

Towards efficient and reliable neural networks



Pau de Jorge Aranda

St Peter's College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

January 2024

Acknowledgements

First of all, I would like to thank Phil and Greg for making this PhD possible and for their constant support and encouragement. You have been incredibly understanding and always strived to help me grow both professionally and personally. I would also like to thank Puneet and Riccardo, for helping me shape ideas into papers.

This thesis would not have been half as fun without my collaborators; more or less in chronological order: Harkirat, Amartya, Adel and Guillermo. Working with you has been really inspiring and, again, lots of fun. A special thanks to Guillermo, for having been an incredible source of support and friendship, as well as an amazing researcher.

As part of the thesis, I have been fortunate to have a long-term collaboration with Naver Labs Europe, a place full of incredible people. I would like to thank Mert Bulent, Ginger Delmas, Juliette Bertrand, Anilkumar Swamy, Juliette Marrie, Vincent Leroy and Matthieu Armando; for their friendship when I was in Grenoble. And thanks again to Greg and Riccardo for making sure my stays in Grenoble were a great experience.

I would also like to thank the great people in Torr Vision Group, with whom I have enjoyed sunny Oxford together, and have made the cloudy days a bit less cloudy, especially Francesco Pinto, Arnab Ghosh, Jimmy (Yuge) Shy, Tom Joy, Jishnu Mukhoti, Csabi Botos, Kevin (Shuyang) Sun, Will (Chen) Lin, Ameya Prabhu, Francisco Girbal Eiras, Jindong Gu, Alasdair Paren, Ashkan Khakzar and Fabio Pizzati.

Last but not least, I would like to thank my parents, for being there, always. My partner, Jesusa, for showing me the bright side. And to my son, Jan, for smiling back.

Abstract

Deep neural networks have achieved remarkable results in various applications, but when deployed to autonomous agents in the real world several challenges arise. Computational resources are limited and navigating through the real world requires low latency. Moreover, for many safety-critical applications like autonomous driving, deployed models must be reliable. In this thesis, we tackle these challenges with four main contributions:

The first contribution addresses the constraints in computational resources via pruning at initialization. Unlike traditional pruning methods that prune fully trained networks, reducing the number of parameters before training improves memory and computational efficiency both at training and inference time. The proposed pruning mechanism can find trainable sub-networks at much higher sparsity constraints.

The second contribution focuses on the reliability concerns raised by the existence of adversarial examples. We analyse single-step adversarial training methods and propose a simple modification to the Fast Gradient Signed Method that allows us to train robust neural networks more efficiently.

Shifting the focus to reliability under natural domain shifts, the third contribution conducts a comprehensive study evaluating a large number of models in semantic segmentation. We find that while recent models enjoy increased robustness to distribution shifts, uncertainty estimation does not follow suit. We then identify methods from the literature to improve model reliability out of domain.

The fourth contribution focuses on model reliability in the presence of unknown objects. Leveraging recent advancements in text-conditioned image generation, we present a pipeline to automatically add objects into images realistically. We show the resulting images can be used to test and fine-tune anomaly segmentation methods as well as to extend existing datasets to learn new classes.

Contents

Contributions	xi
1 Introduction	1
2 Background and Literature Review	6
2.1 Pruning	6
2.2 Robustness	9
2.2.1 Robustness to natural distribution shifts	9
2.2.2 Adversarial Robustness	10
2.3 Uncertainty estimation	12
3 Progressive Skeletonization: Trimming more fat from a network at initialization	17
3.1 Introduction	19
3.2 Related work	20
3.3 Problem Formulation: Pruning at Initialization	22
3.4 Foresight Connection Sensitivity	24
3.5 Experiments	28
3.5.1 Results on CIFAR-10	28
3.5.2 Results on larger datasets	30
3.5.3 Analysis	32
3.6 Discussion	35

4	Make some noise: Reliable and Efficient Single-Step Adversarial Training	36
4.1	Introduction	38
4.2	Related Work	40
4.3	Preliminaries on Single-Step Adversarial Training	41
4.4	Noise and FGSM	43
4.5	Robustness Evaluations and Comparisons	47
4.5.1	Comparison against Single-Step Methods	47
4.5.2	Comparison against Multi-Step Attacks	48
4.5.3	Analysis of Gradients and Adversarial Perturbations	49
4.6	Increasing Adversarial Perturbations	50
4.7	Additional Ablations	53
4.8	Conclusion	56
5	Reliability in Semantic Segmentation: Are We on the Right Track?	57
5.1	Introduction	59
5.2	Related work	62
5.3	Experimental settings and preliminaries	65
5.3.1	Datasets	65
5.3.2	Architectures	65
5.3.3	Reliability metrics	66
5.4	Are modern segmentors more <i>reliable</i> ?	68
5.4.1	Robustness	69
5.4.2	Calibration error	70
5.4.3	Can we improve out-of-domain calibration?	71
5.4.4	Misclassification detection	77
5.4.5	Out-of-domain detection	78
5.4.6	Can calibration improve misclassification and out-of-domain detection?	79
5.5	Conclusion	79

6	Placing Objects in Context via Inpainting for Out-of-distribution Segmentation	81
6.1	Introduction	83
6.2	Related work	86
6.3	Placing Objects in Context (POC)	88
6.3.1	The POC pipeline	89
6.3.2	Generating datasets with POC	90
6.4	POC for anomaly segmentation	92
6.4.1	Experimental setting	92
6.4.2	Results	94
6.5	POC to learn new classes	96
6.5.1	Experimental settings	96
6.5.2	Results	98
6.6	Conclusion	101
7	Concluding remarks	103
7.1	Summary	103
7.2	Limitations and future work	105
	Bibliography	108
	Appendices	
A	Appendix of Progressive Skeletonization: Trimming more fat from a network at initialization	130
A.1	Pruning implementation details	130
A.2	Additional accuracy-sparsity plots	132
A.3	Further analysis of pruning at initialization	132
A.3.1	Saliency vs T	132
A.3.2	Pruning vs Sparsification	134

A.3.3	Network structure after pruning	134
A.3.4	Evolution of pruning masks	136
A.3.5	Comparison with early pruning	137
A.3.6	Mobilenet experiments	138
A.4	Local optimal masks	139
A.5	Iterative pruning to maximize the gradient norm	141
A.5.1	Maximizing the gradient norm using the gradient approximation	141
A.5.2	Iterative GRASP (Pruning consistency)	142
B	Appendix of Make some noise: Reliable and efficient single-step adversarial training	145
B.1	Additional plots for PreActResNet18 experiments	145
B.2	Experiments with WideResNet28-10 architecture	145
B.3	Increasing adversarial perturbations during training	147
B.4	Longer training schedule	148
B.5	Randomized Alpha	149
B.6	Further visualizations of adversarial perturbations and gradients	151
B.7	Robust evaluations with autoattack	151
B.8	Catastrophic Overfitting outside the ResNet family	153
B.9	Further increasing the attack radii	154
B.10	Testing other norms	154
B.11	Combining N-FGSM with additional regularizers	154
B.12	Imagenet experimental details	157
B.13	Visualization of the loss surface	157
B.14	Magnitude of N-FGSM perturbations	159
B.15	N-FGSM with Gaussian noise	162
B.16	Training with noise augmented samples	163
B.17	Comparison of adversarial training cost	164

B.18 Infrastructure details and GPU hours	165
B.19 Detailed numerical results	166
C Appendix of Reliability in Semantic Segmentation: Are We on the Right Track?	173
C.1 Ablation of calibration metrics	173
C.2 Ablation of number of pixels for calibration	173
C.3 Ablation of confidence score: max probability vs entropy	175
C.4 Ablation number of clusters	175
C.5 Visualization of cluster samples	175
C.6 Theoretical insights on adaptive temperature via clustering	181
C.7 Per-class clustering	185
C.8 Ablation LTS: image vs logits	187
C.9 Ablation of the effect of training settings	189
C.10 Architectures for Universal Image Segmentation	189
C.10.1 Calibration with Mask2Former	191
C.10.2 OOD detection with Mask2Former	191
C.10.3 PRR with Mask2Former	191
C.11 Visualization of uncertainty and temperature maps	192
C.12 Local OOD detection	194
C.13 Additional plots: adaptive temperature via clustering	195
C.14 Additional plots: local temperature scaling	198
C.15 Additional plots: Comparison calibration methods	201
C.16 Additional plots: misclassification detection	204
C.17 Additional plots: out-of-distribution detection	207

D Appendix of Placing Objects in Context via Inpainting for Out-of-distribution Segmentation	209
D.1 POC datasets object list	209
D.2 Anomaly segmentation methods: mIoU on Cityscapes	210
D.3 Adding objects with Instruct Pix2Pix	210
D.4 AUPRC plots	211
D.5 Additional Pascal training samples	213
D.6 Additional anomaly score maps	214
D.7 Additional qualitative results	227
E Catastrophic overfitting can be induced with discriminative non-robust features	234
E.1 Introduction	236
E.2 Preliminaries and related work	238
E.3 Inducing catastrophic overfitting	240
E.4 Networks prefer injected features	243
E.5 Geometry of CO	245
E.5.1 Curvature explosion drives CO	246
E.5.2 Curvature increase is a result of feature interaction	248
E.6 A mechanistic explanation of CO	250
E.7 Analysis of CO prevention methods	251
E.8 Concluding remarks	254
F Appendix of Catastrophic overfitting can be induced with discriminative non-robust features	255
F.1 Analysis of the separability of the injected datasets	255
F.2 Robust classification can require non-linear features	258
F.3 Experimental details	264
F.4 Inducing catastrophic overfitting with other settings	265
F.4.1 Other datasets	265

F.4.2	Other norms	267
F.4.3	Other injected features	267
F.4.4	Other architectures	269
F.5	Performance on $\mathcal{V}$	269
F.6	Learned features at different β	270
F.7	Analysis of curvature in different settings	272
F.8	Adversarial perturbations before and after CO	272
F.9	Further results with N-FGSM, GradAlign and PGD	275
F.10	Further details of low-pass experiment	275

Contributions

This thesis is composed of four main contributions, each corresponding to a publication where I am the sole first author and performed the implementation and ideation of the experiments as well as manuscript writing. The co-authors contributed by discussing results and providing feedback on the research direction and manuscript narrative. The first three publications have been accepted to top conferences while the last contribution is under submission at the time of writing. In particular:

- Chapter 3 is based on de Jorge et al. (2021), which was accepted to *ICLR 2021*.
- Chapter 4 is based on de Jorge et al. (2022), which was accepted to *NeurIPS 2022*.
- Chapter 5 is based on de Jorge et al. (2023), which was accepted to *CVPR 2023*.
- Chapter 6 is under review at the time of writing.

Additionally, Appendix E is based on Ortiz-Jimenez et al. (2023) in which I share first co-authorship with Guillermo Ortiz Jiménez. This work was accepted at TMLR but is not included in the main thesis given that I am not the sole first author.

1

Introduction

Deep neural networks have proven to be very versatile Machine Learning models and are now at the core of many AI applications (LeCun et al., 2015). In recent years, the availability of larger datasets (Russakovsky et al., 2015; Schuhmann et al., 2022) and the increase in compute power Thompson et al. (2020) has allowed the community to scale up model and dataset size, leading to a boost in performance (Sun et al., 2017). However, when deploying deep learning models in real-world applications several issues arise.

Consider the case of a self-driving car, where the perception system needs to accurately estimate the environment in real-time. The low-latency requirements coupled with limited computational resources can render many powerful but heavy models unsuited for the task. On the other hand, deployed cars will be exposed to the open nature of the world, eventually facing previously unseen conditions (*e.g.* a flooded street) or unknown classes (*e.g.* a wild animal crossing the road) whose misclassification could lead to dangerous situations. Unfortunately, despite achieving impressive performance when samples follow the training distribution (usually known as in-distribution or ID samples), current deep learning models are quite brittle to changes in the input

(out-of-distribution or OOD samples), leading to a significantly degraded performance (Recht et al., 2019; Hendrycks and Dietterich, 2019).

One intriguing result which highlights this brittleness is the presence of *adversarial examples*. Szegedy et al. (2014) discovered that deep neural networks would misclassify examples that were almost identical to correctly classified images sampled from the dataset. Although the particular patterns of adversarial examples are unlikely to spontaneously happen in the real world, the existence of such examples poses a security threat when deploying models in sensitive scenarios, especially if models are open-source (with a public training set or model parameters) and indicates a potential issue with the procedures we rely on to design and train neural networks.

On the other hand, considering the fragility of neural networks, it is crucial that models are uncertainty aware for a safe deployment in the real world; *i.e.* that they know what they do not know. However, deep neural networks are known to be overconfident, they predict with high confidence even when they are wrong Guo et al. (2017), and this effect is even more severe under domain shifts Ovadia et al. (2019).

In this thesis, we investigate different avenues to tackle the aforementioned issues by obtaining more efficient and reliable models. In particular, this thesis is divided into four main contributions which are organized as follows:

In Chapter 2 we provide the necessary background and notation as well as a literature review to give context to the different contributions.

In Chapter 3 we present the first contribution focused on memory and computational efficiency via network pruning, *i.e.* removing weights (connections) in a neural network (Mozer and Smolensky, 1989; LeCun et al., 1990; Hassibi et al., 1993). Typically, pruning methods were aimed at removing parameters to obtain a lighter model for inference. Thus, they required training the model with all its connections, followed by (several) iterations of pruning and fine-tuning (Mozer and Smolensky, 1989; LeCun et al., 1990; Hassibi et al., 1993). Recently, Lee et al. (2019) presented a variation of the seminal work by Mozer and Smolensky (1989) and showed it was possible to find

sub-networks at initialization that would reach competitive test accuracy compared to their dense counterparts. Pruning neural networks at initialization is significantly faster than conventional pruning methods. Moreover, its benefits can be leveraged not only for inference but also during training.

We show that previous methods to prune at initialization (Lee et al., 2019; Wang et al., 2020a) have a strong drop in performance when we prune at high sparsity (removing more than 95% of the parameters). Both methods are based on approximations that assume the effect of removing one weight is independent of the other weights and this does not work well for high sparsity levels, where *most* of the parameters are removed. To overcome this, we present a new criteria to estimate the importance of each connection and propose to optimize it iteratively, as we progressively remove more connections. This allows us to obtain trainable sub-networks at much higher sparsity levels.

Chapter 4 is focused on studying computationally efficient ways of training models to be robust to adversarial examples, namely worst-case perturbations. Currently, the most popular methods to obtain adversarially robust models are based on *adversarial training* (AT), which mainly consists of training the model on adversarial examples as a sort of data augmentation (Madry et al., 2018; Zhang et al., 2019; Rice et al., 2020). Unfortunately, these methods increase the training cost by roughly an order of magnitude due to the additional cost of finding worst-case perturbations.

One of the earliest and cheapest methods proposed to find adversarial examples (FGSM) consists in performing a single “gradient ascent” step by following the direction of the sign of the gradient (Goodfellow et al., 2015); however, performing adversarial training with this method was later shown to be ineffective (Madry et al., 2018). Motivated by these findings, several works have focused on making single-step adversarial training effective (Wong et al., 2020; Andriushchenko and Flammarion, 2020; Li et al., 2020; Kim et al., 2021; Park and Lee, 2021; Golgooni et al., 2021), however, the suggested modifications were either only partially effective or required the use of very expensive regularizers. In this second contribution, we present a simple modification of FGSM

that does not need any regularizer and matches the performance of the previous state-of-the-art single-step method while being $3\times$ faster.

Following up on this work, in Appendix E we present a more in-depth analysis of why single-step adversarial training methods fail to lead to robust models. In particular, we discover that we can trigger single-step adversarial training failure by injecting very discriminative but non-robust features. This work is an extension of the work presented in Chapter 4 and is published with shared first authorship, hence, we do not include it in the main body of the thesis.

While adversarial examples can pose a major security threat when designed by a malicious entity, they are unlikely to spontaneously occur in real-world images. On the other hand, natural domain shifts are very common in many applications, such as self-driving cars or medical imaging. The last two contributions are focused on the reliability of neural networks under natural domain shifts (*e.g.* changes in weather) and the presence of unknown classes (*e.g.* a wild animal) respectively.

In Chapter 5 we present a thorough study on the reliability of semantic image segmentation models. The growing popularity of transformer models in computer vision has led to a rapid development of novel architectures (Dosovitskiy et al., 2021; Touvron et al., 2021; Carion et al., 2020; Liu et al., 2021). While in-domain performance follows a constant, upward trend, properties like robustness or uncertainty estimation are often less explored—which are crucial to improve model *reliability*.

We analyze a broad variety of models and observe that recent models are significantly more robust to natural distribution shifts, however, they are not better in terms of uncertainty estimation, which deteriorates significantly out of domain. This motivates us to further explore additional methods that can tackle the issue, and identify promising directions to improve uncertainty estimation under domain shifts.

In Chapter 6 we focus on the task of out-of-distribution (OOD) segmentation, where out-of-distribution refers to any pixel belonging to a semantic class not present in the training set (*e.g.* for a model trained to segment cats and dogs, an elephant would be

OOD). This task is also commonly referred to as anomaly segmentation. In particular, we leverage recent models that perform text-conditioned image generation (Rombach et al., 2022) and open-vocabulary segmentation (Grounded-SAM Contributors, 2023) to automatically add realistic (OOD) objects into images and annotate them.

Capturing and annotating images for semantic segmentation is expensive and the distribution of objects that would be considered anomalous is long-tail. Hence, existing anomaly segmentation datasets capture a limited number of anomalies, lack realism or have strong domain shifts. In contrast, our pipeline can be leveraged to generate anomaly segmentation evaluation sets without the need for further data acquisition or labelling. Moreover, we also show it can be used to boost the performance of recent anomaly segmentation methods (Nayal et al., 2023; Rai et al., 2023; Liu et al., 2023b) as well as learning new semantic classes. For instance, we learn a subset of Pascal classes from scratch by adding them to Cityscapes and achieve similar performance to a Pascal-trained baseline.

All in all, this thesis tackles two main challenges that arise when deploying deep learning models in autonomous agents, namely efficiency and reliability. We address efficiency both by proposing a pruning mechanism to reduce network parameters and by reducing the cost of adversarial training. We address reliability by proposing methods to obtain adversarially robust models as well as understanding the shortcomings of recent models to natural distribution shifts. Additionally, we propose solutions to improve uncertainty estimation under domain shifts and deal with the open-set nature of the world, where novel classes are encountered at test time.

2

Background and Literature Review

The literature review is divided along three main axes: pruning, robustness and uncertainty estimation. The literature on neural network pruning is relevant to the first contribution in Chapter 3, particularly the works that focus on pruning at initialization. The literature on robustness is relevant to the rest of the contributions; adversarial robustness being particularly relevant to Chapter 4 and Appendix E while robustness to natural distribution shifts, together with uncertainty estimation, are more relevant to the last two chapters. With Chapter 5 more focused on global domain shifts (*e.g.* extreme weather conditions) and Chapter 6 focused on unknown objects present at inference time (*e.g.* a dumped piece of furniture in the middle of the street).

2.1 Pruning

Drawing inspiration from neuroscience literature, pruning consists in removing connections (parameters) from a network while preserving (or minimally degrading) its performance on a given task (Hassibi et al., 1993). Generally, given a dataset

$\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ and a network f with parameters $\theta \in \mathbb{R}^m$ the objective of most pruning methods can be written as obtaining:

$$\underset{\theta}{\operatorname{argmin}} \quad \frac{1}{n} \sum_i \mathcal{L}((f(\mathbf{x}_i; \theta)), \mathbf{y}_i) \quad \text{s.t. } \theta \in \mathcal{C}, \quad (2.1)$$

where $\mathcal{L}$ would correspond to the loss function and $\mathcal{C}$ denotes a constraint set for the network parameters. There are two main families of pruning methods: *unstructured pruning* removes individual parameters and usually defines a *sparsity* level (*i.e.* the percentage of removed parameters). In this case, the constrained set is defined as $\mathcal{C} := \{\theta \in \mathbb{R}^m : \|\theta\|_0 < k\}$ only allowing $k < m$ non-zero parameters. If particular patterns in weight pruning are followed (*e.g.* only removing full channels) then it is known as *structured pruning*. Structured pruning has the advantage that it does not require any special library to leverage the sparsity in matrix multiplications, but unstructured pruning can reach higher sparsity rates without losing performance. In this work we focus on unstructured pruning.

In order to optimize the objective in (2.1), different approaches exist that vary in the moment they start imposing sparsity constraints.

Pruning trained models. Most of the pruning works follow the *train – prune – fine-tune* cycle (Mozer and Smolensky, 1989; LeCun et al., 1990; Hassibi et al., 1993; Han et al., 2015; Molchanov et al., 2017; Guo et al., 2016), which requires training the dense network until convergence, followed by multiple iterations of pruning and fine-tuning until a target sparsity is reached. At the end of every training cycle, they devise different criterions to select the most relevant weights (*i.e.* those whose removal would lead to a larger drop in performance). One of the simplest yet better performing criterions is magnitude pruning, which removes the weights with the smallest ℓ_1 norm (Han et al., 2015).

Frankle and Carbin (2019) showed that it was possible to find sparse sub-networks that, when trained from scratch or an early training iteration, were able to match or

even surpass the performance of their dense counterparts. Nevertheless, to find them they use a costly procedure based on iterative magnitude pruning.

Induce sparsity during training. Another popular approach is to induce sparsity during training. This can be achieved by modifying the loss function to consider sparsity as part of the optimization Chauvin (1989); Carreira-Perpiñán and Idelbayev (2018); Louizos et al. (2018) or by dynamically pruning during training Bellec et al. (2018); Mocanu et al. (2018); Mostafa and Wang (2019); Dai et al. (2019); Dettmers and Zettlemoyer (2020); Lin et al. (2020); Kusupati et al. (2020); Evci et al. (2019). These methods are usually cheaper than pruning after training, but they still require to train the network to select the final sparse sub-network.

Pruning at initialization. These methods present a significant leap with respect to other pruning strategies. While traditional pruning mechanisms focused on computational and memory efficiency at inference time, pruning-at-initialization methods bring the same gains also during training. Lee *et al.* presented SNIP Lee et al. (2019), a method to estimate, at initialization, the importance that each weight could have later during training. In a nutshell, SNIP analyses the effect that each weight has on the loss function when perturbed at initialization. Recently, Wang et al. (2020a) proposed GRASP, a different method based on the gradient norm after pruning and showed a significant improvement for higher levels of sparsity. However, neither SNIP nor GRASP perform sufficiently well when a larger fraction of the weights need to be pruned. In Chapter 3, we analyse the approximations made by SNIP and GRASP, and present a more suitable solution to approximate the importance of weights after pruning.

