accuracy for increased robustness with large $\beta = 12$, but decrease to $\beta = 6$ with increasing size of $|A|$. We compare *ESAT-hardest first* with and without TRADES in table 3.4. We observe quicker robust accuracy convergence to the baseline. 66% robust accuracy is achieved at $20k$ samples, instead of at $25k$. However, our choice of β also induces a drop in clean accuracy below full AT clean accuracy (85.6% clean accuracy at $20k$ vs 89.7% at $25k$ vs 89.2% for full AT).

Single-step Adversarial Training Our main experiments use AT with 7 PGD-steps, yet recent works have suggested to reduce the number of steps without impairing performances. One of the most prominent approaches is fast-AT [WRK20] which builds upon single step fast gradient sign method (FGSM) [GSS15]. We show here, that FGSM for SAT provides similar robustness transfer to PGD-7. This has appealing implications for improving efficiency of AT. We can reduce the number of attack steps *and* can reduce the number of examples. We use *FGSM-RS* [WRK20], with a step-size of 0.625 for $\epsilon = 0.5$ and 3.75 for $\epsilon = 3.0$. All other training settings are consistent with previous experiments.

Fast ESAT. Results on all datasets, shown in figure 3.13 show highly similar results (squares on dotted line) to vanilla ESAT with 7 PGD steps (circles on solid line), except when $|A| \leq 10k$ for which robust accuracy is near 0%. This is likely due to catastrophic overfitting [WRK20, AF20],

which could be mitigated with additional regularization [AF20].
Fast S-ESAT. Similarly, for task transfer, we show results in figure 3.14 and observe very similar clean and robust accuracies (lower and upper row) across all architectures. Specifically, FGSM-RS achieves slightly higher clean accuracies and slightly lower robust accuracies – especially for small $|A|$. Nonetheless, single-step AT converges to the full AT baseline (gray line) in a similar fast rate, i.e. generating AEs for around 30% of the training set is sufficient.

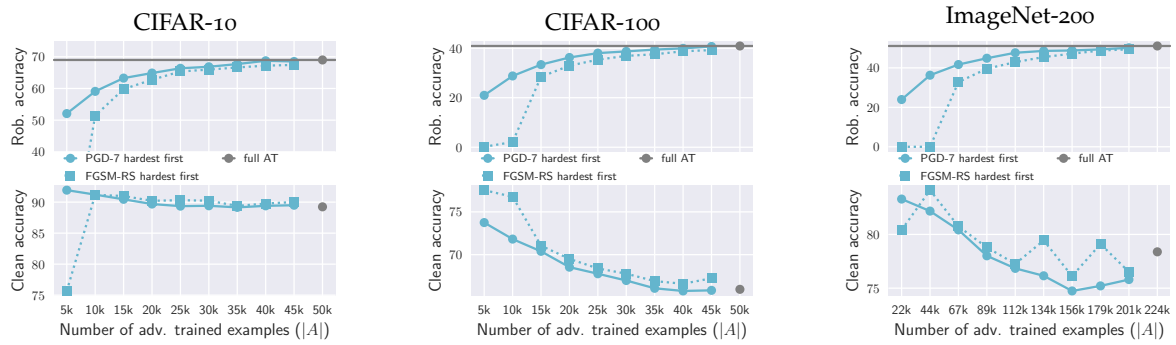


Figure 3.13: Single step AT (FGSM-RS [WRK20]) in the ESAT setting. We observe similar characteristics to ESAT with PGD-7, albeit slightly lower robust accuracies. Moreover, we observe very low robust accuracies for small $|A|$, indicating robust overfitting [WRK20, AF20].)

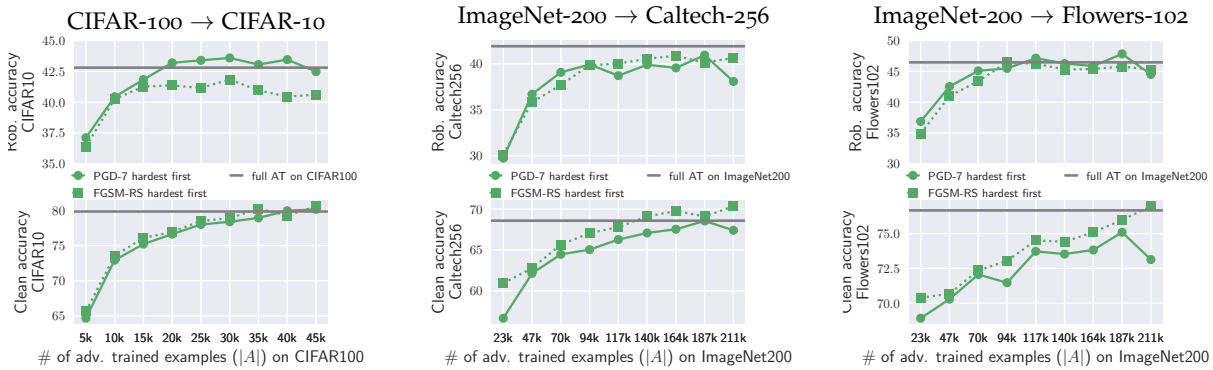


Figure 3.14: Comparison between PGD-7 and single step S-ESAT on the transfer setting to three different downstream tasks. Training and evaluation using L_2 norm.

Conventional setting						
Dataset	**Architecture**	**Epochs**	**Batchsize**	**lr**	**lr-decays**	**L_2 decay**
CIFAR-10	PreActResNet-18	200	128	0.1	100, 150	$5 \cdot 10^{-4}$
CIFAR-100	PreActResNet-18	200	128	0.1	100, 150	$5 \cdot 10^{-4}$
ImageNet-200	ResNet-50	150	256	0.1	50, 100	$1 \cdot 10^{-4}$
Transfer setting						
Dataset	**Architecture**	**Epochs**	**Batchsize**	**lr**	**lr-decays**	**L_2 decay**
CIFAR-10	PreActResNet-18 + [512,10]	40	128	0.1	20, 30	$5 \cdot 10^{-4}$
SVHN	PreActResNet-18 + [512,10]	40	128	0.1	20, 30	$5 \cdot 10^{-4}$
Caltech-256	ResNet-50 + [2048,257]	100	128	0.1	50, 75	$1 \cdot 10^{-4}$
Flowers-102	ResNet-50 + [2048,102]	100	102	0.1	50, 75	$1 \cdot 10^{-4}$

Table 3.2: Training settings for all used dataset for the conventional (upper rows) and the transfer setting (lower rows). In the transfer setting, the last classifier layer is replaced with two linear layers of size $K \times K$ and $K \times N$, abbreviated as $[K, N]$. K defines the number of feature channels and N the number of classes.

		clean acc. in %	**robust acc. in %**		
$A = \{\text{cat}\}$	β	**on CIFAR-10**	**on A**	**on B**	**on CIFAR-10**
w/o TRADES	N/A	91.0	49.6	37.8	39.0
w/ TRADES	1.0	92.4	41.5	32.0	33.0
	6.0	85.9	70.2	42.1	44.9
	12.0	**81.0**	**78.2**	**42.9**	**46.4**
	24.0	81.6	80.1	41.8	45.6

Table 3.3: TRADES [ZYJ+19], applied to CSAT improves robustness transfer from single class *cat* to B over baseline CSAT training. That is, we gain 5.1%-points to achieve a robust accuracy on B of 42.9%.

TRADES on ESAT	$\lvert A\rvert =$	**5k**	**10k**	**15k**	**20k**	**25k**	**30k**	**35k**	**40k**	**45k**	**full AT**
	$\beta =$	12.0	11.25	10.5	9.75	9.0	8.25	7.5	6.75	6.0	6.0
yes	Rob. acc	57.2	62.1	65.7	66.1	67.7	67.6	68.2	66.4	69.0	69.0
	Clean acc	87.0	87.4	84.2	85.6	84.3	87.0	84.0	88.0	88.4	88.6
no	Rob. acc	52.1	59.1	63.3	64.9	66.3	66.8	67.7	68.7	68.5	69.0
	Clean acc	91.9	91.2	90.5	89.7	89.4	89.4	89.2	89.4	89.5	89.2

Table 3.4: ESAT on CIFAR-10 combined with TRADES [ZYJ+19]. Trading off clean for robust accuracy, we gain $\geq 66.0\%$ robust accuracy earlier at $20k$ samples in contrast to $25k$ for vanilla ESAT. However, our β choice decreases clean accuracy below full AT.

3.4 Conclusion

In this paper, we presented an analysis of how adversarial robustness transfers between classes, examples and tasks. To this end, we proposed the use of Subset Adversarial Training (SAT), which splits the training data into A and B and constructs AEs on A only. Trained on CIFAR-10, CIFAR-100 and ImageNet-200, SAT revealed a surprising generalizability of robustness between subsets, which we found to be based on the following observations: (i) adv. robustness transfers among classes even if some or most classes have never been attacked during training and (ii) hard classes and examples provide better robustness transfer than easy ones. These observations remained largely valid in the transfer to downstream tasks like Flowers-102 and Caltech-256 for which we found that overemphasizing loss minimization of AEs in A provided fast convergence to baseline AT robust accuracies, even though transfer to B was severely reduced. Specifically, it appears that only few AEs (A containing 30% of the training set) learn all of the robust features which generalize to downstream tasks. This finding could be particularly interesting for AT in the foundational setting, in which very large datasets render training computationally demanding.

Our findings shed new light onto the properties of adversarial training and may lead to more efficient robustness transfer approaches which would allow easier deployment of robust models. We provided an account on a broad variety of datasets and used models commonly evaluated in related work. It needs to be seen whether our findings generalize to other threat models [LSF21] as well.

4 Certified Robust Models with Slack Control and Large Lipschitz Constants

Contents

This second chapter again not really the second chapter... moves away from adversarial training and investigates an alternative optimization approach to improve adversarial robustness: regularizing the Lipschitz continuity of deep networks. This type of regularization has two advantages over vanilla AT. AEs don't need to be constructed during training and if specified correctly, models can be made certifiably robust. That is, for any given input, the model can certify whether adding a perturbation of norm ϵ will retain the original prediction. This is of course highly desirable for acquiring additional trust in a system. Unfortunately, this comes at the cost of significantly decreased accuracy. In this paper, we propose a Calibrated Lipschitz-Margin Loss (CLL) that addresses this issue and improves certified robustness by tackling two problems: Firstly, commonly used margin losses do not adjust the penalties to the shrinking output distribution; caused by minimizing the Lipschitz constant K. Secondly, and most importantly, we observe that minimization of K can lead to overly smooth decision functions. This limits the model's complexity and thus reduces accuracy. Our CLL addresses these issues by explicitly calibrating the loss w.r.t.margin and Lipschitz constant, thereby establishing full control over slack and improving robustness certificates even with larger Lipschitz constants. On CIFAR-10, CIFAR-100 and Tiny-ImageNet, our models consistently outperform losses that leave the constant unattended. On CIFAR-100 and Tiny-ImageNet, CLL improves upon state-of-the-art deterministic L_2 robust accuracies. In contrast to current trends, we unlock potential of much smaller models without K=1 constraints.

This chapter is based on [LSSF23]: As first author, Max Losch ran all experiments and was the main writer of the paper. no need to repeat what is said in the intro about your contribution - also - if I am not mistaken - there is no such statement in chapter 3? In case you want to keep this statement, it should be done in all chapters for consistency

The code for this work can be found on GitHub[1].

[1] https://github.com/mlosch/CLL

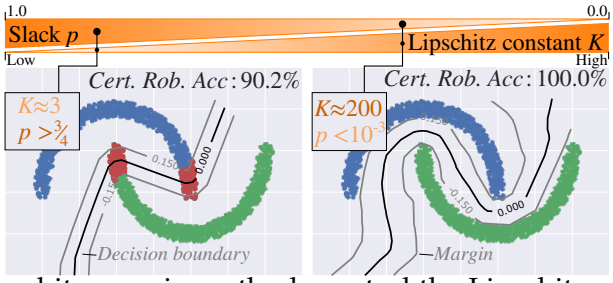


Figure 4.1: Existing Lipschitz margin methods control the Lipschitz constant K to be low, yet we observe decision functions becoming overly smooth when K is too low (left) – impairing accuracy. Our CLL loss provides slack control, which we show is inversely proportional to K (see gradients on top). We can control K to be high, avoid the smoothing and achieve improved clean and robust accuracies (right). Incorrect or not robust samples marked red.

4.1 Introduction

The Lipschitz constant K of a classifier specifies the maximal change in output for a given input perturbation. This simple relation enables building models that are certifiably robust to constrained perturbations in the input space, i.e., those with L_2-norm below ϵ. This is because the resulting output distance $K \cdot \epsilon$ can be calculated efficiently (i.e., in a closed form) without expensive test-time randomization, such as randomized smoothing [CRK19], or network bound relaxations, e.g. [GDS+19, ZCX+20]. This way of obtaining certifiable robustness is also directly linked to large margin classifiers, as illustrated in figure 4.1, where a training sample free area of radius ϵ is created around the decision boundary. The earliest practical examples implementing Lipschitz margins for certified robustness include [HA17], which provided first guarantees on deep networks or LMT [TSS18] and GloRo [LWF21] that design Lipschitz margin losses.

A limitation with Lipschitz margin classifiers is their decreased performance, as has been shown on standard datasets like CIFAR-10 or TinyImageNet, both in terms of empirical robustness and clean accuracy. This is emphasized in particular when comparing to approaches such as adversarial training [MMS+18, CAP+19] and its variants [ZYJ+19, CRS+19, WXW20]. Thus, recent work proposed specialized architectures [CBG+17, LHA+19, ALG19, SSF21, MDAA22, XLL22, TK20, ZCL+21, PL22, AHD+23, WM23] to enforce specific Lipschitz constraints, adjusted losses [TSS18, LWF21, SSF21, ZCL+21, ZJHW22, PL22] or tighter upper Lipschitz bounds [LLP20, HZS+21, DBAA23] to address these shortcomings. Despite existing efforts, clean and robust accuracy still lags behind state-of-the-art empirical results.

We relate this shortcoming to two interlinked properties of used losses: Firstly, logistic based margin losses, including GloRo [LWF21], SOC [SSF21] and CPL [MDAA22] do not adjust their output penalties to the output scaling caused by minimization of K. In practice, this results in samples remaining in under-saturated regions of the loss and effectively within margins. And secondly, we identify that controlling the Lipschitz constant K to be too low, may result in overly smooth decision functions – impairing clean and robust accuracy. The result is exemplarily illustrated in figure 4.2b (middle) for GloRo on the two moons dataset. The induced overly smooth decision surface eventually leads to reduced clean and robust accuracy. **Contributions:** In this work, we propose Calibrated Lipschitz-Margin Loss (CLL), a loss with improved control over slack and Lipschitz constant K to instrument models with a margin of width 2ϵ. Key to our loss is integrating the scale of the logistic loss with K, allowing calibration to the margin. This calibration endows CLL with explicit control over slack and

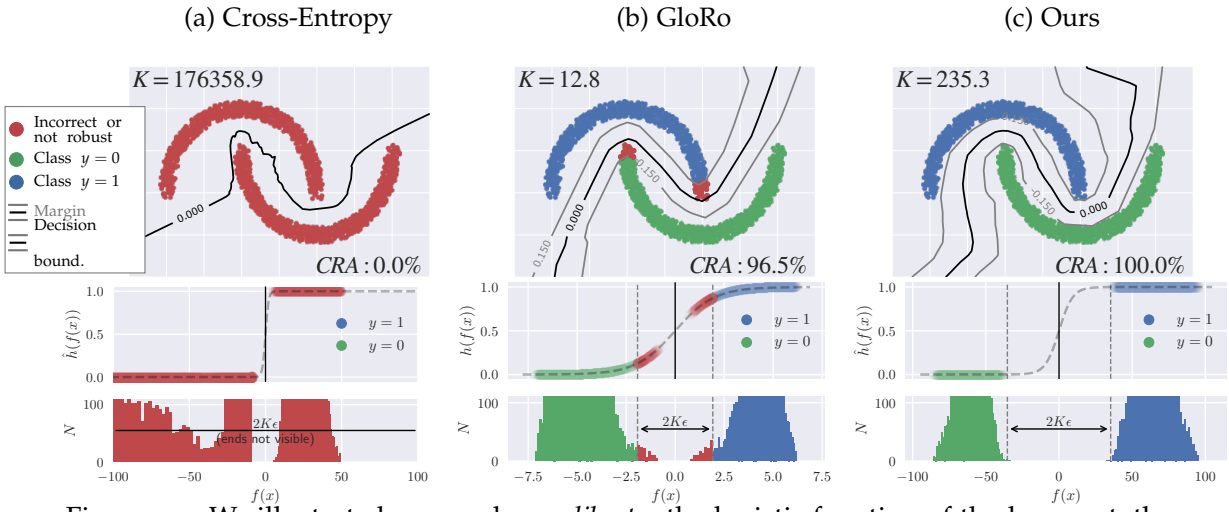


Figure 4.2: We illustrate how our loss *calibrates* the logistic function of the loss w.r.t. the margin, leading to improved clean and certified robust accuracy (CRA). (a) Cross-Entropy does not guarantee margins, the Lipschitz constant K is large. (b) Uncalibrated margin losses such as GloRo may produce violated margins (gray) and decision boundaries (black) on the two-moons dataset. (c) In contrast, our calibrated loss is better visually and quantitatively for robust and accurate samples. The middle row plots their output probabilities.

K, which can improve certified robust accuracy. As a result, the classifier avoids learning an overly smooth decision boundary while obtaining optimal margin, as illustrated for two moons in figure 4.2c. On CIFAR-10, CIFAR-100 and Tiny-ImageNet, our method allows greater Lipschitz constants while at the same time having tighter bounds and yielding competitive certified robust accuracies (CRA). I.e. applying CLL to existing training frameworks consistently improves CRA by multiple percentage points while maintaining or improving clean accuracy.

4.2 Calibrated Lipschitz-Margin Loss (CLL)

We propose a new margin loss called CLL that calibrates the loss to the width of the margin – a property not explicitly accounted for in existing large margin Lipschitz methods. Specifically, we calibrate the logistic functions at the output (sigmoid and softmax), by integrating the Lipschitz constant K into the definition of the logistic scale parameter. This new formulation reveals two properties important for margin training: (i) the scale parameter controls slackness and (ii) slackness influences classifier smoothness. This slack control allows to trade-off certified robust and clean accuracy. But more importantly, we find that this slackness determines K of the whole model. This is illustrated on the two moons dataset in figure 4.1 (left) with high slack implying small K and right with low slack implying large K. Given this improved control, we train models with large constants that produce new state-of-the-art robust accuracy scores.

4.2.1 Background

Let $(x, y) \in \mathcal{D}_{\mathcal{X}}$ be a sample-label pair in a dataset in space $\mathcal{X}$, f be a classifier $f : \mathcal{X} \to \mathcal{S}$ where $\mathcal{S} \subseteq \mathbb{R}^N$ is the logit space with N logits and h be a non-linearity like the logistic function or softmax mapping, e.g. $h : \mathcal{S} \to \mathbb{R}^N$.

Lipschitz continuity. The Lipschitz continuity states that for every function f with bounded first derivative, there exists a Lipschitz constant K that relates the distance between any two points x_1, x_2 in input space $\mathcal{X}$ to a distance in output space. That is, for any input sample distance $\|x_1 - x_2\|_p$, the resulting output distance $\|f(x_1) - f(x_2)\|_p$ is at most $K \cdot \|x_1 - x_2\|_p$. Consequently, this inequality enables the construction of losses that measure distances in input space: e.g. the margin width – and importantly: quick input certification. E.g. assume a classifier with two logits f_1, f_2. An input x is certified robust w.r.t.radius ϵ when the Lipschitz bounded distance ϵK is less than the distance between the two logits: $\epsilon K < |f_1(x) - f_2(x)|$. Note that K is not required to be small to allow certification. The inequality also holds when K is large, as long as the distance between logits is greater, as we will see in our experiments in section 4.3. The exact Lipschitz constant, though, is non-trivial to estimate for highly non-linear functions such as deep networks [VS18]. Fortunately, it is tractable to calculate an upper bound. For this, it is sufficient to be able to decompose the classifier f, e.g., a (convolutional) neural network, into its L individual layers, i.e. $f = g^{(L)} \circ g^{(L-1)} \circ \cdots \circ g^{(1)}$. Then K is upper bounded by the product of norms [SZS+14],

$$K \leq \prod_{l=1}^{L} \|g^{(l)}\|_p =: \hat{K}. \tag{4.1}$$

In general, this bound is extremely loose without any form of Lipschitz regularization[VS18], yet recent work has shown substantially improved tightness, rendering uses in losses tractable (e.g. [LWF21, SSF21, MDAA22, PL22, AHD+23, WM23, HZS+21]).

Lipschitz Margin losses. To produce certified robust models, we strive to train models with large margins. This can be achieved by adapting the training objective and specifying a minimal distance ϵ between any training sample and the decision boundary. A typical example is the hinge loss formulation in SVMs [CV95]:

$$\min_f \mathbb{E}_{(x,y)\sim\mathcal{D}_\mathcal{X}} \left[h\left(\epsilon - yf(x)\right) + \lambda \|f\|^2 \right], \tag{4.2}$$

where $\epsilon = 1$ and $h(\,\cdot\,) = \max\{\,\cdot\,, 0\}$. $\|f\|$ denotes a generic measure of classifier complexity, which in the linear SVM case is the norm of the weight matrix. This formulation can be generalized to Lipschitz classifiers, e.g., by minimizing $\|f\| = K$ or by multiplying ϵ with its Lipschitz factor ϵK. The latter being used in the GloRo loss [LWF21]. All variants of formulation (4.2) require strict minimization of K to produce margins. We find these types of losses can be improved when h belongs to the logistic family – as sigmoid and softmax are. Since logistic functions assume a fixed output distribution, we observe that minimizing K can leave samples too close to the decision boundary We present exemplary evidence in figure 4.2b in which red samples remain within the margins. This is specifically true for GloRo, but also for methods that hard constrain K (e.g. [SSF21, MDAA22, XLL22]). In the following, we address these issues with CLL.

4.2.2 Binary CLL

We base our construction of CLL on the general margin formulation in equation (4.2). Key is calibrating the output non-linearity h to the margin. In the binary case, it is common practice in deep learning to set h in equation (4.2) to a logistic function $h(x) = [1 + \exp\{-x/\sigma\}]^{-1}$ with fixed $\sigma = 1$ and minimize the binary cross-entropy loss. Yet the underlying logistic distribution assumes a fixed distribution width $\sigma = 1$, which can be detrimental for training Lipschitz classifiers. To demonstrate the limitation of this assumption, we look at figure 4.2 illustrating margin training on the two-moons dataset. Figure 4.2a illustrates vanilla cross-entropy and no Lipschitz regularization. The decision boundary attains an irregular shape without margin.

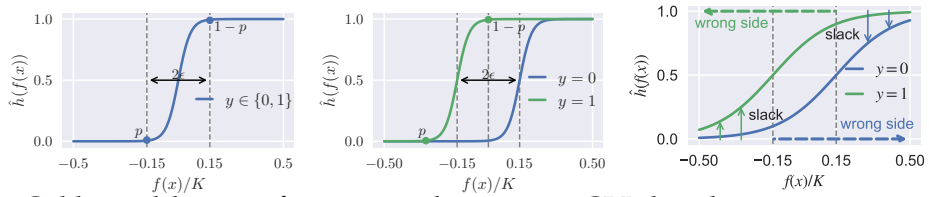


Figure 4.3: Calibrated logistic function as key to our CLL loss by parameterizing the scale parameter σ to margin width 2ϵ. Left: without margin offset; middle: with margin offset ϵ; right: slack in CLL is governed by parameter p. That is to increase slack, we increase p which decreases loss for samples on the wrong margin side (indicated by arrows).

The Lipschitz constant is very high, the output values attain high probabilities since they are pushed to the tails of the distribution (see $p = h(f(x))$ in middle row). In contrast, a Lipschitz margin loss (GloRo [LWF21], fig. 4.2b) produces a margin, yet non-robust samples (red) remain within margin boundaries. This is a consequence of minimizing K, which limits the spread of the distribution. Under this condition, the assumption $\sigma = 1$ is inefficient. Additionally, since σ of the logistic is fixed, the final probability p at the margin $\pm\epsilon$ can only be determined post-hoc, after the Lipschitz constant K finds a minimum. We can be more efficient about this process by calibrating the width to $K\epsilon$ at each training step and requiring p at $\pm\epsilon$ to be a specific value. The result is shown in figure 4.2c which produces the desired margin with no errors. Our proposed loss is defined as follows.

Let $y \in \{-1, 1\}$, and $\mathcal{L}$ be the binary cross entropy loss $\mathcal{L}(y, h(f(x))) = -\log h(f(x)) - (1 - y)\log(1 - h(f(x)))$[2]. We propose the objective:

$$\min_f \mathbb{E}_{(x,y)\sim\mathcal{D}_\mathcal{X}} \left[\mathcal{L}(y, \hat{h}(f(x); y)) + \lambda K^2\right] \tag{4.3}$$

$$\text{with } \hat{h}(f(x); y) = h\left(-\frac{y\epsilon}{\sigma_\epsilon(p)} + \frac{1}{\sigma_\epsilon(p)}\frac{f(x)}{K}\right) \tag{4.4}$$

$$\text{and } \sigma_\epsilon(p) = \frac{2\epsilon}{h^{-1}(1-p) - h^{-1}(p)}, \tag{4.5}$$

where $\hat{h}$ is our *calibrated logistic* and h^{-1} is the inverse cdf: $h^{-1}(p) = \log(p/1-p)$. Our proposal follows from calibrating σ to 2ϵ. For its realization we only require 4 values to uniquely determine σ: (i) the two positions of the margin boundaries $\pm\epsilon$ and (ii) the two probabilities $p_{-\epsilon}, p_\epsilon$ that the logistic distribution should attain at these positions. We illustrate these 4 values in figure 4.3, which displays an already calibrated logistic function to $\epsilon = 0.15$ (vertical lines) and $p_{-\epsilon} = p = 10^{-3}$ and $p_\epsilon = 1 - p = 1 - 10^{-3}$. The theoretical derivation follows from the logistic distribution, which we discuss in the appendix B.1. We denote the calibrated scale as $\sigma_\epsilon(p)$ as it depends on ϵ and a probability p (equation (4.5)). So far, this does not express the integration of the Lipschitz constant K. Recall that the input distance 2ϵ can be related to the corresponding output distance via K, i.e. $2K\epsilon$. We consequently acquire the Lipschitz calibrated version shown in figure 4.3: $f(x)/\sigma_\epsilon(p)K$. Next, we integrate hard margin offsets $\pm\epsilon$ as in formulation (4.2) to maximize the penalty on samples within the margin. To integrate, we add $\pm\epsilon K$, depending on the class sign: $[-yK\epsilon + f(x)]/[\sigma_\epsilon(p)K]$. This places the probability on the margin to the worst case value 0.5. Note that this does not invalidate our calibration: the offset is a tool to improve margin training and is removed during inference.

[2]Transformation to $y \in \{0, 1\}$ is omitted for readability.

4.2.3 Multiclass CLL

The general intuition developed in the binary case above can readily be applied to multiclass classification. The main difference being the application to pairs of output logits (before we considered a scalar output). To this end, let $y \in \{1, 2, ..., N\}$ and $f : \mathcal{X} \to \mathbb{R}^N$. To calibrate a classifier with multinomial output, we need to measure the output distances between pairs of logits, e.g. $f_{i,y}(x) = f_i(x) - f_y(x)$ for logit i and logit of label y. We denote the Lipschitz constant of $f_{i,y}(x)$ as $K_{i,y}$ and generalize binary-CLL to the cross-entropy loss. To construct CLL, we make use of softmaxs shift invariance property – any constant can be subtracted from the input without changing the output. That is, by subtracting $f_y(x)$ from all output logits, we do not change the softmax output distribution and we can calibrate the margin widths for all one-vs-one decision boundaries.

Definition 2 - Multinomial CLL. *Let $\mathcal{L}$ be the negative log-likelihood loss $\mathcal{L}(y, h(f(x); y)) = -\log h(f(x); y)$, then the calibrated softmax is defined as follows:*

$$\hat{h}(f(x); y) = \frac{1}{\sum_{i=1}^{N} e^{g_y(i,x)}} \begin{bmatrix} e^{g_y(1,x)} \\ e^{g_y(2,x)} \\ \vdots \\ e^{g_y(N,x)} \end{bmatrix} \tag{4.6}$$

$$\text{with } g_y(i, x) = \begin{cases} \frac{\epsilon}{\sigma_\epsilon(p)} + \frac{f_{i,y}(x)}{\sigma_\epsilon(p) K_{i,y}} & \text{if } i \neq y \\ -\frac{\epsilon}{\sigma_\epsilon(p)} & \text{if } i = y \end{cases}. \tag{4.7}$$

Here, the core implementation is $g_y(i, x)$. Equal to binary-CLL, the multinomial formulation retains the same properties. That is, (i) normalization by K, and (ii): calibration with $\sigma_\epsilon(p)$. Since we consider logit differences $f_{i,y}$ we obtain two cases: $i \neq y$ and $i = y$. For the latter, $f_{y,y}$ equals 0 for all x, hence $g_y(y, x)$ reduces to $-\epsilon/\sigma_\epsilon(p)$. To assert, equation 4.6 is still a generalization of the logistic function, we provide proof in the appendix B.1.

4.2.4 Discussion

CLL is derived from joining the definition of σ with the Lipschitz inequality while adjusting for the margin distance 2ϵ. By utilizing the Lipschitz constant, this scale parameter can be calibrated to the margin width as if measured in input space. This is feasible because of the normalization by the Lipschitz constant $f(x)/K$ in equations (4.4). This normalization has two additional ramifications. First, it decouples the classifier constant from the loss. K can attain any value, while CLL ensures calibration. And secondly, the Lipschitz constant of the whole model $h(f(x))$ is solely dependent on $\sigma_\epsilon(p)$. That is $K^{(h \circ f)} = K^{(h)} = 1/\sigma_\epsilon(p)$. Below, we discuss the implications of CLL with respect to the tightness of Lipschitz bounds, the interpretation of σ as allowed slack and the classifier's complexity.

Tightness. Tightness dictates the utility of the used Lipschitz bound. To measure, we can estimate a naïve lower bound to K by finding the pair of training samples x_1, x_2 that maximizes the quotient $\|f(x_1) - f(x_2)\|/\|x_1 - x_2\|$. Since the input sample distances remain fixed, tightness can only be increased by increasing output distances *and* bounding K – conceptually bringing the two quantities closer together. Recall that typical losses assume a fixed output distribution $\sigma = 1$. The capacity to push values into the saturated area of the loss is thereby limited. Tightness is thereby mainly achieved by minimizing K because output distances cannot be increased significantly. CLL instead, does not minimize K but bounds it via normalization.

Since our loss is calibrated to $K\epsilon$, we can achieve increased tightness by only maximizing output distances. With the difference of increasing output distances much farther than possible with other margin losses (an example is illustrated in figure 4.2 (compare x-axis of (b) and (c)). We find this to converge faster (see figure 4.7 in the analysis in section 4.3.4) and achieve better robust accuracies than related work, as we present in section 4.3. However, since increasing output distances has a growth influence on K, we find it necessary to put slight regularization pressure on K via factor λ, as stated on the RHS of equation (4.3).

Slack. Since the logistic function has a smooth output transition from 0 to 1, values on the wrong margin side can be interpreted as slackness. We illustrate slack in figure 4.3 which shows a calibrated logistic as in figure 4.3 and indicates wrong margin sides for each class by dashed arrows. Consider class $y = 1$ in green. If a sample falls on the wrong margin side (anywhere along the green arrow at the top) the probability decreases and consequently the loss increases (i.e., negative log-likelihood $\lim_{p\to 0} -\log(p) = \infty$). The governing factor for slack in our loss is the probability p in $\sigma_\epsilon(p)$ of equation 4.4, indicated by the green vertical arrows, which implicitly defines the loss magnitude for samples on the wrong margin side. Note that without calibration, we find the logistic σ or temperature in softmax to have no slack effect. We provide empirical evidence in section 4.3.4.

Classifier complexity. [BM02] derive complexity bounds for a 2-layered neural network (theorem 18). I.e., given 2 layers with weight norm B and 1 and non-linearity with Lipschitz constant L, the Gaussian complexity of the model is bounded by their product $K = B \cdot L \cdot 1$. This directly applies to hard-constrained K=1 models, but we find it applies to soft-constrained models as well. Effectively, this bound has implications for training Lipschitz classifiers: if K is too small, it restricts the models ability to fit the data. While theory provides only an upper bound, we find empirical evidence as support. That is, clean and certified robust accuracy can be improved when L of the loss is adjusted – and thus K. We find strongest evidence on two-moons (figure 4.4, bottom row), and consistent support on image data (section 4.3). With decreasing K (top), the model becomes overly smooth, losing performance. CLL offers direct control over K, simply by adjusting slack. That is, K is inversely proportional to the slack p: $K^{(h\circ f)} = 1/\sigma_\epsilon(p)$. Consequently, for a fixed ϵ, slack bounds the complexity of the model. Models that can separate the data well require little slack, implying a larger $K^{(h\circ f)}$. The next section discusses this in more detail.

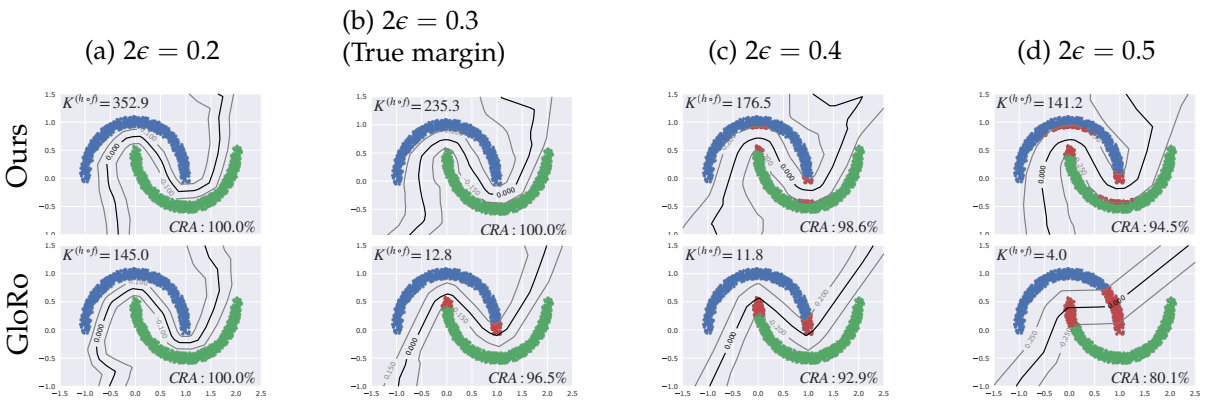


Figure 4.4: Our CLL (top row) vs margin loss GloRo (bottom) on two-moons. **Red points** denote incorrect or non-robust samples. GloRos formulation leads to consistently smaller Lipschitz constant $K^{(h \circ f)}$ restricting the classifier complexity. Already for the true margin $2\epsilon = 0.3$ it fits inefficiently. For "too-large" margins, GloRo eventually degenerates to a failure case with a near linear decision boundary. In contrast, our CLL produces a 100% accurate and robust model for the true margin and retains sensible margins for increased ϵ.

4.3 Evaluation

CLL offers increased control over slack and classifier complexity, discussed in section 4.2.4. In this section, we present empirical evidence for these claims. We show that decreasing slack leads to models with less smooth decision boundaries, resulting in higher clean and certified robustness. To this end, we present results on the synthetic two-moons dataset by visualizing the produced margins and discuss the application on natural images: CIFAR-10, CIFAR-100 [KH+09] and Tiny-ImageNet [LY14]. Implementation of our method involves computing the upper Lipschitz bound. We measure the width of the margin with the L_2 distance. Consequently, we implement equation (4.1) with the product of spectral norms [SZS+14] and calculate them by performing power iterations. Our training strategy follows the respective method we compare with. That is, we reuse published code (where available) but use our loss. To evaluate, we measure certified robust accuracies (CRA) for three margin widths (36/255, 72/255, 108/255) and Lipschitz bound tightness. CRA represents the average number of accurately classified samples that are also robust (fall outside the margin). The latter is the fraction between empirical lower bound and upper bound. All training details are listed in the appendix B.2. Our code is made publicly available at github.com/mlosch/CLL.

4.3.1 Metrics

Certified robust accuracy (CRA). To compare models, we measure the number of accurately classified samples that also fall outside the margin–hence are certified ϵ-robust. Certification of input samples is straight forward with Lipschitz models. That is, an input sample is certified when the smallest difference between highest predicting logit f_i and any other logit f_j is greater than $\epsilon K^{(f)}_{i,j}$:

$$\mathcal{M}_f(x) > \epsilon K^{(f)}_{i,j} \text{ , where } \mathcal{M}_f(x) = f_i(x) - \max_{j \neq i} f_j(x) \tag{4.8}$$

We report the total of certified samples that are also accurately classified as CRA. For simplicity, we perform CRA evaluation on the classifier f without $\hat{h}$.

Tightness. Recall that the exact Lipschitz constant is NP-hard to determine (section 3.1, main paper). As remedy, we resort to upper bounds, for which high tightness is important to avoid overestimation of output distances. In order to demonstrate improved tightness using our loss over existing alternatives, we measure this tightness by estimating an empirical lower bound $K^{(f)}_{\text{lobo}}$ on the training dataset, such that $K^{(f)}_{\text{lobo}} \leq K^{(f)} \leq \hat{K}^{(f)}$. We state the tightness $K^{(f)}_{\text{lobo}}/\hat{K}^{(f)}$ in percent. Note that while this ratio is an approximation, the true tightness is strictly greater or equal than our reported quantity. To estimate $K^{(f)}_{\text{lobo}}$, we follow the description in [LWF21]:

$$\max_{x_1,x_2} \max_i \left\{ \frac{|f_{j_1}(x_1) - f_i(x_1) - (f_{j_1}(x_2) - f_i(x_2)|}{||x_1 - x_2||_2} \right\}, \tag{4.9}$$

where $j_1 = \arg\max_i f_i(x_1)$. Intuitively, we sample two inputs x_1, x_2, and perturb both to maximize the quotient which is a lower bound to $K_{i,j}$:

$$\frac{|f_j(x_1) - f_i(x_1) - (f_j(x_2) - f_i(x_2))|}{||x_1 - x_2||_2} \leq K_{i,j} \tag{4.10}$$

To optimize equation (4.9), we use the *Adam* optimizer and perform 1000 iterations with a learning rate of $3 \cdot 10^{-4}$. We sample 4000 pairs from the training dataset and report the largest value obtained over all pairs.

4.3.2 Two-moons dataset

We start with an analysis on two-moons in order to visualize learned decision boundaries and investigate the connection to the Lipschitz constant. using *uniform* sampled noise of radius 0.1 around each sample. This results in a true-margin of exactly $2\epsilon = 0.3$. In figure 4.4, we exemplarily compare CLL with GloRo, training 7-layered networks for different target margin widths (columns). Individual plots display the decision boundary (black line) and the Lipschitz margin (gray lines). CRAs and Lipschitz constants $K^{(h \circ f)}$ are reported in the corners, non-robust or misclassified training samples are marked red. GloRo already loses CRA at the true margin 2ϵ and the decision boundary becomes very smooth with decreasing $K^{(h \circ f)}$. In contrast, CLL retains 100% CRA for the true margin and only slowly loses CRA beyond. The key is in the control over slack and hence $K^{(h \circ f)}$ – we set $p = 10^{-12}$ in equation 4.4. Our decision functions do not become overly smooth.