2.2 Robustness

2.2.1 Robustness to natural distribution shifts

Consider a model trained on a *source* dataset $\mathcal{D}_s$, for instance, comprising images of different street scenes acquired in European cities with good weather conditions. Now let us consider a *target* dataset $\mathcal{D}_t$ with similar urban scenes but acquired in India, or in the same European cities but under bad weather conditions. Even if the two datasets contain the same classes (let us assume for the sake of the example that they do) the images from $\mathcal{D}_s$ and $\mathcal{D}_t$ have been sampled from two different distributions: European *vs.* Indian cities, or good *vs.* bad weather conditions. This is commonly referred to as a *distribution shift*—also known as *domain shift*. Samples from the distribution used to sample $\mathcal{D}_s$ are known as *in distribution* (ID) while samples that do not follow the training distribution would be *out of distribution* (OOD). We say a model is *robust* under domain shifts if it does not suffer a significant drop in performance when evaluated out of domain compared to its performance in domain.

The brittleness of neural networks to distribution shifts is a well-known problem, and many sub-formulations exist Taori et al. (2020). Robustness against *natural* shifts focuses on changes that arise from natural (unmanipulated) images, as opposed to *artificially* generated shifts (*e.g.* by manipulating the data with color jittering or noise) (Recht et al., 2019; Hendrycks et al., 2021). If (unlabeled) data from the target distribution $\mathcal{D}_t$ is available different strategies have been devised to improve the performance of neural networks in new domains, known as *domain adaptation*. For instance, adapting (fine-tuning) some components of the model, or modifying the input samples (in $\mathcal{D}_t$) to lie closer to the training domain Csurka (2017); Volpi et al. (2019); Toldo et al. (2020). In Chapter 5 and Chapter 6 we study the performance of models under natural distribution shifts while in Chapter 4 we focus on a specific kind of artificial corruptions called *adversarial perturbations* which we discuss in more detail next.

2.2.2 Adversarial Robustness

Adversarial perturbations. Szegedy et al. (2014) show that state-of-the-art models are vulnerable to very small perturbations (in magnitude), usually imperceptible to the human eye, which, when applied to an image can modify the prediction of the model.

More formally, let $f : \mathcal{X} \rightarrow \mathbb{R}^d$ be an image classification model which maps input images $\mathbf{x} \in \mathcal{X}$ to output probabilities $\mathbf{y} \in \mathbb{R}^d$. Given an image $\mathbf{x}$ and some ball $\mathcal{B}(\mathbf{x})$ centered at $\mathbf{x}$, we define an *adversarial example* as any image $\mathbf{x}' \in \mathcal{X}$ such that:

1. $\mathbf{x}' \in \mathcal{B}(\mathbf{x})$ it is similar to the clean image,
2. $\operatorname{argmax} f(\mathbf{x}) \neq \operatorname{argmax} f(\mathbf{x}')$ modifies the model's prediction.

The difference between the original image $\mathbf{x}$ and the adversarial example $\mathbf{x}'$ is the adversarial perturbation, denoted as $\boldsymbol{\delta} = \mathbf{x}' - \mathbf{x}$. The image $\mathbf{x}$ to which we apply the perturbation is usually referred to as *clean image* to specify that it has not been perturbed. We usually refer to either an adversarial example or its corresponding adversarial perturbation interchangeably.

Adversarial attacks. An adversarial attack is any method used to find adversarial examples. Adversarial attacks can vary according to their definition of $\mathcal{B}(\mathbf{x})$, which formalizes the intuition of "small perturbation". $\mathcal{B}(\mathbf{x})$ is typically defined using an ℓ_p -norm, that is, $\mathcal{B}(\mathbf{x}) = \{\mathbf{x}' \mid \|\mathbf{x}' - \mathbf{x}\|_p \leq \epsilon\}$ for some $\epsilon > 0$. The most popular attacks use the ℓ_∞ norm. We also distinguish the attacks according to the amount of information they have on the model. *White-box* attacks, can use any information about the model, including its parameters, gradients, etc. In contrast, *black-box* attacks only have information about the input and output and treat the model as a black box. While black-box attacks try to mimic an *attacker* that can query the predictions of the model through an API, white-box attacks can be much more powerful since they have access to more information. In this thesis, we focus on white-box attacks.

To evaluate the robustness of a model against adversarial perturbations, we would need to ensure its predictions are constant inside a given ball $\mathcal{B}(\mathbf{x})$ for every sample in the dataset. However, it is computationally unfeasible to empirically measure this for large neural networks. Instead, we usually measure robustness against adversarial attacks, *i.e.* the accuracy on a test set where all samples have been adversarially attacked. This has the caveat that the robustness will depend on the attack, thus, the attack is reported along the accuracy obtained.

Adversarial training. Currently, the most effective methods to obtain empirically robust models are based on adversarial training (Madry et al., 2018; Carlini and Wagner, 2017; Croce and Hein, 2020). Following previous notation, the objective of adversarial training can be written as:

$$\min_{\theta} \sum_{i=1}^n \max_{\delta} \ell(f_{\theta}(\mathbf{x}_i + \delta), \mathbf{y}_i), \quad \text{subject to } \|\delta\|_p \leq \epsilon. \quad (2.2)$$

Where f_{θ} is used to explicitly denote the parametrization of the network with $\theta \in \mathbb{R}^m$. Solving this min-max problem directly is non-trivial, the most common procedure is to solve the outer minimization via some standard optimizer (SGD) while augmenting the training samples with an adversarial attack to approximate the solution to the inner maximization problem.

Catastrophic Overfitting. One of the main drawbacks of adversarial training is that finding the worst-case perturbations at each training iteration is costly. One of the most popular ways of solving the inner maximization in (2.2) is by performing gradient *ascent* (as opposed to traditional gradient *descent*). Goodfellow et al. (2015) introduced the Fast Gradient Sign Method (FGSM) attack, which approximates an adversarial example by moving one step in the direction of the sign of the gradient. Later works introduced an iterative version of FGSM by performing several steps and recomputing the gradient at each iteration (Kurakin et al., 2017b,a). Madry et al. (2018) note applying FGSM iteratively is a particular case of the well-known Projected Gradient Descent method to solve constrained optimization problems. Moreover, they show performing adversarial training with FGSM does not lead to robust models when evaluated against multi-step attacks (*i.e.* several gradient ascent steps). The main

downside of PGD adversarial training is that the computational cost increases linearly with the number of steps.

Recently, Wong et al. (2020) proposed a modification to the FGSM attack that performs a random step prior to computing the gradient (RS-FGSM) and showed that adversarial training with RS-FGSM could yield robustness against PGD attacks with a single gradient ascent step (as opposed to the original FGSM adversarial training). Moreover, they found that plain FGSM adversarial training has a characteristic failure mode they coined *Catastrophic Overfitting* (CO). When training with FGSM adversarial training the model gradually improves its robustness, but when CO occurs, the model suddenly becomes vulnerable to multi-step attacks. Andriushchenko and Flammarion (2020), find that RS-FGSM adversarial training does not prevent catastrophic overfitting in all settings and propose to use a regularizer that, enforces local linearity of the loss. However, although technically still a single-step method, the added regularizer makes the training $3\times$ slower than RS-FGSM. Inspired by Wong et al. (2020), in Chapter 4 we present an in-depth analysis of how can noise help prevent CO and propose a simple modification of RS-FGSM that matches the performance of expensive GradAlign at no extra cost.

2.3 Uncertainty estimation

Uncertainty estimation is a broad term to refer to a model’s capability to quantify if a certain prediction can be trusted. There are different tasks which rely on good uncertainty estimates. The most popular ones are calibration, misclassification detection and out-of-distribution detection.

Calibration. A model is said to be calibrated when the confidence aligns with the accuracy. For instance, if one gathers all predictions made with 80% confidence, then the percentage of correct predictions on that subset of predictions should also be 80%. More formally, Consider a classifier $f : \mathcal{X} \rightarrow \mathcal{Y}$ with $\mathcal{Y} = \{1, 2, 3, \dots, k\}$. We model

our classifier as $f = \operatorname{argmax} \hat{p}(\mathbf{y}|\mathbf{x})$ where $\hat{p}(\mathbf{y}|\mathbf{x})$ are the probabilities estimated by the model for each class given the input. We say that a model is calibrated when

$$P(\mathbf{y} \mid \hat{p}(\mathbf{y}|\mathbf{x}) = \mathbf{p}) = \mathbf{p}, \quad \forall \mathbf{x} \sim \mathcal{D} \quad (2.3)$$

where P is the true probability of the classes and $\mathcal{D}$ the data distribution. Note that in the case of (2.3), the probabilities of all possible classes are taken into account, however, most works actually focus on a simplification of this problem where only the probability of the predicted class is taken into account, that is:

$$P(y = \operatorname{argmax} \hat{p}(\mathbf{y}|\mathbf{x}) \mid \max \hat{p}(\mathbf{y}|\mathbf{x}) = p^*) = p^* \quad \forall \mathbf{x}, y \sim \mathcal{D}. \quad (2.4)$$

The most common metric to measure calibration is the *Expected Calibration Error* (ECE). Which looks at the expected difference between the predicted and actual probabilities:

$$\mathbb{E}_{p^* \in [0,1]} \left[\left| p^* - \mathbb{E}_{(\mathbf{x},y) \sim \mathcal{D}} \left[P(\operatorname{argmax} \hat{p}(\mathbf{y}|\mathbf{x}) = y) \mid \max \hat{p}(\mathbf{y}|\mathbf{x}) = p^* \right] \right| \right]. \quad (2.5)$$

In order to empirically estimate the ECE, it is standard practice to quantize the output probabilities given by the model and compute the mean predicted probability (confidence) and accuracy in each bin. That is,

$$\widehat{\text{ECE}}_f = \sum_{i=1}^m \frac{\#B_i}{n} \left| \text{accuracy}_f(B_i) - \text{confidence}_f(B_i) \right| \quad (2.6)$$

where $\#B_i$ denotes the number of elements in the i^{th} bin, m denotes the number of bins and $n = \sum_{i=1}^m \#B_i$ the total number of elements used to estimate the ECE. We also use f to indicate the dependency of the ECE on the classifier.

Guo *et al.* Guo et al. (2017) have shown that deep learning models tend to be overconfident, that is, they predict with high confidence even when they are wrong. They propose a simple, yet effective solution known as temperature scaling, where the output logits are divided by a temperature parameter before applying the softmax layer. Other calibration methods have been proposed (e.g. Naeini et al. (2015); Kull

et al. (2019); Gupta et al. (2020)), but temperature scaling is still very popular due to its simplicity and the fact that it does not modify the accuracy.

Calibration with temperature scaling is effective in samples which follow the same distribution as the calibration set, however, Ovadia *et al.* Ovadia et al. (2019) have shown that model calibration degrades significantly out of domain. Some methods have been proposed that address this problem by assuming access to unlabeled out of distribution images beforehand Pampari and Ermon (2020); Park et al. (2020); Wang et al. (2020b). On the other hand, Gong *et al.* Gong et al. (2021) have proposed methods that improve out-of-distribution calibration without any data from the target domain. They propose to cluster the calibration set in different “domains” and find a different temperature value for each. At test time, images are calibrated using the temperature from the closest cluster. Specifically for semantic segmentation, Ding *et al.* Ding et al. (2021) propose a content-dependent calibration strategy that learns a small calibration network to predict a temperature for each pixel in an image.

Misclassification. Although calibration is one of the most popular reliability metrics, it has some blind spots. For instance, a random binary classifier that always predicts with 50% confidence will be perfectly calibrated in a balanced set, yet the model is by no means reliable. A desiderata for a reliable model is to assign a larger confidence to correct outputs than to incorrect ones. In the ideal case, if we sorted all predictions from least to most confident, we would have all the incorrect predictions first and, after some confidence threshold, only correct ones. *Misclassification detection* measures how far away are we from such an ideal case. One of the standard metrics are the Rejection-Accuracy curves (Fumera and Roli, 2002; Hendrycks et al., 2021). In a nutshell, we reject samples from least to most confident and compute the accuracy *vs.* amount of rejected samples. However, these are biased in favour of better-performing models (as the base accuracy will already be higher). To avoid that, Malinin *et al.* Malinin et al. (2019) suggest normalizing the area under the curve by that of an oracle that sorts all incorrect samples first and subtracts a baseline score where samples are sorted randomly. The resulting metric, known as the Prediction Rejection Ratio (PRR), will be positive if the model confidence is better than the random baseline and

will have a maximum score of 100% which would correspond to a sorting carried out by an oracle.

Out-of-distribution detection. Another important aspect for reliability is that models are aware of their “domain of expertise” (i.e., their training distribution). When a sample differs significantly from the training distribution, we would expect the model to be more uncertain of its prediction. Similarly to misclassification detection, in out-of-distribution detection we try to separate in-distribution (ID) from out-of-distribution (OOD) samples based on the network confidence. Depending on the problem, the definition of what is considered OOD might vary. For instance, we might consider covariate shifts where the distribution of the input features $p(x)$ changes but the $p(y|x)$ is fixed. These shifts can be due to artificial corruptions (*e.g.* adding noise to the image) or natural shifts (*e.g.* changes in the background, using different cameras, etc).

Another kind of shift might be due the appearance of new classes at test time, this is usually referred to as an open-set scenario. For instance, consider a model that classifies cats and dogs and is presented with a picture of an elephant. In a closed-set scenario, the model would only be able to predict cat or dog (and in both cases would be incorrect). One desiderata for a reliable model would be that the confidence when faced with a sample outside of the classification set would be low (with maximum entropy on the probability distribution).

In the case of image classification, like the previous example, note that the model outputs a single prediction for the whole image. Thus, in this case we would consider the full image to be OOD. However, in dense prediction tasks such as semantic segmentation or object detection, we could consider a part of the image (with an unknown object) to be OOD. For instance, consider a segmentation model trained to distinguish different pieces of furniture (*e.g.* sofa, chair, table, wall, etc.) that is faced with an image that has a cat on the sofa. The pixels belonging to the cat would be considered as OOD pixels. In the case of semantic segmentation, this task is also known as anomaly segmentation.

A popular way to detect OOD inputs is based on model confidence estimated via softmax probabilities (Hendrycks and Gimpel, 2016; Liang et al., 2017). Lakshminarayanan et al. (2017) show that an ensemble of models can capture uncertainty better than a single model and Mukhoti and Gal (2018); Gal and Ghahramani (2016) leverage dropout as an efficient way to obtain an "ensemble" within a single model. For the particular task of anomaly segmentation, which we address in Chapter 6, another body of works reconstructs images with generative models and detects anomalies as discrepancies between original images and their reconstructions (Haldimann et al., 2019; Lis et al., 2019, 2020; Xia et al., 2020). Currently, the most promising methods for anomaly segmentation use OOD data to fine-tune the models. In particular, once the models have been trained on the target segmentation dataset, Chan et al. (2021b); Tian et al. (2022); Grcić et al. (2022) crop OOD objects from other sources (*e.g.* a different segmentation dataset) and *stitch* them into the training images to further fine-tune the model to have low confidence on OOD samples. This has been reported to work well even when the test-time OOD objects are different from the ones used during fine-tuning (Blum et al., 2019).

3

Progressive Skeletonization: Trimming more
fat from a network at initialization

Abstract

Recent studies have shown that skeletonization (pruning parameters) of networks at initialization provides all the practical benefits of sparsity both at inference and training time, while only marginally degrading their performance. However, we observe that beyond a certain level of sparsity (approx 95%), these approaches fail to preserve the network performance, and to our surprise, in many cases perform even worse than trivial random pruning. To this end, we propose an objective to find a skeletonized network with maximum foresight connection sensitivity (FORCE) whereby the trainability, in terms of connection sensitivity, of a pruned network is taken into consideration. We then propose two approximate procedures to maximize our objective (1) Iterative SNIP: allows parameters that were unimportant at earlier stages of skeletonization to become important at later stages; and (2) FORCE: iterative process that allows exploration by allowing already pruned parameters to resurrect at later stages of skeletonization. Empirical analysis on a large suite of experiments show that our approach, while providing at least as good a performance as other recent approaches on moderate pruning levels, provide remarkably improved performance on higher pruning levels (could remove up to 99.5% parameters while keeping the networks trainable).

3.1 Introduction

The majority of pruning algorithms for Deep Neural Networks require training dense models and often fine-tuning sparse sub-networks in order to obtain their pruned counterparts. In Frankle and Carbin (2019), the authors provide empirical evidence to support the hypothesis that there exist sparse sub-networks that can be trained from scratch to achieve similar performance as the dense ones. However, their method to find such sub-networks requires training the full-sized model and intermediate sub-networks, making the process much more expensive.

Recently, Lee et al. (2019) presented SNIP. Building upon almost a three decades old saliency criterion for pruning trained models (Mozer and Smolensky, 1989), they are able to predict, at initialization, the importance each weight will have later in training. Pruning at initialization methods are much cheaper than conventional pruning methods. Moreover, while traditional pruning methods can help accelerate inference tasks, pruning at initialization may go one step further and provide the same benefits at train time Elsen et al. (2019).

Wang et al. (2020a) (GRASP) noted that after applying the pruning mask, gradients are modified due to non-trivial interactions between weights. Thus, maximizing SNIP criterion before pruning might be sub-optimal.

They present an approximation to maximize the gradient norm after pruning, where they treat pruning as a perturbation on the weight matrix and use the first order Taylor’s approximation. While they show improved performance, their approximation involves computing a Hessian-vector product which is expensive both in terms of memory and computation.

We argue that both SNIP and GRASP approximations of the gradients after pruning do not hold for high pruning levels, where a large portion of the weights are removed at once. In this work, while we rely on the saliency criteria introduced by Mozer and Smolensky (1989), we optimize what this saliency would be *after* pruning, rather than *before*. Hence, we name our criteria *Foresight Connection sEnsitivity* (FORCE).

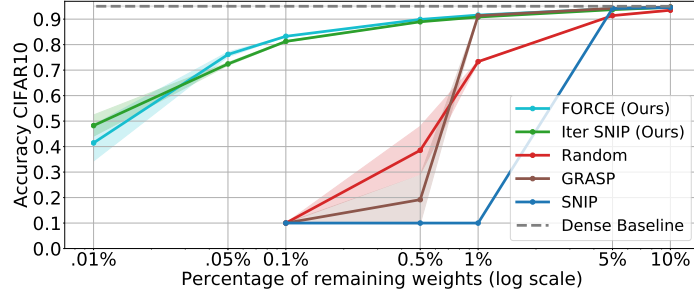


Figure 3.1: Test accuracies on CIFAR-10 (ResNet50) for different pruning methods. Each point is the average over 3 runs of prune-train-test. The shaded areas denote the standard deviation of the runs (too small to be visible in some cases). Random corresponds to removing connections uniformly.

We introduce two approximate procedures to progressively optimize our objective. The first, which turns out to be equivalent to applying SNIP iteratively, removes a small fraction of weights at each step and re-computes the gradients after each pruning round. This allows to take into account the intricate interactions between weights, re-adjusting the importance of connections at each step. The second procedure, which we name FORCE, is also iterative in nature, but contrary to the first, it allows pruned parameters to resurrect. Hence, it supports exploration, which otherwise is not possible in the case of iterative SNIP. Moreover, one-shot SNIP can be viewed as a particular case of using only one iteration. Empirically, we find that both SNIP and GRASP have a sharp drop in performance when targeting higher pruning levels. Surprisingly, they perform even worse than random pruning as can be seen in Fig 3.1. In contrast, our proposed pruning procedures prove to be significantly more robust on a wide range of pruning levels.

3.2 Related work

Pruning trained models Most of the pruning works follow the *train – prune – fine-tune* cycle (Mozier and Smolensky, 1989; LeCun et al., 1990; Hassibi et al., 1993; Han et al., 2015; Molchanov et al., 2017; Guo et al., 2016), which requires training the dense network until convergence, followed by multiple iterations of pruning and fine-tuning until a target sparsity is reached. Particularly, Molchanov et al. (2017) present a criterion very similar to Mozier and Smolensky (1989) and therefore similar

to Lee et al. (2019) and our FORCE, but they focus on pruning whole neurons, and involve training rounds while pruning. Frankle and Carbin (2019) and Frankle et al. (2020) showed that it was possible to find sparse sub-networks that, when trained from scratch or an early training iteration, were able to match or even surpass the performance of their dense counterparts. Nevertheless, to find them they use a costly procedure based on Han et al. (2015). All these methods rely on having a trained network, thus, they are not applicable before training. In contrast, our algorithm is able to find a trainable sub-network with randomly initialized weights. Making the overall pruning cost much cheaper and presenting an opportunity to leverage the sparsity during training as well.

Induce sparsity during training Another popular approach has been to induce sparsity during training. This can be achieved by modifying the loss function to consider sparsity as part of the optimization (Chauvin, 1989; Carreira-Perpiñán and Idelbayev, 2018; Louizos et al., 2018) or by dynamically pruning during training (Bellec et al., 2018; Mocanu et al., 2018; Mostafa and Wang, 2019; Dai et al., 2019; Dettmers and Zettlemoyer, 2020; Lin et al., 2020; Kusupati et al., 2020; Evci et al., 2019). These methods are usually cheaper than pruning after training, but they still need to train the network to select the final sparse sub-network. We focus on finding sparse sub-networks before any weight update, which is not directly comparable.

Pruning at initialization These methods present a significant leap with respect to other pruning methods. While traditional pruning mechanisms focused on bringing speed-up and memory reduction at inference time, pruning at initialization methods bring the same gains both at training and inference time. Moreover, they can be seen as a form of Neural Architecture Search (Zoph and Le, 2016) to find more *efficient* network topologies. Thus, they have both a theoretical and practical interest. Lee et al. (2019) presented SNIP, a method to estimate, at initialization, the importance that each weight could have later during training. SNIP analyses the effect of each weight on the loss function when perturbed at initialization. In Lee et al. (2020), the authors studied pruning at initialization from a signal propagation perspective,

focusing on the initialization scheme. Recently, Wang et al. (2020a) proposed GRASP, a different method based on the gradient norm after pruning and showed a significant improvement for higher levels of sparsity. However, neither SNIP nor GRASP perform sufficiently well when larger compressions and speed-ups are required and a larger fraction of the weights need to be pruned. In this paper, we analyse the approximations made by SNIP and GRASP, and present a more suitable solution to maximize the saliency after pruning.

3.3 Problem Formulation: Pruning at Initialization

Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, the training of a neural network f parameterized by $\boldsymbol{\theta} \in \mathbb{R}^m$ can be written as minimizing the following empirical risk:

$$\operatorname{argmin}_{\boldsymbol{\theta}} \frac{1}{n} \sum_i \mathcal{L}((f(\mathbf{x}_i; \boldsymbol{\theta})), \mathbf{y}_i) \quad \text{s.t. } \boldsymbol{\theta} \in \mathcal{C}, \quad (3.1)$$

where $\mathcal{L}$ and $\mathcal{C}$ denote the loss function and the constraint set, respectively. Unconstrained (standard) training corresponds to $\mathcal{C} = \mathbb{R}^m$.

Assuming we have access to the gradients (batch-wise) of the empirical risk, an optimization algorithm (*e.g.* SGD) is generally used to optimize the above objective, that, during the optimization process, produces a sequence of iterates $\{\boldsymbol{\theta}_i\}_{i=0}^T$, where $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_T$ denote the initial and the final (optimal) parameters, respectively. Given a target sparsity level of $k < m$, the *general* parameter pruning problem involves $\mathcal{C}$ with a constraint $\|\boldsymbol{\theta}_T\|_0 \leq k$, *i.e.*, the final optimal iterate must have a maximum of k non-zero elements. Note that there is no such constraint with the intermediate iterates.

Pruning at initialization, the main focus of this work, adds further restrictions to the above mentioned formulation by constraining all the iterates to lie in a *fixed* subspace

of $\mathcal{C}$. Precisely, the constraints are to find an initialization $\boldsymbol{\theta}_0$ such that $\|\boldsymbol{\theta}_0\|_0 \leq k$ ¹, and the intermediate iterates are $\boldsymbol{\theta}_i \in \bar{\mathcal{C}} \subset \mathcal{C}$, $\forall i \in \{1, \dots, T\}$, where $\bar{\mathcal{C}}$ is the subspace of $\mathbb{R}^m$ spanned by the natural basis vectors $\{\mathbf{e}_j\}_{j \in \text{supp}(\boldsymbol{\theta}_0)}$. Here, $\text{supp}(\boldsymbol{\theta}_0)$ denotes the support of $\boldsymbol{\theta}_0$, *i.e.*, the set of indices with non-zero entries. The first condition defines the sub-network at initialization with k parameters, and the second fixes its topology throughout the training process. Since there are $\binom{m}{k}$ such possible sub-spaces, exhaustive search to find the optimal sub-space to optimize (3.1) is impractical as it would require training $\binom{m}{k}$ neural networks. Below we discuss two recent approaches that circumvent this problem by maximizing a hand-designed data-dependent objective function. These objectives are tailored to preserve some relationships between the parameters, the loss, and the dataset, that *might* be sufficient to obtain a reliable $\boldsymbol{\theta}_0$. For the ease of notation, we will use $\boldsymbol{\theta}$ to denote the dense initialization.

SNIP Lee et al. (2019) present a method based on the saliency criterion from Mozer and Smolensky (1989). They add a key insight and show this criteria works surprisingly well to predict, at initialization, the importance each connection will have during training. The idea is to preserve the parameters that will have maximum impact on the loss when perturbed. Let $\mathbf{c} \in \{0, 1\}^m$ be a binary vector, and $\odot$ the Hadamard product. Then, the *connection sensitivity* in SNIP is computed as:

$$\mathbf{g}(\boldsymbol{\theta}) := \left. \frac{\partial \mathcal{L}(\boldsymbol{\theta} \odot \mathbf{c})}{\partial \mathbf{c}} \right|_{\mathbf{c}=\mathbf{1}} = \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \odot \boldsymbol{\theta}. \quad (3.2)$$

Once $\mathbf{g}(\boldsymbol{\theta})$ is obtained, the parameters corresponding to the top- k values of $|\mathbf{g}(\boldsymbol{\theta})_i|$ are then kept. Intuitively, SNIP favors those weights that are far from the origin *and* provide high gradients (irrespective of the direction). We note that SNIP objective can be written as the following problem:

$$\max_{\mathbf{c}} S(\boldsymbol{\theta}, \mathbf{c}) := \sum_{i \in \text{supp}(\mathbf{c})} |\theta_i \nabla \mathcal{L}(\boldsymbol{\theta})_i| \quad \text{s.t. } \mathbf{c} \in \{0, 1\}^m, \|\mathbf{c}\|_0 = k. \quad (3.3)$$

It is trivial to note that the optimal solution to the above problem can be obtained by selecting the indices corresponding to the top- k values of $|\theta_i \nabla \mathcal{L}(\boldsymbol{\theta})_i|$.

¹In practice, as will be done in this work as well, a subset of a *given* dense initialization is found using some saliency criterion (will be discussed soon), however, note that our problem statement is more general than that.

GRASP Wang et al. (2020a) note that the SNIP saliency is measuring the *connection sensitivity* of the weights *before* pruning, however, it is likely to change after pruning. Moreover, they argue that, at initialization, it is more important to preserve the gradient signal than the loss itself. They propose to use as saliency the gradient norm of the loss $\Delta\mathcal{L}(\boldsymbol{\theta}) = \nabla\mathcal{L}(\boldsymbol{\theta})^T \nabla\mathcal{L}(\boldsymbol{\theta})$, but measured *after* pruning.

To maximize it, Wang et al. (2020a) adopt the same approximation introduced in LeCun et al. (1990) and treat pruning as a perturbation on the initial weights.

Their method is equivalent to solving:

$$\max_{\mathbf{c}} G(\boldsymbol{\theta}, \mathbf{c}) := \sum_{\{i: c_i=0\}} -\theta_i [\mathbf{H}\mathbf{g}]_i \quad \text{s.t. } \mathbf{c} \in \{0, 1\}^m, \|\mathbf{c}\|_0 = k. \quad (3.4)$$

Where $\mathbf{H}$ and $\mathbf{g}$ denote the Hessian and the gradient of the loss respectively.

3.4 Foresight Connection Sensitivity

Since removing connections of a neural network will have a significant impact on its forward and backward signals, we are interested in obtaining a pruned network that is easy to train. We use *connection sensitivity* of the loss function as a proxy for the so-called trainability of a network.

To this end, we first define connection sensitivity *after* pruning which we name *Foresight Connection sEnsitivity* (FORCE), and then propose two procedures to optimize it to obtain the desired pruned network. Let $\bar{\boldsymbol{\theta}} = \boldsymbol{\theta} \odot \mathbf{c}$ denotes the pruned parameters once a binary mask $\mathbf{c}$ with $\|\mathbf{c}\|_0 = k \leq m$ is applied. The FORCE at $\bar{\boldsymbol{\theta}}$ for a given mask $\hat{\mathbf{c}}$ is then obtained as:

$$\mathbf{g}(\bar{\boldsymbol{\theta}}) := \left. \frac{\partial\mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial\mathbf{c}} \right|_{\mathbf{c}=\hat{\mathbf{c}}} = \left. \frac{\partial\mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial\bar{\boldsymbol{\theta}}} \right|_{\mathbf{c}=\hat{\mathbf{c}}} \odot \left. \frac{\partial\bar{\boldsymbol{\theta}}}{\partial\mathbf{c}} \right|_{\mathbf{c}=\hat{\mathbf{c}}} = \left. \frac{\partial\mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial\bar{\boldsymbol{\theta}}} \right|_{\mathbf{c}=\hat{\mathbf{c}}} \odot \boldsymbol{\theta}. \quad (3.5)$$

The last equality is obtained by rewriting $\bar{\boldsymbol{\theta}}$ as $\text{diag}(\boldsymbol{\theta})\mathbf{c}$, where $\text{diag}(\boldsymbol{\theta})$ is a diagonal matrix with $\boldsymbol{\theta}$ as its elements, and then differentiating w.r.t. $\mathbf{c}$. Note, when $k < m$, the sub-network $\bar{\boldsymbol{\theta}}$ is obtained by *removing* connections corresponding to all the weights for which the binary variable is zero. Therefore, only the weights corresponding

to the indices for which $\mathbf{c}(i) = 1$ contribute in equation (3.5), all other weights do not participate in forward and backward propagation and are to be ignored. We now discuss the crucial differences between our formulation (3.5), SNIP (3.2) and GRASP (3.4).

- When $\hat{\mathbf{c}} = \mathbf{1}$, the formulation is exactly the same as the connection sensitivity used in SNIP. However, $\hat{\mathbf{c}} = \mathbf{1}$ is too restrictive in the sense that it assumes that all the parameters are active in the network and they are removed one by one *with replacement*, therefore, it fails to capture the impact of removing a group of parameters.
- Our formulation uses weights and gradients corresponding to $\bar{\boldsymbol{\theta}}$ thus, compared to SNIP, provides a better indication of the training dynamics of the pruned network. However, GRASP formulation is based on the assumption that pruning is a small perturbation on the Gradient Norm which, also shown experimentally, is not always a reliable assumption.
- When $\|\hat{\mathbf{c}}\|_0 \ll \|\mathbf{1}\|_0$, *i.e.*, extreme pruning, the gradients before and after pruning will have very different values as $\|\boldsymbol{\theta} \odot \hat{\mathbf{c}}\|_2 \ll \|\boldsymbol{\theta}\|_2$, making SNIP and GRASP unreliable (empirically we find SNIP and GRASP fail in the case of high sparsity).

FORCE saliency Note FORCE (3.5) is defined for a given sub-network which is unknown *a priori*, as our objective itself is to find the sub-network with maximum connection sensitivity. Similar to the reformulation of SNIP in (3.3), the objective to find such sub-network corresponding to the foresight connection sensitivity can be written as:

$$\max_{\mathbf{c}} S(\boldsymbol{\theta}, \mathbf{c}) := \sum_{i \in \text{supp}(\mathbf{c})} |\theta_i \nabla \mathcal{L}(\boldsymbol{\theta} \odot \mathbf{c})_i| \quad \text{s.t. } \mathbf{c} \in \{0, 1\}^m, \|\mathbf{c}\|_0 = k. \quad (3.6)$$

Here $\nabla \mathcal{L}(\boldsymbol{\theta} \odot \mathbf{c})_i$ represents the i -th index of $\frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}} \Big|_{\mathbf{c}}$. As opposed to (3.3), finding the optimal solution of (3.6) is non-trivial as it requires computing the gradients of all possible $\binom{m}{k}$ sub-networks to find the one with maximum sensitivity. To this end, we present two approximate solutions to the above problem that primarily involve (i)

progressively increasing the degree of pruning, and (ii) solving an approximation of (3.6) at each stage of pruning.

Progressive Pruning (Iterative SNIP) Let k be the number of parameters to be kept after pruning. Let us assume that we know a schedule (will be discussed later) to divide k into a set of natural numbers $\{k_t\}_{t=1}^T$ such that $k_t > k_{t+1}$ and $k_T = k$. Now, given the mask $\mathbf{c}_t$ corresponding to k_t , pruning from k_t to k_{t+1} can be formulated using the connection sensitivity (3.5) as:

$$\mathbf{c}_{t+1} = \underset{\mathbf{c}}{\operatorname{argmax}} S(\bar{\boldsymbol{\theta}}, \mathbf{c}) \quad \text{s.t.} \quad \mathbf{c} \in \{0, 1\}^m, \quad \|\mathbf{c}\|_0 = k_{t+1}, \quad \mathbf{c} \odot \mathbf{c}_t = \mathbf{c}, \quad (3.7)$$

where $\bar{\boldsymbol{\theta}} = \boldsymbol{\theta} \odot \mathbf{c}_t$. The additional constraint $\mathbf{c} \odot \mathbf{c}_t = \mathbf{c}$ ensures that no parameter that had been pruned earlier is activated again. Assuming that the pruning schedule ensures a smooth transition from one topology to another ($\|\mathbf{c}_t\|_0 \approx \|\mathbf{c}_{t+1}\|_0$) such that the *gradient approximation* $\left. \frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}} \right|_{\mathbf{c}_t} \approx \left. \frac{\partial \mathcal{L}(\bar{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}} \right|_{\mathbf{c}_{t+1}}$ is valid, (3.7) can be approximated as solving (3.3) at $\bar{\boldsymbol{\theta}}$. Thus, for a given schedule over k , our first approximate solution to (3.6) involves solving (3.3) iteratively. This allows re-assessing the importance of connections after changing the sub-network. For a schedule with $T = 1$, we recover SNIP where a crude *gradient approximation* between the dense network $\mathbf{c}_0 = \mathbf{1}$ and the final mask $\mathbf{c}$ is being used. This approach of ours turns out to be algorithmically similar to a concurrent work (Verdenius et al., 2020). However, our motivation comes from a novel objective function (5) which also gives place to our second approach (FORCE). Tanaka et al. (2020) also concurrently study the effect of iterative pruning and report, similar to our findings, pruning progressively is needed for high sparsities.

Progressive Sparsification (FORCE) The constraint $\mathbf{c} \odot \mathbf{c}_t = \mathbf{c}$ in (3.7) (Iterative SNIP) might be restrictive in the sense that while re-assessing the importance of *unpruned* parameters, it does not allow previously *pruned* parameters to resurrect (even if they could become important). This hinders exploration which can be unfavourable in finding a suitable sub-network. Here we remove this constraint, meaning, the weights for which $\mathbf{c}(i) = 0$ are *not* removed from the network, rather they are assigned a value of zero. Therefore, while not contributing to the forward signal, they might have a

non-zero gradient. This relaxation modifies the saliency in (3.5) whereby the gradient is now computed at a *sparsified* network instead of a *pruned* network. Similar to the above approach, we sparsify the network progressively and once the desired sparsity is reached, all connections with $\mathbf{c}(i) = 0$ are *pruned*. Note, the step of removing zero weights is valid if removing such connections does not adversely impact the gradient flow of the unpruned parameters. We, in fact, found it to be true in our experiments shown in Fig A.3 (Appendix). However, this assumption might not hold always.

An overview of Iterative SNIP and FORCE is presented in Algorithm 1.

Sparsity schedule Both the above discussed iterative procedures approximately optimize (3.5), however, they depend on a sparsity/pruning schedule favouring *small* steps to be able to reliably apply the mentioned *gradient approximation*. One such valid schedule would be where the portion of newly removed weights with respect to the remaining weights is small. We find a simple exponential decay schedule, defined below, to work very well on all our experiments:

$$\text{Exp mode: } k_t = \exp \{ \alpha \log k + (1 - \alpha) \log m \}, \quad \alpha = \frac{t}{T}. \quad (3.8)$$

In section 3.5.3 we empirically show that these methods are very robust to the hyperparameter T .

Some theoretical insights When pruning weights gradually, we are looking for the best possible sub-network in a neighbourhood defined by the previous mask and

Algorithm 1 FORCE/Iter SNIP algorithms to find a pruning mask

- 1: **Inputs:** Training set $\mathcal{D}$, final sparsity k , number of steps T , weights $\boldsymbol{\theta}_0 \in \mathbb{R}^m$.
 - 2: Obtain $\{k_t\}_{t=1:T}$ using the chosen schedule (refer to Eq (3.8))
 - 3: Define initial mask $\mathbf{c}_0 = \mathbf{1}$
 - 4: **for** $t = 0, \dots, T - 1$ **do**
 - 5: Sample mini-batch $\{z_i\}_{i=1}^n$ from $\mathcal{D}$
 - 6: Define $\bar{\boldsymbol{\theta}} = \boldsymbol{\theta} \odot \mathbf{c}_t$ (as *sparsified* (FORCE) vs *pruned* (Iter SNIP) network)
 - 7: Compute $\mathbf{g}(\bar{\boldsymbol{\theta}})$ (refer to Eq (3.5))
 - 8: $I = \{i_1, \dots, i_{k_{t+1}}\}$ are top- k_{t+1} values of $|\mathbf{g}_i|$
 - 9: Build $\mathbf{c}_{t+1}$ by setting to 0 all indices not included in I .
 - 10: **end for**
 - 11: **Return:** $\mathbf{c}_T$.
-

the amount of weights removed at that step. The problem being non-convex and non-smooth makes it challenging to prove if the mask obtained by our method is globally optimal. However, in Appendix A.4 we prove that each intermediate mask obtained with Iterative SNIP is indeed an approximate local minima, where the degree of sub-optimality increases with the pruning step size. This gives some validation on why SNIP fails on higher sparsity. We can not provide the same guarantees for FORCE (there is no obvious link between the step size and the distance between masks), nevertheless, we empirically observe that FORCE is quite robust and more often than not improves over the performance of Iterative SNIP, which is not able to recover weights once pruned. We present further analysis in Appendix A.3.4.

3.5 Experiments

In the following we evaluate the efficacy of our approaches accross different architectures and datasets. Training settings, architecture descriptions, and implementation details are provided in Appendix A.1.

3.5.1 Results on CIFAR-10

Fig 3.2 compares the accuracy of the described iterative approaches with both SNIP and GRASP. We also report the performance of a dense and a random pruning baseline. Both SNIP and GRASP consider a single batch to approximate the saliencies, while we employ a different batch of data at each stage of our gradual skeletonization process. For a fair comparison, and to understand how the number of batches impacts performance, we also run these methods averaging the saliencies over T batches, where T is the number of iterations. SNIP-MB and GRASP-MB respectively refer to these multi-batch (MB) counterparts. In these experiments, we use $T = 300$. We study the hyper-parameter robustness regarding T later in section 3.5.3.

We observe that for moderate sparsity levels, one batch is sufficient for both SNIP and GRASP as reported in Lee et al. (2019); Wang et al. (2020a). However, as we increase the level of sparsity, the performance of SNIP and GRASP degrades

dramatically. For example, at 99.0% sparsity, SNIP drops down to 10% accuracy for both ResNet50 and VGG19, which is equivalent to random guessing as there are 10 classes. Note, in the case of randomly pruned networks, accuracy is nearly 75% and 82% for ResNet50 and VGG19, respectively, which is significantly better than the performance of SNIP. However, to our surprise, just using multiple batches to compute the connection sensitivity used in SNIP improves it from 10% to almost 90%. This clearly indicates that a better approximation of the connection sensitivity is necessary for good performance in the case of high sparsity regime. Similar trends, although not this extreme, can be observed in the case of GRASP as well. On the other hand, gradual pruning approaches are much more robust in terms of sparsity for example, in the case of 99.9% pruning, while one-shot approaches perform as good as a random classifier (nearly 10% accuracy), both FORCE and Iterative SNIP obtain more than 80% accuracy. While the accuracies obtained at higher sparsities might have degraded too much for some use cases, we argue this is an encouraging result, as no approach before has pruned a network at initialization to such extremes while keeping the network trainable and these results might encourage the community to improve the performance further. Finally, gradual pruning methods consistently improve over other methods even at moderate sparsity levels (refer to Fig A.1), this motivates the use of FORCE or Iterative SNIP instead of other methods by default at any sparsity regime. Moreover, the additional cost of using iterative pruning instead

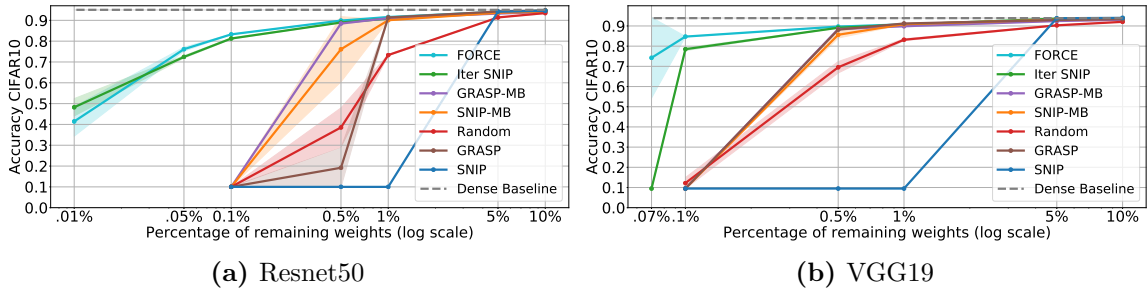


Figure 3.2: Test accuracies on CIFAR-10 for different pruning methods. With increased number of batches (-MB) one-shot methods are more robust at higher sparsity levels, but our gradual pruning approaches can go even further. Moreover, FORCE consistently reaches higher accuracy than other methods across most sparsity levels. Each point is an average of ≥ 3 runs of prune-train-test. The shaded areas denote the standard deviation of the runs (too small to be visible in some cases). Note that in (b), GRASP and GRASP-MB overlap.

of SNIP-MB is negligible compared to the cost of training and is significantly cheaper than GRASP-MB, further discussed in section 3.5.3.

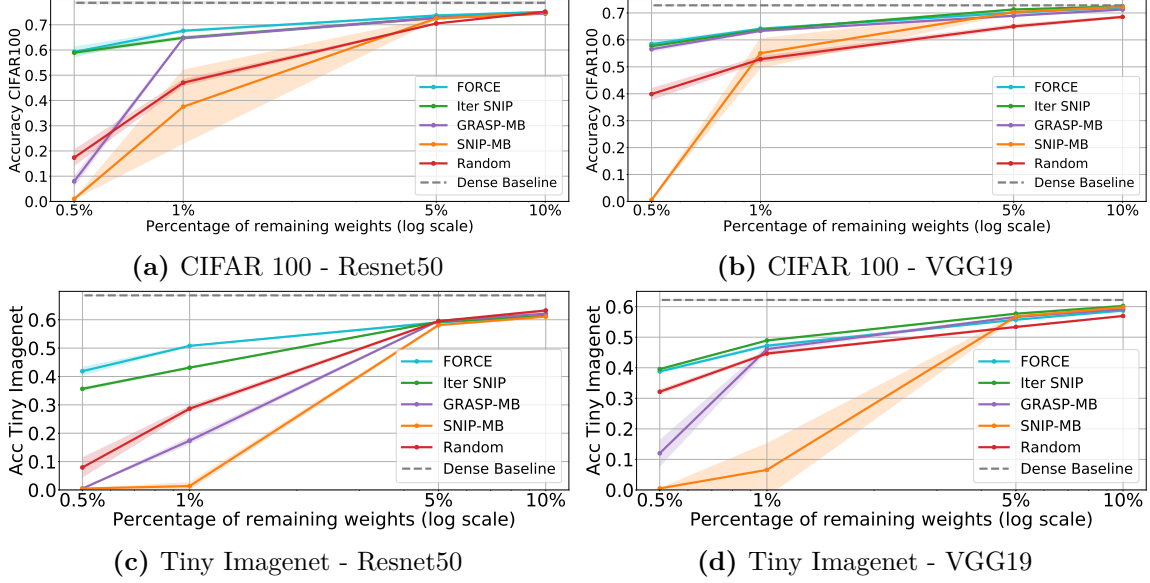


Figure 3.3: Test accuracies on CIFAR-100 and Tiny Imagenet for different pruning methods. Each point is the average over 3 runs of prune-train-test. The shaded areas denote the standard deviation of the runs (too small to be visible in some cases).