4.3.3 Image datasets

We continue our discussion on CIFAR-10, CIFAR-100 and Tiny-ImageNet, evaluating multiple architectures: On CIFAR-10/100, we evaluate *6C2F*[LLP20], *4C3F*[WSMK18], *LipConv*[SSF21] and *XL*[MDAA22, AHD$^+$23]. On Tiny-ImageNet, we consider *8C2F*[LLP20], *LipConv*, XL and LBDN[WM23]. We report Tiny-ImageNet results in table 4.1 and CIFAR results in table 4.2, considering CRA for three different margin widths and clean accuracy. Hereby, all values produced with CLL are averages over 9 runs with different random seeds. Standard deviations are reported in the appendix B.3. Additionally, we report tightness and Lipschitz constants of both the classifier f, as well as the composition with the loss $h \circ f$ – stating the effective complexity of the model. $\overline{K}^{(f)}$ and $\overline{K}^{(h \circ f)}$ state the largest constant between pairs of classes, e.g. $\overline{K}^{(f)} = \max_{i,j} \hat{K}^{(f)}_{i,j}$. Hereby, data scaling factors (e.g. normalization) influence $K^{(h \circ f)}$. E.g. a normalization factor of 5, increases $K^{(h \circ f)}$ by the same factor.

Table 4.1: Results on Tiny-ImageNet on clean and certified robust accuracy (CRA) for six different methods using Lipschitz bounds $\overline{K}^{(h \circ f)}$. Applying CLL to existing methods consistently improves certified robust accuracy (CRA). Architectures evaluated: *8C2F*[LLP20], *LipConv*[SSF21], *XL*[MDAA22, AHD$^+$23] and *Sandwich*[WM23] $\top$ indicates model is additionally trained with TRADES-loss[ZYJ$^+$19]. †-flagged numbers are reproduced values with our own code. CLL numbers are averaged over 9 runs.

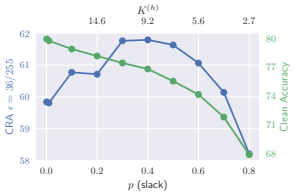
	Method	Model	Clean (%)	CRA $\frac{36}{255}$ (%)	CRA $\frac{72}{255}$ (%)	CRA $\frac{108}{255}$(%)	$\overline{K}^{(f)}$	$\overline{K}^{(h \circ f)}$	Tightness (%)
Tiny-ImageNet	GloRo[LWF21]	8C2F$^\top$	35.5	22.4	-	-	12.5		47
		8C2F$^\top$†	39.5	23.9	14.3	9.0	3.9		53
	Local-Lip-B[HZS$^+$21]	8C2F	36.9	23.4	12.7	6.1	-	-	-
	Ours $\epsilon = 0.5, p = 0.01$	8C2F	**39.8 (+0.3)**	**25.9 (+2.0)**	**16.5 (+2.2)**	**10.7 (+1.7)**	288.3	10.6	**64 (+11)**
	SOC[SSF21]	LipConv-10	32.1	21.5	12.4	7.5	6.4		**85**
		LipConv-20	31.7	21.0	12.9	7.5	6.4		81
	CLL	LipConv-20	**32.6 (+0.5)**	**26.0 (+3.5)**	**20.2 (+7.3)**	**15.5 (+8.0)**	4.9	11.6	**84 (+3)**
	SLL[AHD$^+$23]	XL	32.1	23.2	16.8	12.0	-	-	-
	LBDN[WM23]	Sandwich	33.4	24.7	18.1	13.4	-	-	-
	Ours $\epsilon = 1.0, p = 0.025$	8C2F	**33.5 (+0.1)**	**25.3 (+0.6)**	**19.0 (+0.9)**	**13.8 (+0.4)**	24.4	4.4	73

Certified robust accuracy. For fair comparison, we adjust ϵ and p of CLL to match or outperform clean accuracy of the respective baseline (discussed in section 4.3.4). First, we consider Tiny-ImageNet (table 4.1). *8C2F* is used to compare GloRo, *Local-Lip-B* and CLL, *LipConv* is used to compare *SOC* and CLL and *SLL* on *XL* and *LBDN* on *Sandwich* is compared to CLL on *8C2F*. Here, GloRo on *8C2F* achieves 23.9% CRA for $\epsilon = 36/255$ and 39.5% clean accuracy. Trained with CLL, we achieve a substantial increase for the same margin of 25.9%$(+2.0)$ while improving clean accuracy 39.8%$(+0.3\%)$. We note that GloRo additionally uses the TRADES-loss [ZYJ$^+$19] on *8C2F* to trade-off clean for robust accuracy. CLL simplifies this trade-off control via slack parameter p, see section 4.3.4. Regarding *SOC* on *LipConv*, we find 10 layers to perform slightly better than 20 (training setup in appendix B.2). This is in line with the observation in [SSF21]: adding more layers can lead to degrading performance. CLL, in contrast, clearly outperforms *SOC* on 20 layers with 26.0%$(+3.5)$ CRA ($\epsilon = 36/255$), 15.5%$(+8.0)$ CRA ($\epsilon = 108/255$) and 32.6%$(+0.5)$ clean accuracy on *LipConv-20*. These CRAs outperform the recent best methods *SLL* and *LBDN*. Differently to the AOL loss used in *SLL* and *LBDN* CLL also enables soft-constrained architectures like *8C2F* to achieve sota-performances. I.e., when choosing a lower slack value $p = 0.025$ and $\epsilon_{train} = 1.0$, CLL on *8C2F* out-competes even *LBDN* [WM23]. I.e., we increase CRAs for $\epsilon = 36/255$ to 25.3%$(+0.6)$ and for $\epsilon = 108/255$ to 13.8%$(+0.4)$ while maintaining clean accuracy 33.5%$(+0.1)$. Interestingly, *8C2F* has fewer parameters than *Sandwich* and *XL*: $4.3M$ vs $39M$ vs $1.1B$ [WM23],

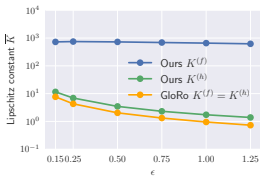
Next, we consider CIFAR-10 and CIFAR-100 (table 4.2) GloRo applied to *6C2F* and *4C3F* produces a CRA ($\epsilon = 36/255$) of under 60% (58.4% and 59.6% respectively) on CIFAR-10. Note that our reimplementation of GloRo improves CRA to 59.6%$(+1.2)$ upon the reported baseline in [LWF21]. Replaced with CLL, we report gains to 61.3%$(+2.9)$ and 61.4%$(+1.8)$ respectively. Additionally, we increase clean accuracy to 77.6% for both ($+0.6$ and $+0.2$ respectively). Thereby, outperforming the local Lipschitz bound extension *Local-Lip-B*[HZS$^+$21] (60.7% CRA), which utilizes expensive sample dependent local Lipschitz bounds to increase tightness. We also compare to *SOC* [SSF21], *CPL* [MDAA22] and *SLL* [AHD$^+$23], which constrain all layers to have Lipschitz constant 1. Here, we consider the *LipConv-20* architecture for *SOC* and the *XL*-architectures for *CPL* and *SLL*. *LipConv-20* contains 20 layers and *XL* 85. *SOC* achieves a clean accuracy of 76.3% on CIFAR-10, which CLL improves to 77.4%$(+1.1)$ while also improving CRA on all tested margins. E.g. for $\epsilon = 36/255$ we report a gain to

Table 4.2: Continuation of table 4.1 but on CIFAR-10 and CIFAR-100. Additional architectures evaluated: *4C3F*[WSMK18], *6C2F*[LLP20]. Local-Lip-B tightness is estimated by reading values off of figure 2a [HZS+21].

	Method	Model	Clean (%)	CRA $\frac{36}{255}$ (%)	CRA $\frac{72}{255}$ (%)	CRA $\frac{108}{255}$(%)	$\overline{K}^{(f)}$	$\overline{K}^{(h\circ f)}$	Tightness (%)
CIFAR-10	Local-Lip-B[HZS+21]	6C2F	77.4	60.7	-	-	7.5		≈80
	GloRo[LWF21]	6C2F	77.0	58.4	-	-	15.8		70
		4C3F†	77.4	59.6	40.8	24.8	7.4		71
	CLL	6C2F	**77.6 (+0.6)**	61.3 (+2.9)	43.5	27.7	709.3	11.6	78 (+8)
		4C3F	77.6 (+0.2)	**61.4 (+1.8)**	**44.2 (+3.4)**	**29.1 (+4.3)**	72.1	11.6	**80 (+9)**
	SOC[SSF21]	LipConv-20	76.3	62.6	48.7	**36.0**	5.6		86
	CLL	LipConv-20	**77.4 (+1.1)**	**64.2 (+1.6)**	**49.5 (+0.8)**	**36.7 (+0.7)**	35.8	14.7	86
	CPL[MDAA22]	XL	78.5	64.4	48.0	33.0	$\sqrt{2}$		78
	CLL	XL	**78.8 (+0.3)**	**65.9 (+1.5)**	**51.6 (+3.6)**	**38.1 (+5.1)**	34.6	11.8	**80 (+2)**
	SLL[AHD+23]	XL	**73.3**	**65.8**	**58.4**	**51.3**	$\sqrt{2}$	6.7	88
	CLL	XL	73.0	65.5	57.8	51.0	59.4	10.6	88.0
CIFAR-100	SOC[SSF21]	LipConv-20	47.8	34.8	23.7	15.8	6.5		**85 (+1)**
	CLL	LipConv-20	**48.2 (+0.4)**	**35.1 (+0.3)**	**25.3 (+1.6)**	**18.3 (+2.5)**	45.4	9.2	84
	CPL[MDAA22]	XL	47.8	33.4	20.9	12.6	1.6		74
	CLL	XL	**47.9 (+0.1)**	**36.3 (+2.9)**	**28.1 (+7.2)**	**21.5 (+8.9)**	42.0	7.6	**79 (+5)**
	SLL[AHD+23]	XL	46.5	36.5	29.0	23.3	1.5	6.0	**81**
	CLL	XL	**46.9 (+0.4)**	**36.6 (+0.1)**	29.0	**23.4 (+0.1)**	1.3	6.5	80



ϵ_{train}	GloRo	Ours
0.15	58.8	60.6 (**+3.0%**)
0.25	49.6	51.6 (**+4.0%**)
0.50	35.9	37.0 (**+3.0%**)
0.75	27.0	28.7 (**+6.3%**)
1.00	21.2	22.5 (**+6.3%**)
1.25	16.9	17.4 (**+3.3%**)



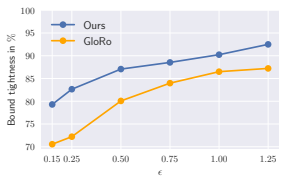


Figure 4.5: Left column: Slack governs robust accuracy trade-off in *4C3F* on CIFAR-10. Remaining columns: Under increasing $0.15 \leq \epsilon \leq 1.25$ on CIFAR-10, we compare our method with GloRo on *4C3F*. We report consistently better CRA with at least 3% improvement (left table). Our Lipschitz bounds are less constrained and $K^{(f)}$ is decoupled from the loss (middle figure). And lastly, our method produces tighter bounds (right).

64.2%(+1.6) and for $\epsilon = 128/255$ a gain to 36.7%(+0.7). Similarly, when applying CLL to *CPL-XL*, we report CRA gains to 65.9%(+1.5) and 38.1%(+5.1) ($\epsilon = 36/255$ and $\epsilon = 128/255$ respectively) while retaining clean accuracy (+0.3). However, CLL on *SLL-XL* provides no improvements. This is due to *SLL* using the *AOL*-loss [PL22], which has similar properties to CLL on K=1 constrained models. We provide a discussion in section 4.3.4. On CIFAR-100 though, CLL provides improvements on all tested models. On *CPL-XL*, we improve CRA on $\epsilon = 108/255$ to 21.5%(+8.9) while retaining clean accuracy 47.9%(+0.1). On *SLL*, we report slight gains in clean accuracy to 46.9%(+0.4) and CRA (+0.1 for both $\epsilon = 36/255$ and $\epsilon = 128/255$).

Lipschitz bound and tightness. CLL offers increased control over the Lipschitz bound of the model $K^{(h\circ f)}$. Across all datasets, we find CLL to increase the Lipschitz constant $K^{(h\circ f)}$ over the respective baselines (while improving clean and robust accuracies). On soft-constrained models like *8C2F* on Tiny-ImageNet, CLL allows a doubling of the constant over GloRo (7.3 vs 3.9). Similarly, on the hard-constrained model *LipConv-20* on Tiny-Imagenet, we observe another doubling over SOC (11.6 vs 6.4). This is consistent with $K^{(h\circ f)}$ on CIFAR. $K^{(h\circ f)}$ of *LipConv-20* is increased with CLL from 5.6 to 14.7. We note an exception for methods that

utilize the *AOL*-loss [PL22]: SLL and LBDN. On these models, we find the constant $K^{(h \circ f)}$ to be highly similar, e.g. SLL-XL on CIFAR-100 ($K^{(h \circ f)} = 6$ vs 6.5). Importantly, these constant changes come with increased tightness as well. On Tiny-ImageNet, we increase tightness from 53% to 64%$(+11)$ on *8C2F* over GloRo and from 81% to 84%$(+3)$ on *LipConv-20* over SOC. Similarly, on CIFAR-10, we increase tightness from 71% to 80% on *4C3F* over GloRo. In general, we observe the largest tightness improvements on soft-constrained models, although improvements on *CPL* are substantial. Here we gain, $+2$ and $+5$ percent points on CIFAR-10 and CIFAR-100 respectively.

4.3.4 Analysis and ablation

We considered different design choices when training with CLL. An important aspect being the robust accuracy trade-off, which we investigate in the following by controlling slack. We also investigate the bound, tightness and CRA over increasing margin width on CIFAR-10 and analyze the impact of the regularizing coefficient λ. Furthermore, we discuss selecting ϵ and p on *8C2F* for Tiny-ImageNet.

Slack. As discussed in section 4.2.4, we can regard the calibration probability p as slack, which trades-off CRA and clean accuracy. We report both on *4C3F* for $p \in [0.01, 0.8]$ in figure 4.5 (left). An increase in p decreases clean accuracy (green) from 80% to 68% but CRA increases to a peak at $p = 0.4$ with 61.9%.

ϵ – p ablation. We showed, that slack p can be used to trade-off clean and robust accuracies on *4C3F* on CIFAR-10 for a fixed ϵ_{train}. For completeness, we extend this experiment with the question: can we maximize the margin while retaining clean accuracy by reducing slack. We evaluate combinations of ϵ_{train} and slack on *8C2F* on Tiny-ImageNet. Results for clean accuracies and CRA for $\epsilon = 36/255$ and $\epsilon = 108/255$ is presented in table 4.3. Generally, we observe a similar trend to the main paper results. With decreasing p and ϵ, we observe an increase of clean accuracy while CRA decreases and in reverse: a decrease in clean accuracy while CRA increases with increasing p and ϵ. Importantly though, we can indeed find a configuration that increases the margin, improves CRA and retains clean accuracy. Take $\epsilon = 0.5$ and $p = 0.05$ for example. This configuration achieves 37.1% clean and 25.7% and 12.0% robust accuracies. We can find a configuration with higher ϵ but most importantly lower p that outperforms all metrics: $\epsilon = 0.75$ and $p = 0.005$. This configuration achieves 37.5%$(+0.4)$ clean accuracy and 26.0%$(+0.3)$ and 12.0% robust accuracies. And, as discussed in section 3.3 (main paper), we also find the better configuration to obtain a higher Lipschitz constant, i.e. $K^{(h \circ f)} = 8.0$ vs $K^{(h \circ f)} = 7.4$. Consequently, training for larger ϵ margins can still result in overall better clean and robust accuracies when the slack is reduced appropriately. It needs to be seen, whether this relationship can be harnessed further for additional improvements.

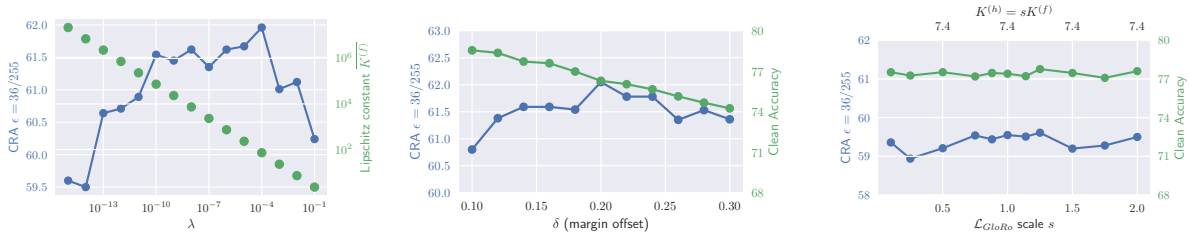
Comparison to fixed temperature scaling. We note that scaling the outputs of existing margin losses is not sufficient to control slack – and thus $K^{(h \circ f)}$, as the optimization is able to compensate for this factor. We run a simple experiment utilizing GloRo and varying temperature $s = 1/T$ in their loss $\mathcal{L}_{\text{GloRo}}(y; h(s \cdot f(x)))$ and present results in figure 4.6c. In contrast to our loss, we observe that such scaling (bottom x-axis) has no impact on $K^{(h \circ f)}$ (top x-axis) and little to no impact on accuracies.

Increasing margin width. In addition to our main results, we compare CLL and GloRo for larger ϵ. We train all experiments on *4C3F*. Different from before, we evaluate not for $\epsilon = 36/255$ but for the trained target, such that $\epsilon_{\text{train}} = \epsilon_{\text{test}}$. The table in figure 4.5, reports CRA and relative improvement over GloRo. With a minimum of 3%, we report consistent

Table 4.3: Clean and certified robust accuracies for 25 combinations of ϵ_{train} and p for CLL on *8C2F* on Tiny-ImageNet. The numbers shown are all trained with a random seed value of 1.

	Clean accuracy					CRA 36/255					CRA 108/255				
$p \setminus \epsilon_{\text{train}}$	0.15	0.25	0.5	0.75	1.0	0.15	0.25	0.5	0.75	1.0	0.15	0.25	0.5	0.75	1.0
0.1	43.7	40.3	36.2	32.4	30.4	22.5	25.2	25.9	25.1	24.1	4.9	8.7	12.8	14.1	14.8
0.05	44.2	42.0	37.1	34.5	32.1	21.8	25.1	25.7	26.0	25.2	4.2	7.9	12.0	13.7	14.5
0.01	45.4	43.0	39.7	36.4	35.1	20.0	23.4	25.9	25.6	25.7	3.4	6.7	10.7	12.4	13.2
0.005	44.4	43.4	39.5	37.5	35.5	19.5	23.2	25.5	26.0	26.0	2.8	6.5	10.4	12.0	12.9
0.001	45.3	43.9	40.7	38.5	37.2	18.5	21.7	24.8	25.5	25.9	2.7	5.8	9.4	11.1	12.0

	$K^{(h \circ f)}$				
$p \setminus \epsilon_{\text{train}}$	0.15	0.25	0.5	0.75	1.0
0.1	19.6	11.8	5.9	3.9	2.9
0.05	24.4	14.7	7.4	4.9	3.7
0.01	35.3	21.2	10.6	7.1	5.3
0.005	40.0	24.0	12.0	8.0	6.0
0.001	50.7	30.4	15.2	10.1	7.6



(a) CLL is insensitive to a broad range of λ choices.

(b) Tuning δ can improve CRA further.

(c) Naïvely scaling outputs in GloRo has no effect on $K^{(h \circ f)}$ (top x-axis) or CRA (blue).

Figure 4.6: Hyperparameter sweeps for λ (left) and margin offset δ (middle) evaluated on CIFAR-10. λ specifies minimization pressure on $K^{(f)}$ and δ can be tuned beyond the canonical value to increase robust accuracy. The right figure shows that GloRo is not endowed with CLLs slack property.

relative improvement for all ϵ. The two right most plots of figure 4.5, display the Lipschitz constants across ϵ and tightness respectively. We see CLL utilizing higher $K^{(h \circ f)}$ throughout while maintaining higher tightness. Note that $K^{(f)}$ remains fairly unconstrained at $\approx 10^3$ with CLL, which is regularized via parameter λ. We analyze its effect in the next paragraph.

Regularizing coefficient λ. As discussed, λ is required to add slight regularization pressure on $K^{(f)}$ – the Lipschitz constant of the classifier f. We subsequently investigate the impact on CRA when choosing different values for λ. For this experiment, we use the *4C3F* architecture and train on CIFAR-10. Figure 4.6a shows CRA in blue (left vertical axis) and Lipschitz constant $\overline{K}$ in green (right axis) over parameter λ which is increased from 10^{-15} to 10^{-1}. Note that $K^{(f)}$ rises exponentially with decreasing λ. We observe a wide range of values leading to state-of-the-art CRA results ($10^{-10} \leq \lambda \leq 10^{-4}$), indicating that CRA is fairly insensitive to the choice of λ. Yet we also see two substantial drops in performance when λ is set too high ($\geq 10^{-3}$) or too low $\leq 10^{-11}$). Choosing λ too high leads to reduced complexity in f, which is undesired. On the other hand, it is not directly evident why choosing λ too low also impairs performance. This observation necessitates further investigation in future work.

Margin offset as tuneable parameter. In our CLL definition (section 4.2.2), we add or subtract ϵ as hard margin offsets to improve margin training. We can consider this offset as hyper-parameter δ, such that the calibrated logistic is now defined as:

$$\hat{h}(f(x); y) = h\left(-\frac{y\delta}{\sigma_\epsilon(p)} + \frac{1}{\sigma_\epsilon(p)}\frac{f(x)}{K}\right).$$

Instead of the canonical value $\delta = \epsilon$, we choose 11 values between 0.1 and 0.3 and plot robust and clean accuracy in figure 4.6b. While robust accuracy (blue curve) can be improved with greater δ, clean accuracy consistently decreases. Here, $\delta = 0.2$ achieves a robust accuracy of 62.0%, improving on our main results (+0.6%). Concurrently, clean accuracy decreases to 76.2%(−1.4%).

Slack control on two-moons. In our discussion on two-moons we have highlighted the connection between slack and Lipschitz constant. Specifically, we showed that choosing low slack can train models with perfect CRA. Here, we show that choosing too much slack does indeed result in increased classifier smoothness. We follow the setup in the main paper but train only for $2\epsilon = 0.3$ – the true margin width. The results for decreasing p from 0.75 to 0.0001 are shown in figure 4.8. While the smallest shown $p = 0.0001$ nearly achieves perfect CRA, a higher slack of $p = 0.25$ produces very similar results to GloRo (see figure 4b, main paper), with training points remaining within the margins. Increasing slack even further $p = 0.75$, we see a near piecewise-linear classifier, unsuitable to generalize to the data – as discussed in section 3.3 (main paper).

SOC on two-moons. In section 4.3.2 we evaluated GloRo and CLL on two-moons to highlight the influence of K on classifier smoothness. For completeness, we additionally present results with 1-Lipschitz constrained models in figure 4.9. We train models of three different depths: 3, 7 and 25 (in columns) using the *SOC* method [SSF21]. *SOC* employs the use of a margin regularization parameter γ, which we evaluate for three different values: 0.0, 0.3 and 1.0 (in rows). Overall, we find none of the tested configurations perform better than 90% CRA. All decision functions are too smooth to fit the data, while the Lipschitz constant $K^{(h \circ f)} = 2.0$ is very low. These observations are in line with our discussions in section 4.2.4 on the link between K and classifier complexity.

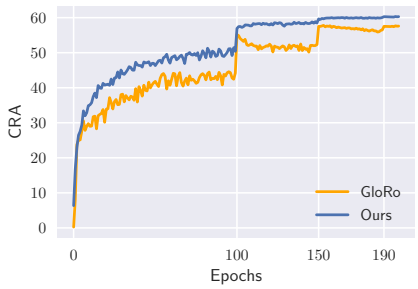


Figure 4.7: Our loss converges faster. Here for *4C3F* on CIFAR-10.

For training CLL we considered different hyperparameters in order to maximize performance. In the following, we provide additional material on convergence speed improvements over GloRo, finding optimal values for the regularization coefficient λ on CIFAR-10 and evaluating the impact of slack (p) on CRA on the constrained *LipConv* architectures. Finally, we show empirical evidence that existing Lipschitz margin losses like GloRo do not possess slack control.

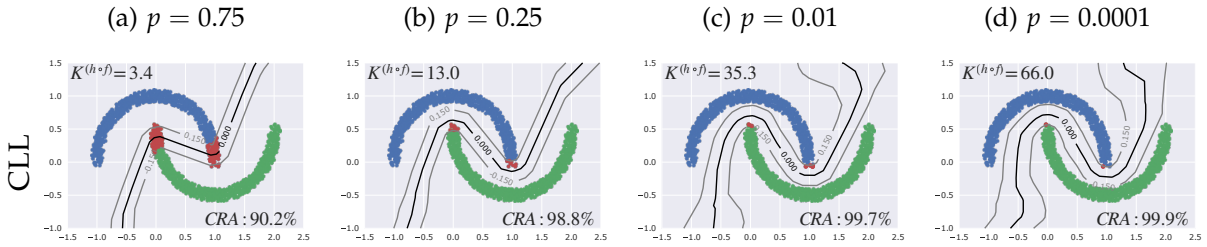


Figure 4.8: Slack control on two moons with CLL. With high p and thus high slack, the decision boundary becomes overly smooth. To reach a perfect CRA score as shown in the main paper, p needs to be smaller than 0.0001 (right most figure).

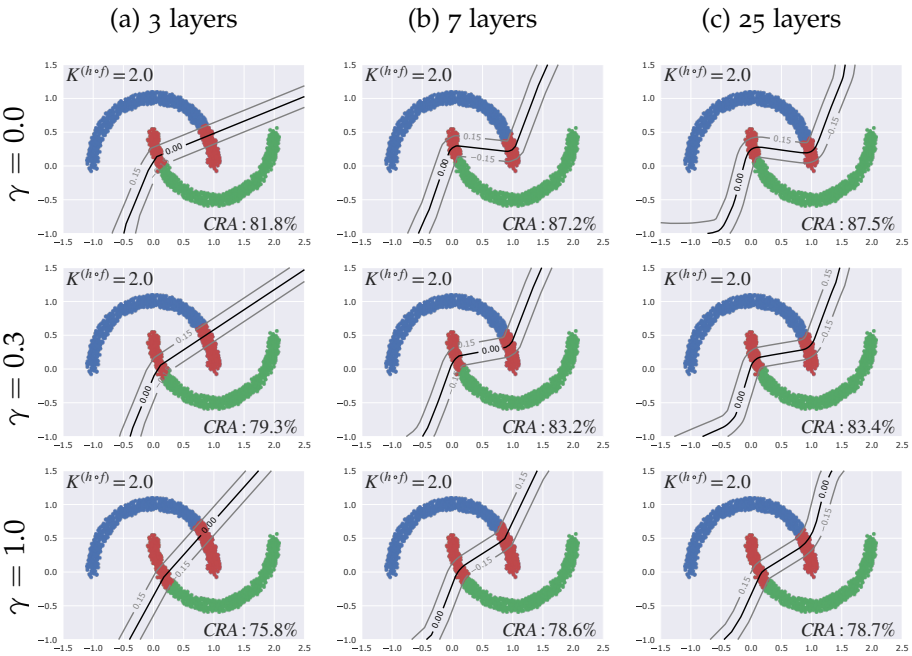


Figure 4.9: 1-Lipschitz constrained SOC is too smooth to fit two-moons, irrespective of depth or its margin controlling parameter γ. All networks use last-layer normalization [SSF21], which increases $K^{(h \circ f)}$ to 2.0

Faster convergence with CLL. For training deep models we highlight an additional benefit provided by CLL that is of practical importance: **faster convergence**. We plot the certified robust accuracy (CRA) over number of epochs in figure 4.7 and find that our loss converges faster and is less noisy over the alternative large margin loss: GloRo. Note that in this instance, we train GloRo with a fixed ϵ schedule to draw a fair comparison.

4.4 Conclusion

We proposed a new loss, Calibrated Lipschitz-Margin Loss (CLL), for Lipschitz margin training that is calibrated to the margin width. CLL reveals two intriguing properties: (i) the calibrated distribution width can be interpreted as slackness and (ii) slackness governs the smoothness of the model – and thereby the Lipschitz constant. The ramifications are important for improving certified robustness with Lipschitz constants. The constant can be large – implying increased model complexity and accuracy – if the model is capable of separating the data well. We illustrated these mechanics on two-moons, highlighting the implications for Lipschitz margin training and provided additional results on CIFAR-10, CIFAR-100 and Tiny-ImageNet. Applied across a wide range of datasets and methods, CLL consistently improved clean and certified robustness.

II

Part 2

In the second part of this dissertation, we move from improving adversarial robustness to addressing the *black-box* property of deep networks. In this dissertation, we specifically look at the *inspectability* of representations. That is, we consider a representation *inspectable* if we can understand its semantic. We pursue a pragmatic form of understanding, which is, the outputs should align with visual concepts in images – a wheel representation should provide a high output value for wheel pixels and a low value otherwise. This definition of inspectability has wide implications for lifting the *black-box* property: predictions can be traced back to their semantics and their respective evidences. Failure cases can be better analyzed, models could be improved and importantly: we could designate input regions as error-prone, omitting a prediction altogether.

In chapter 5, we propose the foundation: Semantic Bottlenecks (SBs). Semantic Bottlenecks (SBs) introduce a single additional layer into a model, for which each output channel is aligned with a single visual concept. We show that constructing SBs is possible in two variants without impairing performance: supervised, requiring additional concept annotation and unsupervised.

In chapter 6, we utilize SBs to investigate failure cases on the task of street scene segmentation. Tracing each individual prediction back to its receptive field on the SB, we cluster the concept evidences to find error modes and their respective differences to nearest true positives. We utilize concept intervention at the bottleneck to test whether the final outputs rely on the expected input concepts.

The previous chapters focus on semantic segmentation, requiring pixel-level annotations for the supervised SB. In the final chapter 7, we provide an indication that scalar image-level annotations (e.g. there is a wheel in the image) will suffice for image classification tasks. We base this conjecture on a related study: we find up to 30% of the last convolutional layers can be reduced to a spatial size of 1×1 without impairing clean accuracy. This falls in line with related work showing convolutional networks can be trained with very small receptive fields and still achieve competitive performances.

5 Semantic bottlenecks: Quantifying and improving inspectability of deep representations

Contents

In this first chapter of part II, we address the black box issue of deep learning, which reduces its trustworthiness, especially, in high stake applications. This black-box issue is, in part, the result of hundreds or thousands of representations that are not inspectable. We argue that there are at least two reasons making inspectability challenging: (i) representations are distributed across hundreds of channels and (ii) a unifying metric quantifying inspectability is lacking.

In this chapter, we address both issues by proposing Semantic Bottlenecks (SBs), which can be integrated into pretrained networks, to align channel outputs with individual visual concepts and introduce the model agnostic *Area Under inspectability Curve (AUiC)* metric to measure the alignment. We present a case study on semantic segmentation to demonstrate that SBs improve the AUiC up to six-fold over regular network outputs. We explore two types of SB-layers in this work. First, concept-supervised SB-layers (SSB), which offer inspectability w.r.t. predefined concepts that the model is demanded to rely on. And second, unsupervised SBs (USB), which offer equally strong AUiC improvements by restricting distributedness of representations across channels. Importantly, for both SB types, we can recover state of the art segmentation performance across two different models despite a drastic dimensionality reduction from 1000s of non aligned channels to 10s of semantics-aligned channels that all downstream results are based on.

5.1 Introduction

While end-to-end training is key to top performance of deep learning, learned intermediate representations with typical training methods remain opaque to humans. Furthermore, assessing inspectability has remained a fairly elusive concept since its framing has mostly been qualitative (e.g. saliency maps). Given the increasing interest in using deep learning in real

world applications, inspectability and a quantification of such is critically missing.

Goals for inspectability. To address this, prior work on inspectability has proposed to improve the spatial coherency of activation maps [ZNWZ18] or to cluster representations to acquire outputs of low dimensionality either with supervision [BHJ18] or without [CLT+19, YKA+19]. In contrast, we demand information in each channel to be represented by a single semantic concept. This is derived from a simple observation: distributed representations do not lend themselves to trivial interpretation (see bottom of figure 5.1). In order to reduce distributedness and improve inspectability we propose to adapt deep networks via three criteria. (i) Reduce the number of channels to a minimum, (ii) associate them with semantic concepts, and, at the same time, (iii) aim to lose as little overall performance as possible. While goal (i) is not necessary to reduce distributedness, it substantially reduces the amount of information that needs to be interpreted by a human observer. In our view such semantics based inspectability can be seen as a way towards achieving true interpretability of deep networks.

Semantic Bottlenecks. We propose to implement these three goals via Semantic Bottlenecks (SBs). SBs are single (convolutional) layers that are easy to integrate into existing, pretrained, networks which map the highly distributed outputs to a semantically aligned space of reduced dimensionality. To achieve this alignment, SBs can be constructed either supervised or unsupervised by regularizing the distributedness of outputs and encourage the network to learn specialized features interpretable by humans. Our unsupervised SBs activate only a single channel for any given spatial location of its input map. While goal (ii), the semantic alignment, is not explicitly solvable in an unsupervised fashion, we find our regularization results in drastically improved channel inspectability. We show this qualitatively and more importantly: quantitatively with a metric comparing activation maps and concept annotations.

Our contributions are three-fold. Firstly, we introduce two network layers we term Semantic Bottlenecks to improve alignment with semantic concepts as described in the last paragraph. The first—supervised SBs—improves alignment by aligning each channel with a single visual concept and the second—unsupervised SBs— align by restricting distributedness. Secondly, we show SBs can be integrated into two state-of-the-art models at early, intermediate or late layers without impairing performance, even when reducing the number of channels, e.g., from 4096 to 20. Finally, we introduce the novel Area Under inspectability Curve (AUiC) metric to quantify alignment between channel outputs and visual concepts for any model and show our SBs improve over baselines up to six-fold.

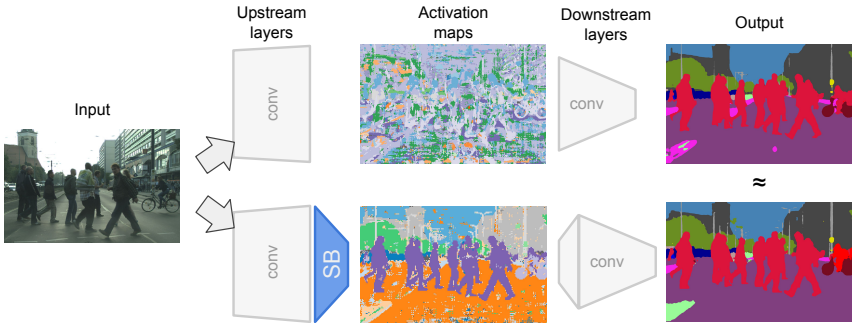


Figure 5.1: Semantic Bottleneck (SB) layers can be integrated into any model and act as inspectable basis for subsequent layers. Their activation maps are coherent and well aligned with semantic concepts (left). The alternative, regular deep networks, have layers with highly distributed representations across hundreds of channels, making them hard to inspect (right).

5.2 Semantic Bottlenecks

To approach more inspectable intermediate representations we demand information in each channel to be represented by a single semantic concept. We propose two variants to achieve this goal: (i) supervise channels to represent unique concepts and (ii) restrict the distributedness of representations across channels to produce concept-specialized channels. We construct both variants as layers that can be integrated into pretrained models, mapping non-inspectable representations to an inspectable semantic space. We name these supervised and unsupervised Semantic Bottlenecks (SB).

Formal definition. Before we start introducing the supervised and unsupervised variants in detail, we first define the integration of SBs in an arbitrary network. For an architecture with L-layers, we define its functional composition:

$$f_{\text{Net}}(x_0) = (f_L \circ f_{L-1} \circ ... \circ f_1)(x_0), \tag{5.1}$$

where x_0 defines the input to the first layer. For clarity let's also name the output of each layer:

$$x_i := (f_i \circ f_{i-1} \circ ... \circ f_1)(x_0), \tag{5.2}$$

where $0 < i \leq L$. We denote the output dimensionality of each layer as $x_i \in \mathbb{R}^{N_i}$ and design our SB such that it acts as bottleneck, reducing this dimensionality:

$$f_{\text{SB}} \ : \ \mathbb{R}^{N_i} \rightarrow \mathbb{R}^{N_{\text{SB}}}, \tag{5.3}$$

applied to a layer f_i, such that $N_{\text{SB}} < N_i$. To integrate an SB after layer f_i (where $0 < i < L$), we extend our network definition from above:

$$f_{\text{SBNet}}(x_0) = (f_L \circ ... \circ f_{i+1} \circ f_{\text{r}} \circ f_{\text{SB}} \circ f_i \circ ... \circ f_1)(x_0). \tag{5.4}$$

Note that we also integrated an additional layer f_{r} that ensures its output dimensionality matches the original layer: $f_{\text{r}} \ : \ \mathbb{R}^{N_{\text{SB}}} \rightarrow \mathbb{R}^{N_i}$. To emphasize which parameters of the network are subject to training, we finally define the following operator Θ:

$$\theta_i = \Theta(f_i), \tag{5.5}$$

which returns the set of all parameters θ for a given function f. How we setup f_{SB} and f_{r} and train them supervised or unsupervised is the subject of the following sections 5.2.1 and 5.2.2.

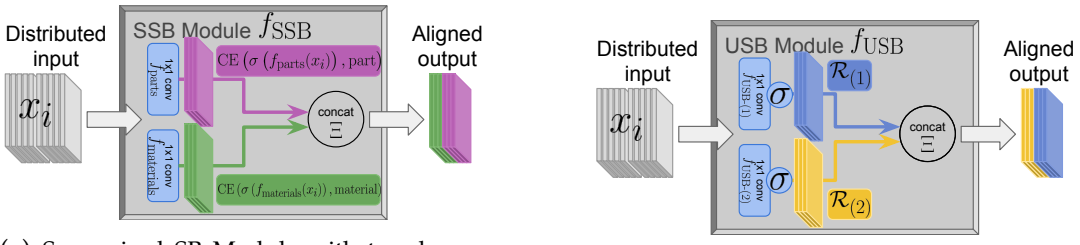
Case Study. To show the utility of SBs, we choose street scene segmentation on the Cityscapes dataset [COR+16] since it is a difficult task that traditionally requires very large models and has a practical application that has direct benefits from inspectability: autonomous driving. Cityscapes consists of 19 different classes, 2 975 training images and 500 validation images, both densely labeled. We use PSPNet [ZSQ+17] and the recent MS-OCRNet [TSC20], both based on ResNet-101 [HZRS16], due to their strong performance on Cityscapes and because residual networks are abundantly used in computer vision tasks today. PSPNet uses a pyramid pooling module before classification and MS-OCRNet uses a hierarchical multi-scale object-contextual representation module [YCW20, TSC20]. Since both architectures are residual, we define their general functional composition w.r.t. their stages (also see figure 5.3):

$$f_{\text{ResNet}}(x_0) = \left(f_{\text{stage-}L} \circ f_{\text{stage-}(L-1)} \circ ... \circ f_{\text{stage-1}}\right)(x_0), \tag{5.6}$$

where a stage contains M residual blocks $\mathcal{B}$, all retaining the dimensionality:

$$f_{\text{stage-i}}(x_{i-1}) = (\mathcal{B}_M \circ \mathcal{B}_{M-1} \circ ... \circ \mathcal{B}_1)(x_{i-1}). \tag{5.7}$$

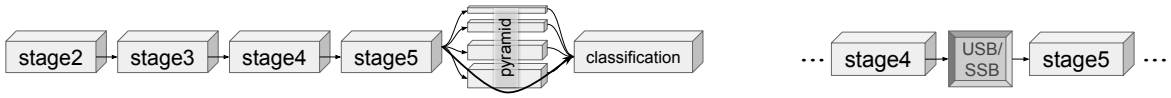
Each block implementing the residual function $\mathcal{B}(x) = \mathcal{F}(x) + x$, for a function $\mathcal{F}$.