3.5.2 Results on larger datasets

We now present experiments on large datasets. Wang et al. (2020a) and Lee et al. (2019) suggest using a batch of size ~ 10 times the number of classes, which is very large in these experiments. Instead, for memory efficiency, we average the saliencies over several mini-batches. For CIFAR100 and Tiny-ImageNet, we average 10 and 20 batches per iteration respectively, with 128 examples per batch. As we increase the number of batches per iteration, computing the pruning mask becomes more expensive. From Fig 3.4, we observe that the accuracy converges after just a few iterations. Thus, for the following experiments we used 60 iterations. For a fair comparison, we run SNIP and GRASP with $T \times B$ batches, where T is the number of iterations and B the number of batches per iteration in our method. We find the results, presented in Fig 3.3, consistent with trends in CIFAR-10.

In the case of Imagenet, we use a batch size of 256 examples and 40 batches per iteration. We use the official implementation of VGG19 with batch norm and Resnet50

from Paszke et al. (2017). As presented in Table 3.1, gradual pruning methods are consistently better than SNIP, with a larger gap as we increase sparsity. We would like to emphasize that FORCE is able to prune 90% of the weights of VGG while losing less than 3% of the accuracy, we find this remarkable for a method that prunes before any weight update. Interestingly, GRASP performs better than other methods at 95% sparsity (VGG), moreover, it also slightly surpasses FORCE for Resnet50 at 90%, however, it under-performs random pruning at 95%. In fact, we find all other methods to perform worse than random pruning for Resnet50. We hypothesize that, for a much more challenging task (Imagenet with 1000 classes), Resnet50 architecture might not be extremely overparametrized. For instance, VGG19 has 143.68M parameters while Resnet50 uses 25.56M (refer to Table A.1). On the other hand, the fact that random pruning can yield relatively trainable architectures for these sparsity levels is somewhat surprising and might indicate that there still is room for improvement in this direction. Results seem to indicate that the FORCE saliency is a step in the right direction and we hypothesize further improvements on its optimization might lead to even better performance. In Appendix A.3.6, we show superior performance of our approach on the **Mobilenet-v2 architecture** (Sandler et al., 2018) as well, which is much more "slim" than Resnet and VGG ².

²Mobilenet has 2.3M params compared to 20.03M and 23.5M of VGG and Resnet, respectively.

Table 3.1: Test accuracies on Imagenet for different pruning methods and sparsities.

Network	VGG19				Resnet50			
Sparsity percentage	90%		95%		90%		95%	
Accuracy	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
(Baseline)	73.1	91.3			75.6	92.8		
FORCE (Ours)	70.2	89.5	65.8	86.8	64.9	86.5	59.0	82.3
Iter SNIP (Ours)	69.8	89.5	65.9	86.9	63.7	85.5	54.7	78.9
GRASP	69.5	89.2	67.6	87.8	65.4	86.7	46.2	66.0
SNIP	68.5	88.8	63.8	86.0	61.5	83.9	44.3	69.6
Random	64.2	86.0	56.6	81.0	64.6	86.0	57.2	80.8

3.5.3 Analysis

Saliency optimization To experimentally validate our approach (3.7), we conduct an ablation study where we compute the FORCE saliency after pruning (3.5) while varying the number of iterations T for different sparsity levels. In Fig 3.4 (left) we present the relative change in saliency as we vary the number of iterations T , note when $T = 1$ we recover one-shot SNIP. As expected, for moderate levels of sparsity, using multiple iterations does not have a significant impact on the saliency. Nevertheless, as we target higher sparsity levels, we can see that the saliency can be better optimized when pruning iteratively. In Appendix A.3.1 we include results for FORCE where we observe similar trends.

Hyperparameter robustness As shown in Figures 3.2 and 3.3, for low sparsity levels, all methods are comparable, but as we move to higher sparsity levels, the gap becomes larger. In Fig 3.4 (middle) we fix sparsity at 99.5% and study the accuracy as we vary the number of iterations T . Each point is averaged over 3 trials. SNIP ($T = 1$) yields sub-networks unable to train (10% acc), but as we move to iterative pruning ($T > 1$) accuracy increases up to 90% for FORCE and 89% for Iter SNIP. Moreover, accuracy is remarkably robust to the choice of T , the best performance for both FORCE and Iter SNIP is with more iterations, however a small number of iterations already brings a huge boost. This suggests these methods might be used by default by a user without worrying too much about hyper-parameter tuning, easily adapting the amount of iterations to their budget.

Pruning cost As shown in Fig 3.2, SNIP performance quickly degrades beyond 95% sparsity. Wang et al. (2020a) suggested GRASP as a more robust alternative, however, it needs to compute a Hessian vector product which is significantly more expensive in terms of memory and time. In Fig 3.4 (right), we compare the time cost of different methods to obtain the pruning masks along with the corresponding accuracy. We observe that both SNIP and GRASP are fragile when using only one batch (red accuracy indicates performance below random baseline). When using multiple batches their robustness increases, but so does the pruning cost. Moreover, we find that gradual

3. Progressive Skeletonization: Trimming more fat from a network at initialization

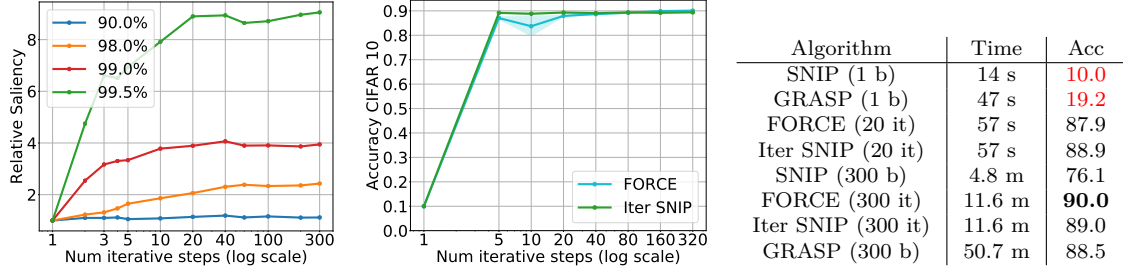


Figure 3.4: **Left:** FORCE saliency (3.6) obtained with (3.7) when varying T normalized by the saliency with one-shot SNIP ($T = 1$). Pruning iteratively brings more gains for higher sparsity levels. Error bars not shown for better visualization. **Middle:** Test acc pruning with FORCE and Iter SNIP at 99.5% sparsity for different T . Both methods are extremely robust to the choice of T . **Right:** Wall time to compute pruning masks for CIFAR10/Resnet50/TeslaK40m vs acc at 99.5% sparsity; (x b) means we used x batches to compute the gradients while (x it) denotes we used x pruning iterations, with one batch per iteration. Numbers in **red** indicate performance below random pruning.

pruning based on the FORCE saliency is much cheaper than GRASP-MB when using equal amount of batches, this is because GRASP involves an expensive Hessian vector product. Thus, FORCE (or Iterative SNIP) would be preferable over GRASP-MB even when they have comparable accuracies.

FORCE vs Iterative SNIP Empirically, we find that FORCE tends to outperform Iter SNIP more often than not, suggesting that allowing weights to recover is indeed beneficial despite having less theoretical guarantees (see *gradient approximation* in Section 3.4). Thus, we would make FORCE algorithm our default choice, especially for Resnet architectures. In Appendix A.3.4 we empirically observe two distinct phases when pruning with FORCE. The first one involves exploration (early phase) when the amount of pruned and recovered weights seem to increase, indicating exploration of masks that are quite different from each other. The second, however, shows rapid decrease in weight recovery, indicating a phase where the algorithm converges to a more constrained topology. As opposed to Iter SNIP, the possibility of the exploration of many possible sub-networks before converging to a final topology might be the reason behind the slightly improved performance of FORCE. But this exploration comes at a price, in Fig 3.4 (middle and left) we observe how, despite FORCE reaching a higher accuracy when using enough steps, if we are under a highly constrained computational budget and can only afford a few pruning iterations, Iter SNIP is more likely to obtain a better pruning mask. This is indeed expected as FORCE might need

more iterations to converge to a good sub-space, while Iter SNIP will be forced to converge by construction. A combination of FORCE and Iter SNIP might lead to an even better approach, we leave this for future work.

Early pruning as an additional baseline Our gradual pruning approaches (SNIP-MB and GRASP-MB as well) use multiple batches to obtain a pruned mask, considering that pruning can be regarded as a form of training (Mallya et al., 2018), we create another baseline for the sake of completeness. We train a network for one epoch, a similar number of iterations as used by our approach, and then use magnitude pruning to obtain the final mask, we call this approach **early pruning** (more details in Appendix A.3.5). Interestingly, we find that early pruning tends to perform worse than SNIP-MB (and gradual pruning) for Resnet, and shows competitive performance at low sparsity level for VGG but with a sharp drop in the performance as the sparsity level increases. Even though these experiments support the superiority of our approach, we would like to emphasize that they do not conclude that any early pruning strategy would be suboptimal compared to pruning at initialization as an effective approach in this direction might require devising a well thought objective function.

Iterative pruning to maximize the Gradient Norm In Sec 3.4, we have seen Iterative SNIP can be used to optimize the FORCE saliency. We also tried to use GRASP iteratively, however, after a few iterations the resulting networks were not trainable. Interestingly, if we apply the *gradient approximation* to GRASP saliency (instead of Taylor), we can come up with a different iterative approximation to maximize the gradient norm after pruning. We empirically observe this method is more robust than GRASP to high sparsity levels. This suggests that 1) Iterative pruning, although beneficial, can not be trivially applied to any method. 2) The gradient approximation is more general than in the context of FORCE/SNIP sensitivity. We present further details and results in Appendix A.5.

3.6 Discussion

Pruning at initialization has become an active area of research both for its practical and theoretical interests. In this work, we discovered that existing methods mostly perform below random pruning at extreme sparsity regime. We presented FORCE, a new saliency to compute the connection sensitivity after pruning, and two approximations to progressively optimize FORCE in order to prune networks at initialization. We showed that our methods are significantly better than the existing approaches for pruning at extreme sparsity levels, and are at least as good as the existing ones for pruning at moderate sparsity levels. We also provided theoretical insights on why progressive skeletonization is beneficial at initialization, and showed that the cost of iterative methods is reasonable compared to the existing ones. Although pruning iteratively has been ubiquitous in the pruning community, it was not evident that pruning at initialization might benefit from this scheme. Particularly, not every approximation could be used for gradual pruning as we have shown with GRASP. However, the *gradient approximation* allowed us to gradually prune while maximizing either the gradient norm or FORCE. We consider our results might encourage future work to further investigate the exploration/exploitation trade-off in pruning and find more efficient pruning schedules, not limited to the pruning at initialization.

4

Make some noise: Reliable and Efficient Single-Step Adversarial Training

Abstract

Recently, Wong et al. (2020) showed that adversarial training with single-step FGSM leads to a characteristic failure mode named Catastrophic Overfitting (CO), in which a model becomes suddenly vulnerable to multi-step attacks. Experimentally they showed that simply adding a random perturbation prior to FGSM (RS-FGSM) could prevent CO. However, Andriushchenko and Flammarion (2020) observed that RS-FGSM still leads to CO for larger perturbations, and proposed a computationally expensive regularizer (GradAlign) to avoid it. In this work, we methodically revisit the role of noise and clipping in single-step adversarial training. Contrary to previous intuitions, we find that using a stronger noise around the clean sample combined with not clipping is highly effective in avoiding CO for large perturbation radii. We then propose Noise-FGSM (N-FGSM) that, while providing the benefits of single-step adversarial training, does not suffer from CO. Empirical analyses on a large suite of experiments show that N-FGSM is able to match or surpass the performance of previous state-of-the-art GradAlign, while achieving 3× speed-up. Code can be found in <https://github.com/pdejorge/N-FGSM>

4.1 Introduction

Deep neural networks have achieved remarkable performance on a variety of tasks (He et al., 2015; Silver et al., 2016; Devlin et al., 2019). However, it is well known that they are vulnerable to small worst-case perturbations around the input data – commonly referred to as *adversarial examples* (Szegedy et al., 2014). The existence of such adversarial examples poses a security threat to deploying models in sensitive environments (Biggio and Roli, 2018; Boloor et al., 2019). This has motivated a large body of work towards improving the *adversarial robustness* of neural networks (Goodfellow et al., 2015; Papernot et al., 2016; Tramèr et al., 2018; Sanyal et al., 2018; Cisse et al., 2017).

The most popular family of methods for learning robust neural networks is based on the concept of *adversarial training* (Goodfellow et al., 2015; Madry et al., 2018). In a nutshell, adversarial training can be posed as a min-max problem where instead of minimizing some loss over a dataset of *clean* samples, we augment the inputs with worst-case perturbations that are generated online during training. However, obtaining such perturbations is NP-hard (Weng et al., 2018) and hence, different *adversarial attacks* have been suggested that approximate them. In their seminal work, Goodfellow et al. (2015) proposed the *Fast Gradient Sign Method* (FGSM), that generates adversarial attacks by performing a gradient ascent step on the loss function. Yet, while FGSM-based adversarial training provides robustness against single-step FGSM adversaries, Tramèr et al. (2018) showed that these models are still vulnerable to multi-step attacks, namely those allowed to perform multiple gradient ascent steps. Given their better (robust) performance, multi-step attacks such as *Projected Gradient Descent* (PGD) Madry et al. (2018) have now become the de facto standard for adversarial training.

The main downside of multi-step adversarial training is that the cost of these attacks increases linearly with the number of steps, making their applicability often computationally prohibitive. For this reason, several works have focused on reducing

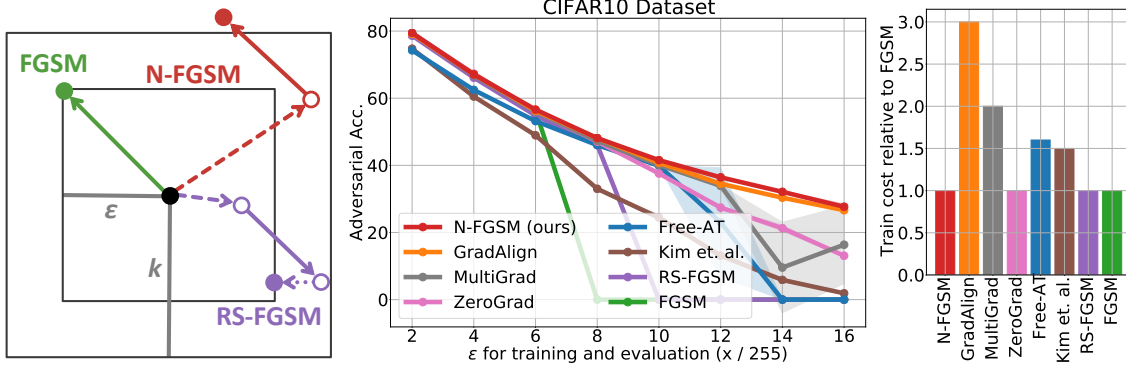


Figure 4.1: Left: Visualization of FGSM (Goodfellow et al., 2015), RS-FGSM (Wong et al., 2020) and N-FGSM (ours) attacks. While RS-FGSM is limited to noise in the $\epsilon - l_\infty$ ball, N-FGSM draws noise from an arbitrary $k - l_\infty$ ball. Moreover, N-FGSM does not clip the perturbation around the clean sample. **Middle:** Comparison of single-step methods on CIFAR-10 with PreactResNet18 over different perturbation radii (ϵ is divided by 255). Our method, N-FGSM, can match or surpass state-of-the-art results while *reducing the cost by a $3\times$ factor*. Adversarial accuracy is based on PGD-50-10 and experiments are averaged over 3 seeds. **Right:** Comparison of training costs relative to FGSM baseline based on the number of Forward-Backward passes, see Appendix B.17 for details.

the cost of adversarial training by approximating the worst-case perturbations with single-step attacks (Wong et al., 2020; Shafahi et al., 2019; Vivek and Babu, 2020). In particular, Wong et al. (2020) studied FGSM adversarial training and discovered that it suffers from a characteristic failure mode, in which a model suddenly becomes vulnerable to multi-step attacks despite remaining robust to single-step attacks. This phenomenon is referred to as *Catastrophic Overfitting* (CO). As a solution, they argued that adding a random perturbation prior to FGSM (RS-FGSM) seemed sufficient to prevent CO and produce robust models. Yet, Andriushchenko and Flammarion (2020) recently observed that RS-FGSM still leads to CO as one increases the perturbation radii of the attacks. They suggested a regularizer (GradAlign) that can avoid CO in the settings they considered, but at the expense of computing a double derivative – significantly increasing the computational cost with respect to RS-FGSM.

In this paper, we revisit the idea of including noise in single-step attacks. Differently from previous methods that consider the noise as part of the attack, we propose an adversarial training procedure where the noise is used as a form of *data augmentation*. As we detail in Section 4.4, this motivates us to introduce two main changes with

respect to previous methods: 1) We center adversarial perturbations with respect to noise-augmented samples and therefore, unlike previous RS-FGSM, we do not clip around the clean samples. 2) We use noise perturbations larger than the ϵ -ball, since they are not restricted by the strength of the attack anymore. Our experiments show that performing data augmentation with sufficiently *strong noise* and removing the *clipping step* improves model robustness and prevents CO, even against large perturbation radii. Our new method, termed N-FGSM, matches, or even surpasses, the robust accuracy of the regularized FGSM introduced by Andriushchenko and Flammarion (2020) (GradAlign), while *providing a $3\times$ speed-up*.

To corroborate the effectiveness of our solution, we present an experimental survey of recently proposed single-step attacks and empirically demonstrate that N-FGSM trades-off robustness and computational cost better than any other single-step approach, evaluated over a large spectrum of perturbation radii (see Figure 4.1, middle and right panels), over several datasets (CIFAR-10, CIFAR-100, and SVHN) and architectures (PreActResNet18 and WideResNet28-10). We will release our code to reproduce the experiments.

4.2 Related Work

Since the discovery of adversarial examples, many defense mechanisms have been proposed, *adversarial training* being one of the most popular and empirically validated. We can categorise adversarial training methods based on how they approximate the perturbations applied to training samples. *Multi-step* approaches approximate an inner maximization problem to find the worst-case perturbation with several gradient ascent steps (Zhang et al., 2019; Kurakin et al., 2017a; Madry et al., 2018). While this provides a better approximation, it is also more expensive. At the other end of the spectrum, *single-step* methods only use one gradient step to approximate the worst case perturbation. Goodfellow et al. (2015) first proposed FGSM; Tramèr et al. (2018) proposed a new variant with an additional random step (R+FGSM), but observed that

both methods were vulnerable to multi-step attacks. Shafahi et al. (2019) proposed *Free Adversarial Training* (Free-AT), which successfully reduced the computational cost of training by using a single backward pass to compute both weight update and attack. Motivated by this, Wong et al. (2020) explored a variant of R+FGSM, namely RS-FGSM, that uses a less restrictive form of noise and showed this can improve robustness for moderate perturbation radii at the same cost as FGSM. Recently, Andriushchenko and Flammarion (2020) proposed the GradAlign regularizer. Combining FGSM with GradAlign results in robust models at even larger perturbation radii. However, GradAlign suffers from a *threefold* increase in the training cost to as compared to FGSM. The need for more efficient solutions has motivated a growing body of work whose goal is the design of computationally lighter single-step methods (Golgooni et al., 2021; Kim et al., 2021; Vivek and Babu, 2020; Park and Lee, 2021; Li et al., 2020).

In this work, we revisit the idea of combining noise with the FGSM attack. Our method builds upon FGSM and intuitions from R+FGSM and RS-FGSM to combine it with random perturbations, however, we consider the noise step as data augmentation rather than part of the attack. This motivates us to use a stronger noise *without* clipping. As opposed to Kang and Moosavi-Dezfooli (2021), our thorough study leads to a practically effective approach that yields robustness also against large perturbation radii.

4.3 Preliminaries on Single-Step Adversarial Training

Given a classifier $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ parameterized by θ and a perturbation set $\mathcal{S}$, f_θ is defined as *robust* at $x \in \mathcal{X}$ on the set $\mathcal{S}$ if for all $\delta \in \mathcal{S}$ we have $f_\theta(x + \delta) = f_\theta(x)$. One of the most popular definitions for $\mathcal{S}$ is the $\epsilon - \ell_\infty$ ball, *i.e.* $\mathcal{S} = \{\delta : \|\delta\|_\infty \leq \epsilon\}$. This is known as the ℓ_∞ threat model which we adopt throughout this work.

To train networks that are robust against ℓ_∞ threat models, adversarial training modifies the classical training procedure of minimizing a loss function over a dataset

$\mathcal{D} = \{(x_i, y_i)\}_{i=1:N}$ of images $x_i \in \mathcal{X}$ and labels $y_i \in \mathcal{Y}$. In particular, adversarial training instead minimizes the worst-case loss over the perturbation set $\mathcal{S}$, *i.e.* training is on the adversarially perturbed samples $\{(x_i + \delta_i, y_i)\}_{i=1:N}$. Under the l_∞ threat model, we can formalize adversarial training as solving the following problem:

$$\min_{\theta} \sum_{i=1}^N \max_{\delta} \mathcal{L}(f_{\theta}(x_i + \delta), y_i) \quad \text{s.t. } \|\delta\|_{\infty} \leq \epsilon, \quad (4.1)$$

where $\mathcal{L}$ is typically the cross-entropy loss. Due to the difficulty of finding the exact inner maximizer, the most common procedure for adversarial training is to approximate the worst-case perturbation through several PGD steps (Madry et al., 2018). While PGD has been shown to yield robust models, its cost increases linearly with the number of steps. As a result, several works have focused on reducing the cost of adversarial training by approximating the inner maximization with a single-step.

If the loss function is linear with respect to the input, the inner maximization of (4.1) will enjoy a closed form solution. Goodfellow et al. (2015) leveraged this to propose FGSM, where the adversarial perturbation follows the direction of the sign of the gradient. Tramèr et al. (2018) proposed adding a random initialization prior to FGSM. However, both methods were later shown to be vulnerable against multi-step attacks, such as PGD. Contrary to prior intuition, recent work from Wong et al. (2020) observed that combining a random step with FGSM can actually lead to a promising robustness performance. In particular, most recent single-step methods approximate the worst-case perturbation solving the inner maximization problem in (4.1) with the following general form:

$$\delta = \psi\left(\eta + \alpha \cdot \text{sign}\left(\nabla_{x_i} \mathcal{L}(f_{\theta}(x_i + \eta), y_i)\right)\right), \quad (4.2)$$

where η is drawn from a distribution Ω . For example, when ψ is the projection operator onto the ℓ_∞ ball and Ω is the uniform distribution $[-\epsilon, \epsilon]^d$, where d is the dimension of $\mathcal{X}$, this recovers RS-FGSM. Under a different noise setting where $\Omega = (\epsilon - \alpha) \cdot \text{sign}(\mathcal{N}(\mathbf{0}_d, \mathbf{I}_d))$ and by choosing the step size α to be in $[0, \epsilon]$, we recover R+FGSM by Tramèr et al. (2018). This was among the first works to explore the application of noise to FGSM, but did not report improvements over it. If we consider

Ω to be deterministically 0 and ψ to be the identity map, we recover FGSM. Finally, if we take FGSM and add a gradient alignment regularizer, this recovers GradAlign.

4.4 Noise and FGSM

Previous methods that combined noise with FGSM, *e.g.* R+FGSM (Tramèr et al., 2018) and RS-FGSM (Wong et al., 2020), have considered the noise as a *random step* integrated within the attack. Since it is a common practice to restrict adversarial perturbations to the ϵ -ball, we argue that this introduces a trade-off between the magnitude of the random step and that of the attack. For illustration, consider the purple lines corresponding to RS-FGSM in Figure 4.1 (left). If the initial random step is significantly larger than the ϵ -ball, then the final clipping step will have a noticeable impact on the perturbation, possibly removing a considerable portion of the FGSM step (middle arrow). To prevent this from happening, R+FGSM and RS-FGSM restrict the random step to lie within the ϵ -ball, thereof implicitly entangling the noise magnitude and the attack strength.

Contrary to previous methods, we note that adding noise to the clean sample can be considered as a form of *data augmentation* to be applied independently from the attack. We make two considerations from this perspective 1) When one performs data augmentation during adversarial training, the input after the corresponding transformation is the starting point to compute the adversarial perturbations, therefore, we argue that adversarial attacks should be centered around the noise-augmented samples. This motivates us to *avoid clipping* around the clean sample. 2) We do not need to restrict the noise augmentation to lie inside an ϵ -ball, since its strength is disentangled from that of the attack. Thus, we can use *stronger noise-augmentations* than previous methods.

These modifications lead to a novel adversarial training method that combines noise-based data augmentations with FGSM. We denote it as Noise-FGSM (N-FGSM). Following the notation introduced in Section 4.3, we define the noise augmented

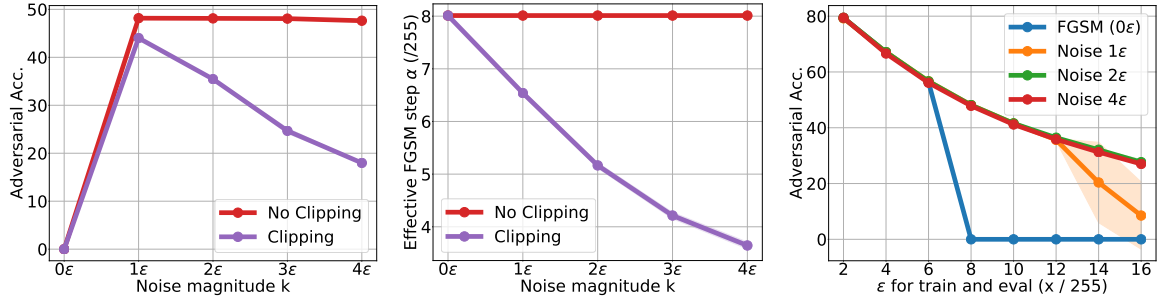


Figure 4.2: **Left:** Ablation of clipping vs not clipping around the clean sample x for $\epsilon = 8/255$. Clipping leads to a significant drop in robustness which increases with the strength of the noise augmentations. **Middle:** Analysis of the effective FGSM step size after clipping. We observe that clipping leads to a decrease in the effective FGSM step size, thus, adversarial perturbations will be more similar to random noise. **Right:** N-FGSM (ours) when varying the noise magnitude k (ϵ is divided by 255). Increasing the amount of noise is key to avoiding CO. For (left) and (right) plots, adversarial accuracy is based on PGD-50-10 and experiments are averaged over 3 seeds.

sample as $x_{\text{aug}} = x + \eta$ where η is sampled from a uniform distribution on $[-k, k]^d$ (where we can have $k > \epsilon$). Then the adversarially perturbed samples have the following form:

$$x_{\text{N-FGSM}} = x_{\text{aug}} + \alpha \cdot \text{sign}(\nabla_{x_{\text{aug}}} \mathcal{L}(f_{\theta}(x_{\text{aug}}), y)). \quad (4.3)$$

This construction corresponds to augmenting the clean sample x with the perturbation defined in (4.2) where ψ is the identity map and Ω is the uniform distribution spanning $[-k, k]^d$. We detail our full adversarial training procedure in Algorithm 2. In what follows, we analyse the effect of treating the noise as data augmentation as opposed to treating it as a random step within the attack. In particular, we show that clipping around the clean sample x (as done in RS-FGSM) can strongly degrade the robustness of the network. Moreover, we show that as we increase the ϵ radii of adversarial attacks, we need stronger noise perturbations than previously used to prevent CO.

Clipping around clean sample x hinders the effectiveness of perturbations.

We analyse two variants, one where perturbations are clipped around the clean sample x (as done in previous methods) and another where no clipping is applied. In Figure 4.2 (left), we report the robust accuracy using PGD-50-10 (*i.e.* PGD attack with 50 iterations and 10 restarts) with $\epsilon = 8/255$ and observe that clipping significantly degrades the effectiveness of FGSM training. To understand this drop,

consider the following perturbations; **(1)** a baseline perturbation where we only use noise $\delta_{\text{random}} = \psi(\eta)$ and **(2)** a perturbation that combines noise with FGSM $\delta_{\text{full}} = \psi(\eta + \alpha \cdot \text{sign}(\nabla_x \mathcal{L}(f_\theta(x + \eta), y)))$. Moreover, we consider two cases in which we either define ψ as a clipping operator or as the identity. We define the effective FGSM step size as the magnitude corresponding to the ratio¹ $\alpha_{\text{effective}} = \|\delta_{\text{full}} - \delta_{\text{random}}\|_2 / \|x\|_2$ which measures the contribution of the FGSM step in the final perturbation compared to simply following the noise direction η . In Figure 4.2 (middle), we observe that the clipping operator reduces the effective magnitude of FGSM, thus, perturbations become more similar to only using random noise. On the other hand, without clipping we always take the full step in the FGSM direction. This highlights the trade-off between noise magnitude and attack strength discussed above.

Larger noise is also necessary to prevent CO. As discussed above, previous work did not investigate the effects of using noise perturbations potentially larger than the attack strength. However, we empirically find that increasing the noise magnitude is key to avoiding CO. In particular, as seen in Figure 4.2 (right), when no clipping is performed, it is crucial that we augment with larger noise magnitude in order to prevent CO in all settings. We find the noise magnitude of $k = 2\epsilon$ to work well in most of our experiments, however, a more extensive hyperparameter tuning might improve our results further.

Note that these results are contrary to previous intuitions: Andriushchenko and Flammarion (2020) suggested that the random step in RS-FGSM is not important per se, arguing that its main role is reducing the ℓ_2 norm of the perturbations, so that the loss remains to be approximately locally linear. In contrast, N-FGSM perturbations are larger on expectation than those of RS-FGSM, while they do not suffer from CO (refer Appendix B.14). We believe that our findings will lead to a better understanding of the role of noise in avoiding CO in future work. Moreover, in Section 4.6 we conduct extensive analyses to show that, despite N-FGSM obtains larger perturbations, clean

¹The denominator $\|x\|_2$ is simply to normalize the ℓ_2 -norm and be comparable to the FGSM step size α .

Algorithm 2 N-FGSM adversarial training

```

1: Inputs: epochs  $T$ , batches  $M$ , radius  $\epsilon$ , step-size  $\alpha$  (default:  $\epsilon$ ), noise magnitude
    $k$  (default:  $2\epsilon$ ).
2: for  $t = 1, \dots, T$  do
3:   for  $i = 1, \dots, M$  do
4:      $\eta \sim \text{Uniform}[-k, k]^d$ 
5:      $x_{\text{aug}}^i = x^i + \eta$  // Augment sample with additive noise.
6:      $x_{\text{N-FGSM}}^i = x_{\text{aug}}^i + \alpha \cdot \text{sign}(\nabla_{x_{\text{aug}}^i} \mathcal{L}(f_{\theta}(x_{\text{aug}}^i), y^i))$  // N-FGSM augmented
       sample.
7:      $\nabla_{\theta} = \nabla_{\theta} \mathcal{L}(f_{\theta}(x_{\text{N-FGSM}}^i), y^i)$  // Compute gradients of model's weights
8:      $\theta = \text{optimizer}(\theta, \nabla_{\theta})$  // Standard weight update, (e.g. SGD)
9:   end for
10: end for

```

accuracy does not degrade and other methods do not benefit from simply increasing the strength of their attacks.

Why does noise augmentation avoid CO? Andriushchenko and Flammarion (2020) found that after CO, the gradients of the loss with respect to the input around clean samples became strongly misaligned, which is a sign of non-linearity. Moreover, Kim et al. (2021) showed that the loss surface of models suffering from CO appears distorted, *i.e.* there is a sharp peak in the loss surface along the FGSM direction, which seems to render FGSM ineffective (observe from Figure 4.5 how after after CO, visually, FGSM perturbations change drastically). In order to prevent CO, GradAlign explicitly regularizes the loss surface so it remains linear. To investigate further, we plot the loss surface at the end of training for different methods (see Figure B.8 in Appendix) and find that, while FGSM or RS-FGSM lead to a distorted loss, N-FGSM obtains a non-distorted loss surface similar to that obtained by GradAlign regularizer. Thus, it seems that adding strong noise-augmentations implicitly regularizes the loss landscape, leading to more effective single-step attacks. This aligns with previous work that theoretically link noise augmentations with a regularizer that encourages Lipschitzness (Bishop, 1995).

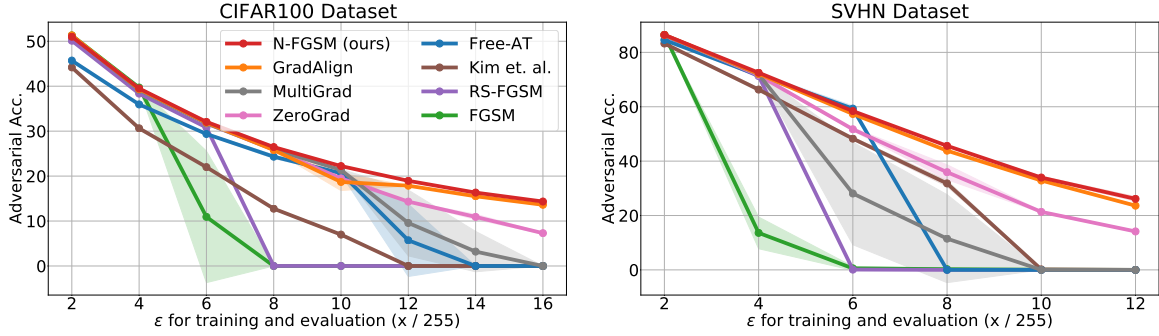


Figure 4.3: Comparison of single-step methods on CIFAR-100 (left) and SVHN (right) with PreactResNet18 over different perturbation radius (ϵ is divided by 255). Our method, N-FGSM, can match or surpass prior art results while *reducing the cost by a $3\times$ factor*. Adversarial accuracy is based on PGD-50-10 and experiments are averaged over 3 seeds. Legend is shared among plots.

4.5 Robustness Evaluations and Comparisons

We compare N-FGSM against several adversarial training methods, on a broad range of $\epsilon - l_\infty$ radii. Following Wong et al. (2020), we evaluate adversarial robustness on CIFAR-10/100 (Krizhevsky et al., 2009) and SVHN (Netzer et al., 2011) with PGD-50-10 attacks, using both PreactResNet18 (He et al., 2016b) and WideResNet28-10 (Zagoruyko and Komodakis, 2016). Evaluations with AutoAttackCroce and Hein (2020) are also in Appendix B.7.

4.5.1 Comparison against Single-Step Methods

We start by comparing N-FGSM against other single-step methods. Note that not all single-step methods are equally expensive, since they may involve more or less computationally demanding operations. For instance, GradAlign uses a regularizer that is considerably expensive, while MultiGrad requires evaluating input gradients on multiple random points. For a comparison of training costs of different single-step methods, we refer the reader to Figure 4.1 (right). We use RS-FGSM and Free-AT with the settings recommended by Wong et al. (2020). We apply GradAlign with hyperparameters reported in the official repository². ZeroGrad and Kim et al.

²<https://github.com/tml-epfl/understanding-fast-adv-training/>

(2021) do not have a recommended set of hyperparameters; for a fair comparison we ablate them and select the ones with highest adversarial accuracy (for every ϵ and dataset). We train on CIFAR-10/100 for 30 epochs and on SVHN for 15 epochs with a cyclic learning rate. Only for Free-AT, we use 96 and 48 epochs for CIFAR-10/100 and SVHN, respectively, to obtain comparable results following Wong et al. (2020). CIFAR-10 results are in Figure 4.1 (middle), whereas CIFAR-100 and SVHN are in Figure 4.3.

As observed in Figure 4.1 and Figure 4.3, FGSM and RS-FGSM suffer from CO for larger ϵ attacks on all reported datasets. For instance, RS-FGSM fails against attacks with $\epsilon = 8/255$ on CIFAR-10 and CIFAR-100 and against $\epsilon = 6/255$ on SVHN. With appropriate hyperparameters, ZeroGrad is able to consistently avoid CO. However, it obtains sub-par robustness compared to N-FGSM and GradAlign, especially against large ϵ attacks. Neither MultiGrad nor Kim et al. (2021) avoid CO in all settings despite being more expensive. Free-AT also suffers from CO on all three datasets as also observed by Andriushchenko and Flammarion (2020). In contrast, N-FGSM avoids CO on all datasets, achieving comparable or superior robustness to GradAlign *while being 3 times faster*.

4.5.2 Comparison against Multi-Step Attacks

In Section 4.5.1, we compared the performance of single-step methods and observed that N-FGSM is able to match or surpass the state-of-the-art method, *i.e.* GradAlign, while reducing the computational cost by a factor of 3. In this section, we compare the performance of N-FGSM against multi-step attacks. In particular, we compare against PGD-2 with $\alpha = \epsilon/2$ and PGD-10 with $\alpha = 2/255$, keeping the same training settings as described in Section 4.5.1. PGD-x denotes x iterations and no restarts.

In Figure 4.4, we observe that PGD-2, despite being a multi-step method, still suffers from CO for larger ϵ as opposed to our proposed N-FGSM. On the other hand, despite achieving comparable clean accuracies, there is a gap in adversarial accuracies between

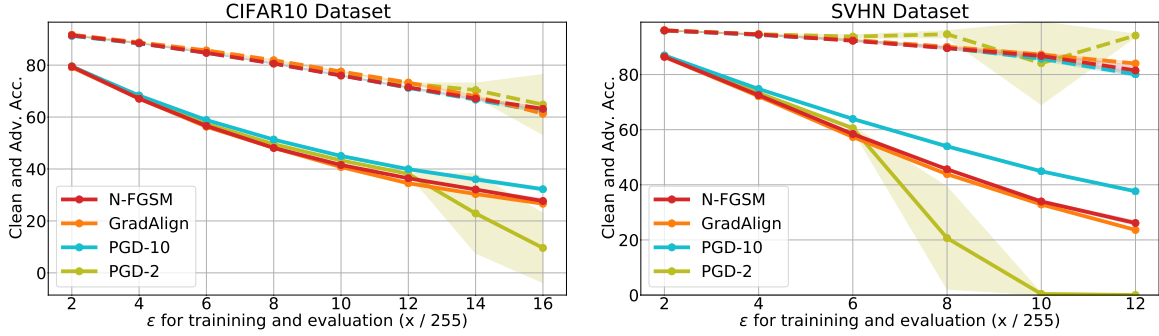


Figure 4.4: Comparison of N-FGSM and GradAlign with multi-step methods on CIFAR-10 (Left) and SVHN (Right) with PreactResNet18 over different perturbation radii (ϵ is divided by 255). Despite all methods achieving comparable clean accuracy (dashed lines), there is a gap in robust accuracy between PGD-10 and single-step methods. However, note that PGD-10 is $10\times$ more expensive than N-FGSM. Adversarial accuracy is based on PGD-50-10 and experiments are averaged over 3 seeds.

PGD-10, and other single-step methods that grows with perturbation size. This can be partially expected since the search space for adversaries grows exponentially with ϵ ; and PGD, with more iterations, can explore it more thoroughly. Nevertheless, *computing a PGD-10 attack is $10\times$ more expensive to N-FGSM*. An important direction for future work would be addressing this gap and analysing, both theoretically and empirically, whether single-step methods can match the performance of their multi-step counterparts.

4.5.3 Analysis of Gradients and Adversarial Perturbations

To gain further insights into CO, we visually explore the perturbations generated with FGSM, RS-FGSM, N-FGSM, and PGD-10 attacks. We show that N-FGSM generates perturbations that exhibit behavior similar to PGD-10. In particular, for a given test sample, we average the adversarial perturbations (δ) and gradients across several epochs at the beginning of training (Epoch 2 to 8) and at the end (Epoch 24 to 30) and visualise them in Figure 4.5 (see also Figure B.6 in Appendix for more examples). We observe that, during early stages in training all, methods generate consistent and interpretable δ . However, after CO, FGSM and RS-FGSM generate δ that are harder to interpret, similarly to their gradients. On the other hand, we observe that N-FGSM

provides consistent and interpretable δ throughout training, similar to those generated by PGD-10. This provides further evidence that N-FGSM enjoys similar properties to the more expensive PGD-10 training.

Figure 4.5 analyzes the gradients and δ throughout the test set. Aside from loosing interpretability, post-CO the gradient norm increases by several orders of magnitude for FGSM and RS-FGSM while it remains low for N-FGSM and PGD-10. We also compute the effective rank³ of δ for each example before and after CO to measure the consistency of δ before and after CO. We consider three training intervals, (Epoch 2 to 8): before CO for all methods; (Epoch 16 to 22): after FGSM suffers CO but not RS-FGSM; (Epoch 24 to 30): after both FGSM and RS-FGSM suffer CO. Prior to CO, PGD-10 has a larger effective rank (*i.e.* the perturbations span a larger subspace) than FGSM and RS-FGSM. N-FGSM has the highest effective rank, arguably due to the higher noise magnitude. Note that RS-FGSM, which has a smaller noise magnitude and clipping, also has a larger effective rank than FGSM, however, the difference is much lower. When either FGSM or RS-FGSM suffer from CO, the effective rank of their δ increases significantly above that of PGD-10 and N-FGSM. This would suggest that δ loose consistency after CO and is aligned with our visualizations in Figure 4.5. All of these show properties of δ and gradients that are consistent across methods (N-FGSM and PGD) that avoid CO and different from methods like RS-FGSM and FGSM, which suffer from CO.

4.6 Increasing Adversarial Perturbations

In Section 4.4, we observed that removing clipping and increasing the noise magnitude were both necessary for the improved performance of N-FGSM. However, as discussed in Theorem B.14.2 this will result in an increase of the squared norm of the training perturbation $\delta_{\text{N-FGSM}}$ as compared to FGSM. In this section, we perform further

³We compute effective rank as the number of singular vectors required to explain 90% of the variance.

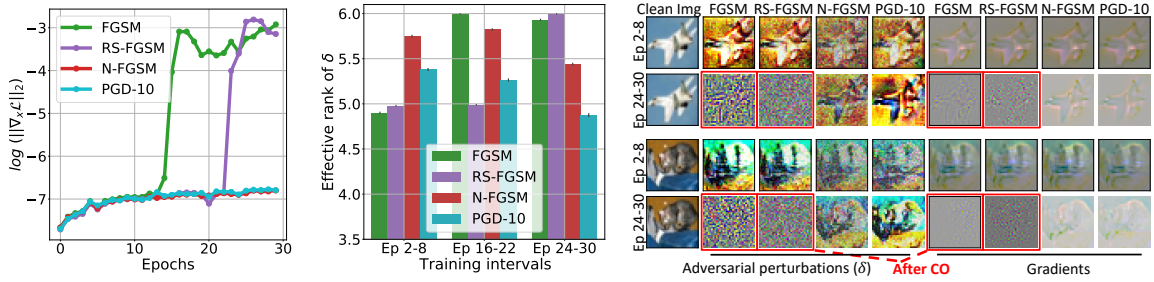


Figure 4.5: **Left:** Mean ℓ_2 norm of per-sample-gradients across all test set samples. After CO, both FGSM and RS-FGSM gradients increase by several orders of magnitude. **Middle:** Effective rank of perturbations (δ) across three training intervals *Ep 2-8* before CO for all methods; *Ep 16-22*: after FGSM presents CO but not RS-FGSM; *Ep 24-30*: after both FGSM and RS-FGSM had CO. **Right:** Visualization of δ and gradients averaged across several epochs at the beginning (top) and end (bottom) of training. After CO, FGSM and RS-FGSM obtain δ and gradients that are hard to interpret.

ablations to corroborate that it is indeed the increase in noise magnitude – and not the mere increase of the perturbation’s magnitude – that helps to stabilize N-FGSM.