(a) Supervised SB Module with two layers: parts- and material-specialization.

(b) Unsupervised SB Module with two layers.

Figure 5.2: Schematics for both Semantic Bottleneck (SB) types. The highly distributed activations x_i from the original layer are fed through one or multiple 1×1-conv layers to specialize channels by semantic alignment. SSBs are concept-supervised and USBs are regularized ($\mathcal{R}$).



(a) Simplified PSPNet architecture indicating stages. Each stage consisting of multiple layers or residual blocks.

(b) PSPNet with SB after stage4.

Figure 5.3: Sequence of stages in PSPNet architecture and integration of an SB. MS-OCRNet follows the same structure, but replaces the pyramid and last classification module with a context integration module.

5.2.1 Supervised Semantic Bottlenecks (SSBs)

SSBs (f_{SSB}) perform concept supervision on each SB channel using additional annotations. Ideally, we possess pixel-annotations for an exhaustive set of subordinate concepts like colors, textures and parts to decide which are required to recover performance on particular tasks. Yet, we show that an encouragingly small task-specific selection is sufficient to satisfy both desiderata of performance and inspectability.

Choosing concepts for Cityscapes. For our supervised SB-layer we choose concepts based on task relevancy for Cityscapes. *Broden+* [XLZ+18] is a recent collection of datasets which serves as a starting point for the concept annotations we require for the SSBs. It offers thousands of images for objects, parts, materials and textures for which the first three types come with pixel level annotations. Here, a pixel can have multi-label annotations. Based on the 377 part and material concepts available (351 parts sourced from ADE [ZZP+17] and Pascal-Part [CML+14] and 26 materials sourced from OpenSurfaces [BUSB13]), we compile a subset of 70 Cityscapes-relevant concepts listed in table 5.1.

Setup. Let's specify the SSB setup formally for semantic segmentation. That is, our inputs and outputs are two dimensional. In our experiments, we compose the SSB to have two bottlenecks: one for materials, one for parts. In the general case, an SSB may have N bottlenecks, each containing channels of a specific concept category:

$$f_{\text{SSB}} := \Xi\left(f_{\text{cat-1}}, f_{\text{cat-2}}, ..., f_{\text{cat-}N}\right), \tag{5.8}$$

where Ξ defines the concatenation along channels, such that $f_{\text{SSB}} : \mathbb{R}^{N_i \times H \times W} \rightarrow \mathbb{R}^{\sum_{j=1}^{N} N_{\text{cat-}j} \times H \times W}$ for outputs with spatial extent W and H. Recall from our f_{SBNet} definition in equation 5.4 that we need an additional function f_{r} to revert the dimensionality reduction. Instead of adding an additional layer, potentially increasing representational power (not part of our goals), we

Table 5.1: Selection of 70 Broden-concepts that we deemed relevant for street scene segmentation. The first row lists all used materials, the second row all parts. Each part is organized into part (right) of object/concept (middle).

Materials	Brick, Fabric, Foliage, Glass, Metal, Plastic, Rubber, Skin, Stone, Tile, Wood	
Parts	Sky	Cloud
	Building	Window, Door, Roof, Shop, Wall
	Person	Leg, Head, Torso, Arm Eye, Ear, Nose, Hand, Hair, Mouth, Foot, Eyebrow, Back
	Road	Crosswalk
	Car	Window, Door, Wheel, Headlight, Mirror, Roof, Taillight, Windshield, Bumber
	Van	Window, Door, Wheel, Headlight, Taillight, Windshield
	Truck	Wheel, Windshield
	Bus	Window, Door, Wheel, Headlight, Mirror
	Train	Head, Headlight, Headroof, Coachroof
	Lamp	Arm, Shade, Bulb
	Bike	Wheel, Handle, Saddle, Chain
	Motorbike	Wheel, Headlight, Handle

softmax	mIoU
without	74.7%
with	43.0%

Table 5.2: PSPNet-SSB@pyramid trained once with and without softmax activation. With softmax activation performance during inference is impaired substantially.

can also adapt the very first residual block of the subsequent stage $f_{\text{stage-}(i+1)}$ to accept the SSB-output dimensionality. That is, the very first convolutional layer is replaced with a layer working on the SSB dimensionality $\mathbb{R}^{\sum_{j=1}^{N} N_{\text{cat-}j} \times H \times W}$. The full SSB-ResNet architecture can be defined as follows:

$$f_{\text{SSB-ResNet}}(x_0) = (f_{\text{stage-}L} \circ ... \circ \hat{f}_{\text{stage-}(i+1)} \circ f_{SSB} \circ f_{\text{stage-}(i)} \circ ... \circ f_{\text{stage-1}})(x_0), \tag{5.9}$$

where $\hat{f}_{\text{stage-}(i+1)}$ is the adapted stage. Note, that we do not apply a non-linearity to SSB-outputs. We observed inferior performance during inference when doing so (see table 5.2). Nonetheless, softmax (σ) is applied when calculating the cross entropy loss:

$$\mathcal{L}_{\text{cat-}j}(x_i, label) = \text{CE}\left(\sigma\left(f_{\text{cat-}j}(x_i)\right), label\right). \tag{5.10}$$

Implementation details. For all of our SSBs, we define two bottlenecks: one for parts and one for materials:

$$f_{\text{SSB}} := \Xi\left(f_{\text{parts}}, f_{\text{materials}}\right). \tag{5.11}$$

We follow figure 5.2a, which shows the structure of our SSBs. Two linear bottleneck layers (blue rectangles) receive the distributed input x_i from a pretrained model and are supervised by an auxiliary loss to map them to target concepts (colored boxes). Given the dense prediction task of our case study, we use 1×1-convolutions to retain the spatial resolution:

$$f_{\text{parts}}(x_i) = \mathcal{W}_{\text{parts}} \cdot x_i + b_{\text{parts}} \tag{5.12}$$

$$f_{\text{materials}}(x_i) = \mathcal{W}_{\text{materials}} \cdot x_i + b_{\text{materials}} \tag{5.13}$$

In our case study, we place SSBs at three different locations in the network. Early, namely after stage3 (see figure 5.3), middle, namely stage4 and late. Late differs between PSPNet and MS-OCRNet. We apply the SSB after the pyramid module for PSPNet and for MS-OCRNet after stage5.

Table 5.3: Segmentation results on Cityscapes validation set for different placements of SSB. ‡ denote results are obtained with models trained from scratch by our setup.

location	layer	#concepts (materials, parts)	mIoU PSPNet‡	mIoU MS-OCRNet‡
Vanilla		N/A	77.6	79.6
early	stage3 (512 input feat.)	70 (11, 59)	76.4	78.2
middle	stage4 (1024 input feat.)	70 (11, 59)	77.4	77.8
late	stage5 (2048 input feat.)	70 (11, 59)	–	77.3
	pyramid (4096 input feat.)	70 (11, 59)	74.7	–

Training details. The authors description of training the MS-OCRNet is fairly extensive in contrast to PSPNet. It is pretrained on ImageNet as well as Mapillary [NORBK17], utilizes the validation set for training and performs auto-segmentation on the coarse training set to increase the number of images [TSC20]. Since we are mainly interested in rendering existing networks more inspectable – instead of outperforming baselines –, we choose the training setup of PSPNet to enable comparisons between both models. To construct baselines, we train both models for 200 epochs on the finely annotated Cityscapes data only, with a learning rate of 0.01, weight decay of 5e−4 and "poly" learning rate policy with power 0.9 [CPK+18, ZSQ+17, XLZ+18]. Integration and training of SSBs is done in two phases. The SSB-ResNet is constructed according to equations 5.9-5.13. In the first phase, only the SSB is trained on the out-of-Cityscapes-domain Broden+ by updating the parameters θ_{Phase1}:

$$\theta_{\text{Phase1}} := \Theta\left(f_{\text{SSB}}\right), \tag{5.14}$$

where Θ returns all SSB parameters (as introduced in eq. 5.5). We run the training for 5 epochs with a learning rate of 0.01, weight decay $5e-3$ and the same policy as for the base models. In the second phase, all subsequent layers are finetuned. The parameters subject to optimization are as follows:

$$\theta_{\text{Phase2}} := \bigcup_{k=i+2}^{L} \Theta(f_{\text{stage-}k}) \cup \Theta(\hat{f}_{\text{stage-}(i+1)}), \tag{5.15}$$

where L defines the number of stages. Phase 2 is trained again on Cityscapes with a lower learning rate of 0.002, keeping the policy and weight decay 5e−4. The number of epochs is fixed to 60. We train both the PSPNet and the MS-OCRNet on 4 GPUs with a total batchsize of 16 and using mixed-precision. The images are cropped for the PSPNet to size 713×713. Our PSPNet achieves 77.6% mIoU on the validation set, on par with the official reported number (78.4%). MS-OCRNet achieves 79.6% mIoU with our training, which is on par with the regular OCRNet performance (also 79.6% [YCW20]) but not as good as the best reported number (85.1%) involving the full training scheme.

5.2.1.1 *Recovering performance using SSBs*

As one of our 3 goals for inspectable deep networks, we strive to lose as little performance as possible. We test our SSB-augmented PSP- and MS-OCRNets on the original Cityscapes task and compare mIoUs (see table 5.3). We denote an SSB after *stageX* as *SSB@stageX*. We compare the PSPNet integrations first. Given our baseline mIoU of 77.6%, SSB@stage4 is able

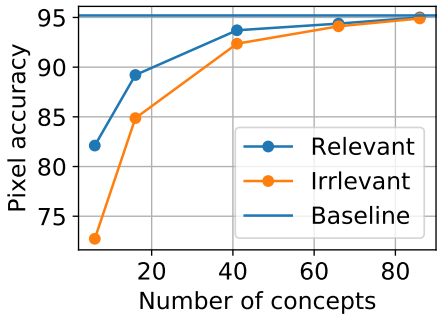


Figure 5.4: Task relevant concepts outperform irrelevant ones on PSPNet-SSB@pyramid.

to recover the full performance, while the applications to stage3 and the pyramid layer result in a slight decrease (76.4% and 74.7% respectively). Similarly for the MS-OCRNet integrations, we observe only slight performance drops from 79.6% down to 77.3% for stage5. As to be expected, MS-OCRNet achieves overall higher performance on all tested locations. Importantly, for both models: the reduction in the number of channels is substantial (e.g. 1024 reduced to 70 for stage4), indicating room to render complex networks more inspectable. This addresses point 1 of our 3 goals (channel reduction).

Bottleneck Tradeoff. Clearly, the performance is strongly dependent on the number of concepts used in the bottleneck. To re-establish the original performance we investigate the idea of increasing the bottleneck size, yet note that increasing the number of concepts is dependent on availability of annotations. In our case, there are no more than our 70 chosen concepts that we deemed necessary for the task. Objects we excluded, that could be considered relevant, are *mirror*, *column*, *stairway* or *door*. Most of these do already exist as parts of objects (see table 5.1) or are annotated on very different scenes (e.g. *mirror* is annotated on indoor scenes). In light of our first two goals—dimensionality reduction and semantic alignment—we here choose a minimal set of concepts for our experiments to reduce semantic overlap. To investigate the segmentation performance impact when using over- complete sets as well as reduced sets, we consider the PSPNet- SSB@pyramid configuration and vary the number of concepts from 6 to 86 - selected manually from a set of task relevant concepts and randomly from a disjoint irrelevant set. The set of irrelevant concepts contain all 377 Broden+ concepts minus the 86 relevant ones. We report the pixel accuracy results in figure 5.4 and find that relevant concepts result in improved accuracies over irrelevant ones – at least for small number of concepts.

5.2.2 Unsupervised Semantic Bottlenecks (USBs)

Clearly, the requirement for additional annotation and uncertainty regarding concept choice is a limitation of SSBs. To address this, we investigate the use of annotation free methods to (i) reduce number of channels, (ii) increase semantic association and (iii) lose as little performance as possible. Similar to SSBs, we address point (i) by integrating layers with low dimensionality. To address (ii) we propose to enforce *non*-distributed representations by performing one-hot relaxation. That is, we strive to achieve one-hot encoded outputs for which only a single channel is active for any given spatial location. We simplify this problem by relaxing an arg max objective and utilize appropriate regularization.

Unsupervised semantic alignment. One-hot relaxation does not explicitly enforce semantic alignment, yet it adapts a high-level idea implemented implicitly using supervision: each channel should be specialized in representing one concept only. Thus, we strive to maximally specialize each channel via approaching one-hot encodings. As we will later discuss, we find our method resolves implicit semantic alignment well, such that channels represent very specific concepts.

5.2.2.1 *Construction of USBs*

As for SSBs, we also integrate the USBs into pretrained models.

Setup. Let's specify the USB setup formally for a residual architecture with integration after stage $f_{\text{stage-}i}$. We define the output of stage-i as x_i (see equation 5.2). The composition of the network is similar to the SSB variant (equation 5.9):

$$f_{\text{USB}} := \Xi\left(f_{\text{USB-}(1)}, f_{\text{USB-}(2)}, ..., f_{\text{USB-}(N)}\right), \tag{5.16}$$

for N parallel bottlenecks whose outputs are concatenated along channels via concatenation operator Ξ, such that

$$f_{\text{USB}} \ : \ \mathbb{R}^{N_i \times H \times W} \rightarrow \mathbb{R}^{\sum_j N_{\text{USB-}(j)} \times H \times W}. \tag{5.17}$$

We present an example setup in figure 5.2b which contains two parallel bottlenecks. As for SSBs, we choose 1×1 convolutional layers with bias for each bottleneck $f_{\text{USB-}(j)}$:

$$f_{\text{USB-}(j)}(x_i) := \sigma\left(\mathcal{W}_{\text{USB-}(j)} \cdot x_i + b_{\text{USB-}(j)}\right). \tag{5.18}$$

Note that, unlike for SSBs, the softmax non-linearity σ is applied both during training and inference as it's integral part to the regularization.

One-hot relaxation. We demand our USBs to produce efficient, sparse codes. By design, we want to have only one active channel per spatial location: That is, $\arg\max_k a_k$, where a_k defines activation of channel k. To relax this non-differentiable problem we utilize softmax and investigate two approaches: adding an additional entropy loss $\mathcal{R}$ or utilizing a parameterization with temperature T. For the first approach using entropy loss, the probability distribution along channels of each $f_{\text{USB-}(j)}$ is used to calculate an additional loss $\mathcal{R}$:

$$\mathcal{R}_{\text{USB-}(j)}(x_i) = \frac{\lambda}{WH} \sum_{w=1}^{W} \sum_{h=1}^{H} \mathcal{H}\left(f_{\text{USB-}(j)}(x_i)_{h,w}\right), \tag{5.19}$$

where $\mathcal{H}$ defines the Shannon entropy and x_i is the output of stage-i ($f_{\text{stage-}i}$) resulting in an USB output tensor with $N_{\text{USB-}(j)}$ channels and spatial extent $H \times W$ ($f^{\text{USB-}(j)}(x_i) \in \mathbb{R}^{N_{\text{USB-}(j)} \times H \times W}$). The loss is scaled with factor λ. Here, the entropy $\mathcal{H}$ is calculated along channels $N_{\text{USB-}(j)}$ to measure the uncertainty w.r.t. the active feature for each spatial location h, w. We want to minimize this uncertainty to ideally reach maximum certainty: one-hot encoded outputs. For the second approach, we utilize the softmax temperature parameter, which we anneal during training. Its parametrization can be reduced to the following:

$$\sigma_T(x) := \sigma(x/T). \tag{5.20}$$

Starting with a high temperature, e.g. $T_0 = 1$ it is reduced quickly in τ training iterations to approach $\arg\max$. We define T at timestep t with polynomial decay: $T_t = T_0 + (T_0 - T_\tau) \cdot \left(1 - \frac{t}{\tau}\right)^\gamma$, where γ specifies how quickly T is decaying.

Table 5.4: One-hot relaxation comparison on PSPNet-USBs. To test distributedness, we replace softmax with arg max during inference and compare segmentation performances. Only annealing recovers performance.

layer	#channels $N \times K$	PSPNet			
		Entropy Loss		Temperature annealing	
		softmax mIoU	arg max mIoU	softmax mIoU	arg max mIoU
stage3	2×50	73.0	0.1	73.6	**71.7**
	4×50	74.0	36.7	74.7	**73.3**
stage4	2×10	75.7	32.9	75.1	**75.0**
	2×50	76.0	25.8	75.8	**75.5**

Table 5.5: Performance comparison between PSPNet and MS-OCRNet after integrating USBs regularized with temperature annealing. $\Sigma_{>0}$ denotes number of active channels – active meaning: the channel has an activation value greater than 0 at least once during validation. Bold numbers highlight best performing configuration per model.

location	layer	#channels $N \times K$	PSPNet		MS-OCRNet	
			$\Sigma_{>0}$	mIoU (ss)	$\Sigma_{>0}$	mIoU (ss)
Vanilla		N/A	*all*	77.6	*all*	79.6
early	stage3	2×50	33	71.7	17	**76.0**
		4×50	60	**73.3**	48	77.4
middle	stage4	1×50	18	72.4	12	75.9
		2×10	16	75.0	11	**77.5**
		2×50	14	**75.5**	20	77.3
late	stage5	1×50	–		23	77.2
		2×10			17	77.4
		2×50			23	**78.0**
	pyramid	1×50	40	**77.5**	–	
		2×10	19	75.4		
		2×50	31	75.6		

Implementation details. Entropy regularization is scaled with factor $\lambda = 0.1$ while T is kept at 1.0. For annealing we set $T_0 = 1.0$, $T_\tau = 0.01$ and $\gamma = 10.0$ for rapid decay. During inference, we compute arg max **instead of softmax** to acquire one-hot encoded outputs. We choose various bottleneck sizes for our USB experiments to evaluate their impact on performance, discussed in section 5.2.2.2.

Training details. Since the training is not dependent on additional concept labels, we can train the USB jointly with all subsequent layers. The set of parameters θ subject to training is specified as:

$$\theta := \bigcup_{k=i+2}^{L} \Theta\left(f_{\text{stage-}k}\right) \cup \Theta(\hat{f}_{\text{stage-}(i+1)}) \cup \Theta\left(f_{\text{USB}}\right), \tag{5.21}$$

where Θ returns all parameters of a function (as introduced in eq. 5.5) and L is the number of all stages.

5.2.2.2 *Recovering performance using USBs*

Here, we show in two parts that first, only USBs trained with temperature annealing produce one-hot encodings and second, that introducing USBs result in little to no performance impact

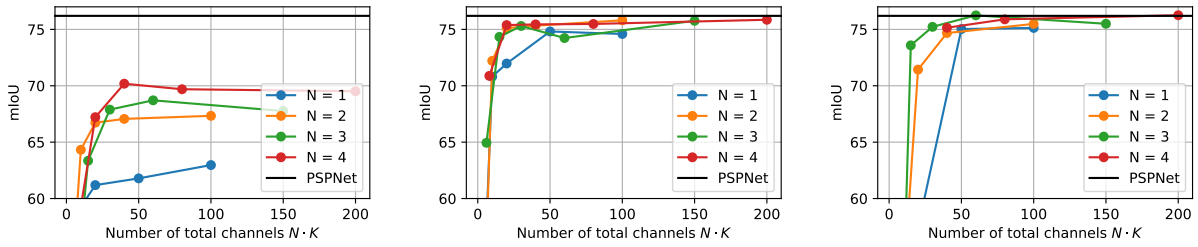
while drastically reducing number of channels. We compare the two regularization methods on two locations of PSPNet (stage3 and stage4) in table 5.4.

Temperature annealing enables one-hot encoding. Table 5.4 reports number of channels and two mIoU values per USB configuration—both evaluated on the Cityscapes validation set. The left column reports the unchanged PSPNet-USB performance, and the right (arg max) reports the performance when replacing the softmax with arg max during inference. This comparison enables a simple test. That is, how distributed (or how specialized) are the learned USB-representations. If the representations are not distributed across channels, taking the arg max will induce no or little performance loss. On the other hand, if representations are distributed, a single channel per spatial location does not express all of the required information—the performance will drop. We compare the two different regularization strategies—entropy loss and temperature annealing—and find that only the latter is able to retain performance. On stage3 we see a slight performance drop from 74.7% to 73.3% for 4×50 channels and only a 0.3 percent point drop for stage4 and 2×50 channels. On the other hand, entropy loss does not reduce distributedness sufficiently. The best configuration—stage3 4×50—drops from 74.0% to 36.7%. Thus, the representations of temperature annealed USBs are not distributed across channels.

USBs reduce dimensionality without performance impact. Next, we investigate the impact of temperature annealing more broadly on multiple layer locations as well as for both models: PSPNet and MS-OCRNet in table 5.5. We report dimensions of the bottlenecks, mIoUs after arg max application as well as the number of active channels $\Sigma_{>0}$. The latter derived from an observation that some channels are never active during evaluation (compare columns $\Sigma_{>0}$ – active channels – and *#channels*). This resembles results from recent work on differential architecture search also utilizing softmax [LSY19, XZLL19]. Overall, this encouragingly indicates further dimensionality reduction. We highlight, that at stage4, stage5 and pyramid it appears that around 20 channels are enough to fully retain the performance – irrespective of the model used. Considering segmentation performance, we observe that only stage3 on PSPNet sees a substantial impact (compare the best 73.3% mIoU vs vanilla PSPNet 77.6%). We conjecture that representations at such an early location in the network are difficult to disentangle, yet we also find USB@stage3 for MS-OCRNet achieves a high 77.4% mIoU. Overall, for MS-OCRNet, we see close to full recovery of mIoU performance across all tested locations.

Parallel bottlenecks are not necessary for good performance. To acquire a better understanding of the impact of using parallel bottlenecks on mIoU performance, we conduct a thorough sweep over USB dimensions and report results for PSPNet in figure 5.5. For each network location, we train USBs with varying number of parallel layers N and width K, where $N \in \{1, 2, 3, 4\}, K \in \{2, 5, 10, 20, 50\}$. For each N we plot the mIoU over the product $N \cdot K$, stating total number of channels. To decrease training time, we reduce the batchsize from 16 to 4, which decreases PSPNets baseline performance from 77.6% to 76.2% mIoU. Stage4 and pyramid reach baseline results approaching 50 total channels with no substantial gains after. N has the greatest impact on stage3, which reaches highest mIoUs using $N = 3$ (compare $(N = 1, K = 50) \rightarrow 61.8\%$ vs $(N = 3, K = 20) \rightarrow 68.7\%$). Overall, we find that using 3 parallel bottlenecks with $K = 10$ is able to retain performance while having the least total number of channels (discounting active channels $\Sigma_{>0}$). However, even a single bottleneck – resulting in one-hot outputs – with $K = 50$ is able to reach 75% mIoU.

We conclude that USBs with parallel bottlenecks ($N > 1$) improve performance quickest, yet $N = 1$ only require few channels more and allow true one-hot encoding.



(a) USB@stage3 (b) USB@stage4 (c) USB@pyramid

Figure 5.5: Impact of USB dimensionality trained with temperature annealing on Cityscapes mIoU score for PSPNet. USBs with parallel bottlenecks ($N > 1$) improve performance quickest, yet $N = 1$ only require few channels more and allow true one-hot encoding.

5.3 Quantification of Layer Output Inspectability

We present the *Area Under inspectability Curve* (*AUiC*) metric enabling model agnostic benchmarking by measuring alignments between channels and visual concepts. We specify three criteria that AUiC has to satisfy: (i) it must be a scalar measuring the alignment between visual concepts and channel outputs. 0 must indicate no alignment is possible and 1 perfect overlap. (ii) The metric must be model agnostic to allow comparisons between two different activation functions. (iii) The quantification must be computed unbiased w.r.t. the concept area in the image. The fundamental ideas inspiring our metric are based on the frequently cited NetDissect method [BZK+17]. To highlight the differences to our metric we will end this section with a discussion.

5.3.1 AUiC metric

Our proposed metric consists of 3 main steps, which we schematically present in figure 5.6. Each channel is considered as binary detector by thresholding its output at various values (top center in figure 5.6). For each threshold, the overlap to concept annotations is evaluated via mIoU (bottom center in same figure). Simply speaking, we want to identify the concepts that best describe each channel output. For the final scalar AUiC metric, we take only the best matching channel-concept pair into account and count how many channels in total have a mIoU over a threshold ξ (right in same figure). In the following, we will describe this process more formally.

Channel-Concept matching: First, each channel needs to be identified as a detector for a single concept. Given dataset $\mathbf{X}$ containing annotations for concept set C, we compare channel activations A_k and pixel annotations L_c, where $c \in C$. We consider binarized channel outputs M_k via thresholding with θ_k, such that $M_k \equiv M(k, \theta_k) = A_k > \theta_k$ (see thresholded activation maps in figure 5.6). Channel output M_k and concept annotation L_c are compared with IoU: $\text{IoU}(\mathbf{x}) = \frac{|M_k \cap L_c|}{|M_k \cup L_c|}$, $|\cdot|$ denoting cardinality of a set. A few things need to be considered w.r.t. to our criteria. The metric must be unbiased w.r.t. size. IoU penalizes small areas more than large ones, since small annotations are disproportionally more susceptible to noise. Consider an annotation of 2 pixels and one false plus one true positive. The IoU scores 1/3, pulling down the average over all samples. This would become an issue later on when we optimize θ. Here, a bias would lead to wrong identifications. We address this issue using the *mean* IoU of positive *and negative* responses to balance the label area by its complement:

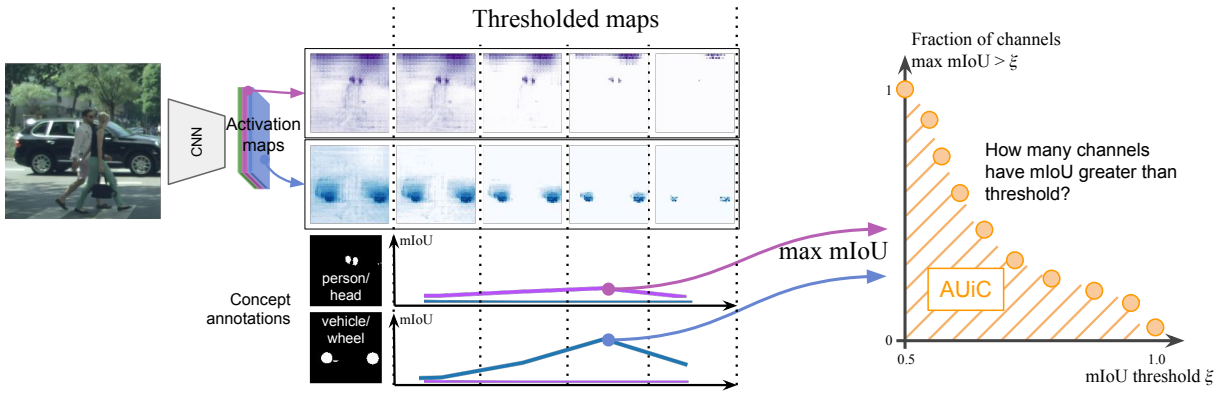


Figure 5.6: Schematic calculation of inspectability score AUiC. First, each channel is considered as binary detector – by thresholding its output at various values – and compared with concept annotations (see blue and magenta channel). The best mIoU scores are used to construct the AUiC score. AUiC is derived by counting how many channels have a mIoU greater than a threshold (see plot to the right)

$\text{mIoU}(\mathbf{x}) = \frac{1}{2}\left(\frac{|M_k \cap L_c|}{|M_k \cup L_c|} + \frac{|\overline{M_k} \cap \overline{L_c}|}{|\overline{M_k} \cup \overline{L_c}|}\right)$. $\overline{M_k}$ and $\overline{L_c}$ are the complements to the binary activation mask and annotation respectively. The alignment score between channel and concept is subsequently defined over the whole dataset $\mathbf{X}$:

$$\text{mIoU}_{k,c}(\mathbf{X}) = \frac{1}{2}\frac{\Sigma_{\mathbf{x}\in\mathbf{X}}|M_k \cap L_c|}{\Sigma_{\mathbf{x}\in\mathbf{X}}|M_k \cup L_c|} + \frac{1}{2}\frac{\Sigma_{\mathbf{x}\in\mathbf{X}}|\overline{M}_k \cap \overline{L}_c|}{\Sigma_{\mathbf{x}\in\mathbf{X}}|\overline{M}_k \cup \overline{L}_c|}. \tag{5.22}$$

We sum over all samples before computing the fraction to include samples not containing concept c.
So far, the choice of θ_k has been undefined. Yet, the alignment between channel and concept is sensitive to θ_k. We keep the determination of θ_k agnostic to the activation distribution by finding critical point $\theta^*_{k,c}$—now per channel and concept— maximizing $\text{mIoU}_{k,c}(\mathbf{X}, \theta_{k,c})$—now parameterized with the threshold:

$$\theta^*_{k,c} = \arg\max_{\theta_{k,c}} \text{mIoU}_{k,c}(\mathbf{X}, \theta_{k,c}). \tag{5.23}$$

An example considering two different concepts (*person/head* and *car/wheel*) is presented in figure 5.6. For each concept, we have identified the threshold $\theta^*_{k,c}$ that maximizes alignment. To assign each channel a single concept, we choose the concept c^* that maximizes mIoU.

$$c^* = \arg\max_{c} \text{mIoU}_{k,c}(\mathbf{X}, \theta^*_{k,c}). \tag{5.24}$$

Each concept can be assigned to multiple channels, but not vise versa.

Scalar quantity. The second step involves summarizing the identifiability to a scalar value – 0 indicating no channel can be identified and 1 all. This is depicted in figure 5.6 to the right. Given a global mIoU threshold ξ we can determine the fraction of channels having a greater mIoU. In order to keep the metric agnostic to the choice of ξ, we define this scalar as the AUC under the indicator function – counting identifiable channels – for all $\xi \in [0.5, 1]$:

$$\text{AUiC} = \frac{2}{K}\int_{0.5}^{1}\sum_{k=1}^{K}\mathbb{1}_{\mathbb{x}\geq\xi}\left(\text{mIoU}_{k,c^*}\right)\, d\xi. \tag{5.25}$$

We coin this scalar AUiC – *Area Under inspectability-Curve*. The factor 2 normalizes the output, such that AUiC ranges from 0 to 1.

Stability w.r.t. $\theta^*_{k,c}$. Since we still choose a single $\theta^*_{k,c}$ to compute our metric, we introduce a second scalar quantity measuring stability when varying $\theta_{k,c}$. For a channel k we retain the selected c^* and marginalize θ out of $\text{mIoU}_{k,c}$. This results in the area under the mIoU curve when varying the threshold:

$$\Delta_{k,c^*} = \int \text{mIoU}_{k,c^*}(\mathbf{X}, \theta)\ d\theta. \tag{5.26}$$

The ideal inspectable channel consistently responds with perfect overlap *only* to concept c^*. In that case Δ_{k,c^*} will be equal to $\text{mIoU}_{k,c^*}(\mathbf{X}, \theta^*_{k,c^*})$ implying maximal stability. In the general case though, a channel may also respond to other concepts but with smaller activations. This results in an inequality $\Delta_{k,c^*} < \text{mIoU}_{k,c^*}(\mathbf{X}, \theta^*_{k,c^*})$, indicating lower stability. Subsequently, the quotient between these two terms enables the quantification of stability. We define this quantity $\mathcal{S}$ aggregating all channels as the fraction between AUC under Δ and AUiC:

$$\mathcal{S} = \frac{1}{\text{AUiC}} \frac{2}{K} \int_{0.5}^{1} \sum_{k=1}^{K} \mathbb{1}_{\mathbb{x} \geq \xi} \left(\Delta_{k,c^*}\right)\ d\xi. \tag{5.27}$$

5.3.2 Discussion

We conclude by showing that AUiC satisfies our three criteria and contrast it to the NetDissect-measure.

Clear definition in $[0, 1]$. 0 must indicate no channel alignment – 1 perfect alignment for all channels. AUiC satisfies this criterion as it integrates over all mIoU thresholds. NetDissect instead chooses a specific IoU threshold $\xi = 0.04$ that results in fuzzy scores at the bounds. At 1, NetDissects measure gives a false sense of security since all channels only require to pass an IoU of 0.04.

Agnostic to network architecture. To enable comparison across diverse types of models, we require a metric agnostic to the distribution of the channel outputs. Our AUiC metric satisfies this criterion since it considers the threshold $\theta^*_{k,c}$ that maximizes mIoU. In NetDissects' measure in contrast, the activation binarization threshold θ_k is chosen based on the top quantile level of activations $a_k \in A_k$ such that $P(a_k > \theta_k) = 0.005$. This 0.005 quantile choice is sensitive to the underlying activation distribution, failing for non-Gaussian distributions – e.g. bi-modals and Bernoulli (USBs have 0/1 outputs), for which θ_k could wrongly be set to 1. This results in M_k being always 0. We observe that NetDissect requires further calibration when evaluating our USBs. For example, only 4 out of 25 channels are assigned for PSPNet-USB@stage4 with default settings while ours assigns 20. Additionally, we observe improved assignment results on SSBs. While the outputs here are normal distributed, the optimization procedure to find the best thresholds is able to improve assignments especially for small concept sizes. We show an example in figure 5.7. We choose MS-OCRNet-SSB@stage5 for which all 70 channels are concept supervised and plot 4 different channels with their respective activation maps. The concept assignment is performed by comparing with 40 concepts from our Cityscapes-Parts dataset (introduced in the next section). The first column shows result for channel 9 which represents *Person/arm*. Note that our thresholding results in activation maps that are much closer to the target concept than NetDissects method. Note that many concepts used for supervision (e.g. *eye*, *ear*, *back*) are not present in Cityscapes-Parts. Nonetheless, our algorithm is able to find the closest matches reliably.

Table 5.6: Channel identification comparison for PSPNet-SSB@pyramid using either IoU or mIoU. The latter reducing size bias substantially, enabling accurate identifications.

channel	trained concept	IoU assignment	Our mIoU assignment
16	person/hair	torso (0.07) painted (0.06)	person (0.57) hair (0.55)
32	lamp/shade	painted (0.07) brown (0.06)	shade (0.58) lamp (0.53)
18	person/foot	torso (0.07) black (0.05)	person (0.53) foot (0.53)
69	wood material	brown (0.07) painted (0.07)	wood (0.53) floor (0.52)

Figure 5.7: Comparison of concept assignments between NetDissects method (lower row) and our calibration-free method (upper row) using Cityscapes-Parts. This comparison is performed on the MSOCRNet-SSB@stage5 for which each channel is concept-supervised. The groundtruth concept is stated above each column. Note that our thresholding (heatmap overlay) is better aligned with the actual concept since its threshold determination is an optimization procedure. Both methods have been evaluated on 100 Cityscapes-Parts images.

Insensitivity to size of concept. To show size bias using IoU, we conduct a comparison between IoU and mIoU. We compare concept assignments on PSPNet-SSB@pyramid since the channels are supervised and hence pre-assigned. Table 5.6 presents the assignments of both methods (columns) for 4 channels (rows). mIoU assignments are consistent with the trained concepts, even identifying concept *wood*. Using IoU instead, concepts like *painted*, *black* or *brown* are among the identified. These concepts cover large areas in Broden images making them less susceptible to noise. The average number of pixels per image of *painted* for example is 1087.5, resulting in an IoU of 0.06, while hair has only 93.8 pixels on average and does not show up when using IoU. mIoU on the other hand computes a score for hair of 0.55 for channel 16, which is trained for *hair*. NetDissects metric also utilizes IoU, for which the authors manually adjusted the threshold to make it unbiased [ZBOT18]. Since this adjustment is done for normal distributions, it's not guaranteed to be unbiased for others.

5.4 Results

Section 3 showed drastically reduced channel numbers while retaining performance—achieving goal (i) and (iii). To assess the semantic alignment of channels (goal (ii)) we utilize our AUiC

metric to show improved inspectability and highlight qualitative examples in this section.

5.4.1 Setup

Datasets. We compare alignments with 3 different datasets: *Cityscapes-Classes*, Broden [BZK+17] and a new dataset: *Cityscapes-Parts*. The broadest dataset we evaluate is Broden which covers 729 various concepts of objects, parts, materials, textures and colors (skipping scene concepts).

Cityscapes-Parts. Since the Broden images are mostly out of domain w.r.t. Cityscapes, we introduce *Cityscapes-Parts* which includes annotation of subordinate concepts to the 19 Cityscapes classes. The new dataset includes 100 densely annotated images from the Cityscapes dataset covering 40 different concepts that we chose based on the Cityscapes classes. Vehicles including bicycles have been decomposed into wheel, head- and taillight, door, seat, steering control and body. Buildings have not only been decomposed into their direct subordinate concepts like wall and window but also into concepts that often obstruct buildings like advertisements. We also distinguish between plain and cobble road as well as their markings. Three example images are presented in figure 5.9 and concept statistics in figure 5.8. We annotated all images with the labelme toolbox [Wad16]. We made all annotations publicly available here: https://doi.org/10.17617/3.B1IZS6.

Compared models. From all USB models trained, we select one per network location that strikes the best balance between mIoU and AUiC: namely retaining performance while reducing number of channels. For PSPNet: stage3 - USB2x50, stage4 - USB1x50, pyramid - USB2x10. For MS-OCRNet: stage3 - USB2x50, stage4 - USB1x50, stage4 - USB2x10.

5.4.2 Quantitative inspectability improvements with SBs

We compare vanilla PSPNet/MS-OCRNet with SSBs and USBs and do so for outputs of an early layer: stage3, a middle layer: stage4 and a late layer: pyramid/stage4. AUiCs are collectively shown per layer in three columns in figure 5.10. Each row shows results for a different dataset indicated by the column headers on the left: Cityscapes-Classes, Cityscapes-Parts and Broden. Vanilla network outputs are indicated by color *red*, SSBs by *yellow* and USBs by *blue*. PSPNet and its SB integrations are indicated by circles, MS-OCRNet by triangles.

SSBs enable inspection for subordinate concepts. On each layer and dataset, SSBs outperform the vanilla baselines. Based on the AUiC improvements on the Broden dataset on all layers—on which the SSBs outperform all tested variants—we highlight that it is possible to align intermediate representations with user defined concepts. SSBs improve the AUiCs on Broden from under 0.05 up to 0.2 for the late layers making a big leap forward towards inspectable representations. In comparison to the application of unsupervised USBs though—as we will discuss shortly—large AUiC improvements are challenging to establish as they heavily depend on the choice of concepts. A priori, it is difficult to know which concepts can be aligned well with supervision whilst also allowing for good segmentation performance.

USBs align with Cityscapes-Classes. In comparison to SSBs, USBs align much better with Cityscapes-Classes than subordinate concepts. We see here the greatest increase in AUiC, e.g. from 0.05 to close to 0.6 on stage4 and 0.1 to 0.6 at stage5/pyramid. Clearly, it is much more effective to use unsupervised semantic bottlenecks to improve inspectability, than to force specific concepts onto the network.