Increasing α alone is not sufficient. N-FGSM combines a noise perturbation with an FGSM step. Thus, we can increase the perturbation magnitude by increasing any of the two. This begs the question: Is it sufficient to increase the N-FGSM step-size α to avoid CO without adding any noise? We observe in Figure 4.6 (A) that training without noise (essentially, FGSM) leads to CO, with robust accuracy equal to zero, even for large values of α . This indicates that it is not an increase in the perturbation norm, but the combination with noise which plays an essential role in circumventing CO for N-FGSM.

Increasing α requires adjusting the noise magnitude. As observed in Figure 4.6 (A), increasing α for N-FGSM leads to CO if the noise magnitude is not large enough. For example, while a noise magnitude $k = 1\epsilon$ and an adversarial step size $\alpha = 1.25\epsilon$ yield a robust accuracy of 49.68%, increasing α to 1.5ϵ while keeping the same noise magnitude results in CO – with robust accuracy equal to zero. This further suggests that an increase in the adversarial step-size α requires a commensurate increase in the noise magnitude. We find that setting the noise magnitude $k = 2\epsilon$ works well for most settings.

Larger noise perturbations preserve clean accuracy. Increasing the norm of training perturbations by increasing α results in a drop in the clean accuracy (discussed later in Section 4.7). This has also been observed in prior works (Wong et al., 2020). However, we show in Figure 4.4 that the clean accuracy for N-FGSM is similar to that of GradAlign, despite the magnitude of the perturbations being larger. We ablate the effects of adversarial and noise perturbations on the clean accuracy in Figure 4.6 (B): we observe that augmenting training samples with noise alone (*i.e.* $\alpha = 0$) has a much milder effect on the clean accuracy than augmenting in an adversarial direction. In general, increasing noise is more forgiving on the clean accuracy than increasing the adversarial step size. This is not surprising, considering that moving in random directions along the input space has a significantly lower impact on the loss than moving along the FGSM direction (see Figure B.8 in the Appendix) and that training with noise alone does not provide any significant robustness against larger attacks (for a more detailed ablation, see Appendix Figure B.10).

Other methods do not benefit from larger training ϵ . As previously mentioned, N-FGSM perturbations have ℓ_∞ -norm larger than ϵ . We have seen that the benefits of N-FGSM can not be reproduced by simply increasing α without increasing the noise. However, for the sake of completeness, we also ablate other single-step baselines by using a larger ϵ during training, while testing with a fixed $\epsilon = 8/255$ on CIFAR10. We observe that increasing ϵ_{train} seems to lead to a decrease in robustness for most methods; for instance, PGD-50-10 accuracy for RS-FGSM drops from 46.08 ± 0.18 when training with $\epsilon = 8/255$ to 0.0 ± 0.0 with $\epsilon = 12/255$. In two cases (GradAlign and MultiGrad), we observe a small increase, with the highest increase being for GradAlign, which improves from 48.14 ± 0.15 to 50.6 ± 0.45 ; yet, the clean accuracy drops from 81.9 ± 0.22 to 73.29 ± 0.23 . This is similar to increasing α for N-FGSM (see Figure 4.6 (C)). However, this is tied to a significant degradation of clean accuracy. All in all, taking into account both clean and robust accuracy, we conclude that all single-step baselines suffer from either CO or a severe degradation in their clean accuracy when increasing the training ϵ . Full results are presented in Table B.1 in Appendix.

4.7 Additional Ablations

Hyperparameter selection. While FGSM relies on a fixed step-size (*i.e.* $\alpha = \epsilon$), Wong et al. (2020) explored different values of α for RS-FGSM, finding that an increase of the step-size improves the adversarial accuracy – up to a point where CO occurs. We also ablate the value of α for N-FGSM in Figure 4.6 (C). We find that by increasing the noise magnitude, N-FGSM can use larger α values than RS-FGSM, without suffering from CO. This leads to an increase in the adversarial accuracy at the expense of a decrease in the clean accuracy. In light of this trade-off, we also use $\alpha = \epsilon$ for N-FGSM. Regarding the noise hyperparameter k , we find that $k = 2\epsilon$ works in all but one SVHN experiment ($\epsilon = 12$, in which we set $k = 3\epsilon$). In comparison, GradAlign regularizer hyperparameter or ZeroGrad quantile value need to be modified for every radius with a noticeable shift between CIFAR-10 and SVHN hyperparameters, suggesting they may require additional tuning when applied to novel datasets.

Long vs fast training schedules. Throughout our experiments, we used the RS-FGSM training setting introduced in Wong et al. (2020). However, Rice et al. (2020) suggest that a longer training schedule coupled with early stopping may lead to a boost in performance. Kim et al. (2021) and Li et al. (2020) report that longer training schedules increase the chances of CO for RS-FGSM and that this limits its performance. We test the longer training schedule with N-FGSM and find that it does not suffer from CO. However, it does suffer from *robust overfitting*, *i.e.* adversarial accuracy on the training set is larger than on the test set as described in Rice et al. (2020) for PGD-10. Notice the difference between the robust accuracy of the final and best models in Figure 4.6 (D). Interestingly, although we observe a slight increase in performance when using the long training schedule, we find the fast training schedule to be remarkably competitive. See more results in Appendix B.4, including a comparison to GradAlign.

Experiments with WideResNet28-10. We also compare the performance of all methods on WideResNet28-10 (Zagoruyko and Komodakis, 2016) architecture in

4. Make some noise: Reliable and Efficient Single-Step Adversarial Training

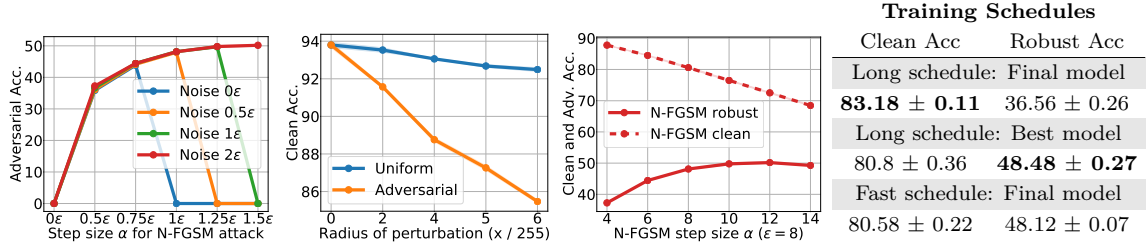


Figure 4.6: Different ablations on N-FGSM parameters and training schedule. From left to right: **A:** Adversarial accuracy when varying step-size α and noise magnitude k ($\epsilon = 8$). Increasing α does not suffice to prevent CO, we must also increase the noise magnitude. **B:** Clean accuracy after training with random or adversarial perturbations. With comparable radius, random perturbations have a much milder effect than adversarial. **C:** Ablation of step size α in N-FGSM $\epsilon = 8, k = 2\epsilon$. As we increase the magnitude of the FGSM perturbation we observe an increase in robustness coupled with a drop on the clean accuracy. **D:** Comparison of the “fast” training schedule from Wong et al. (2020) and “long” training schedule described in Rice et al. (2020). N-FGSM shows robust overfitting but not CO with the long schedule. Adversarial accuracy is based on PGD-50-10 and experiments are averaged over 3 seeds.

Figure B.2 and Figure B.3 in Appendix. As in the experiments with PreActResNet18, N-FGSM obtains the best performance/cost trade-off. We had to increase the regularizer hyperparameter for GradAlign (compared to the settings for PreActResNet18) in order to prevent CO on CIFAR-100 and, to our surprise, *we could not find a competitive hyperparameter setting for GradAlign on the SVHN dataset* for $\epsilon \geq 6$. We tried both increasing the regularizer hyperparameter and decreasing the step size α , but some or all runs led to models close to a constant classifier for each setting. We do not claim that GradAlign will not work, but finding a good configuration might require further tuning. The default configuration for N-FGSM ($\alpha = \epsilon$, $k = 2\epsilon$) works well in all settings except for $\epsilon = 16$ on CIFAR-10 and $\epsilon = 10, 12$ on SVHN. For CIFAR-10, we increase the noise magnitude to $k = 4\epsilon$. For SVHN, we find that decreasing α works better than increasing the noise. In both cases, N-FGSM yields nontrivial adversarial accuracy.

Experiments on Imagenet. We present results on the Imagenet dataset Krizhevsky et al. (2012) in Table 4.1. Due to the high computational demands of Imagenet training and testing we focus on the main baselines of comparable cost to FGSM. Namely FGSM, RS-FGSM and N-FGSM. We observe that FGSM presents CO for $\epsilon = 6/255$ while neither RS-FGSM nor N-FGSM present CO. However, N-FGSM has better robustness.

Table 4.1: Clean accuracy (top) and PGD50-10 accuracy (bottom) of N-FGSM and other same-cost baselines on Imagenet dataset. We observe that FGSM presents CO for $\epsilon = 6/255$ while both RS-FGSM and N-FGSM avoid CO. N-FGSM has consistently better robustness than baselines.

	$\epsilon = 2/255$	$\epsilon = 4/255$	$\epsilon = 6/255$
FGSM	54.72	48.50	48.55
	38.21	25.86	0.08
RS-FGSM	56.29	50.81	47.67
	36.86	25.12	16.49
N-FGSM	54.39	47.56	47.70
	38.07	26.28	17.12

For instance, at $\epsilon = 6/255$ N-FGSM obtains PGD50-10 accuracy of 17.12% while RS-FGSM yields 16.5% and FGSM 0.08% (due to CO). Thus, N-FGSM also avoids CO in ImageNet, improving robustness over same-cost baselines. For experimental details refer to Appendix B.12.

Combining N-FGSM with additional regularizers. Recent works (Sriramanan et al., 2020, 2021) have been proposed to improve performance of single step methods at moderate perturbation radii $\epsilon = 8/255$. However, we observe that with the default settings (which use a version of RS-FGSM with Bernoulli noise) they lead to CO for larger ϵ . Then we compare them with N-FGSM + Regularizer where we apply their proposed regularizers to N-FGSM. If we apply GAT or NuAT regularizers to N-FGSM then we do not observe CO and usually a boost in performance. For instance, at $\epsilon = 10/255$, GAT has a robust acc (with PGD50-10) of 43.34 ± 0.23 while N-FGSM+GAT regularizer obtains 44.97 ± 0.07 , in comparison plain N-FGSM has 41.56 ± 0.16 . This is extremely compelling as it suggests N-FGSM can be combined with other regularizers designed to improve FGSM performance and mutually benefit each other. Full results of the comparison are presented in Table B.5 in Appendix B.11

4.8 Conclusion

In this work, we explore the role of noise and clipping in single-step adversarial training. Contrary to previous intuitions, we show that increasing the noise magnitude and removing the $\epsilon - \ell_\infty$ constraint leads to an improvement in adversarial robustness while maintaining a competitive clean accuracy. These findings led us to propose N-FGSM, a simple and effective approach that can match or surpass the performance of GradAlign (Andriushchenko and Flammarion, 2020), while achieving a $3\times$ speed-up.

We perform an extensive comparison with other relevant single-step methods, observing that all of them achieve sub-optimal performance and most of them are not able to avoid CO for larger ϵ attacks. Moreover, we also analyze gradients and adversarial perturbations during training and observe that they have a similar behaviour for N-FGSM and PGD-10 as opposed to other methods that present CO such as FGSM and RS-FGSM. However, despite impressive improvements of single-step adversarial training methods, there is still a gap between single-step and multi-step methods such as PGD-10 as we increase the ϵ radius. Therefore, future work should put an emphasis on formally understanding the limitations of single-step adversarial training and explore how, if possible, this gap can be reduced.

5

Reliability in Semantic Segmentation: Are We on the Right Track?

Abstract

Motivated by the increasing popularity of transformers in computer vision, in recent times there has been a rapid development of novel architectures. While in-domain performance follows a constant, upward trend, properties like robustness or uncertainty estimation are less explored—leaving doubts about advances in model reliability. Studies along these axes exist, but they are mainly limited to classification models. In contrast, we carry out a study on semantic segmentation, a relevant task for many real-world applications where model reliability is paramount. We analyze a broad variety of models, spanning from older ResNet-based architectures to novel transformers and assess their reliability based on four metrics: robustness, calibration, misclassification detection and out-of-distribution (OOD) detection. We find that while recent models are significantly more robust, they are not overall more reliable in terms of uncertainty estimation. We further explore methods that can come to the rescue and show that improving calibration can also help with other uncertainty metrics such as misclassification or OOD detection. This is the first study on modern segmentation models focused on both robustness and uncertainty estimation and we hope it will help practitioners and researchers interested in this fundamental vision task. Code available at <https://github.com/naver/reliis>.

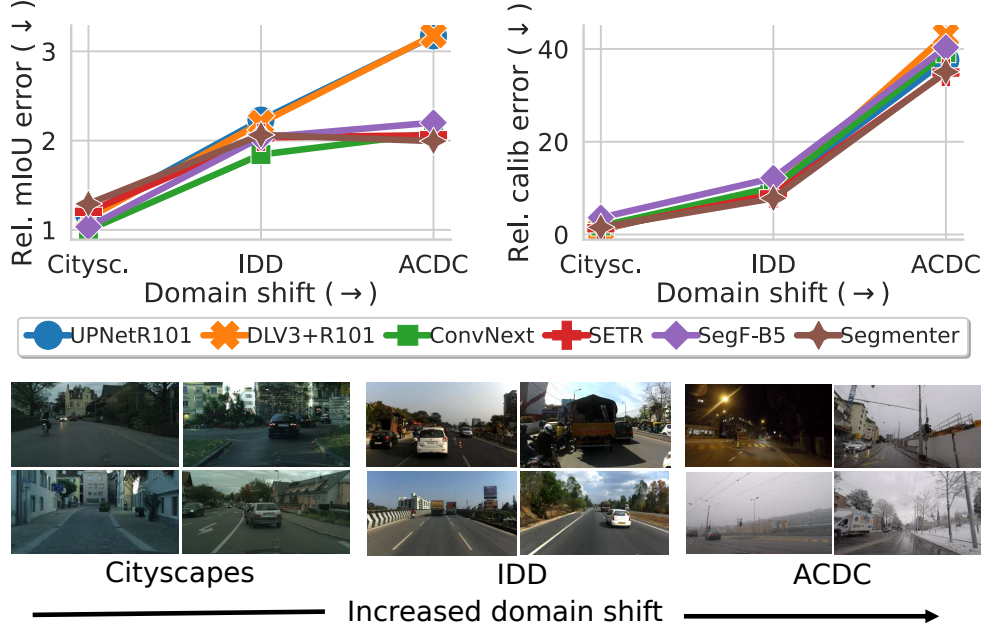


Figure 5.1: Top: mIoU and ECE vs. domain shift. Errors are normalized with respect to the lowest error on the training distribution (Cityscapes). We compare recent segmentation models, both transformer-based (SETR (Zheng et al., 2021), SegFormer (Xie et al., 2021) and Segmenter (Strudel et al., 2021)) and convolution-based (ConvNext (Liu et al., 2022)) with ResNet baselines (UPerNet(Xiao et al., 2018) and DLV3+(Chen et al., 2017b)). All recent models (both transformers and CNNs) are remarkably more robust than ResNet baselines (whose lines in mIoU overlap), however, ECE increases sharply for all methods. **Bottom:** Sample images for each dataset.

5.1 Introduction

Humans tend to overestimate their abilities, a cognitive bias known as Dunning-Kruger effect (Kruger and Dunning, 1999). Unfortunately, so do deep neural networks. Despite impressive performance on a wide range of tasks, deep learning models tend to be overconfident—that is, they predict with high-confidence even when they are wrong (Guo et al., 2017). This effect is even more severe under domain shifts, where models tend to underperform in general (Ovadia et al., 2019; Recht et al., 2019; Hendrycks and Dietterich, 2019).

While these vulnerabilities affect deep models in general, they are often studied for classification models and are comparably less explored for semantic segmentation, a fundamental task in computer vision that is key to many critical applications such as autonomous driving and AI-assisted medical imaging. In those applications, domain

shifts are more the rule than the exception (*e.g.*, changes in weather for a self-driving car or differences across patients for a medical imaging system). Therefore, brittle performance and overconfidence under domain shifts are two important and challenging problems to address for a safe deployment of artificial intelligence systems in the real world.

With that in mind, we argue that a *reliable* model should *i*) be robust to domain shifts and *ii*) provide good uncertainty estimates. The core goal of this study is providing an answer to the following, crucial question: **are state-of-the-art semantic segmentation models improving in terms of robustness *and* uncertainty estimation?**

To shed light on this, we evaluate a large body of segmentation models, assessing their in-domain (ID) *vs.* out-of-domain (OOD) prediction quality (**robustness**) together with their calibration, misclassification detection and OOD detection (**uncertainty estimation**).

We argue that a study of this kind is crucial to understand whether research on semantic segmentation is moving in the right direction. Following the rise of transformer architectures in computer vision (Dosovitskiy et al., 2021; Touvron et al., 2021; Carion et al., 2020; Liu et al., 2021), several studies have compared recent self-attention and CNN-based *classification* models in terms of robustness (Bhojanapalli et al., 2021; Naseer et al., 2021; Bai et al., 2021; Paul and Chen, 2022; Mao et al., 2022; Liu et al., 2022) and predictive uncertainty (Minderer et al., 2021; Pinto et al., 2022). Yet, when it comes to *semantic segmentation*, prior studies (Xie et al., 2021; Zhou et al., 2022) only focused on robustness, using synthetic corruptions as domain shifts (*e.g.*, blur, noise) (Kamann and Rother, 2020). In contrast, we consider natural, realistic domain shifts and study segmentation models both in terms of robustness and uncertainty, leveraging datasets captured in different conditions—see Figure 5.1 (bottom).

Task-specific studies are important, since task-specific architectures and learning algorithms may carry different behaviors and some observations made for classification might not hold true when switching to segmentation. For instance, contrary to

Minderer *et al.* (Minderer et al., 2021), we observe that improvements in calibration are far behind those in robustness, see Figure 5.1 (top). Furthermore, previous analyses only consider simple calibration approaches (Guo et al., 2017) while assessing model reliability; in contrast, we make a step forward and explore content-dependent calibration strategies (Gong et al., 2021; Ding et al., 2021), which show promise to improve reliability out of domain.

Our analysis allows us individuating in which directions we are improving and in which we are lagging behind. This is the first work to systematically study robustness and uncertainty under domain shift for a large suite of segmentation models and we believe it can help practitioners and researchers working on semantic segmentation. We summarize our main observations in the following.

i) Remarkable improvements in robustness, but poor in calibration. Under domain shifts, recent segmentation models perform significantly better (in terms of mIoU)—with larger improvements for stronger shifts. Yet, OOD calibration error increases dramatically for all models.

ii) Content-dependent calibration (Ding et al., 2021) can improve OOD calibration, especially under strong domain shifts, where models are poorly calibrated.

iii) Misclassification detection shows different model ranking in and out of domain. When tested in domain, recent models underperform the ResNet baseline. As the domain shift increases, recent models take the lead.

iv) OOD detection is inversely correlated with performance. Indeed, a small ResNet-18 backbone performs best.

v) Content-dependent calibration (Ding et al., 2021) can improve OOD detection and misclassification out of domain. We observe a significant increase in misclassification detection under strong domain shifts after improving calibration. We also observe improvements for OOD detection, albeit milder.

	Sem. segm.	Robust performance	Uncertainty estimation	Natural shifts	OOD calib methods
Kamann and Rother (2020)	✓	✓			
Bhojanapalli et al. (2021)		✓		✓	
Xie et al. (2021)	✓	✓			
Naseer et al. (2021)		✓			
Bai et al. (2021)		✓		✓	
Minderer et al. (2021)		✓	✓	✓	
Paul and Chen (2022)		✓		✓	
Mao et al. (2022)		✓		✓	
Liu et al. (2022)		✓		✓	
Zhou et al. (2022)	✓	✓			
Pinto et al. (2022)		✓	✓	✓	
<i>Ours</i>	✓	✓	✓	✓	✓

Table 5.1: Studies of recent architectures. While several prior works studied robustness and uncertainty of transformer- and CNN- based *classifiers*, studies on *segmentation* limited to robustness. This is the first study assessing robustness *and* uncertainty of modern segmentation models. Moreover, we consider natural domain shifts and are the only analysis to include content-dependent methods Gong et al. (2021); Ding et al. (2021) to improve calibration in OOD settings.

5.2 Related work

We study robustness and uncertainty in semantic segmentation. In doing so, we touch several fields, which we cover in the following. We further discuss related studies.

Segmentation models. Modern segmentation pipelines typically consist of encoder-decoder architectures (Csurka et al., 2022; Badrinarayanan et al., 2017; Noh et al., 2015; Ronneberger et al., 2015). Decoders are usually designed *ad hoc* for segmentation, with DeepLab (Chen et al., 2017a,b, 2018) and UPerNet (Xiao et al., 2018) being two of the most prominent. On the other hand, the evolution of encoders has been closely related to that of classification models, with ResNet (He et al., 2016a) being one of the most popular for years. The rise of transformers in computer vision (Dosovitskiy et al., 2021) has led to a flurry of works leveraging self-attention for segmentation (Zheng et al., 2021; Strudel et al., 2021; Xie et al., 2021). Novel convolutional architectures

inspired by transformers have also risen (Liu et al., 2022). We compare several recent segmentation models against ResNet baselines, in terms of both robustness and uncertainty.

Robustness. The brittleness of neural networks to changes in the input domain is a well-studied problem and many sub-formulations exist (Taori et al., 2020). Robustness against synthetic shifts takes into account samples crafted by artificially altering images, for example injecting noise or blur (corruption robustness (Hendrycks and Dietterich, 2019; Kamann and Rother, 2020)), or crafting imperceptible perturbations to induce model failure (adversarial robustness (Goodfellow et al., 2015)). Robustness against *natural* shifts focuses on changes that may arise naturally, without human intervention (Recht et al., 2019; Hendrycks et al., 2021).

In this work we are interested in comparing the robustness of different off-the-shelf segmentation models under *natural* domain shifts, since these are particularly relevant in real-world applications. In particular, we focus on semantic segmentation of urban scenes, hence, we evaluate models on samples from unseen geographical locations (Varma et al., 2019) and weather conditions (Sakaridis et al., 2021). Segmentation robustness against natural shifts has been studied before (Yue et al., 2019; Volpi et al., 2022), yet not in tandem with uncertainty and within a large-scale study taking into consideration several recent models.