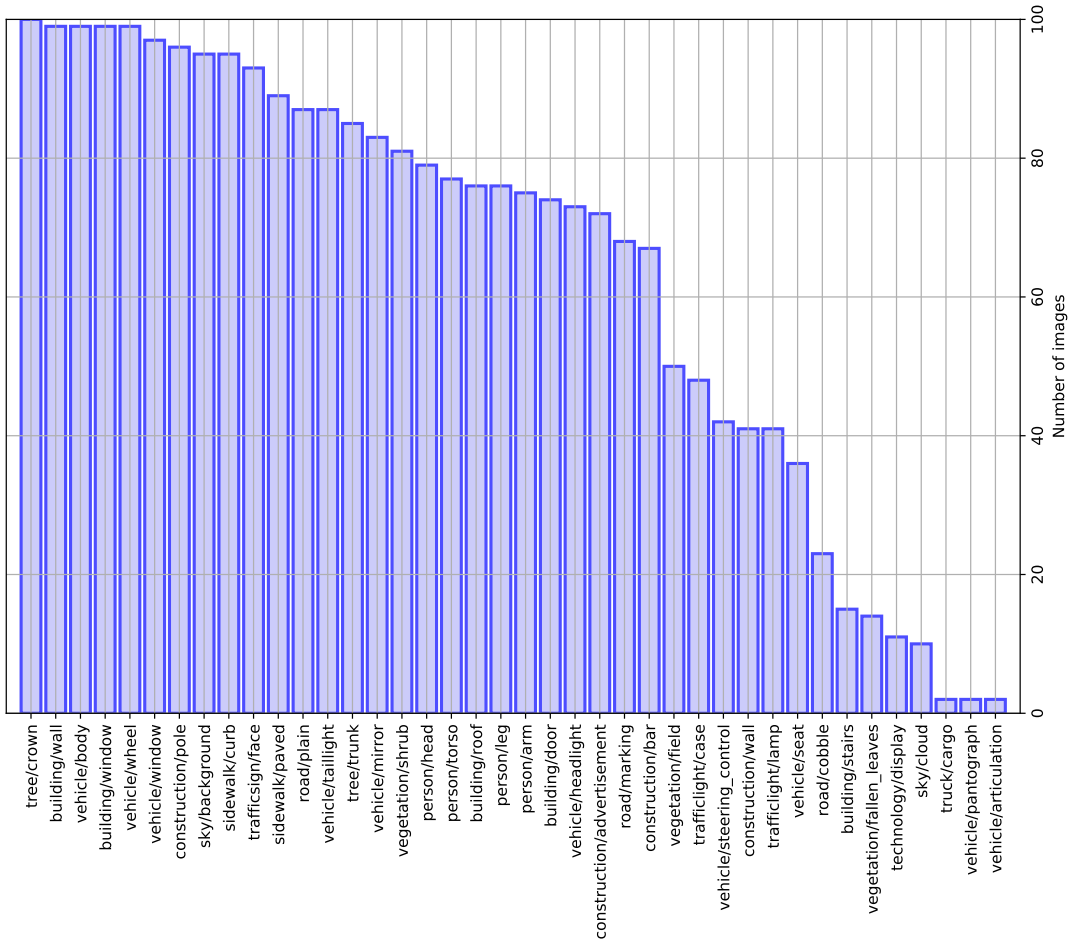


Figure 5.8: Number of images containing each concept in Cityscapes-Parts. 37 out of 40 concepts are present in more than 10 out of 100 images.

Figure 5.9: Examples from our Cityscapes-Parts annotations with 40 different labels focused on parts of Cityscapes objects. 100 randomly selected images from the Cityscapes-dataset have been annotated (excluding test-sets).

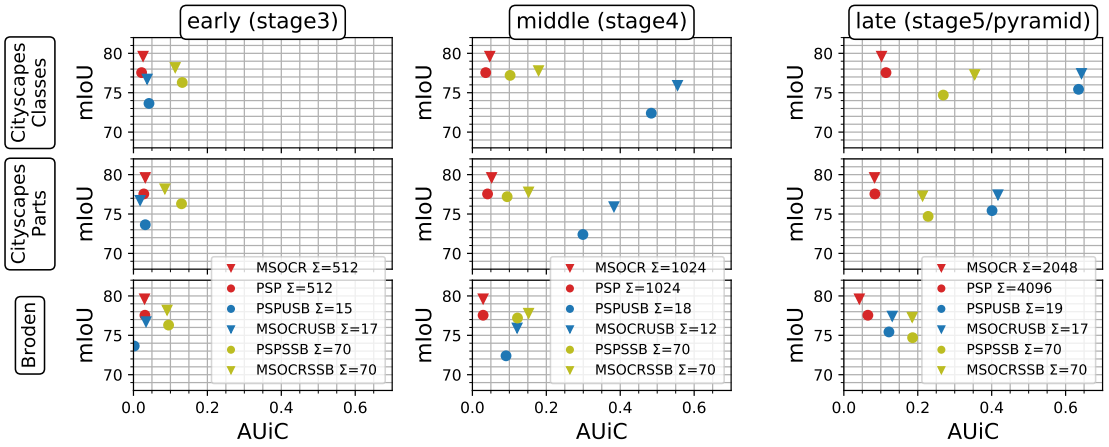


Figure 5.10: AUiC - inspectability scores for SSBs (yellow), USBs (blue) and baselines (red). Higher values are better, 1 being perfect channel-concept alignment. SBs substantially improve that alignment and thus: inspectability. Σ indicates number of active channels.

Table 5.7: Averaged stabilities over datasets.

Stability $\mathcal{S}$	stage3	stage4	pyramid
original	0.034	0.077	0.099
bottleneck	0.004	0.069	0.082
SSB	0.047	0.099	0.099
USB	0.938	0.945	0.947

USBs offer high stability. Shown in table 5.7, we see USBs offering a clear advantage since their outputs are one-hot encoded: alignments are very stable. SSBs on the other hand report only slight stability improvements over baselines. To answer, whether softmax enables greater stability by default (SSBs have no non-linearity), we measure AUiC and $\mathcal{S}$ for SSB with softmax. Measuring with softmax $T = 1$, we find a 2-fold increase of stability to 0.20 but a 3-fold decrease in AUiC to 0.07. While softmax alone increases stability, it does not improve AUiC by default. As noted in section 4, a channel is stable if it responds consistently to the same concept no matter the activation value (arg max USBs have only two states). This is not the case for a regular stage4 and SSB channel, for which the same channel may be active for multiple concepts albeit with low activation. By our definition, this can be inspectable but is not stable. We conclude that the linear SSB-layer is sufficient to align with semantic concepts yet unable to increase stability by a large margin by default. Note that simple bottlenecks show consistently reduced stability (e.g. 0.069 vs 0.077 for bottleneck vs original on stage4).

Representations at stage3 are difficult to align. Comparing the AUiC scores between stage3 and other locations, it becomes evident that only SSBs improve inspectability. This indicates an intrinsic difficulty in aligning individual channels with semantics that early and could imply a necessity for distributed representations. We leave this as a challenge for future work.

Conclusion. Both SSBs and USBs offer clear advantages over baselines. SSBs are semantically supervised and thus can offer the greatest improvements in AUiC. USBs do not require concept supervision, yet form channels that are well aligned with Cityscapes classes offering a different dimension of inspectability.

5.4.3 Qualitative improvements with SBs

To support our quantitative results we compare visualizations of SB-layers and baselines. We show that SB outputs offer substantially improved spatial coherency and consistency.

Top-20 **channels.** To enable comparison between 1000s and 10s of channels, we utilize the mIoU scoring of our AUiC to rank channels. We show the top-20 channels, assign each a unique color and plot the arg max per location. An inspectable – and thus aligned – channel will result in coherent activations for a unique concept. Visualizations are presented for two images in figure 5.12 for all tested layer locations on MS-OCRNet. 9 additional images are shown in figure 5.11 for stage4 only.

MS-OCRNet outputs in the first row (*Vanilla*) show the difficulty in interpreting them, since they are highly distributed across channels (also indicated by [FV18]). While still highly irregular in its appearance, activations at stage4 and stage5 show increasing regularities that resemble the Cityscapes classes (e.g. building in purple or cars in light gray).

SSB & USB outputs. Attending to the first image on the left half of figure 5.12, we see spatial coherency greatly improved for USB outputs over baseline. SSBs offer only slight improvements on stage4, but show clear distinctions on stage5 into wheels (beige color for car wheels and light blue for bicycles), car windows (purple), person-legs (dark blue) and torso (light gray). In relation, the USBs appear to form representations that are early aligned with the output classes, which is already evident for USB@stage4. Comparing USB@stage4 and USB@stage5 it appears the semantics of the representations do not differ but are only refined. This gives a clue to the inner workings of deep segmentation nets: the stage outputs get closer to the class labels the higher we go in the layer. Since USBs are unsupervised, they offer an easy access into what concepts have been learned automatically.

Individual USB channels. Finally, we show activation maps of individual USB channels for MS-OCRNet-USB@stage5 in figure 5.13. These convey four points: (i) the maps are visually sparse, (ii) they are spatially coherent and (iii) they are consistently active for the same image concepts. Find the top-20 visualizations, as well as 5 of $\Sigma_{>0} = 17$ channels – one per column. We consistently see the each channel activate for the same concept irrespective of the input sample. The left most channel (11th) activates for road, the next to the right (8th) for buildings and the last three for person, rider and bicyle/motorcycle. We highlight an interesting redundancy in these last three: Person and rider are already distinguished at this stage5 level and kept as individual representations (compare the 3rd and 15th channel in row 3). It appears to be more difficult to determine a pixel belong to person or rider when only concept person and bicycle is retained.

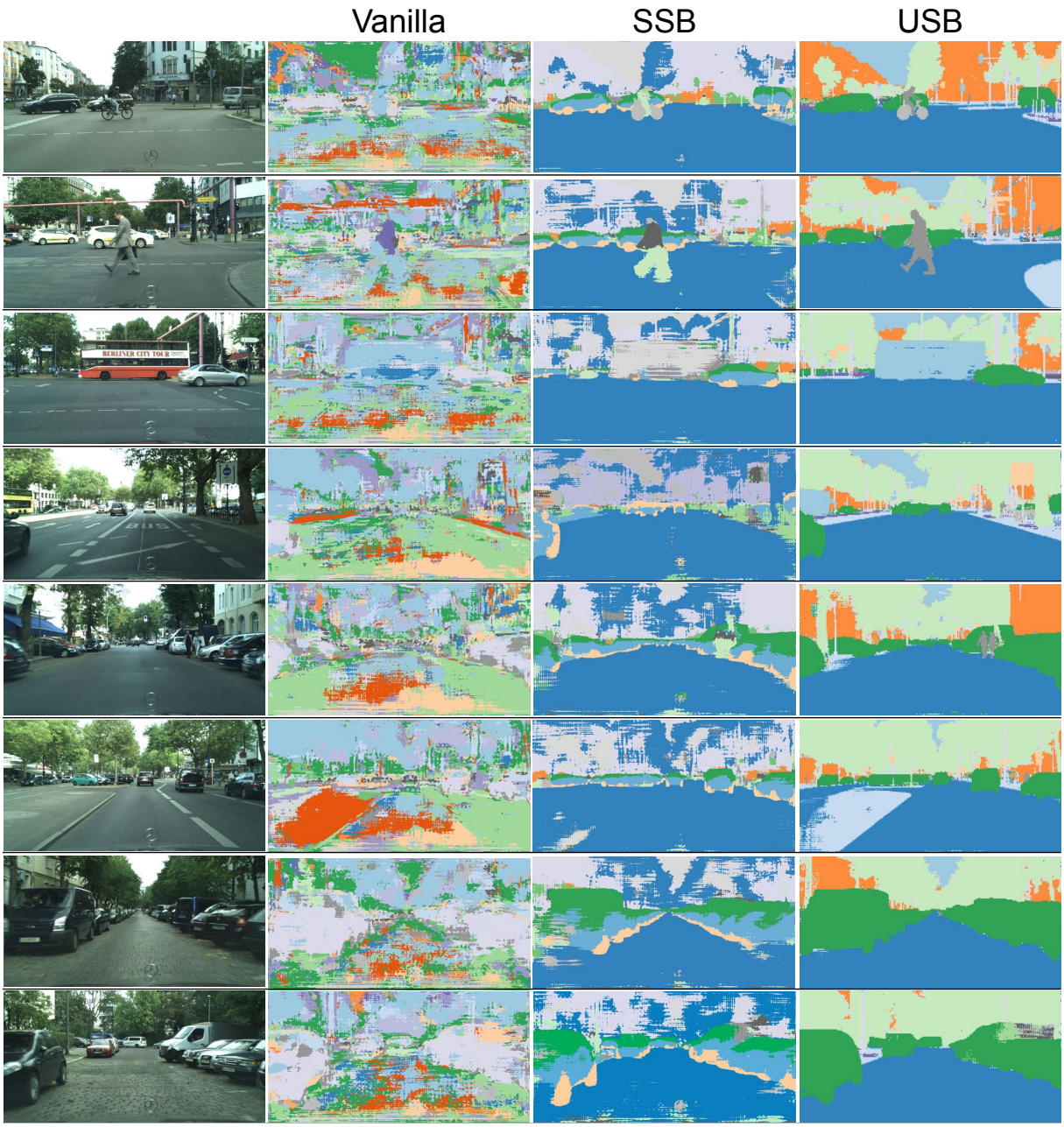


Figure 5.11: Top-20 CS-Parts aligned channels only for stage4 from SSB-, USB- and vanilla MS-OCRNet outputs. USBs and SSBs offer better semantic alignment and are easier to inspect for concept evidence.

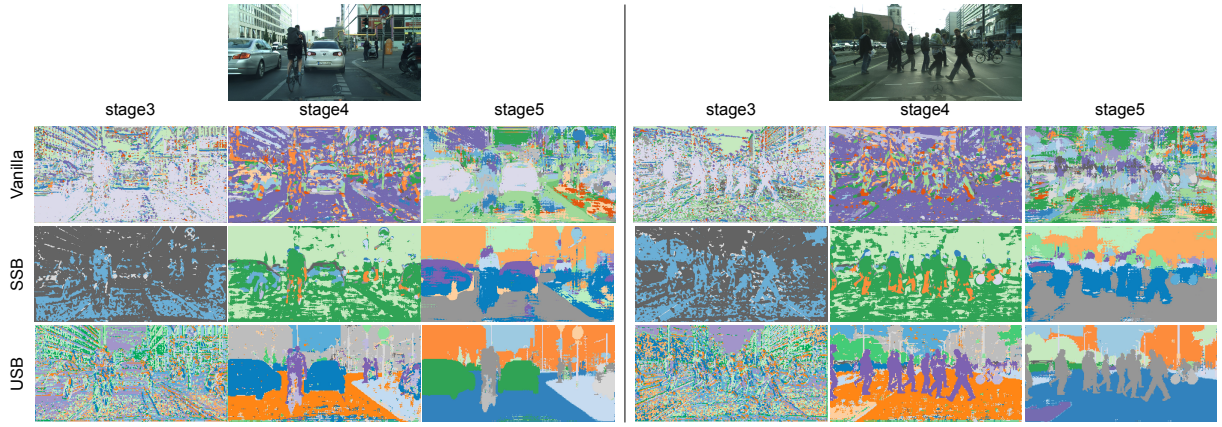


Figure 5.12: Top-20 CS-Parts aligned channels from SSB-, USB- and vanilla MS-OCRNet outputs. Each color is mapped to a single output channel. USBs and SSBs offer better semantic alignment and are easier to inspect for concept evidence.

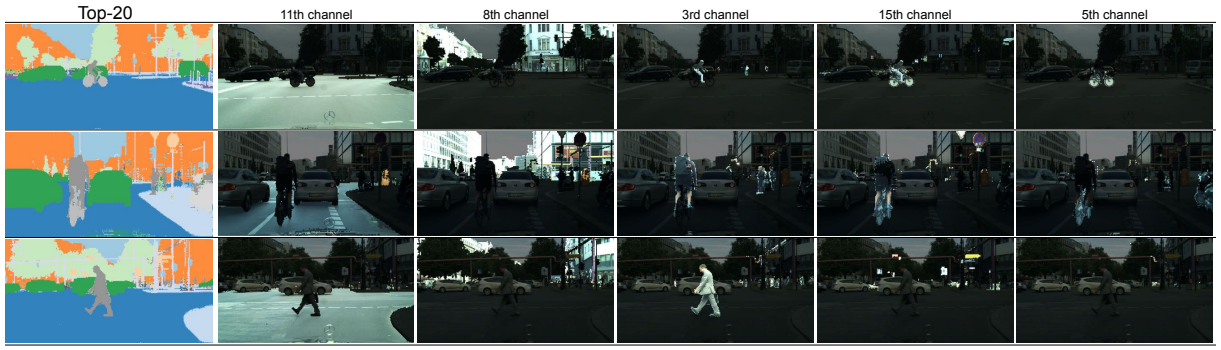


Figure 5.13: Individual channel activations overlaid with the input image of MS-OCRNet-USB@stage5 are coherent and well delineated. Only bright areas are active.

5.5 Conclusion

In light of the increasing number of data driven models in real world applications like autonomous driving, we demand models to be inspectable at intermediate layers. One issue obstructing inspectability in typical deep networks is that representations are distributed across hundreds of channels and are not semantically aligned [BZK+17, FV18, MA20].
Consequently, in this paper, we proposed supervised and unsupervised Semantic Bottlenecks (SSBs and USBs) to reduce number of channels and align each channel with human interpretable semantics while retaining performance. Additionally, we introduced the AUiC metric quantifying the alignment to enable model agnostic benchmarking as well as stability of same alignment. Using our metric, we showed, that SBs improve baseline scores up to six fold while retaining performance. Overall, we identified USBs to offer greatest inspectability since they are not restricted in the choice of concepts and offer great alignment stability given they are one-hot encoded. While SSBs only increase AUiCs over baselines three fold, they enable the alignment to user-defined semantic concepts. SBs are easy to integrate into existing architectures and offer great returns in inspectability, making them a useful extension for models used in real world applications. For future work, we would like to address the following points. First, we want models to have multiple SBs to act as access points to the process pipeline. We conjecture use cases for failure case detection when information along layers indicate conflicting concepts. Secondly, our current methodology involves finetuning the bottleneck as well as all succeeding layers. A combination of USBs with invertible neural networks would spare the need to retrain the target model [ERO20] while maintaining inspectability. Finally, we would like to see our semantic bottlenecks applied for other use cases like pathogen segmentation in the medical domain.

6 Semantic Bottleneck Clustering for Prediction-Error Analysis: A Use-case

Contents

This chapter utilizes SBs, introduced in the previous chapter, to analyze failure cases on the task of street scene segmentation. This can be seen as one of the goals of inspectable representations for improving trust. By providing evidences (or lack-thereof), we desire to get insights into the high level prediction process of deep networks. In this chapter, we specifically look at incorrect predictions and their respective bottleneck evidences. In particular, we cluster these evidences to search for large clusters of similar attributes – indicative of a reoccurring weak spot in the prediction process. Moreover, additionally clustering correct predictions, we show that each failure case can be made explainable by comparing with the closest contrastive example. The latter providing the necessary change in bottleneck evidences to achieve a correct prediction, thereby highlighting which features in the image are under- or overrepresented. We identify three reoccuring error patterns: (i) misinterpretation of image features, (ii) conflicting evidences and (iii) missing evidences.

6.1 Introduction

While deep models have achieved impressive performances on a wide range of vision tasks, they largely remain black boxes that render it difficult to understand failure cases. To lift this black-box property of deep networks and improve the trust in high stake applications like autonomous driving, various inherently interpretable architectures have been proposed, among others [CLT+19, KNT+20, LFS21, BFS22].

By providing attribution to either image features or higher level concepts at intermediate representations, these methods attempt to increase transparency of the models decision. Little focus though has been put on utilizing these methods to analyse failure cases during prediction, thereby providing insights into their utility for interpretability. That is, which prediction errors can be understood and what type of failure cases exist.

In this work, we investigate alluded utility of the recently proposed Semantic Bottlenecks [LFS21] and conduct such a failure case analysis on street scene segmentation – an essential task for autonomous driving. Semantic Bottlenecks (SBs), specifically the supervised

variant (SSBs), reduce the dimensionality of an intermediate layer to only a few channels and align each to a human interpretable concept. As has been shown in [LFS21], this can be integrated into state-of-the-art architectures without impairing segmentation performance. The resulting SSBs provide the opportunity to cluster the concept evidence for all correct and incorrect predictions, thereby providing means to analyze number and types of errors.

Our **contributions** are two-fold: We show that misclassifications can often be ascribed to cases of missing or conflicting evidence and secondly show that directed manipulation of evidence in the SSBs opens the possibility for intervention. That is, we can add or remove semantic evidence to achieve substantial changes in the segmentation.

6.2 Error Analysis on Supervised Semantic Bottlenecks (SSBs)

We conduct our failure case analysis on Cityscapes [COR+16] and utilize Supervised Semantic Bottlenecks (SSBs) [LFS21] that resolve each prediction to a fixed number of high level concepts. The underlying training scheme of SSBs involves two steps. First, a concept predictor is trained on additional part-level annotations. Then, a second predictor for the target task is trained on top. Consequently, concept evidence in the bottleneck can be inspected to investigate reasons for correct or incorrect predictions. In this section, we revisit the construction of SSBs for street scene segmentation and discuss how clustering its activations can be utilized to analyze prediction errors.

6.2.1 Construction of SSBs

Introduced in [LFS21], semantic bottlenecks (SBs) are integrated into state-of-the-art segmentation models by splitting them in two followed by bottleneck insertion. Assume a model with L layers, accepting input x_0:

$$f_{\text{Net}}(x_0) = (f_L \circ f_{L-1} \circ ... \circ f_1)(x_0),$$

and integrate a SB after layer i:

$$f_{\text{SBNet}}(x_0) = (f_L \circ ... \circ f_{i+1} \circ f_r \circ f_{\text{SB}} \circ f_i \circ ... \circ f_1)(x_0).$$

Here, f_{SB} maps to a lower dimensional space, providing concept evidence and f_r consequently maps back to the original space expected as input by f_{i+1}. Formally:

$$f_{\text{SB}} : \mathbb{R}^{N_i \times H \times W} \to \mathbb{R}^{N_{\text{SB}} \times H \times W} \quad \text{and} \quad f_r : \mathbb{R}^{N_{\text{SB}} \times H \times W} \to \mathbb{R}^{N_i \times H \times W},$$

where $N_{\text{SB}} < N_i$ and H and W are spatial dimensions that are retained throughout f_{SB} and f_r.

In [LFS21], f_{SB} and preceding layers are trained on pixel-wise material and part annotations on the Broden dataset [BZK+17], after which they are frozen and all succeeding layers including f_r are trained on Cityscapes. This training strategy and the resulting model is coined Supervised Semantic Bottleneck (SSB).

6.2.2 Case study: Street scene segmentation

We analyze errors on the task of street scene segmentation using Cityscapes. This is a natural choice given that SSBs have been proposed on that same task and street scene segmentation

Table 6.1: 70 concepts from *Broden+* [XLZ+18] are selected to be relevant for the Cityscapes domain, according to [LFS21].

Materials	Brick, Fabric, Foliage, Glass, Metal, Plastic, Rubber, Skin, Stone, Tile, Wood	
Parts	Sky	Cloud
	Building	Window, Door, Roof, Shop, Wall
	Person	Leg, Head, Torso, Arm Eye, Ear, Nose, Hand, Hair, Mouth, Foot, Eyebrow, Back
	Road	Crosswalk
	Car	Window, Door, Wheel, Headlight, Mirror, Roof, Taillight, Windshield, Bumper
	Van	Window, Door, Wheel, Headlight, Taillight, Windshield
	Truck	Wheel, Windshield
	Bus	Window, Door, Wheel, Headlight, Mirror
	Train	Head, Headlight, Headroof, Coachroof
	Lamp	Arm, Shade, Bulb
	Bicycle	Wheel, Handle, Saddle, Chain
	Motorcycle	Wheel, Headlight, Handle

has high relevance for autonomous driving. Equally to [LFS21], we train and evaluate SSBs with $N_{\text{SB}} = 70$ concept dimensions, of which 59 are parts and 11 are materials. These concepts are source from the *Broden+* [XLZ+18] dataset, a collection of multiple datasets containing pixel-level annotations for hundreds of various concepts relevant to scene understanding. As noted by [LFS21], most of the offered concepts (377 parts and materials) are not relevant for the Cityscapes classes. Consequently, the number of concepts is reduced to 70. These are listed in table 6.1. Importantly, we are particularly interested in whether the selected concepts can be utilized to better understand the errors made by the network.

6.2.3 Prediction-error analysis

SSBs provide concept evidences for each predicted pixels. This enables inspection of prediction errors. E.g., given a wrong prediction of a person standing in front of a car, the bottleneck can reveal conflicting or missing evidences. To reveal such errors though, it is typically not sufficient to look at the raw bottleneck output. All bottleneck channels have their own bias and are potentially unrelated to each other. Thus we desire to compare outputs to a true-positive reference. Here, we resort to SB activation clustering as a means to achieve two goals: (i) we get insights into the *type* of errors and (ii) we can compare to true positive cluster centroids as reference.

The error data. As data, we use the bottleneck activations of SSB@pyramid placed at the penultimate layer of PSPNet – the backbone model. For each pixel predicted, the receptive field on the bottleneck constitutes a 3×3 patch. We average these spatial dimensions for simplicity and thus acquire a single 70-dimensional vector for each prediction. We source true-positives, true-negatives, false-positives and false-negatives from hold-out Cityscapes data: its validation and *coarse* training set. This totals $500 + 20,000$ images. We filter single pixel errors and errors on borders by applying dilation and erosion to both prediction and groundtruth masks. These errors often derive from incorrect or inconsistent groundtruth.

Agglomerative clustering is used to cluster the bottleneck activations [Jr.63]. This clustering approach does not require prior knowledge about number of clusters and hence is particularly suitable for our error analysis. It first builds a tree bottom-up, i.e. create leaf nodes for all samples, and successively joins the two most similar samples to the same parent node. Hereby, the similarity is decided by a linkage criterion. In our case, we desire clusters to consist of the

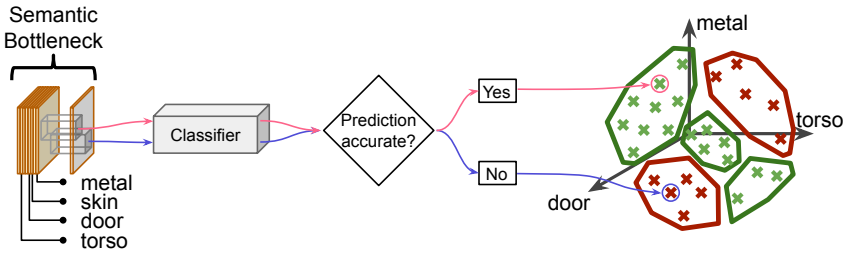


Figure 6.1: For each pixel-prediction on Cityscapes, we extract all activations of the Semantic Bottleneck (SB) within the receptive field. The spatial dimensions are averaged such that we acquire a concept activation vector. These samples are then clustered, but separated by type. That is, we ensure clusters consist purely (at least nearly) of accurate or inaccurate predictions.

same outcome type (of which we consider 4: true-positives, true-negatives, false-positives and false-negatives). This enables the inspection of errors and a direct comparison to the closest cluster of accurate predictions.

6.2.4 Implementation details

Bottleneck training We follow the description in [LFS21] and integrate a linear SSB into a pretrained PSPNet [ZSQ+17]. Concept training is performed on the 70 concepts selected from the *Broden+* dataset [XLZ+18]. We train and evaluate a SSB@pyramid with small adjustments to training hyperparameters due to hardware constraints: we use smaller input cropsize of 651×651 instead of 713×713 and use a smaller batchsize of 4 instead of 16. This results in a slight loss of performance (see results in table 6.2, e.g. SSB@pyramid achieves a mIoU of 72.8% on Cityscapes: a drop of 1.9 percent points over the original results reported in [LFS21].

Table 6.2: Segmentation results on Cityscapes validation set for different placements of SSBs. In contrast to [LFS21], we train with a smaller batchsize of 4, achieving slightly lower performances.

configuration	batchsize	#concepts (materials, parts)	mIoU
PSPNet	16	N/A	77.6
PSPNet	4	N/A	76.2
SSB@stage4 (1024 input feat.)	4	70 (11, 59)	76.2
SSB@pyramid (4096 input feat.)	4	70 (11, 59)	72.8

Agglomerative clustering We use the ward linkage [Jr.63] to decide on the linkage between two nodes. Empirically, we found this to produce the most coherent clusters in conjunction with specifying an additional criterion that satisfies high outcome type purity (4 outcome types: true-positives, true-negatives, false-positives and false-negatives) and a maximum cluster size of 150 (pixel samples). For our analysis, we inspect errors made with the SSB@pyramid configuration. This bottleneck placement is late in the PSPNet architecture and simplifies the clustering as the receptive field of the classifier on the SSB is only 3×3. To further reduce the dimensionality of $70 \times 3 \times 3$, we average over the spatial dimensions to acquire a dimensionality of 70.

6.3 Evaluation

Before beginning with the failure case analysis, we conduct a bottleneck intervention study to ensure class predictions are based on related bottleneck concepts. E.g., class *car* must be sensitive to *car*-related concepts and insensitive to *person*-related concepts. We see this as a necessary requirement for the subsequent failure case inspection.

6.3.1 Bottleneck intervention as sanity check

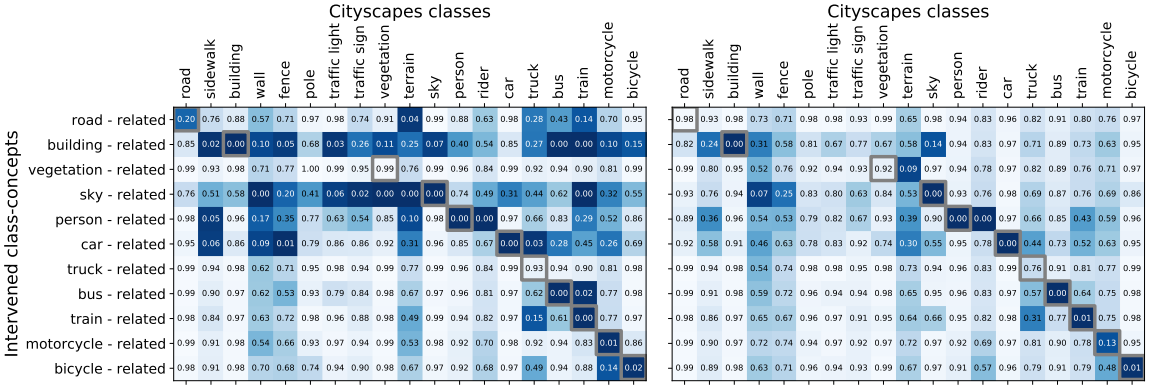
Table 6.3: For 11 out of 19 Cityscapes classes, we relate *Broden+* concepts used during SSB training. These concepts are used for the intervention experiments in section 6.3.1.

Class	Related *Broden+* concepts (cf. table 6.1)
Road	Road/Crosswalk, Material/Stone
Building	Building/{Window, Door, Roof, Shop, Wall}, Material/{Brick, Glass}
Vegetation	Material/Foliage
Sky	Sky/Cloud
Person	Person/{Leg, Head, Torso, Arm, Eye, Ear, Nose, Hand, Hair, Mouth, Foot, Eyebrow, Back}, Material/{Fabric, Skin}
Car	Car/{Window, Door, Wheel, Headlight, Mirror, Roof, Taillight, Windshield, Bumper}
Truck	Truck/{Wheel, Windshield}
Bus	Bus/{Window, Door, Wheel, Headlight, Mirror}
Train	Train/{Head, Headlight, Headroof, Coachroof}
Motorcycle	Motorcycle/{Wheel, Headlight, Handle}
Bicycle	Bicycle/{Wheel, Handle, Saddle, Chain}

Previously used in a similar setting [KNT+20], intervening bottleneck concepts allows to alter the prediction during test time without requiring complex image manipulation operations [SSF19]. We utilize this feature of SSBs to investigate whether the part of the model from bottleneck to prediction (SSB $\rightarrow y$) has learned to rely its predictions on the 'right' concepts. E.g., we want to assert all predictions for class *car* are based on the 9 *car*-related concepts in the SSB (see table 6.3). This is a necessary requirement to render semantic bottlenecks inspectable. Alternatively, the prediction may be dependent on potentially unrelated concepts. Such dependencies would require additional analysis, partially defeating the purpose of designing bottlenecks with a fixed set of concepts. In the following, we perform an ablation study to show that SSBs are inspectable for many classes at a late layer.

Method To ablate concept evidences, we set the activation value to the 5th percentile, determined on the training set (similar to [KNT+20]). We evaluate mIoU on the validation set and test for ablation of concepts for *road*, *building*, *vegetation*, *sky*, *person*, *car*, *truck*, *bus*, *train*, *motorcycle* and *bicycle*. Note that we cannot associate all Cityscapes classes to the 70 concepts provided in the SSB (see table 6.3). We leave these classes to future work. If the network has learned a meaningful mapping between SSB and classes, all or most pixels of the targeted class will be misclassified. Ideally, no additional class is impacted.

Quantitative results For two SSB placements: SSB@stage4 and SSB@pyramid, we present quantitative results in confusion matrices in figure 6.2 and qualitative results in figure 6.3. Quantitatively, in figure 6.2, we observe a near 1-to-1 mapping between concepts and classes for SSB@pyramid. That is, *person*-related concepts have a direct influence on classes *person* and *rider*, while the impact on other classes is relatively low. Some concepts do not exhibit the expected dependencies though. *road*-related concepts (of which there are only two: *crosswalk* and *stone*), have nearly no influence on class *road* (0.98 of mIoU is retained). Similarly, *truck*-related concepts have little impact on class *truck* (0.76 of mIoU is retained). For SSB@stage4,



(a) SSB@stage4 (b) SSB@pyramid

Figure 6.2: Utilizing bottleneck interventions, we show that late SSBs (SSB@pyramid) have the desired 1-1 mapping between concepts (e.g. *car*-related) and class (here: *car*). During test-time, we intervene bottleneck concepts, based on table 6.1, by setting them to a low value (5th percentile) and evaluate the relative drop in mIoU per class. White cells (1.0) designate no mIoU change and dark blue (0.0) designate a maximum drop. Ideally, we have a clear 1-to-1 mapping between concepts and class and observe only a single dark blue cell per row. This desired characteristic is observed only at a late layer like *pyramid*, while some additional dependencies exist. On *stage4*, on the other hand, we see a 1-to-many mapping, where the intervention of (e.g. *car*-related) concepts has a strong influence on many classes (e.g. *sidewalk*).

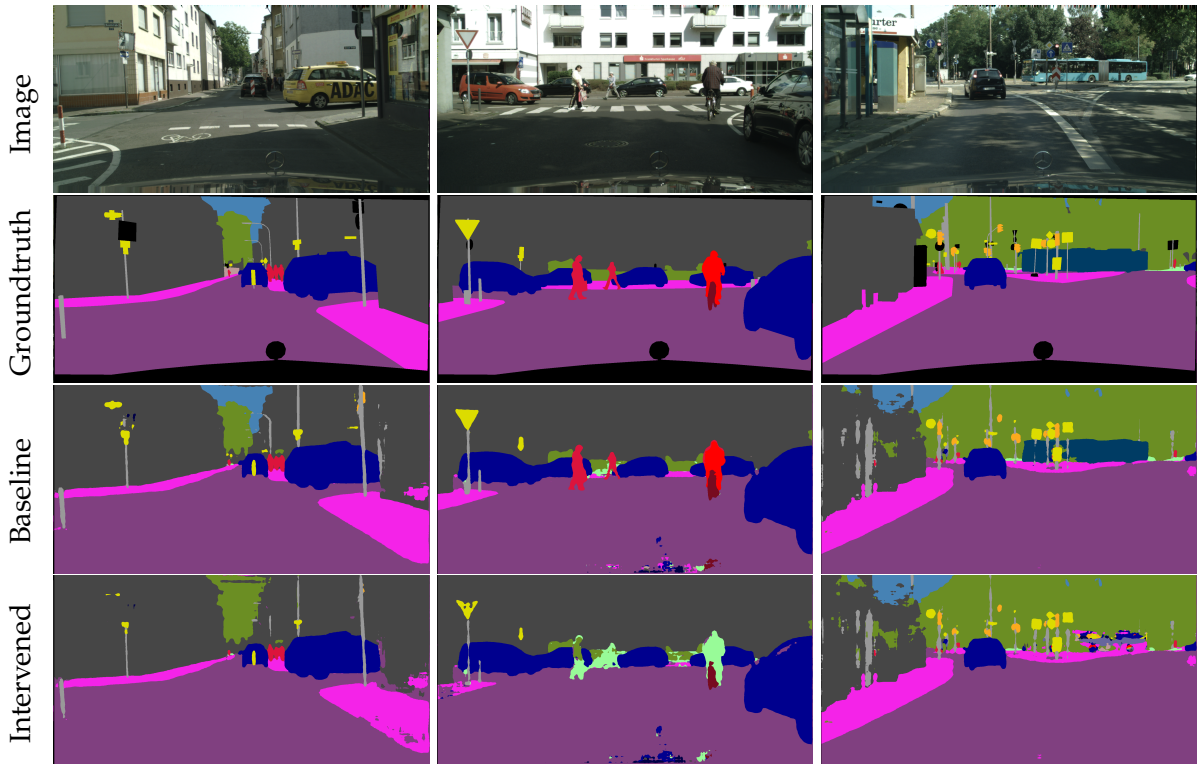
we observe multiple 1-to-many dependencies, especially for *building*-related and *sky*-related concepts. These have substantial impact on multiple classes like *wall*, *vegetation*, *sky*, *train* and more. In contrast to SSB@pyramid, we see a stronger dependency between class *road* and its related concepts (0.2 of mIoU is retained), although terrain and train are influenced as well (0.04 and 0.14 retained mIoU respectively).

Qualitative results Qualitatively, we show three different Cityscapes validation images in figure 6.3 and how concept intervention on SSB@pyramid results in the expected change in segmentation. As before, we target classes like *sky* (left), *person* (center) and *bus* (right) by intervening their related concepts in table 6.1. The first row shows the input image, the second row its groundtruth and the last two rows present non-intervened and intervened segmentations. For all cases, the intervention has negligible effect on unrelated classes, yet results in misclassification of the targeted class.

Conclusion Overall, while the dependencies are not perfectly resolved on SSB@pyramid, many classes can be considered inspectable. Class *person*, *car*, *bus*, *train*, *motorcycle* and *bicyle* have a strong mapping to their respective concepts. We consequently focus our main error-analysis on these classes. Future work should consider whitening of concepts, as in [CBR20].

6.3.2 Error analysis

In the following, we investigate the utility of SSBs to analyze and understand the errors made on the task of street scene segmentation. To this end, we apply agglomerative clustering to all bottleneck activations per class ensuring each cluster consisting of mostly of one outcome type: true-positive (TP), true-negative (TN), false-positive (FP) or false-negative (FN). This



(a) *sky*-related concepts (b) *person*-related concepts (c) *bus*-related concepts

Figure 6.3: Qualitative results for intervened SSB@pyramid on three different Cityscapes validation images. (a) shows segmentation for intervened *sky*-related concepts (note the flip from class *sky* to *vegetation*. (b) intervenes *person*-related concepts and (c) *bus*-related concepts. In all cases, the intervention of concepts has negligible effect on unrelated classes, yet results in misclassifcation of the related class.

Table 6.4: Cluster statistics for three Cityscapes classes. The samples are sourced from the Cityscapes validation and *coarse* training set. By removing single pixel and border errors, we lose a small fraction of samples (e.g. down to 0.969 for class person). Depending on the class, the number of clusters can be large (e.g. 2833 for car) while the number of images – from which samples are extracted from – remains close to 1.

	person	car	bike
Total number of samples	236,346	369,136	215,640
Number of error-samples	36,326	169,116	39,240
#error-clusters of size > 2	1105	2833	855
accounting for fraction of samples	0.969	0.998	0.979
Avg #samples per cluster	31.9	59.1	45.1
Avg. #images per cluster	1.3	1.4	1.3

clustering enables the analysis of two properties: (i) the type of class-errors made and (ii) their comparison to counterfactuals. The latter provided by the closest TP-cluster means, showing the necessary bottleneck changes to obtain a correct prediction. We present a selection of such errors made by PSPNet with integrated SSB@pyramid and discuss (i) and (ii).

Cluster statistics First, we give an overview over the number of samples considered per class and the resulting error-clusters found (number limited to FPs and FNs). In table 6.4, we present statistics for three of the classes found to be highly dependent on a fixed number of concepts (see previous subsection). These classes are: *person*, *car* and *bike*. Note that the number of error clusters is large (855 for class *bike* and 2833 for class *car*), each consisting of roughly 30-50 samples. We also observe the average number of images per error cluster to be close to 1. This indicates a fairly large variation of activations among clusters. Moving forward, we focus on the clusters, that are large and consist of multiple images. These clusters can be considered representative for typical errors made by the network, while small clusters typical represent highly specific errors.

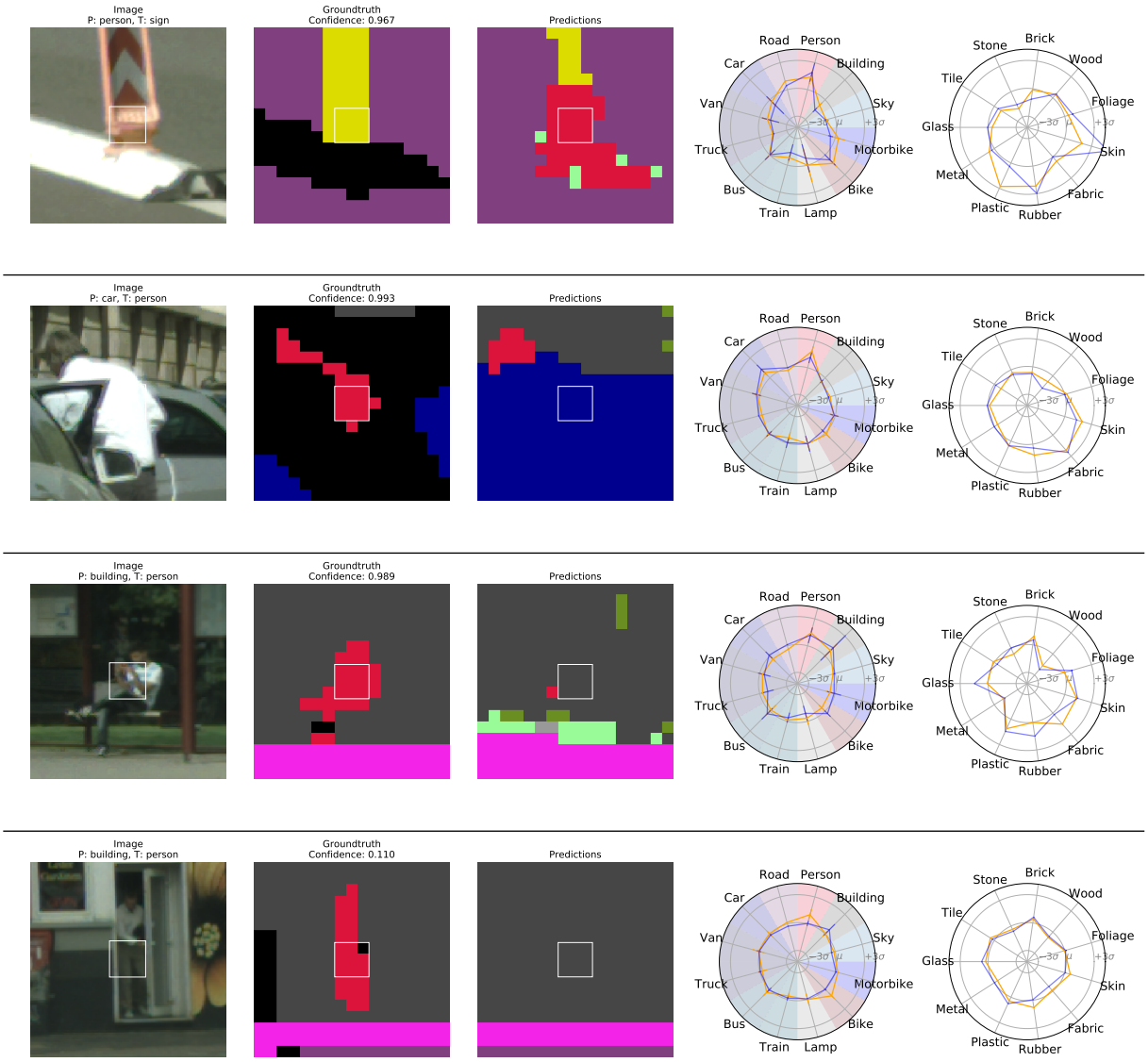
Types of errors A selection of such errors are presented in figure 6.4, organized columnwise into image-region-of-interest, groundtruth and predicted segmentation as well as the 70 concept activations. The latter being split into parts and materials and visualized as spider plots. To simplify, we group the 59 part-concepts by their corresponding object category (as indicated in table 6.1). Per part-group, we report mean concept activation and standard deviation. By standardizing the SSB-activations, on all samples from the Cityscapes validation set, to zero mean and unit variance we gain insight into the relative amplitude of each evidence.

We identify three patterns that result in misclassification:
Misinterpretation of image features. The first row of figure 6.4 originates from a cluster with 41 samples and are all related to the misclassification of construction beacons (target class: *sign*) as *persons* with confidence of over 90% (based on the softmax probabilitiy differences of the top 2 logits). The SB activations give insights into why that might be the case. In particular, the material concept *skin* is with over 4 times the standard deviation highly activated, potentially related to the color of the beacon. We also observe that the part concept *foot* (shown in figure 6.5, top) is typically highly activated for these cases. An additional systematic error could be traced back to the choice of concepts in the bottleneck. The *Broden+* dataset does not offer specific concepts for traffic signs. To improve inspectability, additional related concepts should be added.

Conflicting evidence in the SSB. We show here 2 examples from 2 clusters in the rows 2 and 3 in figure 6.4, in which persons are either misclassified as car or as building. Both with

very high confidence. In particular, in both cases there is clear evidence for the correct concepts, yet conflicting evidence diffuses from the immediate surroundings resulting in superseding evidence for the confused class.

Missing evidence: The last pattern we observe is lack of concept evidence, mostly in dark or blurry regions. In the last row of figure 6.4, the bottleneck has low evidence for the correct class person (confidence of 0.11, displayed below "Groundtruth"). Visually, the person is easy to identify in the image, standing in the door of a building. Inspecting the bottleneck, we see that the part-evidence for person is below average. This is aggravated by the missing evidence for individual parts: *leg*, *torso* and *foot*, shown in figure 6.5 (bottom). This indicates that the information might have been lost upstream.



(a) ROI in image, groundtruth and prediction (b) Part and material activations

Figure 6.4: Selection of error examples from four different clusters. *Blue:* SSB@pyramid activations used to classify the center pixel of the 3x3 rectangle in white. *Orange:* Average SB activations of the closest true positive cluster (counterfactual) of the target class for reference.

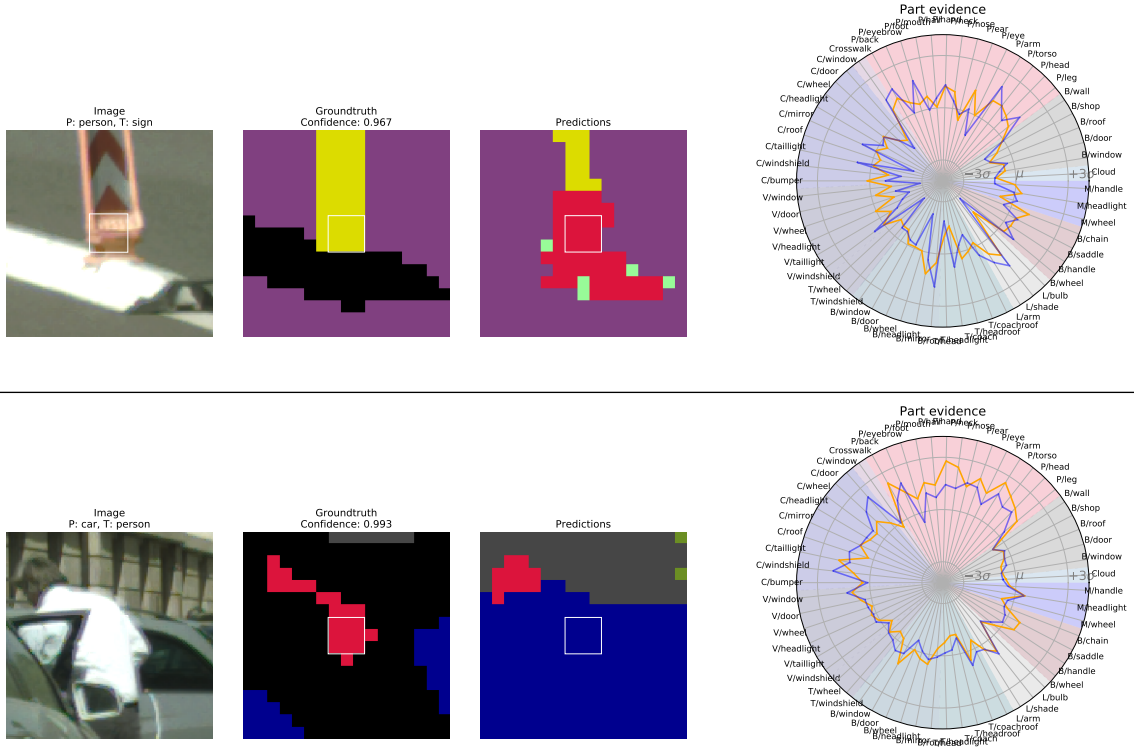


Figure 6.5: Extension to figure 6.4 (first and second row) showing 59 individual part evidences in SSB@pyramid.

6.4 Conclusion

We have investigated the use of Supervised Semantic Bottlenecks (SSBs), as proposed in [LFS21], for inspecting errors on the task of street scene segmentation. To this end, we proposed the use of agglomerative clustering of the bottleneck activations to find modes of errors. These modes can be used to identify repeating errors or to provide counterfactuals – missing concept evidences that correct misclassifications. As sanity check, we used bottleneck interventions, to test whether the learned mapping from concepts to classes is meaningful and "interpretable". While bottlenecks at earlier layers learn such meaningful mappings only for few classes, bottlenecks at later layers provide substantial improvements.

Given that predictions are based on human meaningful concept definitions at an intermediate layer, we have shown that SSBs in conjunction with our proposed error-analysis can build a strong basis to partially lift the "black-box" property of deep networks.

7 Analyzing the Dependency of ConvNets on Spatial Information

Contents

In this last chapter, I guess you mean 'last chapter of this part'? we provide evidence that SBs can be constructed with image-level instead of pixel-level concept information for the task of image classification. This can be of interest, when the exact concept position is of no relevance and expensive pixel-wise annotations are no option. We construct this evidence without using SBs explicitly, but by showing that pixel-wise information is irrelevant at later layers in convolutional networks (CNNs) used for image classification. Intuitively, image classification *should* profit from using spatial information. Recent work, however, suggests that this might be overrated in standard CNNs. In this chapter, we are pushing the envelope and aim to investigate the reliance on spatial information further. We propose to discard spatial information in individual layers via shuffling locations or average pooling during both training and testing phases to investigate impact on performance. In line with prior work showing lack of global context integration [BB19], we observe spatial information to be decreasingly relevant with network depth. We can discard spatial information in the last 30% of layers In particular, discarding spatial information in the last 30% of VGG-16 layers on CIFAR-100, the test accuracy only drops by 0.03%. Moreover, removing 53% of layers only drops accuracy by 2.66%. Evaluation on several object recognition datasets with a wide range of CNN architectures shows an overall consistent pattern.

7.1 Introduction

Despite the impressive performances of convolutional neural networks (CNNs) on computer vision tasks [LSD15, GDDM14, KSH12, HZRS16, SZ14], their inner workings remain mostly obfuscated to us, especially how the information is encoded throughout layers. Generally, the majority of modern CNNs for image classification utilize a collection of filters with local receptive fields to capture hierarchical patterns across all the convolutional layers [KSH12, SZ14, HZRS16]. Such design choices are based on the assumption that spatial information remains important at every convolutional layer, and better representations can be attained by gradually enlarging the receptive field to incorporate more contexts. This further leads to lots

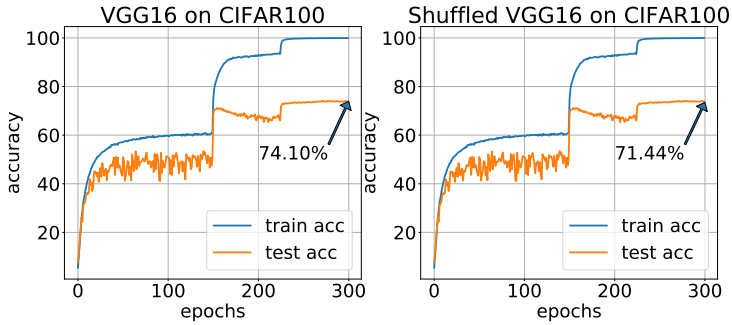


Figure 7.1: Shuffling the feature maps from the last 54% layers in VGG-16 randomly and spatially only reduces the final test accuracy by 2.66% (from 74.10% to 71.44%) on CIFAR-100, and the training processes look surprisingly similar, which implies that spatial information may not be necessary for good classification accuracy.

of approaches that help capture spatial correlations between features in order to improve model performance [SLJ+15, HSS18, BLZBG16]. For example, a popular class of those methods is the visual attention mechanism [JSZ+15, MHG+14] which enables more powerful representations by enhancing the most salient region of the image.

However, recent works on restricting the receptive field of CNN architectures for scrambled inputs [BB19] or using wavelet feature networks of shallow depth [OBZ17], have all found it to be possible to acquire competitive performances on the respective tasks. This raises doubts on the necessity of spatial information for classification and whether the network can still maintain the performance when the spatial information is completely removed from the training process.

In this work, we re-design the structure of the network to separate the spatial information and channel-wise information independently, with the goal of analyzing the dependency of the network on them. Spatial information refers to the spatial ordering on the feature map. To this end, we propose *channel-wise shuffle* to eliminate channel information, and *spatial shuffle*, *patch-wise spatial shuffle* and *GAP+FC* to eliminate spatial information. Surprisingly, we find that the spatial information is not necessary at later layers, and the modified CNNs, i.e. without accessing any spatial information at later layers, can still achieve competitive results on several object recognition datasets. As an example, figure 7.1 shows the training processes of a standard VGG-16 and a modified VGG-16 with spatial shuffle on CIFAR-100. In the shuffled VGG-16, feature maps must first go through a random spatial shuffle operation before convolved with the filters from the last 54% layers. Interestingly, the test accuracy only drops 2.66%, and the training process is nearly identical to the standard VGG-16. This observation generalizes to various CNN architectures: removing spatial information from the last 30% layers gives a surprisingly little test accuracy decrease within 1% across architectures and datasets, and the accuracy decrease is still within 7% even if the last 50% layers are manipulated. This indicates that spatial information is overrated for standard CNNs and not necessary to reach competitive performances. Finally, our investigation on the detection task shows that although the unavailability of spatial information at later layers does hinder the CNN to localize objects, the impact is not as fatal as expected; at the same time, the classification ability of the model is not affected.

The main contributions of our work are as follows: we find that spatial information at later layers is not really necessary for good classification test accuracy and that even though the depth of the network plays an important role, later layers do not require spatial integration. As a side effect, GAP+FC leads to a smaller model with fewer parameters with small test accuracy

drops.

7.2 Methods and Experimental Setup

In this section, we design methods to systematically study the phenomenon found in figure 7.1 that spatial information appears to be neglectable to some extent. We test how information is represented throughout the network's layers by discarding spatial or channel information in different ways in intermediate layers and applying them to well-established architectures. Experiments are conducted on object recognition and detection tasks. Section 7.2.1 elaborates details on our approaches, and the experimental setup is discussed in section 7.2.2.

7.2.1 Approaches to Constrain Information

We propose four different methods, namely *channel-wise shuffle*, *spatial shuffle*, *patch-wise spatial shuffle*, and *GAP+FC*, to remove either spatial or channel information from the training. Spatial information here refers to the awareness of the relative spatial position between activations on the same feature map, and channel information stands for the dependency across feature maps. The left part of figure 7.2 illustrates an example of VGG-16 with its last two layers modified by GAP+FC or any of the three shuffle methods.

Spatial shuffle extends the ordinary convolution operation by prepending a random spatial shuffle operation to permute the input to the convolution. As illustrated in figure 7.2: Given an input tensor of size $c \times h \times w$ with c being the number of feature maps for a convolutional layer, we first take one feature map from the input tensor and flatten it into a 1-d vector with $h \times w$ elements, whose ordering is then permuted randomly. The resulting vector is finally reshaped back into $h \times w$ and substitute the original feature map. This procedure is independently repeated c times for each feature map so that activations from the same location in the previous layer are misaligned, thereby preventing the information from being encoded by the spatial arrangement of the activations. The shuffled output becomes the input of an ordinary convolutional layer in the end. Even though shuffling itself is not differentiable, gradients can still be propagated through in the same way as pooling operations. Therefore it can be embedded into the model directly for end-to-end training. As the indices are recomputed within each forward pass, the shuffled output is also independent across training and testing steps.

Images in the same batch are shuffled in the same way for the sake of simplicity since we find empirically that it does not make a difference whether the images in the same batch are shuffled in different ways.

Patch-wise spatial shuffle is a variant of *spatial shuffle*. In contrast, patch-wise spatial shuffle does not perform on a global scale but a local scale by dividing the feature map into grids. Each patch in the grid is subsequently shuffled independently. Afterwards, an ordinary convolution is performed as usual. Note that the two operations are equivalent when the patch size is the same as the feature map size. figure 7.2 demonstrates an example of patch-wise spatial shuffle with a 2×2 patch size, where the random permutation of pixel locations is restricted within each patch.

Channel-wise shuffle is used to investigate the importance of channel information which is normally deemed as essential [XGD+17, ZQXW17, ZZLS18]. It keeps the spatial ordering of activations and randomly permutes the ordering of feature maps to prevent the model from utilizing channel information. An illustration can be seen in figure 7.2, channel-wise shuffle is

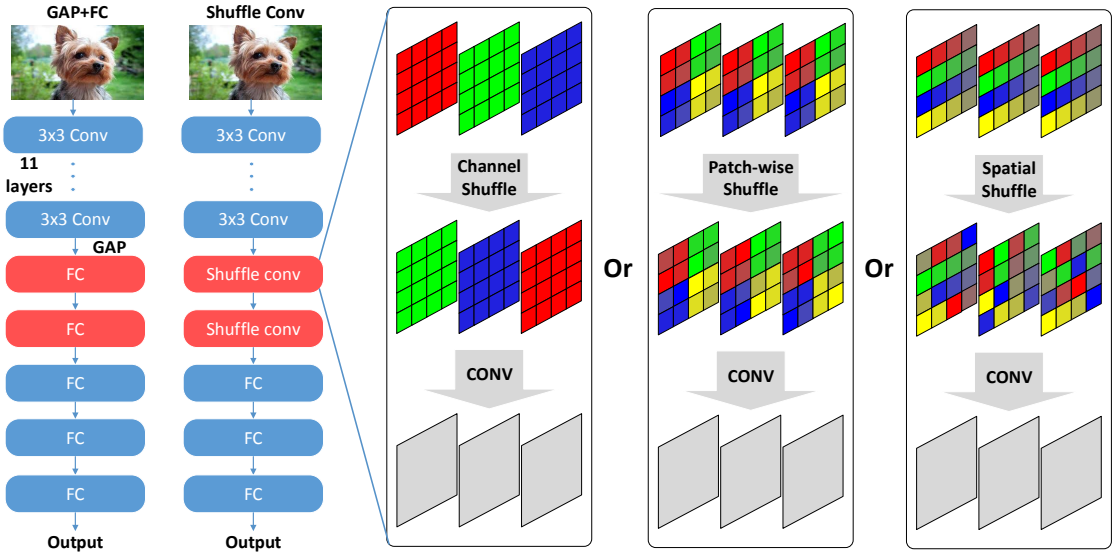


Figure 7.2: An example of VGG-16 modified by our methods. The leftmost architecture shows the modification (in red) from *GAP+FC*, where the last two convolutional layers are replaced by fully-connected layers after a GAP layer. The middle architecture shows the modification (in red) from shuffle conv, where the last two convolutional layers are replaced by one of the shuffling methods and an ordinary convolution. *Spatial shuffle* randomly and independently permutes pixels on each feature map at a global scale in the sense that a pixel can end up anywhere on the feature map. *Patch-wise shuffle* first divides the feature map into grids; then it randomly permutes the pixel locations within each grid independently. *Channel shuffle* randomly permutes the order of feature maps, leaving the spatial ordering unchanged.

also performed independently across training and testing steps.

GAP+FC denotes Global Average Pooling and Fully Connected Layers. *Spatial Shuffle* is an intuitive way of destroying spatial information. However, shuffling introduces undesirable randomness into the model; non-deterministic feature maps from an image lead to fluctuations in the model prediction, so an evaluation needs multiple forward passes to acquire an estimate of the mean of the output. A simple deterministic alternative achieving a similar goal is to deploy Global Average Pooling (GAP) after an intermediate layer, and all the subsequent ones are substituted by fully connected layers. Compared to *Spatial Shuffle* that introduces an extra computational burden at each forward pass, it is a much more efficient way to avoid learning spatial information at intermediate layers because it shrinks the spatial size of all subsequent feature maps to one; therefore, the number of FLOPs and parameters are also reduced.

7.2.2 Experimental Setup

This section details the experimental setup for the classification and object detection tasks. We test different architectures on three datasets: CIFAR-100, Small-ImageNet-32x32 [CLH17], and Pascal VOC 2007 + 2012. Small-ImageNet-32x32 is a down-sampled version of the original ImageNet (from 256×256 to 32×32). We report top-1 accuracy and mAP [EVGW$^+$b, EVGW$^+$a] in classification and detection experiments respectively. We will take an existing architecture and apply the modification to different layers. The rest of the setup and hyper-parameters for modified architectures remain the same as the original architectures.

Classification: For the VGG architecture, the modification is only performed on the convolutional layers, as illustrated in figure 7.2. For the ResNet architecture, one bottleneck sub-module is considered as a single piece, and the modification is applied onto the 3×3 convolutions within the sub-module since they are the only operations with spatial extent. Features that go through the skip connection branch are also shuffled in the shuffle experiments to prevent the model from learning to ignore the information from the residual branch. The rest of the configuration remains the same (see supplemental material for an example of modified ResNet-50 architecture).

For CIFAR-100 and Small-ImageNet-32x32 experiments, the original ResNet architecture down-samples the input image by a factor of 32 and gives 1×1 feature maps at last layers, therefore shuffling is noneffective. To make shuffling non-trivial, we set the first convolution in ResNet to 3×3 with stride 1 and the first max-pooling layer is removed so that the final feature map size is 4×4.

To alleviate the effect of mismatched training details, we first reproduce the reported results for all experiments and then train our modified architectures under the same training setting. All models in the same set of experiments (e.g. VGG-16 on CIFAR-100) use the same set of hyper-parameters, and they share the same initialization from the same random seed. During testing, we make sure to use a different random seed than during training.

Detection: We use the training set and validation set of VOC 2012+2007 as the training data and report mAP on VOC 2007 test set. We shuffle the last layer in the backbone model to test the robustness of localization against the absence of spatial information.

7.3 Results

We first compare the test accuracy of VGG-16 on CIFAR-100 with spatial or channel information missing from a different number of last layers in section 7.3.1. An in-depth study of our main

Table 7.1: Top-1 accuracy of VGG-16 on CIFAR-100 with spatial / channel-wise shuffle enabled at either training or test time for the last 30% layers. A model from standard training does not possess robustness against spatial shuffle (23.49%) and channel-wise shuffle (1.04%). However, when imposed in training, the model achieves 74.07% test accuracy for spatial shuffle and 67.56% for channel-wise shuffle, showing impressive robustness to the loss of spatial information.

Train Scheme	no shuffle	channel shuffle	channel shuffle	no shuffle	spatial shuffle	spatial shuffle	no shuffle
Test Scheme	no shuffle	channel shuffle	no shuffle	channel shuffle	spatial shuffle	no shuffle	spatial shuffle
Top-1(%)	74.10	67.56	67.80	1.04	74.07	73.74	23.49

observations on CIFAR-100 and Small-ImageNet-32x32 for VGG-16 and ResNet-50 is conducted in section 7.3.2. In section 7.3.3, we investigate the model robustness against the loss of spatial information in various degree by controlling the amount of spatial information that passes through the network. Finally, we present the detection results on VOC datasets in section 7.3.4.

7.3.1 Spatial and Channel-wise Shuffle on VGG-16

In this section, we first investigate the invariance of pre-trained models to the absence of the spatial or channel information at test time, then we impose this invariance at training time with methods in section 7.2.1.

Shuffle the Last 30% Layers Channel-wise : Our baseline is a VGG-16 trained on CIFAR-100 that achieves 74.10% test accuracy. We first test its robustness against the absence of the channel information at test time by substituting the last 30% convolutional layers with the channel-wise shuffle convolution. As is expected, the test accuracy drops to 1.04% (table 7.1), which is the same as the random guessing on CIFAR-100. Following the same training scheme of the baseline, we then train another VGG-16 with channel-wise shuffle added to its last 30% convolutional layers. This model can reach around 67% test accuracy no matter whether channel-wise shuffle is applied at test time. However, it still performs significantly worse than the baseline, which indicates that the expressiveness of the model is much limited without utilizing the ordering of feature maps even though the spatial information is preserved.

Shuffle the Last 30% Layers Spatially: As a comparison to channel shuffle, we repeat the same experiment on spatial shuffle, and the result is presented in the second half of Table 7.1. No shuffle $\rightarrow$ spatial shuffle of the pre-trained VGG-16 gives 23.49% test accuracy, which is similar to the test accuracy of a one-hidden-layer perceptron (with 512 hidden units and ReLU activation) on CIFAR-100 (25.61%) when evaluated with the random spatial shuffle. However, if the spatial shuffle is infused into the model at training time, then the baseline test accuracy can be retained no matter whether random spatial shuffle appears at test time (74.07% for spatial shuffle $\rightarrow$ spatial shuffle and 73.74% for spatial shuffle $\rightarrow$ no shuffle).

Shuffle Other Layers: To systematically study the impact of spatial and channel information, we gradually increase the number of modified layers from the last in VGG-16 and report the corresponding test accuracy in Fig. 7.3. All models are trained with the same setup, and shuffling is performed both at training and test time; the x-axis is the percentage of modified layers counting from the last layer on with 0 referring the baseline.

Besides an overall decreasing trend for both shuffling with the increase of the percent of modified layers, the test accuracy of spatial shuffle drops unexpectedly slowly, e.g merely 2.66%

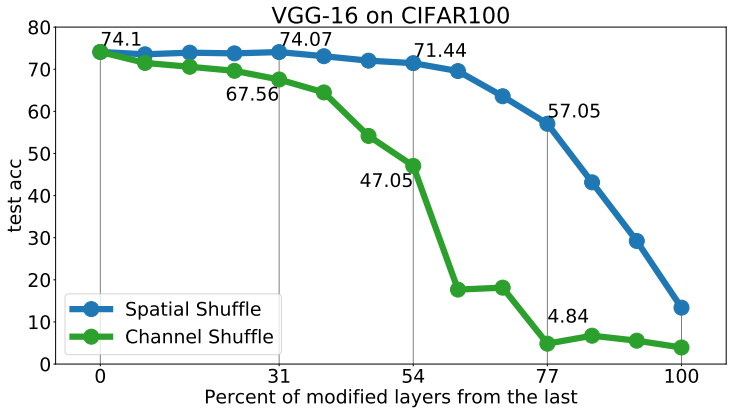


Figure 7.3: Classification accuracy of VGG-16 on CIFAR-100 with different shuffle schemes. The very slow decrease of the test accuracy of spatial shuffle implies a far less important role of spatial information for classification. The test accuracy is not much affected, given that the spatial shuffle modifies 31% of its layers. Even with 54% later layers shuffled spatially, the test accuracy only decreases by 2.66%, and the same number of the test accuracy decrease in channel-wise shuffle happens when the last layer is modified.

test accuracy drop when up to 54% of layers from the last are shuffled spatially. Likewise, when spatial information is removed from the last 77% layers, it still has a reasonable test accuracy (57.05%), whereas the test accuracy of channel-wise shuffle is only 4.84%.

Discussion: This indicates that although a standard model makes use of both spatial dimension and channel dimension to encode information, the spatial information plays a surprisingly less pivotal role than the channel information. The model is even able to adapt to the complete absence of spatial information at later layers if spatial information is removed explicitly at training time, which strengthens the claims from [BB19, SFH17] that CNNs intrinsically possess invariance to the spatial relationship among features to some extent. Moreover, the unsuccessful adaptation to channel-wise shuffle implies that the large model capacity may mainly come from the channel order and shuffling the channel order causes unrecoverable damage to the model.

7.3.2 Spatial Information at Later Layers is Not Necessary

In this section, we design more experiments to study the reliance of different layers on spatial information: we modify the last convolutional or bottleneck layers of VGG-16 or ResNet-50 by *Spatial Shuffle* (both at training and test time) and *GAP+FC* such that the spatial information is removed in different ways. Our modification on the baseline model always starts from the last layer and is consecutively extended to the first layer. The modified networks are then trained on the training set with the same setup and evaluated on the hold-out validation set.

Results on CIFAR-100 and Small-ImageNet-32x32: Results of VGG-16 and ResNet-50 on CIFAR-100 and Small-ImageNet-32x32 are shown in Fig. 7.4. The x-axis is the percent of modified later layers, and 0 is the baseline model test accuracy without modifying any layer.

As we can see, *Spatial Shuffle* and *GAP+FC* have a similar overall behaviour consistently across architectures and datasets: the baseline test accuracy is retained for a long time before it starts to decrease with the increase of the percent of modified layers. When the last 30% layers are modified by GAP+FC or spatial shuffle, there is no or little test accuracy decrease across

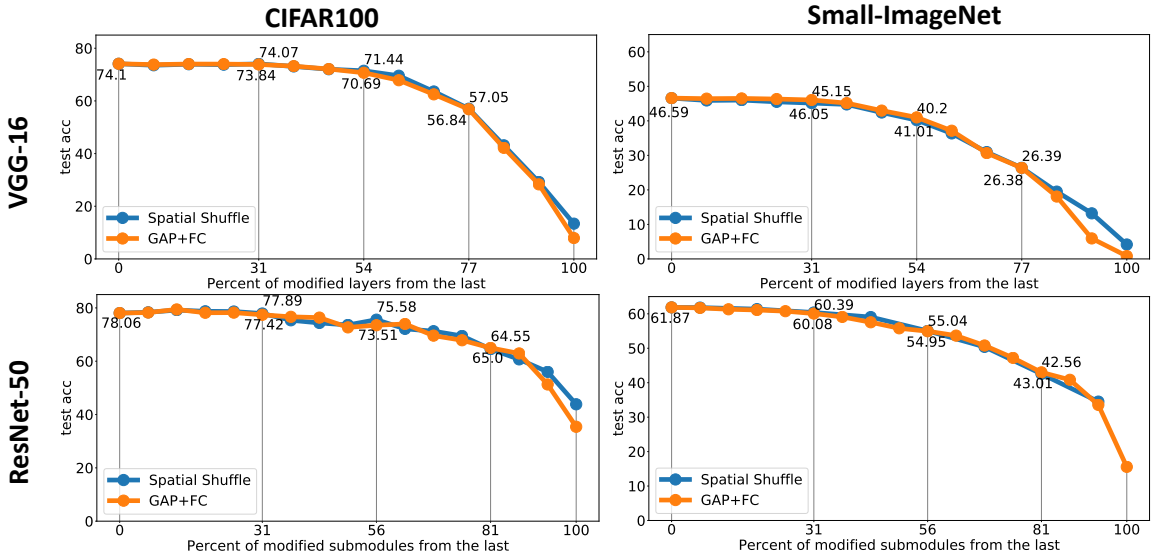


Figure 7.4: Classification results of GAP+FC and spatial shuffle for VGG-16 and ResNet-50 on CIFAR-100 and Small-ImageNet-32x32. The x-axis is the percent of modified layers / sub-modules counting from the last one. Models on the same dataset are trained with the same setup. It can be observed consistently across experiments that the baseline test accuracy is preserved for a large fraction of shuffled (orange) or averaged (blue) layers. This suggests, that spatial information at later layers is not necessary for good test accuracy. The difference between the baseline models and the models whose latter half of the layers are modified by GAP+FC or spatial shuffle is, however, still in a reasonable range between 2.48% (ResNet-50 with spatial shuffle on CIFAR-100) to 6.92% (ResNet-50 with GAP+FC on Small-ImageNet-32x32).

experiments (0.17% for ResNet-50 on CIFAR-100 and 1.44% for VGG-16 on Small-ImageNet with spatial shuffle). And the test accuracy decrease is still in a reasonable range (2.48% with spatial shuffle on CIFAR-100 and 6.92% for GAP+FC on Small-ImageNet-32x32 for ResNet-50), even with around half of the last layers modified. At 77% to 81% of the modified later layers, the test accuracy just starts to show a significant difference to the baseline in the range of 8.58% (ResNet-50 with spatial shuffle on CIFAR-100) to 20.21% (VGG-16 with GAP+FC on Smalll-ImageNet-32x32).

Our experiments here clearly show that spatial information can be neglected from a significant number of later layers with no or small test accuracy drop if the invariance is imposed at training, which suggests that *spatial information at last layers is not necessary for good test accuracy*. We should, however, notice that it does not indicate that models whose prediction is based on spatial information can not generalize well. Besides, unlike the common design manner that layers at different depth inside the network are normally treated equally, e.g. the same module is always used throughout the architecture [HZC+17, SHZ+18, IMA+17], our observation implies it is beneficial to have different designs for different layers since there is no necessity to encode spatial information in the later layers. As a side effect, GAP+FC can reduce the number of model parameters with little test accuracy drop. For example, GAP+FC achieves nearly identical results (46.05%) to the VGG-16 baseline (46.59%), while reducing the number of parameters from 37.70M to 29.31M on Small-ImageNet-32x32.

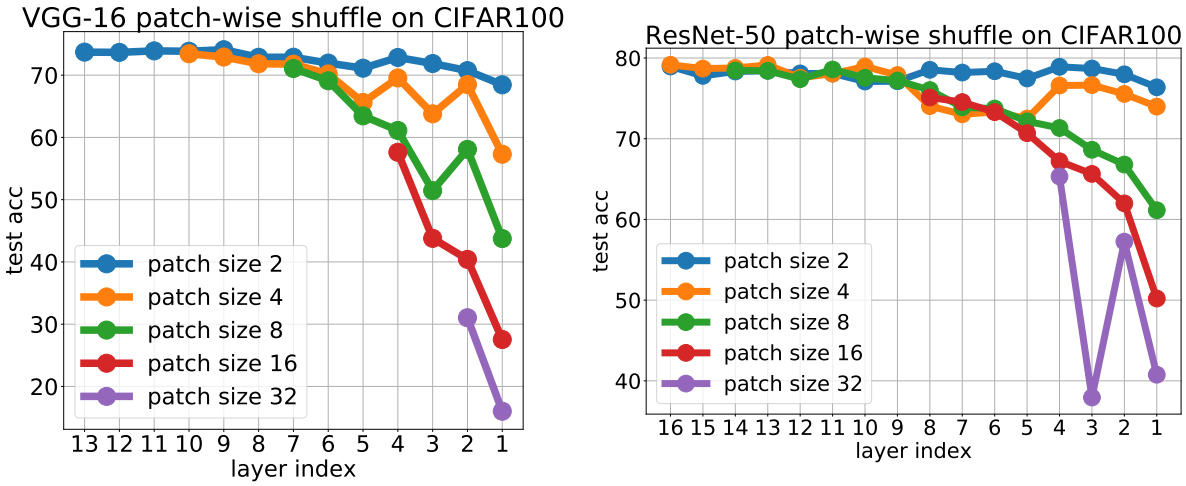


Figure 7.5: The result of patch-wise spatial shuffling of VGG-16 and ResNet-50 on CIFAR-100. Only a single layer is shuffled at a time. Layer index 13 and 16 stand for the last layer of VGG-16 and ResNet-50, respectively. With the increase of the patch size, the test accuracy decreases faster at first layers than that at last layers. It is interesting to see that both models' test accuracy do not fall into the random guess (16.02% for VGG-16 and 40.76% for ResNet-50) at layer index one and patch size 32, where the input image is completely shuffled.

7.3.3 Patch-wise Spatial Shuffle

In this section, we study the relation between the model test accuracy and the amount of spatial information that propagates throughout a network. The latter is controlled by patch-wise spatial shuffle with different patch sizes. The larger the patch size is, the less the preserved spatial information. Patch-wise spatial shuffle reduces to spatial shuffle when the patch size is the same as the feature map size, in which case no spatial information remains. Our experiments are conducted on CIFAR-100 for VGG-16 and ResNet-50, and we only shuffle a single layer at a time since the model is not able to recover the "damage" caused by shuffling an early layer (see more in the supplemental material).

The result of patch-wise spatial shuffling of different patch sizes is shown in Fig. 7.5. We can see that the patch size does not make much difference in terms of the test accuracy at later layers, e.g. results of patch size 2, 4 and 8 for ResNet-50 at 8-14 layers are similar. However, the test accuracy has a rapid decrease with the increase of the patch size at first layers, indicating a relatively important role of spatial information at first layers. Nevertheless, this role might not be as much important as what is commonly believed, as the ResNet-50 still has 40.76% test accuracy when the input image is completely shuffled.

7.3.4 Detection Results on VOC Datasets

Object detection should intuitively suffer more from spatial shuffling than classification since the spatial information should help to localize objects. In this section, we show some initial results on Pascal VOC [EVGW+b, EVGW+a].

We design an analogue to YOLO [RDGF16] as our detection model. The architecture consists of a backbone and a detection head; the backbone is a ResNet-50 without the classifier, and the detection head has three bottlenecks and a 3×3 convolutional layer whose outputs is

Figure 7.6: Left: Qualitative detection results on the VOC 2007 test set. Examples are the first 11 images in the test set. The left result is from the baseline, and the right result is from the shuffled model. Right: Detection error analysis of our baseline and the shuffled model shows a doubled localization error in the shuffled model and the rest types of error are in the same level as the baseline.

in the same format as [RDGF16]. Different to [RDGF16], we deploy a 3×3 convolution instead of a fully connected layer in the end to output the final detection results. The latter gives the model potential access to the object feature, which may be exploited by the model to predict its location. In order to prevent the undesirable shortcut, we use a 3×3 convolution so that the prediction of a bounding box at a certain location does not depend on all activation on the feature map.

By using a pre-trained ResNet-50 on ImageNet, we can reach 66% mAP on VOC2007 test set after fine-tuning, which is the same as the number in [RDGF16]. To avoid pretraining a spatially shuffled model on ImageNet, we compare a spatially shuffled model and a non spatially shuffled model, both trained from scratch on VOC. Our models are trained for 500 epochs with exponentially decaying learning rate starting from 0.001. Our baseline model achieves 50% mAP on VOC2007 test set without using an ImageNet pre-trained backbone. The result of the shuffled model, where we apply random shuffle to the last layer of the backbone, is 34%. While this sounds like a large drop, it turns out that the classification performance is essentially preserved and only the localization performance is suffering. To analyze this effect in detail, we use the method and tools proposed in [HCD12]. The diagnosis tool classifies each prediction from the model as either correct prediction or a type of error based on its class label and IoU with the ground truth. More details can be found in [HCD12].