Uncertainty. Guo *et al.* (Guo et al., 2017) have shown that deep models are overconfident. They have proposed a simple, yet effective solution known as temperature scaling (TS) where the output logits are divided by a temperature parameter before the softmax layer. Other calibration methods have been proposed (*e.g.*, (Naeini et al., 2015; Kull et al., 2019; Gupta et al., 2020)), but TS is still very popular due to its simplicity and the fact that it does not alter predictions.

Calibration with TS is effective in ID settings; yet, Ovadia *et al.* (Ovadia et al., 2019) have shown that model calibration degrades significantly out of domain. Some methods have been proposed that address this problem (Pampari and Ermon, 2020; Park et al., 2020; Wang et al., 2020b), by assuming access to unlabeled OOD images beforehand.

On the other hand, Gong *et al.* (Gong et al., 2021) have proposed methods that improve OOD calibration without any data from the target domain. They propose to cluster the calibration set in different “domains” and find a different temperature value for each. At test time, images are calibrated using the temperature from the closest cluster. *Ad hoc* for semantic segmentation, Ding *et al.* (Ding et al., 2021) propose a content-dependent calibration strategy that learns a small calibration network to predict a temperature for each pixel in an image.

In our study we test ID and OOD performance of several calibration methods, focusing on techniques that do not require access to OOD samples (Guo et al., 2017; Gong et al., 2021; Ding et al., 2021)—as generally robustness is evaluated on unseen domains (Hendrycks and Dietterich, 2019; Kamann and Rother, 2020; Taori et al., 2020; Pinto et al., 2022)

Previous analyses. In Table 5.1 we compare related studies on different aspects of reliability. Several works have suggested that transformer-based *classifiers* are more robust than CNNs (Bhojanapalli et al., 2021; Naseer et al., 2021; Bai et al., 2021; Mao et al., 2022; Paul and Chen, 2022). Yet, the recent ConvNeXt (Liu et al., 2022) has challenged this result and later work have suggested that further investigation is needed (Pinto et al., 2022). Minderer *et al.* (Minderer et al., 2021) have compared calibration of several classifiers, concluding that convolution-free models are more robust *and* better calibrated. In contrast, Pinto *et al.* (Pinto et al., 2022) have compared recent transformers and CNNs, arguing there is “no clear winner”. Some works have compared the robustness of transformers and CNNs for segmentation (Xie et al., 2021; Zhou et al., 2022)—but only against *synthetic* domain shifts. We broadly study robustness *and* uncertainty in segmentation under *natural* domain shifts. Similarly to (Pinto et al., 2022), we do not observe a single model family which is better calibrated in all scenarios. In contrast with (Minderer et al., 2021) though, we observe that robustness and calibration *do not* go hand in hand. This shows that not all trends observed in classification transfer to segmentation, confirming the importance of task-specific studies like ours.

5.3 Experimental settings and preliminaries

5.3.1 Datasets

As discussed in Section 5.2, we use different datasets for semantic segmentation of urban scenes to model natural domain shifts—inspired by prior art (Yue et al., 2019; Choi et al., 2021; Volpi et al., 2022).

Cityscapes (CS) (Cordts et al., 2016a) contains images taken across 50 European cities at day time with overall good weather. Training and validation sets use sequences from disjoint sets of cities. Following this protocol, we further split validation cities into a calibration and a test set. Since CS is a mainstream benchmark in semantic segmentation, we use it as our training set (ID) to leverage available trained weights.

IDD (Varma et al., 2019) was captured in the cities of Hyderabad, Bangalore and their outskirts. Given the different geographical location, it poses a clear domain shift for CS models.

ACDC (Sakaridis et al., 2021) contains images captured in adverse conditions (Fog, Rain, Snow and Night), which translate into strong domain shifts. Similarly to previous work (Pinto et al., 2022; Zhou et al., 2022; Xie et al., 2021), we focus on *covariate* shifts, *i.e.*, changes in the input distribution—keeping the label set fixed. In practice, for OOD settings (IDD and ACDC) we consider the 19 classes from CS, ignoring the others. The only exception is one experiment in Appendix C.12, where we consider label shifts.

5.3.2 Architectures

We implement our models with MMsegmentation (Contributors, 2020). Following prior work (Pinto et al., 2022; Minderer et al., 2021), we use the original training recipes

for each model to compare them at their best. For completeness, we also explore the effects of pre-training dataset and number of training iterations in Appendix C.9.

SETR (Zheng et al., 2021). The first convolution-free segmentation model. It uses a ViT backbone (Dosovitskiy et al., 2021) and different decoders (SETR-Naive, SETR-MLA and SETR-PUP). We use the ViT-Large backbone and analyze all three decoders.

Segmenter (Strudel et al., 2021). Similarly to SETR, it also uses a ViT backbone; yet, unlike the simpler SETR decoders, it carries a transformer-based one. We also test ViT-Large.

SegFormer (Xie et al., 2021). This model incorporates an original self-attention mechanism and several architectural changes to be more efficient. We evaluate all models from this family (B0–B5), gradually increasing the number of parameters.

ConvNeXt (Liu et al., 2022). Convolutional model with changes inspired by transformers. We use ConvNeXt-Large, comparable in size to ViT-Large, with an UPerNet decoder (Xiao et al., 2018).

ResNet-based (He et al., 2019). We use ResNet-V1c model, the default ResNet in MMsegmentation library (Contributors, 2020). Compared to vanilla ResNet(He et al., 2016a) it uses a stem with three 3x3 convs (instead of a 7x7 conv). We use ResNet-18/50/101 models and two popular decoders: **DLV3+** (Chen et al., 2017b) and **UPerNet** (Xiao et al., 2018)

Additionally, in Appendix C.10 we explore **Mask2Former**, an architecture for *universal image segmentation* (Cheng et al., 2022) that does not follow the conventional *logits + softmax* paradigm.

5.3.3 Reliability metrics

We evaluate model reliability on four aspects: robustness, calibration, misclassification detection and OOD detection.

Robustness. We measure robustness by evaluating the standardized mean Intersection-over-Union (mIoU) performance in OOD settings, *i.e.*, on ACDC and IDD. We also provide ID performance, evaluating models on CS.

Calibration. A model is said to be calibrated when the predictive probabilities (*i.e.*, the logits after a softmax) correspond to the true probabilities. For instance, if we group all samples where the predicted probability is 90%, we would expect that 90% of those predictions are correct. The most common calibration metric is the *Expected Calibration Error (ECE)* (Naeini et al., 2015), which looks at the expected difference between the predicted and actual probabilities. To estimate the ECE we quantize the predicted probabilities and compare the accuracy with the mean probability in each bin. Since the binning strategy can affect the results, we test ECE with equally spaced bins (Naeini et al., 2015), equally populated bins (Nixon et al., 2019; Nguyen and O’Connor, 2015) and the Kolmogorov-Smirnov test (Gupta et al., 2020), which gets rid of the binning strategy altogether. We report results with standard ECE, but find that all three aforementioned metrics yield similar conclusions (see Appendix C.1).

Given that in segmentation we have per-pixel predictions, the number of calibration samples explodes (a single CS image contains $2048 \times 1024 \simeq 2\text{M}$ pixels). We ablate the different calibration metrics as we sub-sample the number of pixels per image and observe that 20k pixels per image is enough to estimate the ECE (see Appendix C.2).

Misclassification detection. A desiderata for a reliable model is to assign a larger confidence to correct outputs than incorrect ones¹. In the ideal case, if we sorted all predictions from least to most confident, we would have all the incorrect predictions first and correct ones later. *Misclassification detection* measures how far away are we from such an ideal case. This can be measured with *Rejection-Accuracy curves* (Fumera and Roli, 2002; Hendrycks et al., 2021): we reject samples with low confidence and compute the accuracy *vs.* amount of rejected samples. However, these are biased in favor of better-performing models, since the base accuracy is higher in the first

¹We use max softmax as confidence in the paper; in Appendix C.3 we present similar results with negative entropy.

place. To avoid that, we follow Malinin *et al.* (Malinin et al., 2019) and normalize the area under the curve by that of an oracle and subtracts a baseline score with randomly sorted samples. The resulting metric, known as the Prediction Rejection Ratio (PRR), will be positive if the confidence is better than the random baseline and will have a maximum score of 100% when the model matches the oracle.

Out-of-domain detection. Another important aspect in reliability is that models are aware of their “domain of expertise” (*i.e.*, their training domain). When a sample differs significantly from the training samples, we would expect the model to be more uncertain of its prediction. Similarly to misclassification detection, in OOD detection we try to separate ID from OOD images based on the network confidence. This has broad applicability, *e.g.*, generating alerts for samples too far from the training domain or, connecting to active learning, gathering them for further review and annotation. As in the rest of our work, we consider a whole image to be out of domain if it presents a significant domain shift; that is, we consider CS in domain and IDD/ACDC out of domain. Since we define ID and OOD samples at the image level, we consider the average confidence of all pixels in a given image. We use the Area Under the Receiver Operating Characteristic curve (AUROC) (Murphy, 2012), which goes from 0 to 1 (1 being the best score). Additionally, in Appendix C.12 we consider OOD detection at a region level, considering classes in the IDD dataset not present in CS, *i.e.*, *rickshaw*, *billboard*, *guard rail*, *tunnel* and *bridge*.

5.4 Are modern segmentors more *reliable*?

In the following, we present the main findings of our study on the reliability of semantic segmentation models.

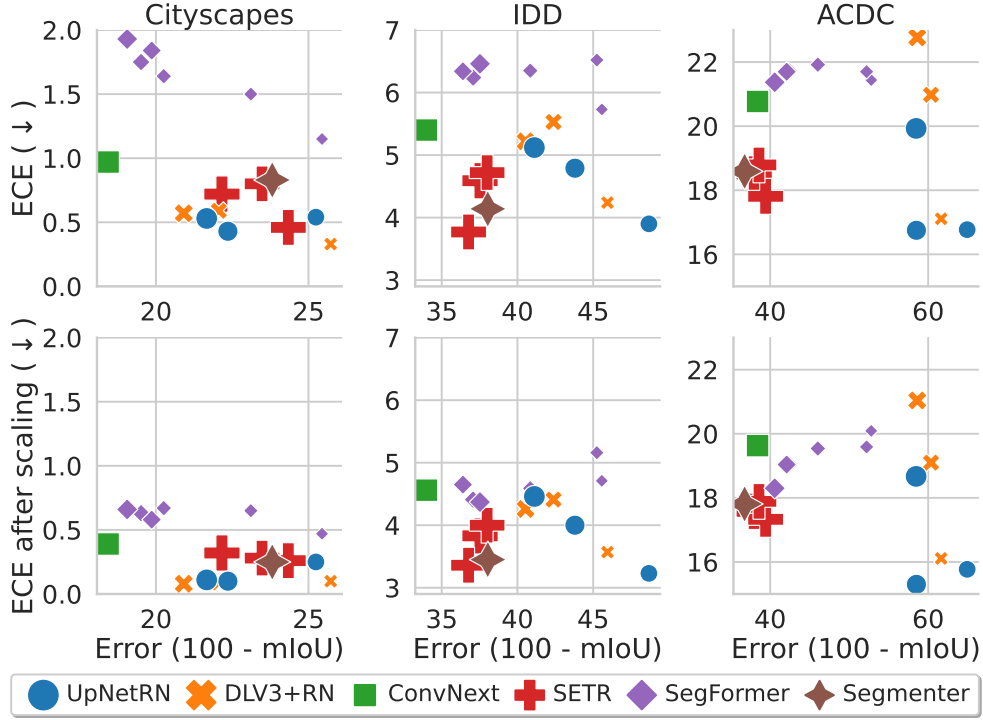


Figure 5.2: Expected mIoU error ($\downarrow$) vs. calibration error ($\downarrow$) before and after TS for different model families (top and bottom, respectively). All models trained and calibrated on CS. Markersize proportional to number of parameters. Notice how TS yields marginal *relative* gains in OOD settings (IDD and ACDC).

5.4.1 Robustness

Since the domain shifts we consider are natural and not synthetically induced, there is no straightforward way to evaluate their strength. To this end, we establish an ordering for the severity of the shift based on performance degradation of the ResNet baselines (DLV3+R101 and UPNetR101), which results in $CS < IDD < ACDC$. This aligns with a qualitative evaluation of the different datasets (see Figure 5.1, bottom, for a few samples). In Figure 5.1 (top left) we present the mIoU error (*i.e.*, $100 - \text{mIoU}$) for several models evaluated on the three datasets. To highlight the loss in performance as we increase the shift, we normalize all errors w.r.t. the best performing CS model (ConvNeXt). The trend is clear: the larger the domain shift, the larger the improvement brought by more recent segmentation models.

We expand on this in Figure 5.2 (top), where we plot the mIoU error (on the x-axis) *vs.*

the calibration error for all the models belonging to the different families (Sec. 5.3.2). The size of the markers is proportional to the number of parameters. Similarly to Figure 5.1, we observe that the gap in mIoU between ResNet baselines and recent models grows larger as we increase the domain shift (*c.f.* marker position on the x-axis). Interestingly, two of the models that perform best under ACDC’s strong shift (SETR and Segmenter) are not significantly better than the ResNet baseline on the training domain (CS). This indicates that, in our setting, only assessing ID performance can hide the real value of newly crafted models and, hence, it is important to evaluate architectures out of domain in order to fully grasp their potential.

While there is no single model family that performs significantly better in all datasets, we can reach the clear conclusion that *all recent models are significantly more robust than well established baselines under natural shifts*.

5.4.2 Calibration error

Off-the-shelf calibration. In Figure 5.1 (top-right) we present the ECE for different models as we increase the domain shift. Similarly to the mIoU error, ECE is normalized by the best CS model (in terms of calibration). Interestingly, despite the remarkable improvements in terms of robustness, recent models are not significantly better calibrated. When moving from CS to ACDC, the *relative* mIoU error increases by a factor $\sim 2\times$ for recent models *vs.* $\sim 3\times$ for ResNet baselines; yet, in terms of *relative* calibration error, all models increase by a $\sim 40\times$ factor. This clearly highlights the need for further advances in model calibration.

In Figure 5.2 (top) we show the ECE *vs.* mIoU error for all models and datasets. When it comes to calibration *vs.* robustness trade-off, there is no clear winner among the model families we consider. Moreover, we do not observe a clear trend between mIoU and ECE in any domain.

Calibration with TS. In Figure 5.2 (bottom) we present the same results after applying TS (Guo et al., 2017), tuned on CS. Comparing top and bottom, we observe

an overall improvement in calibration for all networks. In particular, SegFormer models (◆), which had the largest ECE on CS and IDD, seem to benefit the most from TS. Nevertheless, even after this improvement, OOD calibration error (IDD, ACDC) remains significantly larger than the ID one (CS) for all models. Regarding ECE *vs.* mIoU out of domain, a mild trend emerges after TS: For transformer models, better-calibrated models are also the most robust (before TS, SegFormer (◆) did not follow this trend). On the other hand, for ResNet baselines (✕, ●) the better-calibrated models are generally the most brittle. Regarding ConvNeXt (■), although it is one of the best performing models, it seems to be worse calibrated than other models with similar mIoU. Overall, *recent models are not significantly better calibrated than ResNet baselines neither before nor after TS.*

5.4.3 Can we improve out-of-domain calibration?

Calibration error on samples from the training domain is not alarming, especially after TS. Nonetheless, the sharp increase in ECE out of domain is concerning for many applications, especially since recent segmentation models do not show a clear improvement in this direction. This renders methods that seek to improve calibration out of domain all the more important, but yet this is a rather underexplored research area. As discussed in Section 5.2, to the best of our knowledge, only Gong *et al.* (Gong et al., 2021) tackle OOD calibration *without* additional information about the test domain. They suggest clustering the calibration set into multiple “domains” based on the image features extracted by the network. A different temperature per cluster is then selected and test-time predictions are scaled according to the cluster assigned to the images. This *adaptive* TS method was originally devised for classification, but we extend it to the segmentation task by scaling all the logits of a given image with the same temperature. Regarding the number of clusters, we find 16 to be a reasonable number (see Appendix C.4 for this analysis).

Clustering on different calibration sets. In Figure 5.3 we present the results of calibrating with clusters computed on different calibration sets. Since our training

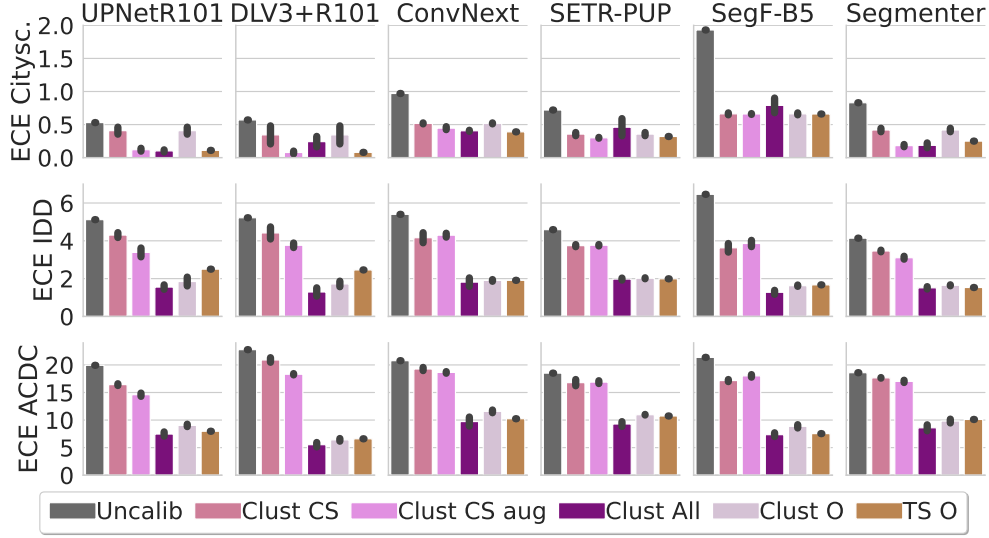


Figure 5.3: ECE ($\downarrow$) after clustering TS for a selection of models (best mIoU on CS per family). If the calibration samples are representative of the test domains *Clust All* can indeed improve ECE, however, without access to OOD samples (*Clust CS* and *Clust CS aug.*) benefits of clustering are limited. Oracle baselines (*O*) always use calibration images from the test domain.

dataset is CS, we assume that only CS images are available for calibration. As a naive alternative to obtain more diverse clusters, we introduce a *CS aug.* dataset where some of the CS calibration images are randomly augmented with different transformations (*e.g.*, color scaling, changes in brightness, contrast, etc). The rationale is that more diverse clusters might generalize better to new domains.

To assess how beneficial OOD samples can be during calibration, we also add another calibration set which contains images from all the datasets (CS, IDD and ACDC) mixed together, we will refer to it as *All*. This serves as a sort of upper bound, since our main goal is still assessing robustness in unseen domains. Furthermore, we introduce two more oracle baselines (*O*), which use calibration images from the test domain. For instance, when evaluating on IDD, the oracle calibration set will consist of *only* IDD while *All* will contain images of IDD, ACDC and CS mixed. One oracle baseline uses clustering, while the other uses vanilla TS (*Clust O* and *TS O*, respectively).

As expected, calibrating on all datasets (*Clust All*) significantly improves ECE, with comparable performance to oracles in most settings. In contrast, without access to OOD samples (*Clust CS*), calibrating with the method by Gong *et al.* yields rather

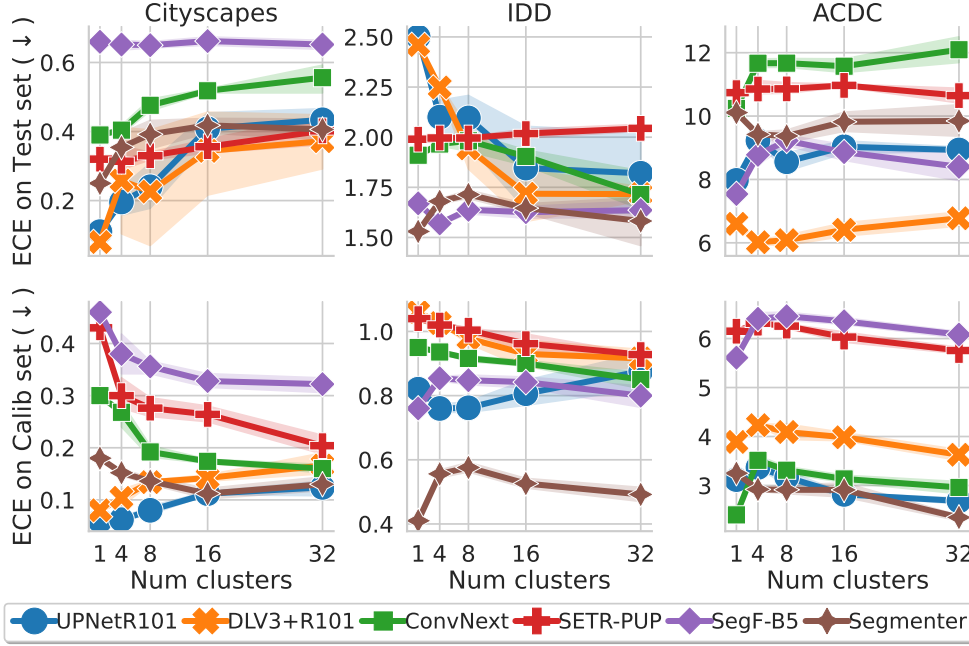


Figure 5.4: ECE ($\downarrow$) vs. number of clusters for oracles (calibration images extracted from the test domain). Even when evaluating on the calibration set, where there can be no overfitting of the temperatures, ECE does not decrease monotonically as we increase the number of clusters (one cluster is equivalent to vanilla TS).

limited improvements. Moreover, increasing cluster diversity via data augmentation (*Clust CS aug.*) is not always beneficial.

To gain more intuition, we visualize the cluster assignments (see Appendix C.5). When using all datasets for calibration, test-time images are qualitatively close to the assigned clusters; yet, with *Clust CS* or *Clust CS aug.*, OOD images do not blend well with the calibration images of their corresponding clusters. We argue that one implicit assumption for clustering to work well is that test-time images are close to one of the clusters (domains) in the calibration set; therefore, under strong domain shifts, it is unlikely to bring much improvement.