The results in Fig. 7.6 right show that the misclassification to the wrong class and background are of similar percents for both models, and the localization error doubles for the shuffled model (an increase from 14.2% to 28.4%). Though random shuffling indeed affects the model's localization ability, it is unexpected that the effect is not fatal. Because random shuffling switches features, it is highly likely the model trained with spatial shuffle has to predict the correct bounding box for one object based on some other features. We should also notice that a prediction is counted as a localization error if it has the correct class label and the IoU to the ground truth is less than 0.5. Therefore, classification-wise speaking, the shuffled model got 73.7% (45.3% + 28.4%) of its predictions correct, which is at the same level as the baseline (73.3% = 59.1% + 14.2%).

Qualitative Results: Fig. 7.6 left shows some qualitative results from both models. Those examples are the first 11 images in the VOC2007 test set. We can see that the localization error actually mainly comes from small objects for which the shuffled model tends to predict several bounding boxes on one object, and the bounding box of the relatively big object is not really off, e.g. the shuffled model managed to localize the dining table in the middle right image and the horse in the middle left image while the baseline can not.

7.4 Conclusion

To conclude, we empirically show that a significant number of later layers of CNNs are robust to the absence of spatial information, which is commonly assumed to be important for object recognition tasks. Modern CNNs can tolerate the loss of spatial information from the last 30% of layers at around 1% accuracy drop; and the test accuracy only decreases by less than 7%, when spatial information is removed from the last half of layers on CIFAR-100 and Small-ImageNet-32x32. Though the depth of the network is essential for good test accuracy, later layers do not require spatial integration.

8 Conclusion and Future Work

Contents

Progress in deep learning has increased fast in recent years, achieving human or even super-human level performance in some vision tasks. However, these recent improvements often rely on optimizing models for accuracy only while disregarding for other factors like adversarial robustness or interpretability. Given the potential that deep learning now offers to society, it quickly becomes evident that these factors need to be improved rapidly. Otherwise, we risk mistrusting systems using deep learning.

8.1 Conclusions

In this thesis, we addressed the trustworthiness issue along two research directions: First, in part I, we improved the *efficiency of adversarial machine learning* by proposing to train on subsets to reduce computational overhead of adversarial training and by utilizing Lipschitz regularization to provide deterministic certified robustness that is particularly cheap to compute during inference. Second, in part II, we show that models on street scene segmentation have poorly aligned intermediate representations that are difficult to inspect and impractical to use for error analysis. To alleviate, we proposed Semantic Bottlenecks (SBs) whose channels are aligned with human meaningful concepts and showed that they can be utilized to identify types of errors.

In the following, we revisit the contributions of each part and chapter and conclude this thesis with a discussion on future work in section 8.2.

Part I, Towards more Efficient Adversarial Robust Training: In the first part, we investigated two paths to improve the efficiency of adversarial robust training over vanilla adversarial training methods. Thereby addressing the vulnerability towards adversarial examples – slight added input perturbations dramatically changing the output – and its deteriorating impact on trust in applied systems.

In chapter 3 we considered perturbing only a fixed subset of training data, thereby restricting the access to the "seen" adversarial examples and reducing the costly inner optimization loop of AT. We showed that such Subset Adversarial Training (SAT) provides non-trivial robustness transfer to all classes, even when perturbing only 10% of data. Counter-intuitively, classes that seem related do not necessarily transfer best across each other. We observed instead, that classes and examples which are hard to classify provide the best transfer, leading to unexpected results. E.g. class *cat* provides over 20%-points better robustness transfer to class *truck* than *car* does to *truck*. Based on the hardness correlation, we showed that just 50% of the hardest

training examples can reach baseline AT performance. In a task transfer setting, this reduces even to 30% with appropriate loss weighting. Overall, SAT gives an encouraging outlook on adversarial training: not all examples are required to be perturbed to reach high adversarial robustness. Most importantly though, answering the "why" robustness transfers so well among classes might shed new light into improving our understanding of adversarial examples and training in the future.

In chapter 4 we focused on an alternative approach to improve adversarial robustness, i.e. Lipschitz regularization which does not require constructing adversarial examples and provides cheap deterministic robustness certification. Based on existing methods on bounding the global Lipschitz constant of a model, we identified two issues that are likely to hold back competitive performances on larger datasets and proposed Calibrated Lipschitz-Margin Loss (CLL) to address them. Firstly, existing work consistently minimizes or constrains the Lipschitz constant to be small, thereby limiting the complexity of the classifier. Secondly, the used cross-entropy losses are not calibrated to the margin width induced by the Lipschitz constant, thereby training models with unnecessarily high slack. CLL calibrates the loss and implicitly returns control over the Lipschitz constant by making slack a tuneable parameter. We provided state-of-the-art results on CIFAR-100 and Tiny-ImageNet. Importantly, the latter being achieved with much smaller models than currently used in related work. CLL is a drop-in replacement for existing work using global Lipschitz bounds and should be a valuable staple for future research and or applications where calibrated outputs count.

Part II, Inspectable Representations with Semantic Bottlenecks: In the second part, we addressed a second issue in deep learning, reducing its trustworthiness. Deep learning models on vision tasks are trained as black-boxes, disregarding the alignment between learned intermediate representations and representations that are meaningful to humans. This misalignment results in uninterpretable models, that are difficult to inspect w.r.t. their inner reasoning process. However, this is essential in high-stakes applications.

In chapter 5, we propose a step towards more inspectable and aligned representations with Semantic Bottlenecks (SBs). A SB is a single layer with reduced number of channels, of which each is explicitly or implicitly aligned with human meaningful representations. SBs are easy to integrate with existing architectures, have only little impact on prediction performance and enable the investigation of intermediate visual concept evidences. We showed a supervised and an unsupervised variant of SB, both achieving much higher improvements in visual concept alignment. For the latter, we proposed a novel metric we called *AUiC*, which supported our conjecture that deep models are poorly aligned by default. Overall, SBs improve alignment up to a factor of six while reducing number of channels from thousands down to dozens.

In chapter 6 we utilized our proposed supervised SBs to analyze failure cases on the task of street scene segmentation. Thereby investigating the utility of SBs for explaining predictions. Via agglomerative clustering of bottleneck activations, we showed that is possible to find clusters containing errors on multiple input images, that they generally fall into three types of errors (misinterpretation, conflicting evidence or missing evidence) and that clusters enable provide counterfactuals for comparison, indicating a minimum change in concept evidences to achieve an accurate prediction.

In chapter 7, we showed in an tangential study to Semantic Bottlenecks, that deep models on image classification tasks depend little on spatial context in the last 30% of convolutional layers. Thereby indicating that SBs for image classification could be integrated by aligning with image-level concepts, like "tail of animal is present", instead of expensive pixel-level concept annotations. We investigated this independency by proposing two architectural changes that explicitly destroy spatial information: randomly shuffling the feature maps and early use

of global average pooling. These results generalized across multiple datasets (CIFAR-100, Small-ImageNet-32 $\times$ 32) and architectures (VGG-16 and ResNet-50).

8.2 Future Work

In this section, we discuss several remaining problems in our and related work, how they could be addressed and how these potential research directions could improve our understanding of adversarial robust and inspectable deep learning. This section is aligned with the organization of chapters in this thesis. The penultimate subsection, however, groups the discussion for chapter 5 to 7.

8.2.1 Regarding "On Adversarial Training without Perturbing all Examples"

Robust overfitting: As shown in related work, adv. training may be prone to memorize adv. examples generated during training [DXY+21] which leads to robust overfitting [RWK20]. In our Subset Adversarial Training (SAT) setting in chapter 3, this is likely to be worse when subset size $|A|$ is small leading to increased overfitting. To improve robust accuracies under such subset constraints, the degree of memorization should be analyzed. Moreover, [DXY+21] proposes a simple regularization term in addition to adv. training that can mitigate this. Its integration with SAT should be investigated.

What constitutes hard examples: In chapter 3, we observed that hard examples transfer robustness best and have shown that they can be informative to augment datasets with additional hard classes (section 3.3.2). To push the envelop of adversarial training, we need to improve our understanding of what constitutes a hard example. We see two plausible hypotheses dictating robustness transfer: (i) closeness to decision boundary and (ii) feature diversity. The closeness to the decision boundary has been utilized in related work to improve adversarial training [WZY+20, HZG+21, ZZN+21b]. The gist similar to our observation regarding example entropy: examples in proximity to the decision boundary appear to contribute differently to robust training than examples far away. However, it seems implausible to assume that class *car* is closer to class *cat* than *truck*. In contrast, point (ii), feature diversity might play a dominant role. We speculate, that classifying hard examples requires the detection of more diverse features, which are trained adv. robust. Given the compositional nature of deep networks, it is plausible to assume that these robust features are reused by other classes. To improve our understanding of adversarial training, we believe these questions warrant more extensive research. We discuss potential implications in the next paragraph.

Synthetic-class Adversarial Training: A striking observation in chapter 3 on CIFAR-10 is that attacking only a single class during training (*cat*) provides over 67% of baseline robust accuracy on all 10 classes (46.4% robust accuracy vs 69.0%). The following questions arise: (i) can we find an additional 11th class that outperforms the results using the *cat*-class?, alternatively, (ii) can we alter the images of class *cat* such that the robust accuracy on all classes is maximized? and (iii) if that is possible, can we construct a synthetic 11th class that serves the same purpose? These questions appear linked to the existence of *universal adversarial examples* (UAEs) [MDFFF17]: a single crafted input perturbation that changes the prediction of all classes. This is a similar effect to our SAT on *cat* example, for which the crafted perturbations have an impact on all other classes. The existence of such a synthetic class, as in (iii), would be very interesting to investigate as it has ramifications for understanding deep networks and adversarial examples. We have shown in chapter 3, that adding an 11th class from existing

datasets is possible (see figure 3.7 in section 3.3.2).

Class-Incremental Learning: A key motivation for conducting Subset Adversarial Training (SAT), was to investigate whether all training examples need to be perturbed to achieve good robust accuracies. In chapter 3, we have clearly shown that this requirement can be substantially relaxed. Less than 50% of data is sufficient to reach baseline AT performance. Our observations have encouraging ramifications for class-incremental learning [MLT+22]. Adversarial robustness transfers to new classes and the class hardness informs the degree of this transfer Here, a particular research question is of interest. Does the class-entropy remain informative enough to decide whether a new class should be attacked? Alternatively, can it inform how many PGD steps should be invested or how many examples should be attacked? Adversarial Robustness in the class-incremental learning setting has not yet been studied in detail, [KK23] being the first to our knowledge. To improve training efficiency, SAT offers a sensible foundation for future investigations on class-incremental learning.

Long-tail distributions: In chapter 3, we observed that the hardest examples to classify contribute the largest fraction of robustness during adversarial training. Hence, it is sufficient to perturb only 40% to 50% of the hardest training examples. This has appealing implications for training on datasets that follow more natural long-tailed distributions with few classes having many examples and most classes having few examples [THZ20, ZKH+23]. Given that the examples in the tail are potentially hard to classify, it should be evaluated whether these examples can provide most of the robustness for the whole model – thereby reducing the demand to perturb all of the data and/or classes. This, of course, assumes that our entropy metric is a reliable proxy for example hardness and that the latter is informative on such long-tail data distributions.

8.2.2 Regarding "Certified Robust Models with Slack Control"

Layer normalization for Lipschitz models: One of the major advances in deep learning, allowing faster convergence and increased training stability, has been the introduction of layer normalizations such as batch norm [IS15]. In the field of Lipschitz regularization though, layer normalization is not typically applied. While the Lipschitz constant of batch norm is trivial to compute, we found in preliminary studies that it decreases stability of training using our CLL or *GloRo* losses and leads to impaired certifiable robust accuracy. That is, we observe some channels to attain very small weights leading to small output standard deviations. This in turn results in large Lipschitz constants. The exact reasons need to be investigated further to eventually enable integration of batch, or any other, norm.

Alternative norms: As already discussed in [RWGM20], the most frequently used L_2 and L_∞ norm might not be the most suitable metrics for computer vision tasks. Under these metrics, it is easy to find two images that are distant from each other, yet look visually similar to humans. This could be a severe issue for Lipschitz regularization, where the relationship between input and output distances are key. This is aggravated when the Lipschitz constant is regularized globally, as in chapter 4. That is, while the L_2 and L_∞ metrics might be suitable in a small ϵ neighborhood around the image, global Lipschitz regularization demands any, potentially distant, example pair-distance are upper bounded by the Lipschitz constant. Given the highly non-linear nature of deep networks, this assumption seems highly conservative. The upper bound is given by the sharpest transition zones between classes [HA17], potentially providing a poor bound on smoother transitions.

Local Lipschitz regularization: As alternative to considering global Lipschitz regularization, we could instead regularize locally for examples that are close to the decision boundary, thereby determining the bound in a small neighborhood zone only. [HZS+21] proposed an extension to the GloRo loss [LWF21] that retains certified robustness involving additional optimization steps. [EKM+18] proposed an alternative loss that implicitly minimizes the local Lipschitz constant by linearizing the region around an input. Both could be combined with our proposed loss CLL, although the latter does not provide a direct mean to certify robustness. In this case, robustness must be measured empirically or by joining the method with randomized smoothing to provide probabilistic guarantees. Irrespectively, local Lipschitz regularization is likely to provide better clean and robust performances and thus should be the focus of additional research.

8.2.3 Regarding "Semantic Bottlenecks"

Multiple bottleneck layers: Our Semantic Bottlenecks in part II were integrated at a single network location only. To provide improved inspectability, additional locations should be utilized. In preliminary investigations, we found multiple SSBs and USBs integrations to substantially impact performances. For SSBs, this is likely due to a mismatch between actual representations and existing concept annotations (we used Broden+ annotations [XLZ+18]). This mismatch can only be resolved when having more diverse concepts, also including low-level image concepts like lines etc. USBs can alleviate this issue, yet they also showed poor performance likely due to the enforced one-hot output encoding – we revisit this issue in the next paragraph on uncertainty.

While challenging, integration of multiple bottlenecks should be feasible. Let's revisit the benefits that we would gain: (i) we would gain low- to high-level concept evidences and (ii) we gain the potential for more fine-grained error analysis over the analysis in chapter 6. The latter entails its own challenge though. Let's assume a bottleneck close to the input image and we want to analyze its evidences w.r.t. all predictions. The receptive field on such a low level bottleneck is large, potentially including the whole image, thereby reducing inspectability dramatically. To resolve, one could slice the model into pieces and only analyze the model piece-wise, that is, *between* two bottlenecks. In this case, the last bottleneck output becomes the input and the subsequent bottleneck is error-analyzed. This way, the receptive field can be controlled to be small. However, it needs to be investigated to what degree such a piece-wise error analysis allows investigations between input image and final prediction.

Uncertainty for USBs: In chapter 5, we have proposed the Unsupervised Semantic Bottleneck (USB), which provides more meaningful outputs by forcing each channel to have a highly specialized representation. We achieved this by enforcing one-hot outputs. While integration of a single USB-layer did not impair performance much, we observed that integrating multiple USBs do have a detrimental effect. We presume this to be based on the limited information bandwidth that is incurred by the one-hot constraint. Particularly on edges in the image, the one-hot encoding cannot convey uncertainty information about the input. To remedy, a parallel bottleneck carrying an uncertainty estimate would be beneficial. The exact implementation warrants additional research, yet it might be feasible to utilize mechanics from variational autoencoders [KW14] and Information Bottlenecks [AFDM17, MAKR21], since they explicitly encode the uncertainty for the models predictions while trading off accuracy for complexity. The latter specifically interesting for distilling a low complexity concept set. Alternatively, the entropy over the softmax logits could be a simple, yet useful proxy for such an uncertainty estimate that does not require a change in architecture. Here, temperature calibration might be

necessary to improve performance [GPSW17].

Causal learning frameworks: In the ideal case, we would like to know the causal dependencies between variables in a deep network. That is, consider the task of face recognition, then an intermediate *eye* predictor depends causally on some number of lower level feature predictors like *circle* or *eyelashes*. In a causal framework, we desire each low level predictor to affect only a subset of subsequent detectors, not all simultaneously [Sch22]. E.g., *eye* should not be dependent on the presence of *mouth*-features even though they are statistically correlated. In deep learning literature, this is often called disentangling factors of variation [BCV13, HMP+17]. Such a framework constructs a directed acyclic graph between these factors, such that their dependencies can be mechanistically understood – a desired trait for interpretability [Lip18]. In chapter 6, we utilized a simplified facet of such a framework: intervening bottleneck evidences to test their dependence on predictions (also used in [KNT+20]). While useful in this setting, this does not provide causal dependencies since we only tested for individual binary detector outputs (presence or absence (w.r.t. a threshold) as independent factors. In reality, the dependence is much more strongly entangled between outputs of multiple detectors and final prediction. While post-training evaluation is expensive (test all possible combinations) and "notoriously difficult"[Sch22], the conditional independence between factors is possible to estimate with adaptation of existing methods like LIME [RSG16] or SHAP [LL17]. Since dimensionality is dramatically reduced in SBs, computational overhead is reduced. Given the appealing properties of knowing causal dependencies, research on combining causal analysis with SBs is warranted.

8.2.4 Regarding "Analyzing the Dependecy of ConvNets on Spatial Information"

Analyzing spatial encoding along channels: In chapter 7 we showed that the spatial output of the last convolutional layers can be collapsed without impairing performance. This indicates that spatial information is not integrated at a late layer but does not guarantee it. In fact, it could be possible that the model learns to encode the necessary spatial information along the channels. E.g., car positioned at the top left is encoded in the first channel, while at the bottom right it is encoded in the second. This would substantially reduce the variety of encoded features and thus potentially impair performance. This hypothesis requires additional analysis.

List of Figures

List of Tables

Bibliography

[ACFH20] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European conference on computer vision*. Springer, 2020. Cited on page 10.

[ACÖG18] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *Proc. of the International Conference on Learning Representations (ICLR)*, 2018. Cited on page 15.

[ADH22] Chirag Agarwal, Daniel D'souza, and Sara Hooker. Estimating example difficulty using variance of gradients. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. Cited on pages 24 and 25.

[AF20] Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on pages 3, 11, 13, 33, 34, 35, and 111.

[AF21] Sahar Abdelnabi and Mario Fritz. What's in the box: Deflecting adversarial attacks by randomly deploying adversarially-disjoint models. In *Proceedings of the 8th ACM Workshop on Moving Target Defense*, 2021. Cited on page 12.

[AF22] Maksym Andriushchenko and Nicolas Flammarion. Towards understanding sharpness-aware minimization. In *Proc. of the International Conference on Machine Learning (ICML)*, 2022. Cited on page 13.

[AFDM17] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2017. Cited on page 107.

[AGM+18] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Cited on page 15.

[AHD+23] Alexandre Araujo, Aaron J Havens, Blaise Delattre, Alexandre Allauzen, and Bin Hu. A unified algebraic perspective on lipschitz neural networks. *Proc. of the International Conference on Learning Representations (ICLR)*, 2023. Cited on pages 4, 11, 12, 40, 42, 47, 48, 49, 117, 154, and 156.

[AJL21] Kjersti Aas, Martin Jullum, and Anders Løland. Explaining individual predictions when features are dependent: More accurate approximations to shapley values. *Artificial Intelligence*, 2021. Cited on page 15.

[ALG19] Cem Anil, James Lucas, and Roger Grosse. Sorting out lipschitz function approximation. *Proc. of the International Conference on Machine Learning (ICML)*, 2019. Cited on pages 11, 40, and 154.

[ALK+19] Lynton Ardizzone, Carsten Lüth, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Guided image generation with conditional invertible neural networks. *arXiv.org*, 2019. Cited on page 18.

[ALSF24] Alaa Anani, Tobias Lorenz, Bernt Schiele, and Mario Fritz. Adaptive hierarchical certification for segmentation using randomized smoothing. *arXiv.org*, 2024. Cited on page 14.

[AMRK20] Lynton Ardizzone, Radek Mackowiak, Carsten Rother, and Ullrich Köthe. Training normalizing flows with the information bottleneck for competitive generative classification. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on page 12.

[ASDX20] Maruan Al-Shedivat, Avinava Dubey, and Eric Xing. Contextual explanation networks. *The Journal of Machine Learning Research (JMLR)*, 2020. Cited on page 18.

[AUH+19] Jean-Baptiste Alayrac, Jonathan Uesato, Po-Sen Huang, Alhussein Fawzi, Robert Stanforth, and Pushmeet Kohli. Are labels required for improving adversarial robustness? *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019. Cited on page 11.

[BB19] Wieland Brendel and Matthias Bethge. Approximating CNNs with bag-of-local-features models works surprisingly well on imagenet. *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on pages 6, 91, 92, and 97.

[BBM+15] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 2015. Cited on pages 5 and 14.

[BCM+13] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. *Machine Learning and Knowledge Discovery in Databases*, 2013. Cited on pages 1 and 2.

[BCV13] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE TPAMI*, 2013. Cited on page 108.

[BFS22] Moritz Böhle, Mario Fritz, and Bernt Schiele. B-cos networks: Alignment is all we need for interpretability. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. Cited on pages 5, 18, and 79.

[BFS23] Moritz Böhle, Mario Fritz, and Bernt Schiele. Optimising for interpretability: Convolutional dynamic alignment networks. *IEEE TPAMI*, 2023. Cited on page 18.

[BGH19] Yogesh Balaji, Tom Goldstein, and Judy Hoffman. Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets. *arXiv preprint*, 2019. Cited on page 11.

[BH20] Mohammad Taha Bahadori and David E Heckerman. Debiasing concept-based explanations with causal analysis. *arXiv preprint arXiv:2007.11500*, 2020. Cited on page 18.

[BHA+21] 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*, 2021. Cited on pages 13 and 33.

[BHJ18] Maxime Bucher, Stéphane Herbin, and Frédéric Jurie. Semantic bottleneck for computer vision tasks. *ACCV*, 2018. Cited on pages 18 and 58.

[BLW+20] David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. Rewriting a deep generative model. *The European Conference on Computer Vision (ECCV)*, 2020. Cited on page 17.

[BLZBG16] Sean Bell, C Lawrence Zitnick, Kavita Bala, and Ross Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. Cited on page 92.

[BM02] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *The Journal of Machine Learning Research (JMLR)*, 2002. Cited on pages 4, 12, and 45.

[BMN+19] Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron C. Courville. Systematic generalization: What is required and can it be learned? *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on pages 14 and 22.

[BMN21] Robert Baldock, Hartmut Maennel, and Behnam Neyshabur. Deep learning through the lens of example difficulty. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. Cited on pages 24 and 25.

[BMR+17] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017. Cited on page 10.

[BPF20] Samyadeep Basu, Phil Pope, and Soheil Feizi. Influence functions in deep learning are fragile. *The International Conference on Learning Representations (ICLR)*, 2020. Cited on page 16.

[BRB18] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations*, 2018. Cited on page 10.

[BRK+19] Wieland Brendel, Jonas Rauber, Matthias Kümmerer, Ivan Ustyuzhaninov, and Matthias Bethge. Accurate, reliable and fast robustness evaluation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 10.

[BTVG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *The European Conference on Computer Vision (ECCV)*, 2006. Cited on page 4.

[BUSB13] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. Opensurfaces: A richly annotated catalog of surface appearance. *ACM TOG*, 2013. Cited on page 60.

[BZK+17] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. Cited on pages 5, 17, 67, 71, 77, and 80.

[BZKK20] Philipp Benz, Chaoning Zhang, Adil Karjauv, and In So Kweon. Robustness may be at odds with fairness: An empirical study on class-wise accuracy. *Advances in Neural Information Processing Systems (NeurIPS) Workshop on Pre-registration in Machine Learning*, 2020. Cited on page 3.

[BZS+19] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on page 17.

[CAP+19] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian J Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on pages 22 and 40.

[CAS+20] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020. Cited on page 10.

[CBG+17] Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. *Proc. of the International Conference on Machine Learning (ICML)*, 2017. Cited on page 40.

[CBR20] Zhi Chen, Yijie Bei, and Cynthia Rudin. Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2020. Cited on pages 18 and 84.

[CCG+20] Nick Cammarata, Shan Carter, Gabriel Goh, Chris Olah, Michael Petrov, Ludwig Schubert, Chelsea Voss, Ben Egan, and Swee Kiat Lim. Thread: circuits. *Distill*, 2020. Cited on pages 16 and 17.

[CCMC18] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng Chau. Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector. *ECML PKDD*, 2018. Cited on pages 1 and 10.

[CH20a] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, 2020. Cited on page 10.

[CH20b] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *Proc. of the International Conference on Machine Learning (ICML)*, 2020. Cited on pages 10 and 12.

[CLH17] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017. Cited on page 95.

[CLMM17] Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. Cited on page 25.

[CLT+19] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: deep learning for interpretable image recognition. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on pages 5, 18, 58, and 79.

[CML+14] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. Cited on page 60.

[CMPL+23] Arthur Conmy, Augustine N Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *arXiv preprint arXiv:2304.14997*, 2023. Cited on pages 16 and 17.

[COR+16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. Cited on pages 59 and 80.

[CPK+18] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2018. Cited on page 62.

[CRA20] Eric Chu, Deb Roy, and Jacob Andreas. Are visual explanations useful? a case study in model-in-the-loop prediction. *arXiv:2007.12248*, 2020. Cited on page 16.

[CRK19] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. *Proc. of the International Conference on Machine Learning (ICML)*, 2019. Cited on pages 3, 12, and 40.

[CRS+19] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on pages 11, 22, and 40.

[CTD+23] Nicholas Carlini, Florian Tramer, Krishnamurthy Dj Dvijotham, Leslie Rice, Mingjie Sun, and J Zico Kolter. (certified!!) adversarial robustness for free! In *Proc. of the International Conference on Learning Representations (ICLR)*, 2023. Cited on pages 12 and 13.

[CTOF20] Alvin Chan, Yi Tay, Yew Soon Ong, and Jie Fu. Jacobian adversarially regularized networks for robustness. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on page 12.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 1995. Cited on page 42.

[CW17] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, 2017. Cited on pages 10 and 12.

[CZL+20] Tianlong Chen, Zhenyu Zhang, Sijia Liu, Shiyu Chang, and Zhangyang Wang. Robust overfitting may be mitigated by properly learned smoothening. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on page 13.

[DBAA23] Blaise Delattre, Quentin Barthélemy, Alexandre Araujo, and Alexandre Allauzen. Efficient bound of lipschitz constant for convolutional layers by gram iteration. *Proc. of the International Conference on Machine Learning (ICML)*, 2023. Cited on pages 12 and 40.

[DBC22] Jon Donnelly, Alina Jade Barnett, and Chaofan Chen. Deformable protopnet: An interpretable image classifier using deformable prototypes. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. Cited on page 18.

[DBK+20] 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. *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on page 4.

[DCY+20] Michael Downs, Jonathan L Chu, Yaniv Yacoby, Finale Doshi-Velez, and Weiwei Pan. Cruds: Counterfactual recourse using disentangled subspaces. *ICML Workshop on Human Interpretability*, 2020, 2020. Cited on page 16.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. Cited on pages 4, 26, 27, and 117.

[DEL22] Hadi M Dolatabadi, Sarah Erfani, and Christopher Leckie. l-inf-robustness and beyond: Unleashing efficient adversarial training. *The European Conference on Computer Vision (ECCV)*, 2022. Cited on pages 11, 13, 22, and 23.

[DFY+20] Yinpeng Dong, Qi-An Fu, Xiao Yang, Tianyu Pang, Hang Su, Zihao Xiao, and Jun Zhu. Benchmarking adversarial robustness on image classification. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 321–331, 2020. Cited on page 10.

[dJABV+22] Pau de Jorge Aranda, Adel Bibi, Riccardo Volpi, Amartya Sanyal, Philip Torr, Grégory Rogez, and Puneet Dokania. Make some noise: Reliable and efficient single-step adversarial training. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. Cited on pages 3 and 33.

[DLP+18] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018. Cited on page 10.

[DMBB20] Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. Multi-objective counterfactual explanations. *International Conference on Parallel Problem Solving from Nature*, 2020. Cited on page 16.

[DMP+19] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *USENIX Security Symposium*, pages 321–338, 2019. Cited on page 10.

[DRTAA23] Meghal Dani, Isabel Rio-Torto, Stephan Alaniz, and Zeynep Akata. Devil: Decoding vision features into language. *GCPR*, 2023. Cited on page 17.

[DSLH20] Gavin Weiguang Ding, Yash Sharma, Kry Yik Chau Lui, and Ruitong Huang. Mma training: Direct input space margin maximization through adversarial training. *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on pages 11, 13, 22, and 24.

[DVK17] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv*, 2017. Cited on pages 1 and 14.

[DXY+21] Yinpeng Dong, Ke Xu, Xiao Yang, Tianyu Pang, Zhijie Deng, Hang Su, and Jun Zhu. Exploring memorization in adversarial training. *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on pages 13, 27, and 105.

[EC] European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council. Cited on pages 1 and 16.

[EGSD18] Gamaleldin F Elsayed, Ian Goodfellow, and Jascha Sohl-Dickstein. Adversarial reprogramming of neural networks. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2018. Cited on page 10.

[EIS+19] Logan Engstrom, Andrew Ilyas, Hadi Salman, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019. Cited on pages 10 and 25.

[EKM+18] Gamaleldin Elsayed, Dilip Krishnan, Hossein Mobahi, Kevin Regan, and Samy Bengio. Large margin deep networks for classification. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Cited on page 107.

[ERO20] Patrick Esser, Robin Rombach, and Bjorn Ommer. A disentangling invertible interpretation network for explaining latent representations. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on pages 18 and 77.

[Eur19] European Commission. Ethics guidelines for trustworthy ai. https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai, 2019. Cited on page 2.

[Eur21] European Commission. Proposal for a regulation of the european parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206, 2021. Cited on pages 1 and 2.

[EVGW+a] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. Cited on pages 95 and 99.

[EVGW+b] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. Cited on pages 95 and 99.

[FPB+23] Thomas Fel, Agustin Picard, Louis Bethune, Thibaut Boissin, David Vigouroux, Julien Colin, Rémi Cadène, and Thomas Serre. Craft: Concept recursive activation factorization for explainability. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. Cited on page 17.

[FPV19] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. *The International Conference on Computer Vision (ICCV)*, 2019. Cited on page 15.

[FRD19] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research (JMLR)*, 2019. Cited on page 15.

[FRH+19] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 12.

[FV18] Ruth Fong and Andrea Vedaldi. Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. Cited on pages 5, 17, 74, and 77.

[FXLS20] Yue Fan, Yongqin Xian, Max Maria Losch, and Bernt Schiele. Analyzing the dependency of convnets on spatial information. *DAGM GCPR*, 2020. Cited on pages 7 and 8.

[GBCB16] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016. Cited on page 4.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014. Cited on page 91.

[GDS+19] Sven Gowal, Krishnamurthy Dj Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. Scalable verified training for provably robust image classification. *The International Conference on Computer Vision (ICCV)*, 2019. Cited on pages 3, 12, and 40.

[GGMF17] Abel Gonzalez-Garcia, Davide Modolo, and Vittorio Ferrari. Do semantic parts emerge in convolutional neural networks? *IJCV*, 2017. Cited on page 17.

[GGY+19] Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. Simple black-box adversarial attacks. In *International Conference on Machine Learning*. PMLR, 2019. Cited on page 10.

[GHP07] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. *Technical Report*, 2007. Cited on pages 27 and 32.

[GHvdO+21] Sven Gowal, Po-Sen Huang, Aäron van den Oord, Timothy A. Mann, and Pushmeet Kohli. Self-supervised adversarial robustness for the low-label, high-data regime. *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on page 11.

[GJM+20] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020. Cited on pages 3 and 17.

[GK22] Paul Gavrikov and Janis Keuper. Adversarial robustness through the lens of convolutional filters. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. Cited on page 31.

[GMFKB23] Rohit Gandikota, Joanna Materzynska, Jaden Fiotto-Kaufman, and David Bau. Erasing concepts from diffusion models. *arXiv preprint arXiv:2303.07345*, 2023. Cited on page 17.

[GPSW17] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. *Proc. of the International Conference on Machine Learning (ICML)*, 2017. Cited on page 108.

[GRCvdM18] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering adversarial images using input transformations. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2018. Cited on page 12.

[GRF+20] Yue Gao, Harrison Rosenberg, Kassem Fawaz, Somesh Jha, and Justin Hsu. Analyzing accuracy loss in randomized smoothing defenses. *arXiv.org*, 2020. Cited on page 12.

[GRM+19] 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. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on pages 6 and 17.

[GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *Proc. of the International Conference on Learning Representations (ICLR)*, 2015. Cited on pages 10, 11, 12, 23, 24, 33, and 34.

[GTR+18] Robert Geirhos, Carlos RM Temme, Jonas Rauber, Heiko H Schütt, Matthias Bethge, and Felix A Wichmann. Generalisation in humans and deep neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Cited on page 3.

[Gui22] Riccardo Guidotti. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, 2022. Cited on page 14.

[GWZK19] Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 17.

[GZB+23] Robert Geirhos, Roland S Zimmermann, Blair Bilodeau, Wieland Brendel, and Been Kim. Don't trust your eyes: on the (un) reliability of feature visualizations. *arXiv preprint arXiv:2306.04719*, 2023. Cited on page 17.

[HA17] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. Cited on pages 11, 40, and 106.

[Hay18] Jamie Hayes. On visible adversarial perturbations & digital watermarking. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1597–1604, 2018. Cited on page 10.

[HCC18] Todd Huster, Cho-Yu Jason Chiang, and Ritu Chadha. Limitations of the lipschitz constant as a defense against adversarial examples. *ECML PKDD*, 2018. Cited on page 12.

[HCD12] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing error in object detectors. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – The European Conference on Computer Vision (ECCV) 2012*, pages 340–353, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. Cited on page 100.

[HCLR19] Peter Hase, Chaofan Chen, Oscar Li, and Cynthia Rudin. Interpretable image recognition with hierarchical prototypes. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, 2019. Cited on page 18.

[HEKK19] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 15.

[HIA21] Frederik Hvilshøj, Alexandros Iosifidis, and Ira Assent. Ecinn: Efficient counterfactuals from invertible neural networks. In *Proc. of the British Machine Vision Conference (BMVC)*, 2021. Cited on page 16.

[HKR+20] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, 2020. Cited on pages 9 and 14.

[HLWF24] Kai Hu, Klas Leino, Zifan Wang, and Matt Fredrikson. Effectively leveraging capacity for improved deterministic robustness certification. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2024. Cited on pages 4 and 12.

[HMP+17] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *Proc. of the International Conference on Learning Representations (ICLR)*, 2017. Cited on page 108.

[HSMR23] Robin Hesse, Simone Schaub-Meyer, and Stefan Roth. Funnybirds: A synthetic vision dataset for a part-based analysis of explainable ai methods. *The International Conference on Computer Vision (ICCV)*, 2023. Cited on page 15.

[HSS18] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7132–7141, 2018. Cited on page 92.

[HZB+21] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. Cited on pages 26, 27, and 117.

[HZC+17] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv e-prints*, page arXiv:1704.04861, Apr 2017. Cited on page 98.

[HZG+21] Weizhe Hua, Yichi Zhang, Chuan Guo, Zhiru Zhang, and G Edward Suh. Bullettrain: Accelerating robust neural network training via boundary example mining. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. Cited on pages 3, 11, 13, 22, 23, 24, 25, and 105.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *The International Conference on Computer Vision (ICCV)*, 2015. Cited on page 14.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. Cited on pages 26, 59, and 91.

[HZS+21] Yujia Huang, Huan Zhang, Yuanyuan Shi, J. Zico Kolter, and Anima Anandkumar. Training certifiably robust neural networks with efficient local lipschitz bounds. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. Cited on pages 11, 12, 40, 42, 48, 49, 107, 117, and 154.

[HZW+23] Kai Hu, Andy Zou, Zifan Wang, Klas Leino, and Matt Fredrikson. Unlocking deterministic robustness certification on imagenet. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. Cited on pages 4 and 12.

[IMA+17] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *ArXiv*, abs/1602.07360, 2017. Cited on page 98.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proc. of the International Conference on Machine Learning (ICML)*, 2015. Cited on page 106.

[IST+19] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 17.

[JD20] Matt Jordan and Alexandros G Dimakis. Exactly computing the local lipschitz constant of relu networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on page 12.

[JG18] Daniel Jakubovitz and Raja Giryes. Improving dnn robustness to adversarial attacks using jacobian regularization. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018. Cited on page 12.

[JGB+20] Joel Janai, Fatma Güney, Aseem Behl, Andreas Geiger, et al. Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends® in Computer Graphics and Vision*, 12(1–3):1–308, 2020. Cited on page 14.

[JIDD17] Ajil Jalal, Andrew Ilyas, Constantinos Daskalakis, and Alexandros G Dimakis. The robust manifold defense: Adversarial training using generative models. *arXiv preprint*, 2017. Cited on page 12.

[JKV+19] Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. Towards realistic individual recourse and actionable explanations in black-box decision making systems. *arXiv preprint arXiv:1907.09615*, 2019. Cited on page 16.

[JPK+21] Jongheon Jeong, Sejun Park, Minkyu Kim, Heung-Chang Lee, Do-Guk Kim, and Jinwoo Shin. Smoothmix: Training confidence-calibrated smoothed classifiers for certified robustness. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. Cited on page 12.

[Jr.63] Joe H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 1963. Cited on pages 81 and 82.

[JSK+23] Saachi Jain, Hadi Salman, Alaa Khaddaj, Eric Wong, Sung Min Park, and Aleksander Madry. A data-based perspective on transfer learning. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. Cited on page 13.

[JSZ+15] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2017–2025, 2015. Cited on page 92.

[JSZ+19] Guoqing Jin, Shiwei Shen, Dongming Zhang, Feng Dai, and Yongdong Zhang. Ape-gan: Adversarial perturbation elimination with gan. In *Proc. of the IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019. Cited on page 12.

[KATL19] Pang Wei W Koh, Kai-Siang Ang, Hubert Teo, and Percy S Liang. On the accuracy of influence functions for measuring group effects. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 16.

[KB15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Proc. of the International Conference on Learning Representations (ICLR)*, 2015. Cited on page 154.

[KH+09] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical report*, 2009. Cited on pages 4, 26, 27, and 46.

[KK23] Minchan Kwon and Kangil Kim. Enhancing accuracy and robustness through adversarial training in class incremental continual learning. *arXiv preprint arXiv:2305.13678*, 2023. Cited on page 106.

[KKG18] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing. *arXiv preprint*, 2018. Cited on page 12.

[KKNN22] Ashkan Khakzar, Pedram Khorsandi, Rozhin Nobahari, and Nassir Navab. Do explanations explain? model knows best. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. Cited on page 15.

[KL17] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. *Proc. of the International Conference on Machine Learning (ICML)*, 2017. Cited on page 16.

[KMB+21] Ashkan Khakzar, Sabrina Musatian, Jonas Buchberger, Icxel Valeriano Quiroz, Nikolaus Pinger, Soroosh Baselizadeh, Seong Tae Kim, and Nassir Navab. Towards semantic interpretation of thoracic disease and covid-19 diagnosis models. *MICCAI*, 2021. Cited on page 17.

[KNT+20] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. *Proc. of the International Conference on Machine Learning (ICML)*, 2020. Cited on pages 18, 79, 83, and 108.

[KR19] Michael Kearns and Aaron Roth. *The ethical algorithm: The science of socially aware algorithm design*. Oxford University Press, 2019. Cited on page 9.

[KSA+18] Pieter-Jan Kindermans, Kristof T. Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: Patternnet and patternattribution. *Proc. of the International Conference on Learning Representations (ICLR)*, 2018. Cited on page 15.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. 2012. Cited on page 91.

[KSH22] Shuming Kong, Yanyan Shen, and Linpeng Huang. Resolving training biases via influence-based data relabeling. *The International Conference on Learning Representations (ICLR)*, 2022. Cited on page 16.

[KSL19] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on page 33.

[KSMD16] Pieter-Jan Kindermans, Kristof Schütt, Klaus-Robert Müller, and Sven Dähne. Investigating the influence of noise and distractors on the interpretation of neural networks. *arXiv preprint arXiv:1611.07270*, 2016. Cited on page 14.

[KTSH21] Minseon Kim, Jihoon Tack, Jinwoo Shin, and Sung Ju Hwang. Entropy weighted adversarial training. *The International Conference on Machine Learning (ICML) Workshop on Adversarial Machine Learning*, 2021. Cited on pages 11, 13, 22, and 24.

[KW14] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *Proc. of the International Conference on Learning Representations (ICLR)*, 2014. Cited on page 107.

[KWG+18] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). *Proc. of the International Conference on Machine Learning (ICML)*, 2018. Cited on page 17.

[KZG18] Danny Karmon, Daniel Zoran, and Yoav Goldberg. Lavan: Localized and visible adversarial noise. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 2507–2515. PMLR, 2018. Cited on page 10.

[KZS+22] Maximilian Kaufmann, Yiren Zhao, Ilia Shumailov, Robert Mullins, and Nicolas Papernot. Efficient adversarial training with data pruning. *arXiv preprint*, 2022. Cited on page 11.

[LAG+19] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. *Proc. of the IEEE Symposium on Security and Privacy*, 2019. Cited on pages 3 and 12.

[Lam81] Diane Lambert. Influence functions for testing. *Journal of the American Statistical Association*, 1981. Cited on page 16.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998. Cited on page 4.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. Cited on page 4.

[LCLS17] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2017. Cited on page 10.

[LFS19] Max Losch, Mario Fritz, and Bernt Schiele. Interpretability beyond classification output: Semantic bottleneck networks, 2019. Cited on pages 7, 8, and 18.

[LFS20] Max Losch, Mario Fritz, and Bernt Schiele. Semantic bottlenecks: Quantifying and improving inspectability of deep representations. *DAGM GCPR*, 2020. Cited on pages 7 and 8.

[LFS21] Max Losch, Mario Fritz, and Bernt Schiele. Semantic bottlenecks: Quantifying and improving inspectability of deep representations. *IJCV*, 2021. Cited on pages 7, 8, 79, 80, 81, 82, 90, and 118.

[LHA+19] Qiyang Li, Saminul Haque, Cem Anil, James Lucas, Roger B Grosse, and Jörn-Henrik Jacobsen. Preventing gradient attenuation in lipschitz constrained convolutional networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on pages 4, 11, 12, and 40.

[LHL15] Chunchuan Lyu, Kaizhu Huang, and Hai-Ning Liang. A unified gradient regularization family for adversarial examples. In *IEEE international conference on data mining*, 2015. Cited on page 12.

[Li24] Linyi Li. Sok leaderboard: Certified robustness for deep neural networks. https://sokcertifiedrobustness.github.io/leaderboard, 2024. Cited on page 12.

[Lip18] Zachary C Lipton. The mythos of model interpretability. *Queue*, 2018. Cited on pages 1, 2, 14, and 108.

[LJL+20] Jie Li, Rongrong Ji, Hong Liu, Jianzhuang Liu, Bineng Zhong, Cheng Deng, and Qi Tian. Projection & probability-driven black-box attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. Cited on page 10.

[LKF23] Tobias Lorenz, Marta Kwiatkowska, and Mario Fritz. Certifiers make neural networks vulnerable to availability attacks. *ACM Workshop on Artificial Intelligence and Security*, 2023. Cited on page 13.

[LL17] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. Cited on pages 15 and 108.

[LLCR18] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. Cited on page 18.

[LLP20] Sungyoon Lee, Jaewook Lee, and Saerom Park. Lipschitz-certifiable training with a tight outer bound. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on pages 11, 40, 47, 48, 49, 117, and 154.

[LOS+24] Max Losch, Mohamed Omran, David Stutz, Bernt Schiele, and Mario Fritz. On adversarial training without perturbin all examples. *Proc. of the International Conference on Learning Representations (ICLR)*, 2024. Cited on pages 7 and 8.

[Low99] David G Lowe. Object recognition from local scale-invariant features. *The International Conference on Computer Vision (ICCV)*, 1999. Cited on page 4.

[LPK20] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 2020. Cited on page 14.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015. Cited on page 91.

[LSF21] Cassidy Laidlaw, Sahil Singla, and Soheil Feizi. Perceptual adversarial robustness: Defense against unseen threat models. *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on pages 11 and 37.

[LSFFX10] Li-Jia Li, Hao Su, Li Fei-Fei, and Eric P Xing. Object bank: A high-level image representation for scene classification & semantic feature sparsification. *Advances in Neural Information Processing Systems (NeurIPS)*, 2010. Cited on page 18.

[LSSF23] Max Losch, David Stutz, Bernt Schiele, and Mario Fritz. Certified robust models with slack control and large lipschitz constants. *DAGM GCPR*, 2023. Cited on pages 7, 8, and 39.

[LSY19] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on page 66.

[LWF21] Klas Leino, Zifan Wang, and Matt Fredrikson. Globally-robust neural networks. *Proc. of the International Conference on Machine Learning (ICML)*, 2021. Cited on pages 4, 11, 40, 42, 43, 47, 48, 49, 107, 118, 154, and 156.

[LWW+23] Jinqi Luo, Zhaoning Wang, Chen Henry Wu, Dong Huang, and Fernando De la Torre. Zero-shot model diagnosis. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. Cited on page 17.

[LXL23] Linyi Li, Tao Xie, and Bo Li. Sok: Certified robustness for deep neural networks. *Proc. of the IEEE Symposium on Security and Privacy*, 2023. Cited on pages 3 and 12.

[LY14] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *Technical report*, 2014. Cited on pages 4 and 46.

[MA20] Jesse Mu and Jacob Andreas. Compositional explanations of neurons. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on pages 17 and 77.

[MAKR21] Radek Mackowiak, Lynton Ardizzone, Ullrich Kothe, and Carsten Rother. Generative classifiers as a basis for trustworthy image classification. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. Cited on pages 12, 16, and 107.

[MAT+18] Marius Mosbach, Maksym Andriushchenko, Thomas Trost, Matthias Hein, and Dietrich Klakow. Logit pairing methods can fool gradient-based attacks. *NeurIPS 2018 Workshop on Security in Machine Learning*, 2018. Cited on page 12.

[MCB19] Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl. Quantifying model complexity via functional decomposition for better post-hoc interpretability. *Machine Learning and Knowledge Discovery in Databases: International Workshops of ECML PKDD 2019*, 2019. Cited on page 17.

[MCL+21] Anita Mahinpei, Justin Clark, Isaac Lage, Finale Doshi-Velez, and Weiwei Pan. Promises and pitfalls of black-box concept learning models. *arXiv preprint arXiv:2106.13314*, 2021. Cited on page 18.

[MDAA22] Laurent Meunier, Blaise J Delattre, Alexandre Araujo, and Alexandre Allauzen. A dynamical system perspective for lipschitz neural networks. *Proc. of the International Conference on Machine Learning (ICML)*, 2022. Cited on pages 11, 12, 40, 42, 47, 48, 49, 117, 154, and 156.

[MDFF16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. Cited on page 10.

[MDFFF17] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. Cited on pages 10 and 105.

[MFL+20] Diego Marcos, Ruth Fong, Sylvain Lobry, Rémi Flamary, Nicolas Courty, and Devis Tuia. Contextual semantic interpretability. In *Proc. of the Asian Conference on Computer Vision (ACCV)*, 2020. Cited on page 18.

[MGB18] Konda Reddy Mopuri, Aditya Ganeshan, and R Venkatesh Babu. Generalizable data-free objective for crafting universal adversarial perturbations. 2018. Cited on page 10.

[MHG+14] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2204–2212, 2014. Cited on pages 14 and 92.

[MJ18] David Alvarez Melis and Tommi Jaakkola. Towards robust interpretability with self-explaining neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Cited on page 18.

[MKN+23] Bálint Mucsányi, Michael Kirchhof, Elisa Nguyen, Alexander Rubinstein, and Seong Joon Oh. *Trustworthy Machine Learning*. University of Tübingen, 2023. Cited on page 9.

[MKR21] Aniek F. Markus, Jan A. Kors, and Peter R. Rijnbeek. The role of explainability in creating trustworthy artificial intelligence for health care: A comprehensive survey of the terminology, design choices, and evaluation strategies. *Journal of Biomedical Informatics*, 2021. Cited on page 16.

[MLT19] Diego Marcos, Sylvain Lobry, and Devis Tuia. Semantically interpretable activation maps: what-where-how explanations within cnns. *The International Conference on Computer Vision (ICCV)W*, 2019. Cited on page 18.

[MLT+22] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost Van De Weijer. Class-incremental learning: survey and performance evaluation on image classification. *IEEE TPAMI*, 2022. Cited on page 106.

[MMDF19] Apostolos Modas, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Sparsefool: a few pixels make a big difference. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on page 10.

[MMKI18] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. 2018. Cited on page 12.

[MMS+18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *Proc. of the International Conference on Learning Representations (ICLR)*, 2018. Cited on pages 1, 10, 11, 14, 21, 22, 23, and 40.

[Mol20] Christoph Molnar. *Interpretable machine learning*. Independently published, 2020. Cited on page 14.

[MPT22] Emanuele Marconato, Andrea Passerini, and Stefano Teso. Glancenets: Interpretable, leak-proof concept-based models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. Cited on page 18.

[MRSF23] Mazda Moayeri, Keivan Rezaei, Maziar Sanjabi, and Soheil Feizi. Text-to-concept (and back) via cross-model alignment. *Proc. of the International Conference on Machine Learning (ICML)*, 2023. Cited on page 17.

[MST20] Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. *Proceedings of the 2020 conference on fairness, accountability, and transparency*, 2020. Cited on page 16.

[MWK20] Pratyush Maini, Eric Wong, and Zico Kolter. Adversarial robustness against the union of multiple perturbation models. In *Proc. of the International Conference on Machine Learning (ICML)*, 2020. Cited on page 11.

[MWYA19] Shuntaro Miyazato, Xueting Wang, Toshihiko Yamasaki, and Kiyoharu Aizawa. Reinforcing the robustness of a deep neural network to adversarial examples by using color quantization of training image data. In *Proc. of the IEEE International Conference on Image Processing (ICIP)*, 2019. Cited on page 12.

[NDS+21] Vedant Nanda, Samuel Dooley, Sahil Singla, Soheil Feizi, and John P Dickerson. Fairness through robustness: Investigating robustness disparity in deep learning. *ACM FAccT*, 2021. Cited on page 3.

[NORBK17] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. *The International Conference on Computer Vision (ICCV)*, 2017. Cited on page 62.

[NWC+11] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. *Advances in Neural Information Processing Systems (NeurIPS) Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. Cited on page 27.

[NZ08] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. *ICVGIP*, 2008. Cited on pages 27 and 32.

[OBZ17] Edouard Oyallon, Eugene Belilovsky, and Sergey Zagoruyko. Scaling the scattering transform: Deep hybrid networks. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, pages 5618–5627, 2017. Cited on page 92.

[OCS+20] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. Cited on page 17.

[OS22] Mohamed Omran and Bernt Schiele. Towards systematic robustness for scalable visual recognition. *The International Conference on Machine Learning (ICML) Shift Happens Workshop*, 2022. Cited on pages 3, 22, and 23.

[OSF19] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019. Cited on page 10.

[OSJ+18] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 2018. Cited on page 17.

[OW23] Tuomas Oikarinen and Tsui-Wei Weng. CLIP-dissect: Automatic description of neuron representations in deep vision networks. *Proc. of the International Conference on Learning Representations (ICLR)*, 2023. Cited on page 17.

[PBE+22] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022. Cited on pages 16 and 17.

[PBK20] Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. Learning model-agnostic counterfactual explanations for tabular data. *Proceedings of the web conference 2020*, 2020. Cited on page 16.

[PDS18] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *BMVC*, 2018. Cited on page 15.

[PHA+18] Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Anna Rohrbach, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach. Multimodal explanations: Justifying decisions and pointing to the evidence. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. Cited on page 5.

[PL22] Bernd Prach and Christoph H Lampert. Almost-orthogonal layers for efficient general-purpose lipschitz networks. *The European Conference on Computer Vision (ECCV)*, 2022. Cited on pages 11, 12, 40, 42, 49, 50, 152, and 153.

[PMG+18] Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. Deflecting adversarial attacks with pixel deflection. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. Cited on page 12.

[PMJ+16] 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. Cited on page 10.

[PPZ+20] William Peebles, John Peebles, Jun-Yan Zhu, Alexei Efros, and Antonio Torralba. The hessian penalty: A weak prior for unsupervised disentanglement. *The European Conference on Computer Vision (ECCV)*, 2020. Cited on page 18.

[PRBB21] Maura Pintor, Fabio Roli, Wieland Brendel, and Battista Biggio. Fast minimum-norm adversarial attacks through adaptive norm constraints. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021. Cited on page 10.

[PTSS21] Ameya Patil, Michael Tuttle, Alex Schwing, and Naresh Shanbhag. Robustifying ℓ_∞ adversarial training to the union of perturbation models. *arXiv.org*, 2021. Cited on page 11.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research (JMLR)*, 2011. Cited on page 154.

[RBPAS23] Sukrut Rao, Moritz Böhle, Amin Parchami-Araghi, and Bernt Schiele. Studying how to efficiently and effectively guide models with explanations. *The International Conference on Computer Vision (ICCV)*, 2023. Cited on page 15.

[RBS22] Sukrut Rao, Moritz Böhle, and Bernt Schiele. Towards better understanding attribution methods. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. Cited on page 15.

[RDGF16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. Cited on pages 99 and 100.

[RDV18] Andrew Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proc. of the Conference on Artificial Intelligence (AAAI)*, 2018. Cited on page 12.

[RGC+21] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan Andrei Calian, Florian Stimberg, Olivia Wiles, and Timothy A Mann. Data augmentation can improve robustness. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. Cited on page 13.

[RNR20] Arash Rahnama, Andre T Nguyen, and Edward Raff. Robust design of deep neural networks against adversarial attacks based on lyapunov theory. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Cited on page 12.

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016. Cited on pages 5, 15, and 108.

[RSS20] Sukrut Rao, David Stutz, and Bernt Schiele. Adversarial training against location-optimized adversarial patches. In *Proc. of the European Conference on Computer Vision (ECCV) Workshops*, pages 429–448. Springer, 2020. Cited on page 10.

[RWGM20] Mihaela Rosca, Theophane Weber, Arthur Gretton, and Shakir Mohamed. A case for new neural networks smoothness constraints. *"I Can't Believe It's Not Better!"Advances in Neural Information Processing Systems (NeurIPS) workshop*, 2020. Cited on pages 12 and 106.

[RWK20] Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. *Proc. of the International Conference on Machine Learning (ICML)*, 2020. Cited on pages 12, 13, 25, 27, and 105.

[SBM+16] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *TNNLS*, 2016. Cited on page 15.

[SCD+17] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, Dhruv Batra, et al. Grad-cam: Visual explanations from deep networks via gradient-based localization. *The International Conference on Computer Vision (ICCV)*, 2017. Cited on pages 5 and 14.

[Sch22] Bernhard Schölkopf. Causality for machine learning. In *Probabilistic and Causal Inference: The Works of Judea Pearl*. Independently published, 2022. Cited on page 108.

[SDBR15] J Springenberg, Alexey Dosovitskiy, Thomas Brox, and M Riedmiller. Striving for simplicity: The all convolutional net. In *Proc. of the International Conference on Learning Representations (ICLR) Workshops*, 2015. Cited on page 14.

[SEE+18] Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramer, Atul Prakash, and Tadayoshi Kohno. Physical adversarial examples for object detectors. *USENIX workshop on offensive technologies (WOOT 18)*, 2018. Cited on pages 1 and 10.

[SF19] Suraj Srinivas and François Fleuret. Full-gradient representation for neural network visualization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on pages 14 and 15.

[SFH17] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS) 30*, pages 3856–3866. Curran Associates, Inc., 2017. Cited on page 97.

[SGK17] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *Proc. of the International Conference on Machine Learning (ICML)*, 2017. Cited on pages 14 and 15.

[SHS19] David Stutz, Matthias Hein, and Bernt Schiele. Disentangling adversarial robustness and generalization. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on pages 16, 22, 23, and 24.

[SHS21] David Stutz, Matthias Hein, and Bernt Schiele. Relating adversarially robust generalization to flat minima. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2021. Cited on page 13.

[SHZ+18] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018. Cited on page 98.

[SIE+20] Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on pages 12, 13, 22, and 33.

[ŠK14] Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 2014. Cited on page 15.

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. Cited on page 92.

[SN22] Yoshihide Sawada and Keigo Nakamura. Concept bottleneck model with additional unsupervised concepts. *IEEE Access*, 2022. Cited on page 18.

[SNG+19] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on pages 11 and 13.

[SNX+20] Ali Shafahi, Mahyar Najibi, Zheng Xu, John Dickerson, Larry S Davis, and Tom Goldstein. Universal adversarial training. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. Cited on page 11.

[SRPR20] Saima Sharmin, Nitin Rathi, Priyadarshini Panda, and Kaushik Roy. Inherent adversarial robustness of deep spiking neural networks: Effects of discrete input encoding and non-linear activations. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2020. Cited on page 12.

[SSF19] Rakshith Shetty, Bernt Schiele, and Mario Fritz. Not using the car to see the sidewalk–quantifying and controlling the effects of context in classification and segmentation. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Cited on page 83.

[SSF21] Sahil Singla, Surbhi Singla, and Soheil Feizi. Improved deterministic l2 robustness on cifar-10 and cifar-100. *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on pages 11, 12, 40, 42, 47, 48, 49, 52, 53, 111, 117, 154, and 155.

[SSY+20] Hadi Salman, Mingjie Sun, Greg Yang, Ashish Kapoor, and J Zico Kolter. Denoised smoothing: A provable defense for pretrained classifiers. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on page 12.

[SSZ+20] Ali Shafahi, Parsa Saadatpanah, Chen Zhu, Amin Ghiasi, Christoph Studer, David Jacobs, and Tom Goldstein. Adversarially robust transfer learning. *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on pages 12, 13, 22, and 31.

[STE+21] Shibani Santurkar, Dimitris Tsipras, Mahalaxmi Elango, David Bau, Antonio Torralba, and Aleksander Madry. Editing a classifier by rewriting its prediction rules. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. Cited on page 18.

[STY17] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *Proc. of the International Conference on Machine Learning (ICML)*, 2017. Cited on page 14.

[SVKT+21] Lukas Schott, Julius Von Kügelgen, Frederik Träuble, Peter Vincent Gehler, Chris Russell, Matthias Bethge, Bernhard Schölkopf, Francesco Locatello, and Wieland Brendel. Visual representation learning does not generalize strongly within the same domain. *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on page 3.

[SvKT+22] Lukas Schott, Julius von Kügelgen, Frederik Träuble, Peter V. Gehler, Chris Russell, Matthias Bethge, Bernhard Schölkopf, Francesco Locatello, and Wieland Brendel. Visual representation learning does not generalize strongly within the same domain. *Proc. of the International Conference on Learning Representations (ICLR)*, 2022. Cited on page 14.

[SVZ14] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *Proc. of the International Conference on Learning Representations (ICLR)*, 2014. Cited on pages 5, 14, and 17.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014. Cited on page 91.

[SZC+18] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is robustness the cost of accuracy?–a comprehensive study on the robustness of 18 deep image classification models. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018. Cited on pages 11 and 12.

[SZS+13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv:1312.6199*, 2013. Cited on pages 10, 11, and 21.

[SZS+14] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *Proc. of the International Conference on Learning Representations (ICLR)*, 2014. Cited on pages 1, 2, 4, 11, 22, 42, 46, and 154.

[TB19] Florian Tramer and Dan Boneh. Adversarial training and robustness for multiple perturbations. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019. Cited on page 11.

[THZ20] Kaihua Tang, Jianqiang Huang, and Hanwang Zhang. Long-tailed classification by keeping the good and removing the bad momentum causal effect. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on page 106.

[TK20] Asher Trockman and J Zico Kolter. Orthogonalizing convolutional layers with the cayley transform. *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on pages 4, 11, 12, and 40.

[TKJ+21] Qi Tian, Kun Kuang, Kelu Jiang, Fei Wu, and Yisen Wang. Analysis and applications of class-wise robustness in adversarial training. *SIGKDD*, 2021. Cited on pages 3 and 22.

[TPG+17] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017. Cited on page 10.

[TSC20] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical multi-scale attention for semantic segmentation. *arXiv:2005.10821*, 2020. Cited on pages 59 and 62.

[TSE+19] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on pages 11, 12, 31, 110, 115, and 149.

[TSS18] Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Cited on pages 3, 4, 11, 40, and 152.

[Var19] Kush R Varshney. Trustworthy machine learning and artificial intelligence. *XRDS: Crossroads, The ACM Magazine for Students*, 2019. Cited on pages 9 and 14.

[VLK21] Arnaud Van Looveren and Janis Klaise. Interpretable counterfactual explanations guided by prototypes. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2021. Cited on page 16.

[VS18] Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Cited on pages 4, 11, 12, and 42.

[Wad16] Ketaro Wada. labelme: Image Polygonal Annotation with Python. https://github.com/wkentaro/labelme, 2016. Cited on page 71.

[WLNN23] Bowen Wang, Liangzhi Li, Yuta Nakashima, and Hajime Nagahara. Learning bottleneck concepts in image classification. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. Cited on pages 17 and 18.

[WLS+23] Ningfei Wang, Yunpeng Luo, Takami Sato, Kaidi Xu, and Qi Alfred Chen. Does physical adversarial example really matter to autonomous driving? towards system-level effect of adversarial object evasion attack. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. Cited on pages 1 and 10.

[WM23] Ruigang Wang and Ian Manchester. Direct parameterization of lipschitz-bounded deep networks. *Proc. of the International Conference on Machine Learning (ICML)*, 2023. Cited on pages 11, 12, 40, 42, 47, 48, and 117.

[WMR17] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harvard Journal of Law & Technology*, 2017. Cited on page 16.

[WRK20] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on pages 3, 11, 13, 33, 34, 35, 111, and 148.

[WSMK18] Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J Zico Kolter. Scaling provable adversarial defenses. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Cited on pages 47, 49, 117, and 154.

[WWX+20] Dongxian Wu, Yisen Wang, Shu-Tao Xia, James Bailey, and Xingjun Ma. Skip connections matter: On the transferability of adversarial examples generated with resnets. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on page 10.

[WWZ+18] Siyue Wang, Xiao Wang, Pu Zhao, Wujie Wen, David Kaeli, Peter Chin, and Xue Lin. Defensive dropout for hardening deep neural networks under adversarial attacks. In *Proc. of the International Conference on Computer-Aided Design*, 2018. Cited on page 12.

[WXW20] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on pages 11, 13, 22, and 40.

[WZD+20] Zifeng Wang, Hong Zhu, Zhenhua Dong, Xiuqiang He, and Shao-Lun Huang. Less is better: Unweighted data subsampling via influence function. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. Cited on page 16.

[WZY+20] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on pages 11, 13, 22, 24, and 105.

[XCDX21] Zhikang Xia, Bin Chen, Tao Dai, and Shu-Tao Xia. Class aware robust training. *ICASSP*, 2021. Cited on pages 3 and 22.

[XGD+17] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1492–1500, 2017. Cited on page 93.

[XLHL20] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10687–10698, 2020. Cited on page 11.

[XLL22] Xiaojun Xu, Linyi Li, and Bo Li. Lot: Layer-wise orthogonal training on improving l2 certified robustness. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. Cited on pages 12, 40, and 42.

[XLZ+18] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *The European Conference on Computer Vision (ECCV)*, 2018. Cited on pages 5, 60, 62, 81, 82, 107, and 118.

[XZLL19] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. *Proc. of the International Conference on Learning Representations (ICLR)*, 2019. Cited on page 66.

[YACH21] Fan Yang, Sahan Suresh Alva, Jiahao Chen, and Xia Hu. Model-based counterfactual synthesizer for interpretation. *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021. Cited on page 16.

[YCN+15] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv:1506.06579*, 2015. Cited on page 17.

[YCW20] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. *The European Conference on Computer Vision (ECCV)*, 2020. Cited on pages 59 and 62.

[YKA+19] Chih-Kuan Yeh, Been Kim, Sercan O Arik, Chun-Liang Li, Pradeep Ravikumar, and Tomas Pfister. On concept-based explanations in deep neural networks. *arXiv:1910.07969*, 2019. Cited on page 58.

[YO22] Yutaro Yamada and Mayu Otani. Does robustness on imagenet transfer to downstream tasks? *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. Cited on pages 12, 13, and 22.

[YRZ+20] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Russ R Salakhutdinov, and Kamalika Chaudhuri. A closer look at accuracy vs. robustness. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. Cited on pages 11 and 12.

[ZBL+18] Jianming Zhang, Sarah Adel Bargal, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. *IJCV*, 2018. Cited on page 15.

[ZBOT18] Bolei Zhou, David Bau, Aude Oliva, and Antonio Torralba. Interpreting deep visual representations via network dissection. 2018. Cited on page 70.

[ZCAW17] Luisa M Zintgraf, Taco S Cohen, Tameem Adel, and Max Welling. Visualizing deep neural network decisions: Prediction difference analysis. *arXiv:1702.04595*, 2017. Cited on page 15.

[ZCL+21] Bohang Zhang, Tianle Cai, Zhou Lu, Di He, and Liwei Wang. Towards certifying l-infinity robustness using neural networks with l-inf-dist neurons. *Proc. of the International Conference on Machine Learning (ICML)*, 2021. Cited on pages 11 and 40.

[ZCX+20] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. Cited on pages 3, 12, and 40.

[ZF14] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *The European Conference on Computer Vision (ECCV)*, 2014. Cited on pages 4 and 15.

[ZJHW22] Bohang Zhang, Du Jiang, Di He, and Liwei Wang. Boosting the certified robustness of l-infinity distance nets. *Proc. of the International Conference on Learning Representations (ICLR)*, 2022. Cited on pages 11 and 40.

[ZKH+23] Yifan Zhang, Bingyi Kang, Bryan Hooi, Shuicheng Yan, and Jiashi Feng. Deep long-tailed learning: A survey. *IEEE TPAMI*, 2023. Cited on page 106.

[ZKL+15] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *Proc. of the International Conference on Learning Representations (ICLR)*, 2015. Cited on page 17.

[ZL19] Yuchen Zhang and Percy Liang. Defending against whitebox adversarial attacks via randomized discretization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019. Cited on page 12.

[ZLB+23] Yang Zhang, Yawei Li, Hannah Brown, Mina Rezaei, Bernd Bischl, Philip Torr, Ashkan Khakzar, and Kenji Kawaguchi. Attributionlab: Faithfulness of feature attribution under controllable environments. *arXiv preprint arXiv:2310.06514*, 2023. Cited on pages 15 and 16.

[ZMM+21] Ruihan Zhang, Prashan Madumal, Tim Miller, Krista A Ehinger, and Benjamin IP Rubinstein. Invertible concept-based explanations for cnn models with non-negative concept activation vectors. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. Cited on page 17.

[ZNR17] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. Efficient defenses against adversarial attacks. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, 2017. Cited on page 12.

[ZNWZ18] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8827–8836, 2018. Cited on page 58.

[ZQXW17] Ting Zhang, Guo-Jun Qi, Bin Xiao, and Jingdong Wang. Interleaved group convolutions. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, pages 4373–4382, 2017. Cited on page 93.

[ZSQ+17] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. Cited on pages 59, 62, and 82.

[ZYJ+19] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. *Proc. of the International Conference on Machine Learning (ICML)*, 2019. Cited on pages 11, 12, 22, 23, 24, 33, 34, 36, 40, 48, and 117.

[ZZL+19] Yue Zhao, Hong Zhu, Ruigang Liang, Qintao Shen, Shengzhi Zhang, and Kai Chen. Seeing isn't believing: Towards more robust adversarial attack against real world object detectors. *ACM SIGSAC conference on computer and communications security*, 2019. Cited on page 10.

[ZZLS18] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6848–6856, 2018. Cited on page 93.

[ZZN+21a] Jingfeng Zhang, Jianing Zhu, Gang Niu, Bo Han, Masashi Sugiyama, and Mohan Kankanhalli. Geometry-aware instance-reweighted adversarial training. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on page 11.

[ZZN+21b] Jingfeng Zhang, Jianing Zhu, Gang Niu, Bo Han, Masashi Sugiyama, and Mohan Kankanhalli. Geometry-aware instance-reweighted adversarial training. *Proc. of the International Conference on Learning Representations (ICLR)*, 2021. Cited on page 105.

[ZZP+17] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. Cited on page 60.

[ZZRP19] Michał Zajac, Konrad Zołna, Negar Rostamzadeh, and Pedro O Pinheiro. Adversarial framing for image and video classification. In *Proc. of the Conference on Artificial Intelligence (AAAI)*, volume 33, pages 10077–10078, 2019. Cited on page 10.

Adversarial Training without Perturbing all Examples

A.1 Full training details

Class order. In the following, we list the order of classes ranked by entropy $\overline{\mathcal{H}}_C$ (equation 3.3). CIFAR-10 can be derived from figure 3.8 in the main paper. In figures A.1 and A.2, we provide the list for CIFAR-100 and ImageNet-200. On CIFAR-100, the first and thus hardest classes consist mostly of animate categories like *otter*, *rabbit* and *crocodile*. The easiest on the other hand are inanimate categories, specifically vehicle related classes, e.g. *road*, *motorcycle* or *pickup-truck*. Overall, the animate-inanimate order is similar to CIFAR-10. On ImageNet-200, we observe a very different order. Inanimate categories like *spatula*, *drumstick* or *umbrella* are among the hardest, while animate classes like *monarch (butterfly)*, *flamingo* or *lorikeet* are among the easiest. Named hard classes may be difficult to distinguish due to a frequent presence of people in the images.

otter, lizard, seal, rabbit, mouse, crocodile, lobster, shrew, shark,
woman, beaver, bowl, turtle, squirrel, possum, snail, girl, kangaroo,
ray, forest, caterpillar, man, baby, dinosaur, lamp, elephant, couch, boy,
porcupine, snake, butterfly, leopard, crab, table, mushroom, dolphin,
willow_tree, beetle, spider, clock, fox, sweet_pepper, bee, house,
raccoon, tulip, bridge, bus, rose, tank, whale, train, worm, lion, poppy,
trout, bed, plate, can, telephone, tiger, hamster, aquarium_fish,
maple_tree, orchid, pear, mountain, tractor, oak_tree, rocket, skunk,
cockroach, television, cup, sea, cloud, lawn_mower, castle, bottle,
palm_tree, keyboard, apple, plain, pickup_truck, bicycle, orange, chair,
wardrobe, motorcycle, road

Figure A.1: CIFAR-100 classes ranked by decreasing entropy $\overline{\mathcal{H}}_C$. Animal classes are hardest, inanimate classes easiest.

spatula, shovel, syringe, drumstick, hand blower, lighter, nail, maraca,
barrow, umbrella, bow, quill, iron, stethoscope, soap dispenser, dumbbell,
mask, reel, toaster, ant, walking stick, envelope, candle, sleeping bag,
sandal, tricycle, cowboy boot, cradle, breastplate, bubble, banjo, chest,
cliff, wine bottle, fountain, crayfish, doormat, Chihuahua, chain, apron,
kimono, cockroach, accordion, sewing machine, ocarina, revolver, torch,
piggy bank, goblet, studio couch, wreck, hermit crab, grand piano, beaker,
snail, marimba, sundial, mantis, vulture, sea lion, flagpole, washer,
acoustic guitar, mongoose, grasshopper, Christmas stocking, bikini, corn,
balance beam, fox squirrel, American alligator, academic gown,
feather boa, suspension bridge, stingray, acorn, common iguana, forklift,
parachute, mushroom, hotdog, American black bear, beacon, garbage truck,
cello, pug, bee, banana, volcano, baboon, centipede, golfcart, marmot,
limousine, African chameleon, leafhopper, canoe, wood rabbit, agama,
starfish, lynx, German shepherd, capuchin, balloon, goose,
submarine, golden retriever, mitten, jeep, hummingbird, armadillo,
weevil, porcupine, puck, snowplow, barn, fly, tarantula, Rottweiler,
pool table, red fox, harvestman, pretzel, ballplayer, American egret,
puffer, ladybug, pelican, obelisk, bald eagle, go-kart, bell pepper,
castle, snowmobile, junco, lemon, spider web, lion, water tower,
basketball, guacamole, toucan, tank, jellyfish, viaduct,
robin, ambulance, broccoli, flatworm, pomegranate, bison, sea anemone,
jay, rugby ball, organ, drake, cheeseburger, mosque, koala, garter snake,
African elephant, lycaenid, oystercatcher, box turtle, cabbage butterfly,
steam locomotive, goldfinch, jack-o'-lantern, school bus, lorikeet,
manhole cover, rapeseed, flamingo, yellow lady's slipper, monarch

Figure A.2: ImageNet-200 classes ranked by decreasing entropy $\overline{\mathcal{H}}_C$. In contrast to the order on CIFAR-10 and CIFAR-100, animate classes are generally not the most frequent among the hardest. Instead its mostly inanimate objects.

A.2 CIFAR-100 Super-classes

To supplement section 3.3.2 in chapter 3, we provide the 20 super-classes in CIFAR-100, listed in figure A.3. We used these 20 classes to augment CIFAR-10 to construct multiple CIFAR-10+1 datasets.

aquatic-mammal: ['beaver', 'dolphin', 'otter', 'seal', 'whale']
fish: ['aquarium_fish', 'flatfish', 'ray', 'shark', 'trout']
flower: ['orchid', 'poppy', 'rose', 'sunflower', 'tulip']
container: ['bottle', 'bowl', 'can', 'cup', 'plate']
fruit: ['apple', 'mushroom', 'orange', 'pear', 'sweet_pepper']
device: ['clock', 'keyboard', 'lamp', 'telephone', 'television']
furniture: ['bed', 'chair', 'couch', 'table', 'wardrobe']
insect: ['bee', 'beetle', 'butterfly', 'caterpillar', 'cockroach']
large-carnivore: ['bear', 'leopard', 'lion', 'tiger', 'wolf']
building: ['bridge', 'castle', 'house', 'road', 'skyscraper']
scene: ['cloud', 'forest', 'mountain', 'plain', 'sea']
large-mammal: ['camel', 'cattle', 'chimpanzee', 'elephant', 'kangaroo']
small-mammal: ['fox', 'porcupine', 'possum', 'raccoon', 'skunk']
crustacean: ['crab', 'lobster', 'snail', 'spider', 'worm']
human: ['baby', 'boy', 'girl', 'man', 'woman']
reptile: ['crocodile', 'dinosaur', 'lizard', 'snake', 'turtle']
rodent: ['hamster', 'mouse', 'rabbit', 'shrew', 'squirrel']
tree: ['maple_tree', 'oak_tree', 'palm_tree', 'pine_tree', 'willow_tree']
vehicle: ['bicycle', 'bus', 'motorcycle', 'pickup_truck', 'train']
utility-vehicle: ['lawn_mower', 'rocket', 'streetcar', 'tank', 'tractor']

Figure A.3: 20 CIFAR-100 Superclasses for reference.

A.3 Full results for CSAT

Results for CSAT can be plotted for three different validation subsets: A_{val}, B_{val} and on the whole dataset $\mathcal{D}_{\text{val}}$. For clarity, we only showed robust accuracies on $\mathcal{D}_{\text{val}}$ and B_{val} in the main paper in figure 3.9. Here, we provide all results. That is, in figure A.4, we show robust accuracies in the upper split and clean accuracies in the lower split for all 3 subsets.

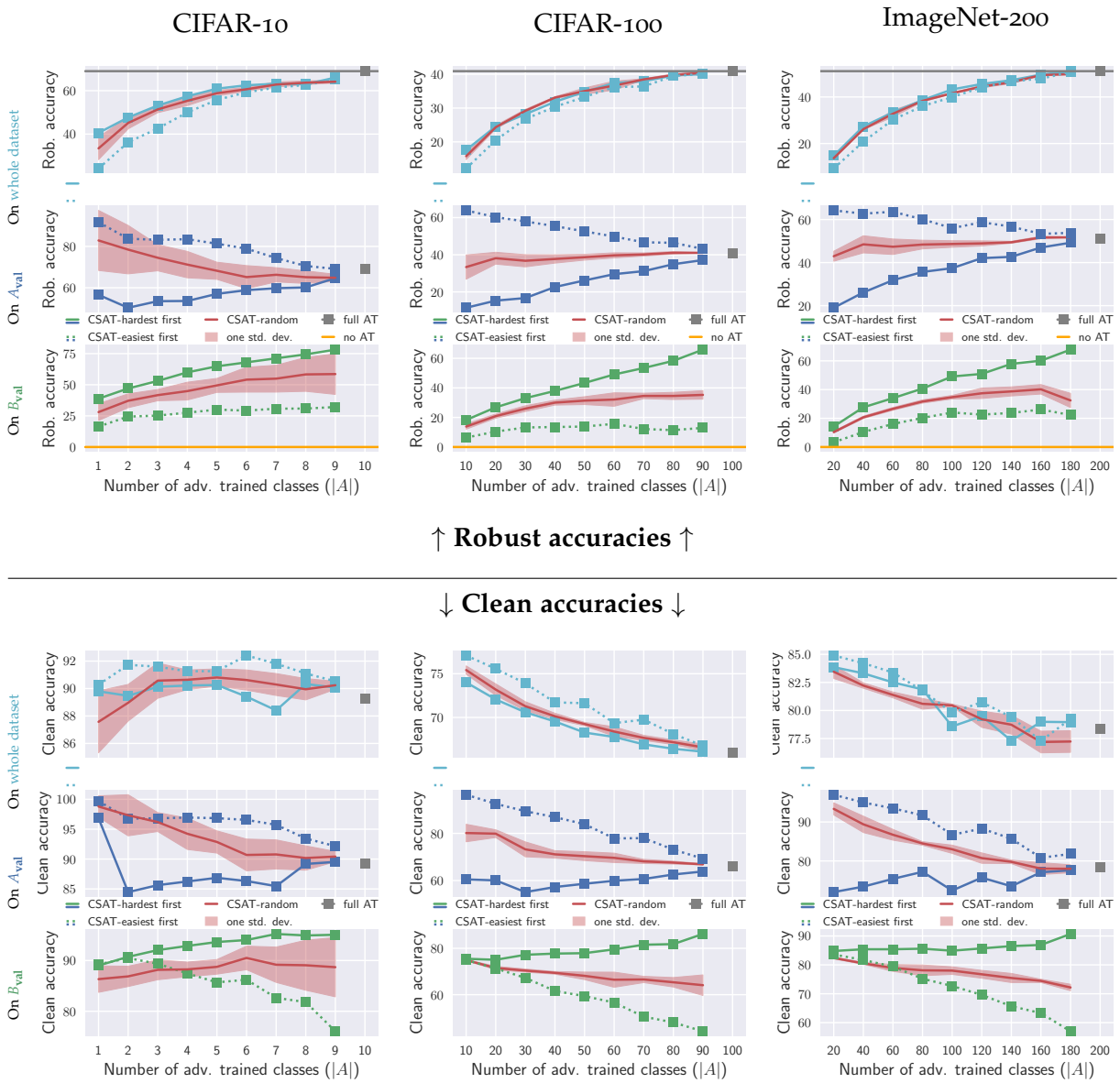


Figure A.4: Full robust (upper split) and clean accuracies (lower split) from CSAT experiments, plotted for the whole dataset, A_{val} and B_{val}. Selecting the hardest classes first (solid lines), clean accuracies and robust accuracies on A_{val} steadily increase, while selecting the easiest in contrast (dotted lines) results in a steady decline. This provides additional support that entropy as metric provides a useful account of difficulty, since easy classes can achieve higher accuracy. Furthermore, we note that clean accuracy on the whole dataset is increasing or mostly stable, while on other datasets it is steadily decreasing. This should be investigated further.

A.4 Clean accuracy independent CSAT results

So far, we presented robust accuracies as well clean accuracies. To highlight, that the hardest examples provide strongest robustness transfer which is independent of clean accuracy gains, we show in figure A.5 an additional robustness metric: *Attack defense rate*. We define *Attack defense rate* as the robust accuracy on purely accurately classified examples. That is, given a dataset, we select examples that are accurately predicted and perform AA. The fraction of robust examples are reported. We observe, that the relationship between hardest, easiest and random examples are retained w.r.t. figure A.4 on all dataset. We thus conclude that the reported gains are independent of clean accuracy.

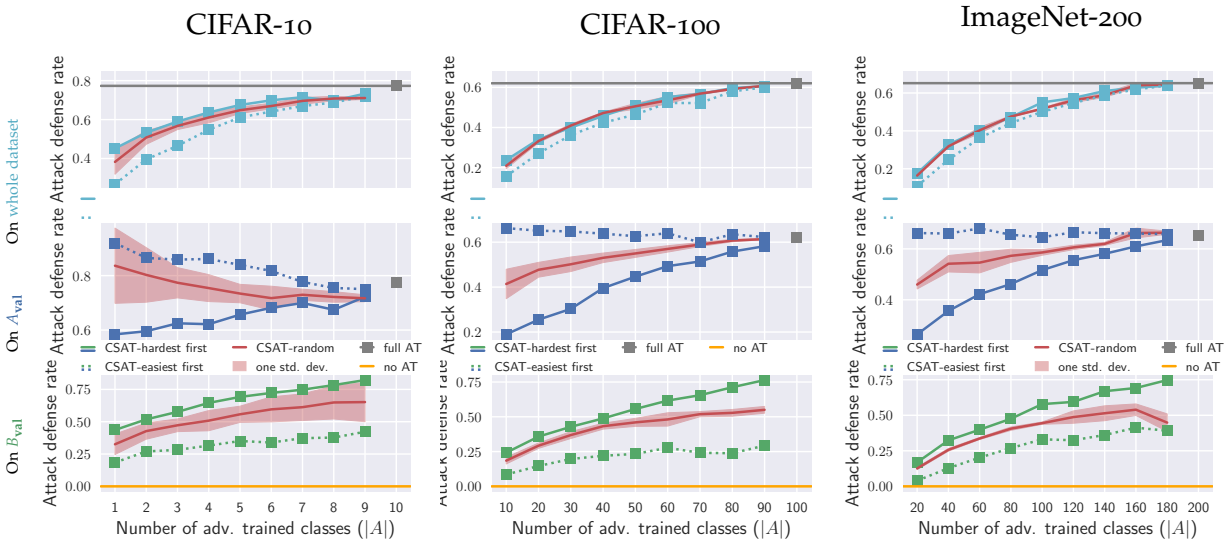


Figure A.5: As complement to the CSAT results in figure A.4, we additionally show the defense rates (for what fraction of accurately classified validation examples does Auto-Attack fail to find adv. examples). As the defense rate shows robustness independent of clean accuracy, we can test whether the robustness gains are in fact based on the improved robustness transfer from hard classes. We see this being the case across all datasets.

A.5 L_∞ results

While the main paper focuses on the L_2 norm, we also provide a corresponding evaluation for L_∞.

ESAT. We evaluate ESAT on CIFAR-10, CIFAR-100 and ImageNet-200 in figure A.6 and S-ESAT on CIFAR-100 $\rightarrow$ CIFAR-10 and ImageNet-200 $\rightarrow$ {Caltech-256, Flowers-102}. For ESAT in figure A.6, we find characteristics similar to the L_2 results in figure 3.10. That is, few examples contribute a large amount of overall robustness to the model and robustness increases quickly with size of A. Noteworthy though, is the low robust accuracy when A contains 10% of data, dropping to 0% on CIFAR-10 and CIFAR-100. We have found this to be similar to catastrophic overfitting described for single-step AT [WRK20]. The effect can be mitigated with more PGD iterations (10) and a smaller step size ($1/255$), but is not resolved entirely. That is,

note that the drop only occurs for the hardest examples, but not for the easiest or a random selection.

S-ESAT. For S-ESAT, we show the same transfer configurations as in the main paper. In figure A.7, we observe similar characteristics as for L_2. That is, fast convergence to baseline AT performance and hard examples providing better robust transfer than easy examples. Also similar to L_2, is the observation that robustness transfer from ImageNet-200 to Flowers-102 is best when using a random selection of examples.

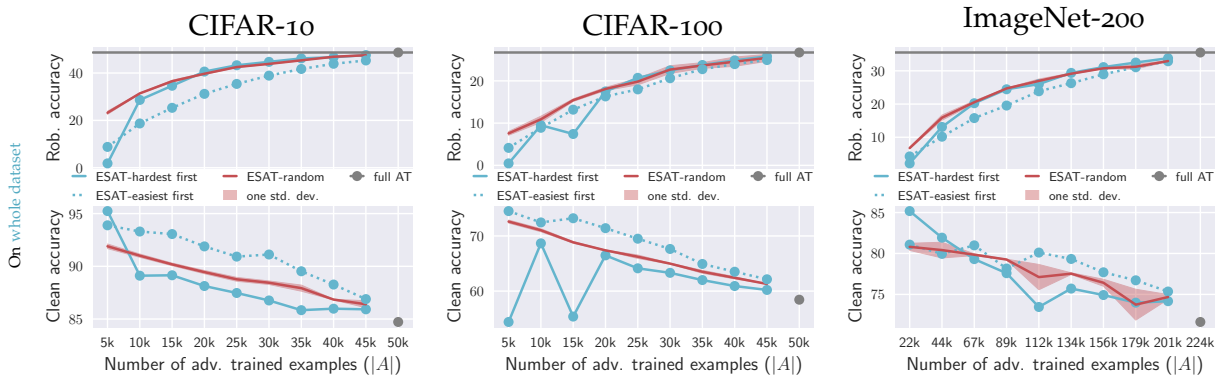


Figure A.6: Example-subset Adversarial Training (ESAT) on CIFAR datasets and ImageNet-200 using L_∞ with norm $\epsilon_\infty = 8/255$, provide quick convergence to a full AT baseline (gray line and dot) with increasing size of A. We report robust accuracy (upper row) and clean accuracy (lower row) and observe similar characteristics as with CSAT (figure 3.9). I.e., selecting the hardest examples first (solid line) provide higher rob. accuracy than easy ones (dashed line), although the gap substantially widens. Random example selection (red) provides competitive performance on average. Across all datasets, we see the common clean accuracy decrease while robust accuracy increases [TSE+19].

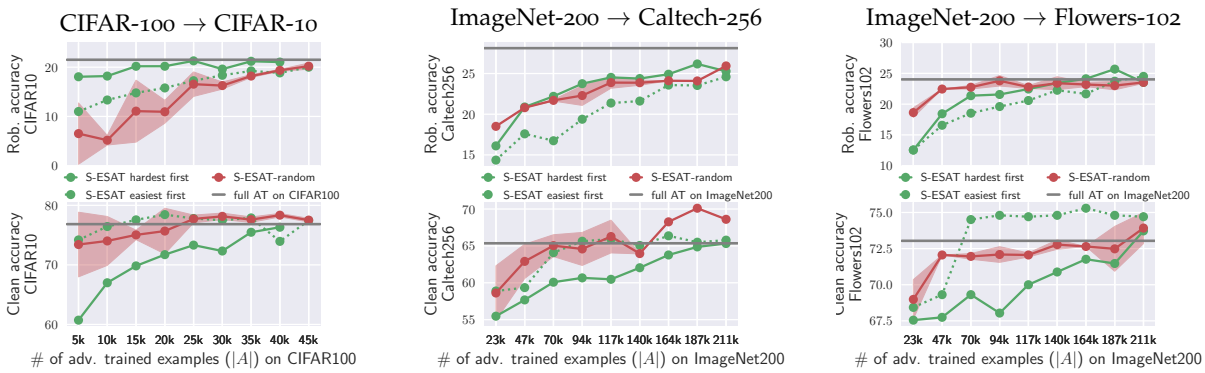


Figure A.7: Transfer from S-ESAT to three different downstream tasks using L_∞ with norm $\epsilon_\infty = 8/255$. S-ESAT is trained on source dataset CIFAR-100 (left) and ImageNet-200 (middle and right). We report robust (top row) and clean (bottom) accuracies for increasing size of A. Similar to our observations for L_2 in figure 3.12, we find that hard examples provide better robustness transfer than easy ones, but random selections (red) achieve competitive performances. Most importantly, "seeing" only few AEs (here 30% of source data) recovers baseline AT performance (gray line).

A.6 Full results for transfer settings

In the main paper, we omitted transfer results to SVHN as well as using S-CSAT. Firstly, we provide the transfer result from CIFAR-100 to SVHN in figure A.8. Robust accuracies are plotted on the upper plot, clean accuracies below. Note that $5k$ examples in A are sufficient to reach baseline AT performance (gray line), while $15k$ provides a substantial improvement in robust accuracy (22% vs 20%). Secondly, transfer results on S-CSAT aligned with the experiments in section 3.3.4 are shown in figure A.9. We observe similar characteristics to the CSAT results in section 3.2.2, i.e. selecting the hardest classes first (solid line) is only advantageous on small A, while generally it draws even with the random baseline (red). Overall, convergence to the full AT baseline is slower than with S-ESAT.

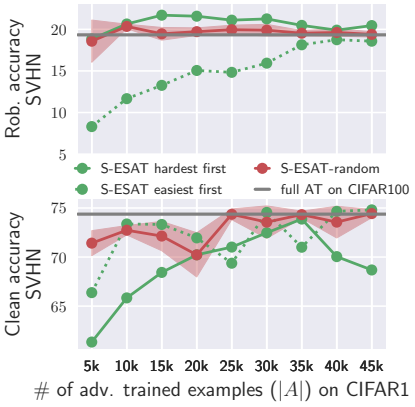


Figure A.8: Robustness transfer from CIFAR-100 to SVHN using S-ESAT

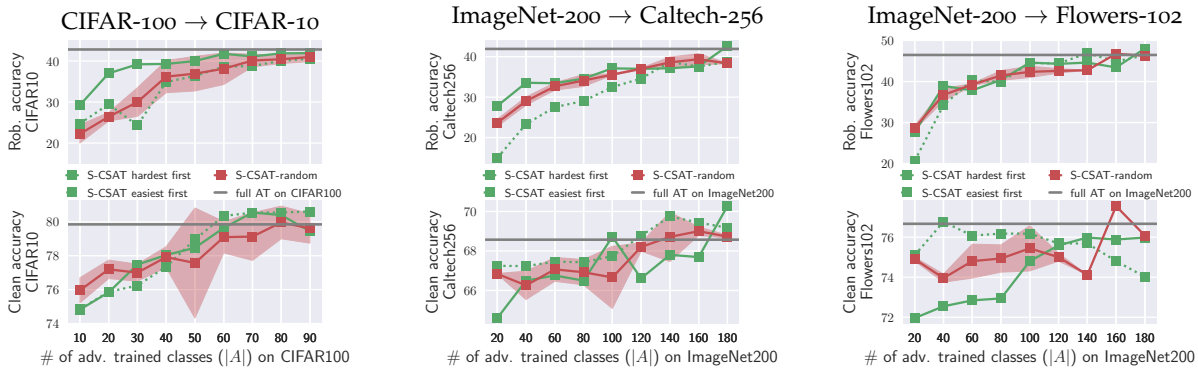


Figure A.9: Transfer from S-CSAT to the same downstream tasks as in figure 3.12. S-CSAT is trained on source dataset CIFAR-100 (left) and ImageNet-200 (middle and right). We report robust (top row) and clean (bottom) accuracies for increasing size of A. We observe similar properties to S-ESAT, yet find convergence to the baseline AT performance to be substantially slower; in line with our discussion on SAT in section 3.2.2.

B Certified Robust Models with Slack Control and Large Lipschitz Constants

B.1 Calibrated Lipschitz-Margin Loss (CLL)

The main paper discussed the calibration of the logistic function for the binary case. To complete this discussion, we provide its generalization to the multiclass case, provide a derivation of σ (equation 5 in the main paper) and a proof for $N = 2$ that our multiclass CLL remains a generalization of the logistic function. Additionally, we delineate CLL from the *AOL*-loss [PL22] which provides similar properties on $K = 1$-constrained models.

Multiclass CLL. It is well known that softmax is a generalization of the logistic distribution. The generalization can be shown easily when the number of logits N is 2. Here, we show, that our calibrated softmax for CLL remains a logistic generalization. We make use of the shift invariance of softmax – any constant can be added or subtracted without changing the output. For simplification, let $\sigma_\epsilon(p)$ simply be σ and $k = 1/K_{1,2}^{(f)} = 1/K_{2,1}^{(f)}$ and $a = \epsilon/\sigma$. The multiclass-CLL for $N = 2$ is then defined as

$$\hat{h}(f(x); y) = \frac{1}{e^{g_y(1,x)} + e^{g_y(2,x)}} \begin{bmatrix} e^{g_y(1,x)} \\ e^{g_y(2,x)} \end{bmatrix} \tag{B.1}$$

$$with\ g_y(i,x) = \begin{cases} a + k/\sigma f_{i,y}(x) & \text{if } i \neq y \\ -a & \text{if } i = y \end{cases}. \tag{B.2}$$

Due to softmaxs shift invariance, we add a to both sides and acquire:

$$g_y(i,x) = \begin{cases} 2a + k/\sigma f_{i,y}(x) & \text{if } i \neq y \\ 0 & \text{if } i = y \end{cases}. \tag{B.3}$$

This formulation encourages a margin width of 2ϵ by adding to all logits where $i \neq y$. We note that a similar (uncalibrated) variant to this formulation has already been used in Lipschitz Margin Training (LMT) [TSS18]. In the following, we develop for both cases of $y \in \{1, 2\}$. Let $y = 2$, then we have $g_{y=2}(2, x) = 0$. Otherwise when $y = 1$, we have $g_{y=1}(1, x) = 0$. It follows:

$$\hat{h}(f(x); y = 2) = \frac{1}{e^{2a+k/\sigma f_{1,2}(x)} + 1} \begin{bmatrix} e^{2a+k/\sigma f_{1,2}(x)} \\ 1 \end{bmatrix} \tag{B.4}$$

$$= \begin{bmatrix} 1 - \frac{1}{1+e^{2a+k/\sigma f_{1,2}(x)}} \\ \frac{1}{1+e^{2a+k/\sigma f_{1,2}(x)}} \end{bmatrix} \tag{B.5}$$

$$\hat{h}(f(x); y = 1) = \frac{1}{1 + e^{2a+k/\sigma f_{2,1}(x)}} \begin{bmatrix} 1 \\ e^{2a+k/\sigma f_{2,1}(x)} \end{bmatrix} \tag{B.6}$$

$$= \begin{bmatrix} \frac{1}{1+e^{2a+k/\sigma f_{2,1}(x)}} \\ 1 - \frac{1}{1+e^{2a+k/\sigma f_{2,1}(x)}} \end{bmatrix}. \tag{B.7}$$

We observe, that the representations for $y = 1$ and $y = 2$ are the same except for the flipped logit pair: $f_{1,2}$, $f_{2,1}$. It is easy to see, that both cases are instances of the logistic distribution when we flip the sign of the exponents:

$$\hat{h}(f(x); y = 2) = \begin{bmatrix} 1 - \frac{1}{1+e^{-(-2a^{-k}/\sigma f_{1,2}(x))}} \\ \frac{1}{1+e^{-(-2a^{-k}/\sigma f_{1,2}(x))}} \end{bmatrix}. \tag{B.8}$$

This derivation shows that our multiclass-CLL is a generalization of the logistic function. Consequently, we can utilize the same method of calibration as for the binary case.

Derivation of σ. We consider the logistic distribution with mean $\mu = 0$:

$$f(x; \sigma) = \frac{e^{-\frac{x}{\sigma}}}{\sigma(1 + e^{-\frac{x}{\sigma}})^2}, \tag{B.9}$$

$$\text{with cdf } \Phi(x; \sigma) = h(x; \sigma) = \frac{1}{1 + e^{-\frac{x}{\sigma}}} \tag{B.10}$$

and want to determine the parameter σ. To this end, let X be the random variable of this distribution and let x_1, x_2 be two values, $x_1 < x_2$. Then the probabilities for these values are given by $P(X < x_1) = p_1$ and $P(X < x_2) = p_2$. The logistic distribution belongs to the location-scale family, which enables reparametrization of X: let X have zero-mean (as assumed) and unit variance. Then we can define a new random variable $Y := \sigma X$, which has the cdf $\Phi_Y(y) = \Phi\left(\frac{y}{\sigma}\right)$. From this follows a set of linear equations:

$$\Phi^{-1}(p_1)\sigma = x_1 \tag{B.11}$$

$$\Phi^{-1}(p_2)\sigma = x_2 \tag{B.12}$$

and the corresponding solution used for calibrating CLL:

$$\sigma = \frac{x_2 - x_1}{\Phi^{-1}(p_2) - \Phi^{-1}(p_1)}. \tag{B.13}$$

B.1.1 Similarity to *AOL* loss.

As mentioned in section 4 (main paper), applying CLL to the *SLL* framework does not improve clean or robust accuracies by the same margins as on the other methods. This is because, the used *AOL* loss [PL22] has similar properties to CLL when used on $K = 1$ constrained models. AOL is defined as: $h\left((-uy + f(x))/t\right) t$, where uy is an offset similar to $y\epsilon$ and t a normalization factor similar to $\sigma_\epsilon(p)K$. Importantly, t is fixed throughout training, while K in CLL is not. This crucial difference is key: K may change in the orders of magnitudes during training and thus must be accounted for appropriately to ensure calibration. CLL is thus a generalization of the *AOL* loss. This has an especially large effect on soft constrained models, as we saw on experiments on two-moons (figures 4 (main paper) and 4.8) and Tiny-ImageNet (table 1, main paper).

B.2 Implementation and Setup

Implementation of our method involves computing the upper Lipschitz bound and training with our calibrated loss. The latter is discussed on different training datasets as listed in the main paper. We list implementation details for two-moons as well as CIFAR-10 and Tiny-ImageNet.

Computing upper Lipschitz bound. We implement the upper bound defined in equation (4.1) by utilizing the product of spectral norms [SZS+14]. We follow a common approach in calculating the spectral norm by performing power iteration and closely follow the strategy in GloRo [LWF21]. That is, initialize the power iterates independently for each layer, update 5 times for each forward pass and reuse the state. [HZS+21] found this to work best over random re-initializations. To ensure the bound is exact during evaluation, we let the power iteration converge once after each training epoch until the error between iterations is smaller than 10^{-9}. During validation and testing, the calculated spectral norms are kept fixed.

Two-moons setup. For our two-moons experiments, we generate the data via the scikit-learn [PVG+11] package, but use uniform sampled noise instead of normal distributed noise. For each point we sample within a circle of radius 0.1, such that the resulting true-margin is exactly $2\epsilon = 0.3$. The training data is generated with 2000 samples.

On this dataset, we train 7-layered MLPs with MinMax activations. All models have two output logits and are trained with variants of the cross-entropy loss. GloRo instruments an additional logit and our method utilizes our calibrated version in equation (4.6). We set p to 10^{-12} in CLL to decrease slack and increase K. We found training to converge faster when annealing p from $p_0 = 10^{-3}$ to $p_T = 10^{-12}$. We utilize polynomial decay

$$p(t) = p_T + (p_0 - p_T) \cdot (1 - {}^{t}/_{T})^{\gamma}, \tag{B.14}$$

where t denotes the training iteration, T the total number of iterations and γ defines the speed of decay. Our γ is set to 50. An overview over given hyperparameters for ours and GloRo is given in table B.1.

Table B.1: Two-moons training configuration for GloRo and CLL (ours).

	architecture	epochs	batchsize	lr	lr decay	p	λ
GloRo	10-20-40-40-20-10-2	1000	100	10^{-3}	to 10^{-4} at epoch 400	N/A	N/A
CLL	10-20-40-40-20-10-2	1000	100	10^{-3}	to 10^{-4} at epoch 400	$10^{-3} \rightarrow 10^{-12}$	10^{-6}

Image dataset setup. We trained models on three different image datasets: CIFAR-10, CIFAR-100 and Tiny-ImageNet. On CIFAR-10, we trained four different architectures: *4C3F*, *6C2F*, *LipConv* and *XL*. On Tiny-ImageNet, we trained on *8C2F*, *LipConv* and *XL*. These architectures follow their definitions in their original papers, see [WSMK18],[LLP20], [SSF21], [LLP20], [MDAA22, AHD+23]. The only difference is in the last layer for K=1 constrained models used with CLL. We use unconstrained final classification layer, instead of last-layer normalization [SSF21]. We found this to provide a slight edge. We list all architectures trained with CLL in table B.2. The syntax in table B.2 is as follows: *c(C,K,S,P)* denotes a convolutional layer with C channels, a kernel size of $K \times K$ with a stride S and padding P. *d(C)* denotes a fully connected layer with C output dimensions. Note that *8C2F* adds no padding when $S = 2$, which is differently to the other architectures.

On CIFAR-10, *LipConv* was trained on the code basis published by Singla et al.[SSF21] extended with our CLL loss. *4C3F*, *6C2F* and *8C2F* were trained on our own code basis. The latter three models, were trained for 800 epochs using a batch size of 128, except *8C2F* for which we used 256. Optimization is done via *Adam* [KB15] using no weight decay and a learning rate of 10^{-3} on CIFAR-10 and $2.5 \cdot 10^{-4}$ on Tiny-ImageNet, which is decayed by factor 0.1 at epoch 400, 600 and 780. In contrast to GloRo, we do not need to anneal the target ϵ to reach best performances. On Tiny-ImageNet, we do anneal the normalization with an additional β factor ${}^{f(x)}/_{(K \cdot \beta)}$, where β is scheduled with the polynomial formulation in equation (B.14). All models utilize the MinMax non-linearity [ALG19] inplace of ReLU. All hyperparameters for *4C3F*, *6C2F*, *8C2F* and *LipConv* are summarized in table B.4.

Table B.2: Architecture configurations on CIFAR-10 (*4C3F*, *6C2F*) and Tiny-ImageNet (*8C2F*).

	Layer 1	2	3	4	5	6	7	8	9	10
4C3F	c(32,3,1,1)	c(32,4,2,1)	c(64,3,1,1)	c(64,4,2,1)	d(512)	d(512)	d(10)			
6C2F	c(32,3,1,1)	c(32,3,1,1)	c(32,4,2,1)	c(64,3,1,1)	c(64,3,1,1)	c(64,4,2,1)	d(512)	d(10)		
8C2F	c(64,3,1,1)	c(64,3,1,1)	c(64,4,2,0)	c(128,3,1,1)	c(128,3,1,1)	c(128,4,2,0)	c(256,3,1,1)	c(256,4,2,0)	d(256)	d(200)

Since *LipConv* had not been evaluated on Tiny-ImageNet in [SSF21], we list performances for a range of γ values (the margin regularization) in table B.3. We identify $\gamma = 0.3$ to work best for *LipConv-10*.

Table B.3: Clean and robust accuracies of *LipConv-10* for different margin influencing parameter γ. The best configuration ($\gamma = 0.3$) is used to compare with CLL in main table 2 (main paper)

γ	Clean (%)	CRA (%)
0.05	32.2	21.1
0.15	32.1	21.4
0.3	32.1	21.5
0.4	32.0	21.1

Data preprocessing. For *4C3F*, *6C2F* and *8C2F* we use the following data preprocessing. All other models use the preprocessing described in their respective papers. We scale the features to the range $[0, 1]$. On CIFAR-10, we use the following **data augmentations** for *4C3F* and *6C2F*:

1. random-crop to 32, with padding of size 4
2. random horizontal flip
3. color jitter on each channel: 0.25
4. random-rotation up to 2 degrees

On Tiny-ImageNet for *8C2F*, we use exactly the same setup, but increase the crop-size to 64 with padding of size 8.

Table B.4: Hyperparameter configuration for all trained models using CLL. logarithmic scheduling as described in [LWF21]. TRADES factor is annealed linearly from 0.5 to 20, with 20 reached at epoch 400.

architecture	dataset	batch size	epochs	ϵ_{train}	initialization	lr	
CLL on	CIFAR-10	128	800	0.15	kaiming	10^{-3}	
	lr-decay	loss	ϵ schedule	power-iter	p (slack)	λ	
4C3F and 6C2F	10^{-4} at 400 10^{-5} at 600 10^{-6} at 780	CLL	fixed	5	0.15	10^{-4}	

architecture	dataset	batch size	epochs	ϵ_{train}	initialization	lr	
CLL on	CIFAR-10	128	200	0.25	kaiming	10^{-3}	
	lr-decay	loss	ϵ schedule	power-iter	p (slack)	λ	LLN
LipConv	10^{-4} at 100 10^{-5} at 150	CLL	fixed	N/A	$5 \cdot 10^{-2}$	10^{-3}	no

architecture	dataset	batch size	epochs	ϵ_{train}	initialization	lr
GloRo on	CIFAR-10	128	800	0.15	kaiming	10^{-3}
	lr-decay	loss	ϵ schedule	power-iter		
4C3F and 6C2F	10^{-4} at 400 10^{-5} at 600 10^{-6} at 780	GloRo	logarithmic	5		

architecture	dataset	batch size	epochs	ϵ_{train}	initialization	lr	
CLL on	Tiny-ImageNet	256	800	0.5 / 1.0	kaiming	$2.5 \cdot 10^{-4}$	
	lr-decay	loss	ϵ schedule	power-iter	p (slack)	λ	K schedule
8C2F	$2.5 \cdot 10^{-5}$ at 400 $2.5 \cdot 10^{-6}$ at 600 $2.5 \cdot 10^{-7}$ at 780	CLL	fixed	5	0.01 / 0.025	10^{-5}	polynomial $0.01 \to 1.0$, $\gamma = 10$

architecture	dataset	batch size	epochs	ϵ_{train}	initialization	lr	
CLL on	CIFAR-10	256	200	0.25	kaiming	10^{-3}	
	lr-schedule	loss	ϵ schedule	power-iter	p (slack)	λ	LLN
(CPL-)XL	triangular[MDAA22]	CLL	fixed	N/A	0.1	10^{-3}	no

architecture	dataset	batch size	epochs	ϵ_{train}	initialization	lr	
CLL on	CIFAR-100	256	200	1.0	kaiming	10^{-3}	
	lr-schedule	loss	ϵ schedule	power-iter	p (slack)	λ	LLN
(CPL-)XL	triangular[MDAA22]	CLL	fixed	N/A	10^{-3}	10^{-3}	no

architecture	dataset	batch size	epochs	ϵ_{train}	initialization	lr	
CLL on	CIFAR-10	256	1000	1.0	kaiming	10^{-2}	
	lr-schedule	loss	ϵ schedule	power-iter	p (slack)	λ	LLN
(SLL-)XL	triangular[AHD+23]	CLL	fixed	N/A	$5 \cdot 10^{-5}$	10^{-3}	no

architecture	dataset	batch size	epochs	ϵ_{train}	initialization	lr	
CLL on	CIFAR-100	256	1000	1.0	kaiming	10^{-2}	
	lr-schedule	loss	ϵ schedule	power-iter	p (slack)	λ	LLN
(SLL-)XL	triangular[AHD+23]	CLL	fixed	N/A	$3 \cdot 10^{-3}$	10^{-3}	no

architecture	dataset	batch size	epochs	ϵ_{train}	initialization	lr	
CLL on	Tiny-ImageNet	128	200	0.15	kaiming	10^{-3}	
	lr-decay	loss	ϵ schedule	power-iter	p (slack)	λ	LLN
LipConv	10^{-4} at 100 10^{-5} at 150	CLL	fixed	N/A	$3 \cdot 10^{-1}$	10^{-5}	no

architecture	dataset	batch size	epochs	ϵ_{train}	initialization	lr
GloRo on	Tiny-ImageNet	256	800	0.15	kaiming	10^{-4}
	lr-decay	loss	ϵ schedule	power-iter		
8C2F	10^{-5} at 400 10^{-6} at 600 10^{-7} at 780	GloRo + TRADES $0.5 \to 20$ at ep400	logarithmic	5		

B.3 Main results under different random seeds

To the main results in table 1 and 2 (main paper), we here report all runs under different seeds when using CLL in table B.5 (CIFAR-10) and table B.7 (Tiny-ImageNet). In the last column, we report additionally, the average value and its standard deviation. We highlight, that the standard deviation is similar to reported values in related work.

Table B.5: CLL results on CIFAR-10 starting from different random seeds.

Model	Metric	Random seed value									Avg
		1	2	3	4	5	6	7	8	9	
CLL@4C3F	Acc	77.3	77.1	77.5	77.3	77.3	77.5	77.3	77.6	77.2	77.3 ± 0.2
	CRA 36/255	61.2	61.5	61.3	61.2	61.6	61.7	61.5	61.5	61.4	61.0 ± 0.2
	CRA 72/255	44.5	44.5	44.1	43.6	44.1	44.5	44.5	44.0	44.2	44.2 ± 0.3
	CRA 108/255	29.3	29.1	29.0	28.7	29.1	29.2	29.3	29.1	29.0	29.1 ± 0.2
	$K^{(f)}$	72.1	72.0	72.0	72.0	72.2	72.2	72.2	71.8	72.1	72.1 ± 0.2
	Tightness	80.6	79.5	81.2	80.8	79.5	79.8	79.8	80.8	79.6	80.2 ± 0.7
CLL@6C2F	Acc	77.5	77.8	77.6	77.7	77.4	77.8	77.5	77.6	77.9	77.6 ± 0.2
	CRA 36/255	61.3	61.1	61.2	61.4	61.1	61.1	61.2	61.7	61.4	61.3 ± 0.2
	CRA 72/255	43.4	43.7	43.6	43.5	43.5	43.3	43.4	43.7	43.6	43.5 ± 0.1
	CRA 108/255	27.6	27.8	27.8	27.7	27.2	27.6	27.6	28.1	28.1	27.7 ± 0.3
	$K^{(f)}$	709.3	704.3	704.6	708.8	722.5	710.2	708.1	707.1	709.1	709.3 ± 5.3
	Tightness	76.6	78.2	77.4	78.7	76.3	77.4	77.5	77.9	78.3	77.6 ± 0.8
CLL@LipConv-20	Acc	77.4	76.8	77.8	77.4	77.3	77.9	77.3	77.1	77.7	77.4 ± 0.3
	CRA 36/255	64.1	64.0	64.2	64.1	64.4	64.4	64.4	63.8	64.4	64.2 ± 0.2
	CRA 72/255	49.5	49.4	49.4	49.7	49.4	49.8	49.7	49.0	49.9	49.5 ± 0.2
	CRA 108/255	37.3	36.3	36.4	36.5	36.5	36.8	37.1	36.5	36.7	36.7 ± 0.3
	$K^{(f)}$	35.6	35.7	35.7	35.7	35.5	36.0	35.8	35.9	35.9	35.8 ± 0.2
	Tightness	85.8	85.6	85.3	86.3	85.1	85.7	85	85.6	86.3	85.6 ± 0.5
CLL@CPL-XL	Acc	78.9	79.0	78.8	78.5	78.9	78.8	78.6	78.7	78.7	78.8 ± 0.1
	CRA 36/255	66.0	66.0	66.0	65.7	65.8	66.0	65.8	65.7	65.9	65.9 ± 0.1
	CRA 72/255	51.6	51.6	51.5	51.7	51.6	51.9	51.5	51.7	51.7	51.6 ± 0.1
	CRA 108/255	38.1	38.2	37.9	37.8	38.0	38.0	38.3	38.2	38.2	38.1 ± 0.2
	$K^{(f)}$	34.6	34.6	34.6	34.6	34.6	34.5	34.6	34.6	34.5	34.5 ± 0.1
	Tightness	80.5	80.6	81.7	80.1	80.3	80.1	79.8	80.2	79.9	80.4 ± 0.5
CLL@SLL-XL	Acc	73.0	72.9	73.1	73.2	72.9	73.1	73.0	72.8	73.1	73.0 ± 0.1
	CRA 36/255	65.5	65.5	65.3	65.7	65.4	65.5	65.4	65.5	65.6	65.5 ± 0.1
	CRA 72/255	57.8	57.6	57.9	57.7	57.5	57.7	57.7	57.9	58.2	57.8 ± 0.2
	CRA 108/255	51.0	50.9	50.7	51.2	50.9	50.8	51.0	51.1	51.0	51.0 ± 0.1
	$K^{(f)}$	58.9	57.3	58.3	60.0	61.1	62.1	60.0	57.7	59.1	59.4 ± 1.6
	Tightness	87.0	89.0	88.0	87.8	87.9	88.2	87.7	88.3	88.0	88.0 ± 0.5

Table B.6: CLL results on CIFAR-100 starting from different random seeds.

Model	Metric	Random seed value									Avg
		1	2	3	4	5	6	7	8	9	
CLL@LipConv-20	Acc	48.8	48.4	48.4	48.0	48.0	48.5	48.4	47.3	47.9	48.2 ± 0.4
	CRA 36/255	35.4	35.2	35.2	35.4	35.0	35.2	35.2	34.3	35.1	35.1 ± 0.3
	CRA 72/255	25.6	25.4	25.5	25.3	25.5	25.6	25.3	24.4	25.2	25.3 ± 0.3
	CRA 108/255	18.6	18.4	18.4	18.3	18.4	18.7	18.1	17.7	18.4	18.3 ± 0.3
	$K^{(f)}$	45.3	45.5	45.6	45.3	45.6	45.5	45.2	45.3	45.7	45.4 ± 0.2
	Tightness	84.4	84.1	85.2	83.2	83.5	85.3	83.8	85.0	84.6	84.3 ± 0.7
CLL@CPL-XL	Acc	48.0	47.8	48.0	48.2	47.9	47.8	47.7	48.2	47.9	47.9 ± 0.2
	CRA 36/255	36.1	36.7	36.4	36.1	36.5	36.1	36.5	36.0	36.3	36.3 ± 0.2
	CRA 72/255	28.0	27.9	28.0	28.9	28.0	28.0	28.0	27.7	28.0	28.1 ± 0.3
	CRA 108/255	21.5	21.5	21.8	21.5	21.5	21.5	21.4	21.4	21.7	21.5 ± 0.1
	$K^{(f)}$	42.0	42.0	42.0	41.9	42.0	42.0	42.0	42.0	42.2	42.0 ± 0.1
	Tightness	79.0	79.4	78.1	78.8	79.0	77.7	79.9	78.6	77.9	78.7 ± 0.7
CLL@SLL-XL	Acc	47.1	46.9	46.9	46.8	47	47.1	46.6	46.6	47.4	46.9 ± 0.3
	CRA 36/255	36.9	36.7	36.3	36.6	36.5	36.6	36.5	36.7	36.7	36.6 ± 0.2
	CRA 72/255	29	29.2	29	28.9	29	29.2	29	29	29	29.0 ± 0.1
	CRA 108/255	23.6	23.6	23.5	23.4	23.8	23.3	23.4	23.3	23.1	23.4 ± 0.2
	$K^{(f)}$	1.2	1.5	1.2	1.2	1.2	1.3	1.2	1.2	1.2	1.3 ± 0.1
	Tightness	79.8	79.5	79.5	80.4	79.1	82.0	78.5	81.5	80.5	80.1 ± 1.1

Table B.7: CLL results on Tiny-ImageNet starting from different random seeds.

Model	Metric	Random seed value									Avg
		1	2	3	4	5	6	7	8	9	
CLL@8C2F $\epsilon = 0.5, p = 0.01$	Acc	39.7	39.8	39.6	39.6	40.0	39.8	39.6	39.9	39.9	39.8 ± 0.1
	CRA 36/255	25.5	25.7	25.7	26.3	26.1	25.9	25.5	26.1	26.3	25.9 ± 0.3
	CRA 72/255	16.3	16.5	16.5	16.7	16.4	16.7	16.5	16.7	16.6	16.5 ± 0.1
	CRA 108/255	10.7	10.7	10.6	10.6	10.9	10.9	10.8	10.7	10.7	10.7 ± 0.1
	$K^{(f)}$	287.7	288.6	287.9	287.7	291.0	291.5	285.5	288.4	286.7	288.3 ± 1.9
	Tightness	63.6	64.8	64.4	64.1	63.9	64.5	64.0	64.9	64.7	64.2 ± 0.5
CLL@8C2F $\epsilon = 1.0, p = 0.025$	Acc	33.6	33.3	33.5	33.0	33.4	33.1	33.6	33.9	33.7	33.5 ± 0.3
	CRA 36/255	25.6	24.9	25.4	25.4	25.2	25.3	25.1	25.3	25.3	25.3 ± 0.2
	CRA 72/255	19.2	18.6	19.0	19.0	18.9	18.9	18.9	19.1	19.0	19.0 ± 0.2
	CRA 108/255	13.9	13.3	13.9	13.6	13.9	14.1	14.1	13.9	13.7	13.8 ± 0.2
	$K^{(f)}$	24.5	24.3	24.5	24.4	24.4	24.4	24.5	24.4	24.3	24.4 ± 0.1
	Tightness	73.5	71.2	72.9	72.1	73.8	71.2	74.1	73.7	72.7	72.8 ± 1.0
CLL@LipConv-10	Acc	32.5	31.8	32.8	32.1	31.6	33.4	32.4	32.6	32.2	32.4 ± 0.5
	CRA 36/255	25.4	25.2	24.8	25.0	24.4	24.9	25.1	24.9	25.0	25.0 ± 0.3
	CRA 72/255	18.9	18.8	18.4	18.4	17.2	18.3	18.8	18.5	18.4	18.4 ± 0.5
	CRA 108/255	14.0	14.1	13.1	13.1	11.6	13.4	13.8	13.5	13.4	13.3 ± 0.7
	$K^{(f)}$	6.8	6.1	6.3	7.3	6.8	6.3	6.9	6.8	6.8	6.7 ± 0.4
	Tightness	84.1	81.3	81.8	84.0	82.3	82.0	81.2	83.8	81.8	82.5 ± 1.2
CLL@LipConv-20	Acc	33.4	32.7	32.5	32.5	32.1	31.7	32.8	33.0	32.6	32.6 ± 0.5
	CRA 36/255	25.9	26.2	26.1	26.2	25.6	26.1	26.4	26.0	25.8	26.0 ± 0.2
	CRA 72/255	19.7	20.4	20.5	20.4	20.1	20.3	20.3	20.3	20.1	20.2 ± 0.2
	CRA 108/255	14.5	16.0	15.6	15.6	15.5	15.8	15.9	15.3	15.5	15.5 ± 0.4
	$K^{(f)}$	4.0	5.2	5.0	5.0	5.0	5.4	4.9	4.6	4.9	4.9 ± 0.4
	Tightness	81.9	84.5	84.4	84.2	82.9	84.5	83.9	87.1	85.1	84.3 ± 1.4