On the bright side, if representative images from the deployment domains are available, clustering could be applied to allow a single model to be calibrated on multiple domains. Of course, with OOD annotated samples, additional fine-tuning or adaptation techniques could be applied, but this is out of the scope of this study, since our focus is *off-the-shelf* model robustness, without adaptation.

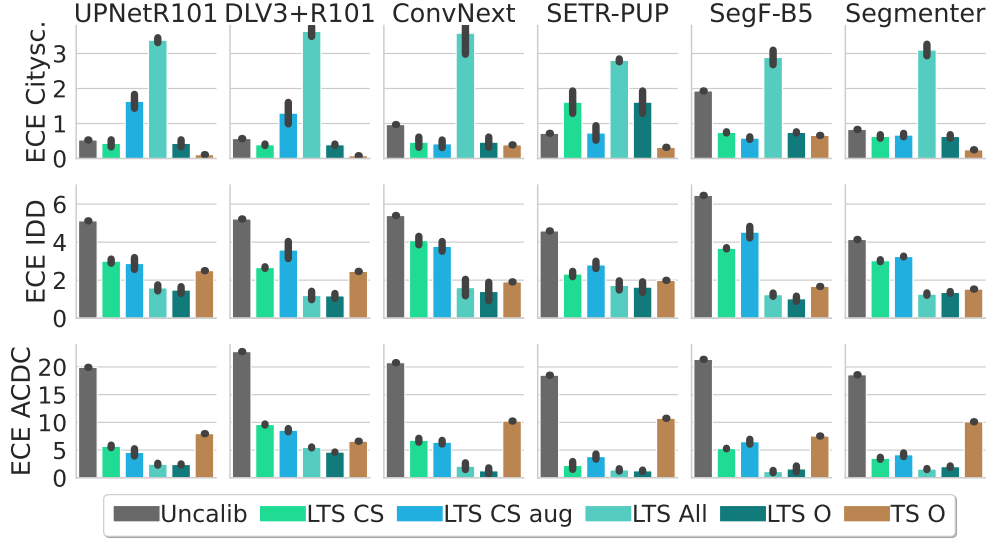


Figure 5.5: ECE ($\downarrow$) after local TS (LTS) for a selection of models (best mIoU on CS per family). Even without access to OOD samples (*LTS CS*), LTS calibration improves ECE out of domain, especially under strong domain shifts (ACDC). With access to OOD samples (*LTS All*), ECE out of domains improves further, albeit it degrades in domain.

Clustering does not improve ECE in domain. Comparing oracles in Figure 5.3, we can observe that clustering does not improve significantly over TS. In some settings, it is even worse. One possible explanation would be that this is due to an overfitting of the temperature parameters to the particular calibration clusters. However, in Figure 5.4 we observe that even when evaluating the ECE in the calibration set, the error does not monotonically decrease with the number of clusters. Although somewhat surprising, this is in fact possible since decreasing the ECE for several disjoint subsets of images (clusters) independently does not guarantee that the ECE on the union set will decrease. We provide a formal theorem in Appendix C.6, in support of this claim. Note that we are not asserting that clustering will not improve ECE *in general* (we empirically observed it can, if provided with a representative calibration set) but rather that it is not guaranteed to do so.

To sum up, *clustering the calibration set does not bring significant improvements unless representative images of the test domain are present in the calibration set. Moreover, it is not better than TS for ID calibration.*

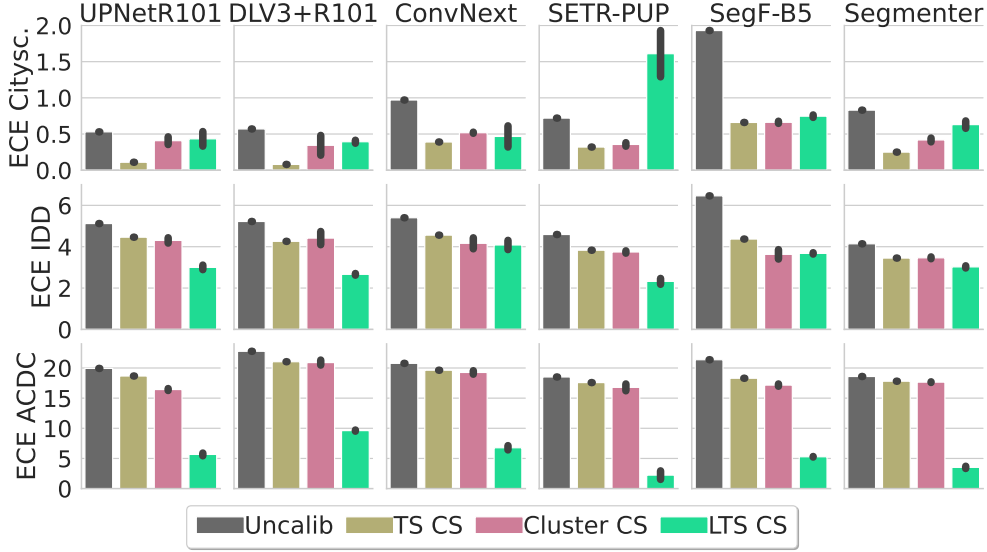


Figure 5.6: Comparison of calibration methods. ECE ($\downarrow$) after calibration for a selection of models (best mIoU on CS per family). All models are calibrated on CS calibration set. LTS is markedly the best calibration method out of domain with remarkable improvements under strong domain shifts.

Adaptive temperature via calibration network

The partial failure of the clustering approach motivates us to investigate other methods that adjust the temperature adaptively w.r.t. the input, since we can expect this to help improving OOD calibration. In this regard, Ding *et al.* (Ding et al., 2021) suggest training a small calibration network that predicts the temperature values as a function of both the input image and the segmentation model logits. This Local Temperature Scaling (LTS) method is specific to segmentation so the output is not a single temperature per image but a “temperature map” with pixel-level temperature values. Despite not being designed for OOD conditions, our intuition is that a network providing sample-dependent temperatures can be beneficial under domain shifts.

LTS using different calibration sets. As in Section 5.4.3, we test LTS on multiple calibration sets (*CS*, *CS aug.*, *All*), which in this case are used to learn the calibration network. Also here, we compare against oracles, for which the calibration network is learned using images from the test domains. Results are shown in Figure 5.5. Interestingly, we find that LTS using only CS images for calibration (*LTS CS*) leads to a noticeable improvement in OOD ECE. In particular, when testing on ACDC—

where the domain shift is stronger—*LTS CS* outperforms even TS with access to images on the test domain (*TS O*) for some models. Also in these experiments, introducing naive data augmentations on CS (*LTS CS aug.*) does not yield substantial improvements.

When using all the datasets for calibration (*LTS All*), OOD results improve even further; yet, there is a noticeable increase in calibration error on CS. Unlike clustering, where the temperature was optimized independently for each cluster, LTS trains the calibration network using all the samples at once and samples with large calibration error (like ACDC or IDD) may dominate the loss. We hypothesize that further improvements in the architecture and training schedule of the calibration network can lead to even better performance and are promising directions for future work.

Focusing on the oracle baselines, LTS outperforms TS on IDD and ACDC, but TS outperforms LTS on CS (*c.f.* *LTS O* and *TS O*). This may be due to the fact that CS is a very homogeneous dataset if compared to the other two, hence, it can be reasonable that a simpler method may perform best.

Comparing all calibration methods. In Figure 5.6 we compare *TS CS*, *Clust CS* and *LTS CS* (all calibrated on CS). LTS is markedly the best calibration method out of domain, especially under stronger domain shifts. In domain, TS works best, but it does not bring significant improvements out of domain. Since LTS predicts the temperature parameter at the pixel level, this motivates an ablation of clustering where we predict different temperatures per image; yet this does not improve results (see details in Appendix C.7). Additionally, we perform an ablation of LTS using only the image or the logits for calibration. Image information seems to be more important for OOD calibration, while the logits are more important in the ID setup (see Appendix C.8).

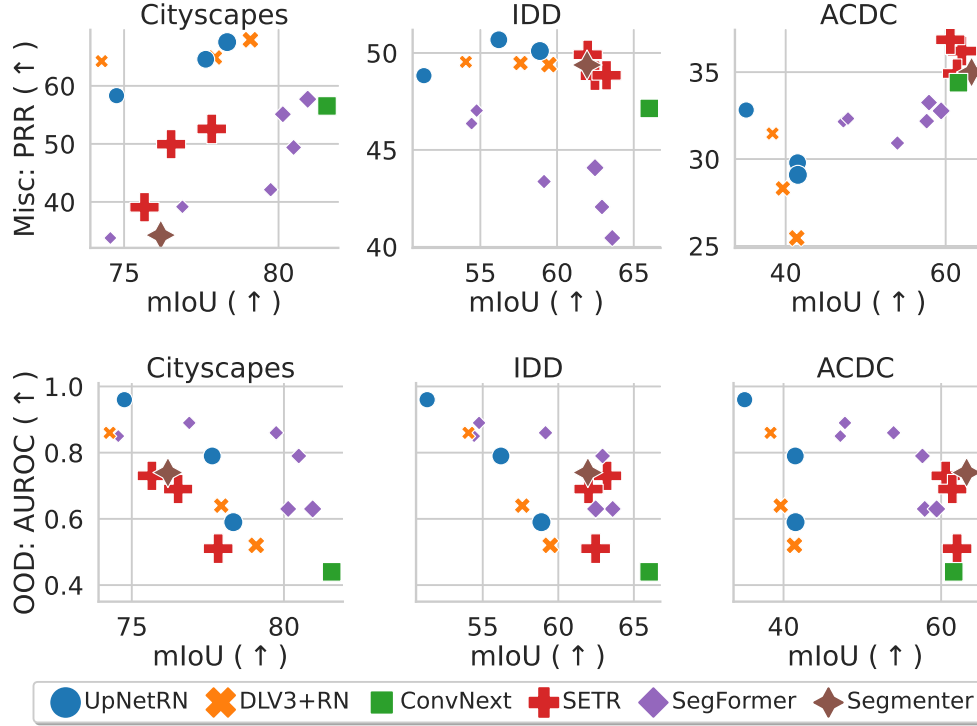


Figure 5.7: Misclassification and OOD detection vs. Robustness for different segmentation models and datasets. **(Top) PRR (↑) vs. mIoU (↑):** ResNet-based models (●, ×) outperform more recent models (other markers) in ID misclassification detection (CS, left), but the trend is opposite under strong domain shifts (ACDC, right). **(Bottom) AUROC (↑) vs. mIoU (↑):** There is no free-lunch between robustness and OOD detection in any considered domain.

5.4.4 Misclassification detection

In Figure 5.7 (top) we compare all models in terms of *misclassification detection vs. robustness* —PRR score (↑) vs. mIoU (↑). In domain, we observe a clear trend: within the same model family, better performing models tend to also show better PRR. However, when considering all models, higher mIoU does not generally imply higher PRR and ResNet-based backbones perform significantly better than more recent architectures. As we increase the domain shift, the trend changes: for ACDC, recent models perform best both in terms of mIoU and PRR. Moreover, out of domain, ResNet families show a negative correlation where better mIoU leads to worse PRR. Overall, *recent models seem to improve misclassification detection under strong domain shifts, but underperform baselines in domain.*

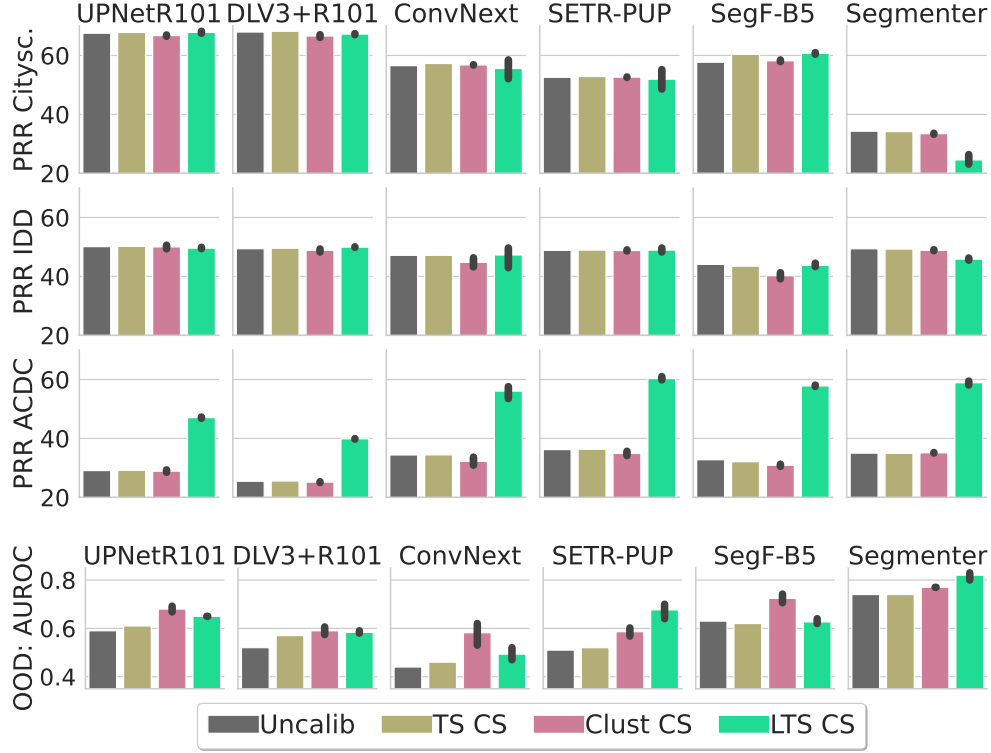


Figure 5.8: Misclassification and OOD detection after calibration for several models after applying different calibration techniques using CS samples. **Misclassification – PRR ($\uparrow$) (first 3 rows)**: Under strong domain shifts (ACDC), LTS calibration significantly improves PRR. **OOD detection – AUROC ($\uparrow$) (last row)**: Both clustering and LTS yield improvements in OOD detection.

5.4.5 Out-of-domain detection

In Figure 5.7 (bottom) we compare *OOD detection vs. robustness* (AUROC score *vs.* mIoU) for all models. To measure OOD detection we separate the ID images (CS) from the OOD ones (IDD and ACDC). Therefore, the y-axis is the same in all three plots and only the mIoU changes. For OOD detection, there is a marked negative trend between CS mIoU and AUROC. When looking at IDD and ACDC, the negative trend continues but there seems to be a distinction between ResNet baselines and other recent models: the latter perform better in terms of mIoU, but at the same time show a drop in AUROC. In short, *there is no free-lunch between robustness and OOD detection. In terms of OOD detection, a small ResNet-18 (●) performs best.*

5.4.6 Can calibration improve misclassification and out-of-domain detection?

In Figure 5.8 we show misclassification (top) and OOD (bottom) detection metrics for different models after calibration using only CS samples. For misclassification, we observe a sharp PRR improvement on ACDC after we calibrate models with LTS. This is encouraging as it indicates that the calibration network learned in LTS can help discern correct from incorrect predictions given its output temperature. We do not observe significant improvements in other datasets or with other methods. This is reasonable, since the largest calibration gain was observed with LTS on ACDC. (see Figure 5.6).

Regarding OOD detection (Figure 5.8 bottom) we observe that both clustering and LTS calibration can improve OOD detection. We find this interesting, since clustering on CS did not improve OOD calibration significantly. Although the clusters using only CS images are not representative enough to produce adequate temperatures for IDD or ACDC, OOD samples are assigned to clusters which have larger temperatures. This is enough to decrease the confidence for OOD samples compared to ID and leads to better OOD detection. Similarly for LTS, the calibration network assigns larger temperatures to OOD images.

In conclusion, *adaptive TS techniques are a promising avenue to improve OOD detection and misclassification detection under strong domain shifts.*

5.5 Conclusion

We have studied the reliability of recent segmentation models—in terms of **robustness** and **uncertainty estimation** under natural domain shifts. Overall, while no single model family is better in all scenarios, recent models are remarkably more robust to domain shifts than ResNet baselines. Yet, this does not translate into better *calibration*—severely degraded out of domain. Thus, it is crucial to find methods to

improve model calibration in OOD settings. To this end, we have explored state-of-the-art methods and found that Local Temperature Scaling (Ding et al., 2021), although originally devised for ID settings, is a promising technique.

Furthermore, we have explored *misclassification* and *OOD detection*—two other important tasks regarding uncertainty estimation. We have shown that recent and more robust models tend to perform better at misclassification under strong domain shifts, but yet they underperform ResNet baselines ID. On the other hand, OOD detection under domain shifts is negatively correlated with the mIoU, which translates into a trade-off between robustness and uncertainty. Finally, we find that adaptive temperature scaling techniques can help beyond calibration and improve OOD detection and misclassification in some settings.

All in all, although we appear to be *on the right track* for what concerns robustness, our findings motivate the need to improve reliability of segmentation models in other dimensions, where results are not equally positive. In that regard, we identify several promising directions which we hope may encourage future research on this important topic.

6

Placing Objects in Context via Inpainting for Out-of-distribution Segmentation

Abstract

When deploying a semantic segmentation model into the real world, it will inevitably be confronted with semantic classes unseen during training. Thus, to safely deploy such systems, it is crucial to accurately evaluate and improve their anomaly segmentation capabilities. However, acquiring and labelling semantic segmentation data is expensive and unanticipated conditions are long-tail and potentially hazardous. Hence, existing anomaly segmentation datasets capture a limited number of anomalies, lack realism or have strong domain shifts. In this paper, we propose the Placing Objects in Context (POC) pipeline to realistically add any object into any image via diffusion models. POC can be used to easily extend any dataset with an arbitrary number of objects. In our experiments, we present different anomaly segmentation datasets based on POC-generated data and show POC can improve the performance of recent state-of-the-art anomaly fine-tuning methods in several standardized benchmarks. POC is also effective to learn new classes. For instance, we learn a subset of Pascal classes from scratch by adding them to Cityscapes and achieve similar performance to a Pascal-trained baseline.

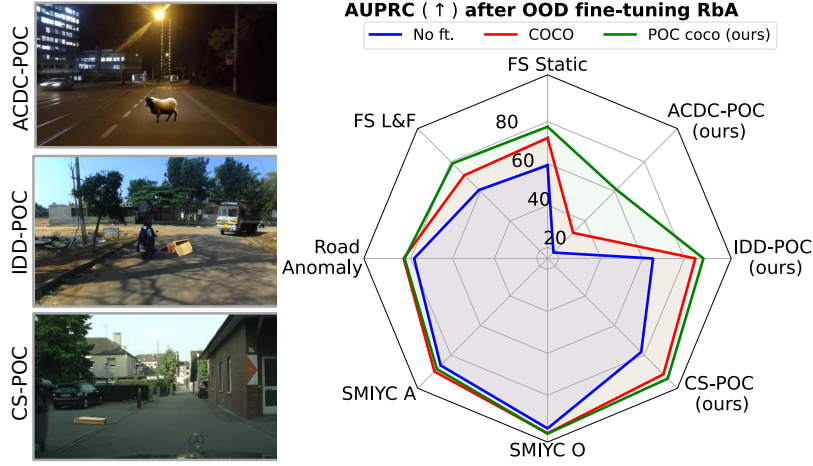


Figure 6.1: **Left:** Images of our POC-generated datasets. Top to bottom, inserted anomalies are: “sheep”, “dumped furniture” and “carton box”. **Right:** AUPRC on different anomaly segmentation datasets. We evaluate RbA Nayal et al. (2023) prior to fine-tuning, and after fine-tuning with COCO objects or our POC-generated images. Fine-tuning with our POC improves results on several benchmarks.

6.1 Introduction

When autonomous agents such as robots or self-driving cars are deployed, they often face an open-world settings. That is, they encounter novel objects that were not present in the training set. This can lead to particularly dangerous situations: for example, consider an unknown wild animal crossing the street and processed by the model as “road”.

To tackle this issue, it is important to distinguish out-of-distribution (OOD) categories—*i.e.* novel objects unseen during training—from in-distribution (ID) ones. In the context of semantic image segmentation, this task is often referred to as *anomaly segmentation*.

Although several methods have been proposed that allow segmenting anomalies (Grcić et al., 2022; Lis et al., 2020; Jung et al., 2021; Liang et al., 2022; Di Biase et al., 2021; Besnier et al., 2021), accurately evaluating the performance of such methods is a challenge in itself. Given a model trained on a particular dataset, the aim is to test its ability to process OOD categories in conditions that resemble the training domain. For example, to test a model trained for semantic segmentation of urban

scenes, the ideal test set would be constituted by images showing OOD categories within an urban environment similar to the training one. Yet, the distribution of OOD objects is long-tail and it is highly inefficient, or even hazardous, to acquire and label images with arbitrary categories.

Previous approaches to generate anomaly segmentation datasets can be grouped into three families: *Stitching and blending* OOD objects from other sources into images from the original dataset Blum et al. (2019); *Collecting images* from driving scenes and annotating OOD objects Pinggera et al. (2016); Chan et al. (2021a); Lis et al. (2019); *Full simulation* of urban scenes with anomalies Hendrycks et al. (2019). *Stitching and blending* is relatively inexpensive if OOD objects are segmented elsewhere, but it often leads to unrealistic insertions (*e.g.* size, illumination). *Collected images* contain real anomalies, but are expensive to acquire and have a significant distribution shift w.r.t. the original dataset (it is not easy to find or replicate images following the original setup). *Full simulation* allows for perfectly blended objects but bears a high set up cost and results in a severe drift from the train distribution. See Figure 6.2 for illustrative examples.

Ideally, methods to generate anomaly segmentation datasets should satisfy four main desiderata: (i) Minimal domain shift w.r.t. the training set—since large domain shifts may lead to underestimating anomaly segmentation capabilities; (ii) Generating realistic images; (iii) Allowing for a dynamic generation of new images with arbitrary OOD objects; (iv) Bearing low set-up costs.

This motivates us to introduce the **Placing Objects in Context** pipeline (POC) to allow practitioners to generate anomaly segmentation test sets by realistically inserting *any* (OOD) object into *any* image on the fly. See a comparison along our desiderata between approaches followed to generate previous benchmarks and the proposed POC in Table 6.1.

In order to insert new objects realistically, we control the location of the added object and only apply local changes to preserve the overall scene semantics. We resort on open-vocabulary segmentation Grounded-SAM Contributors (2023) to select valid

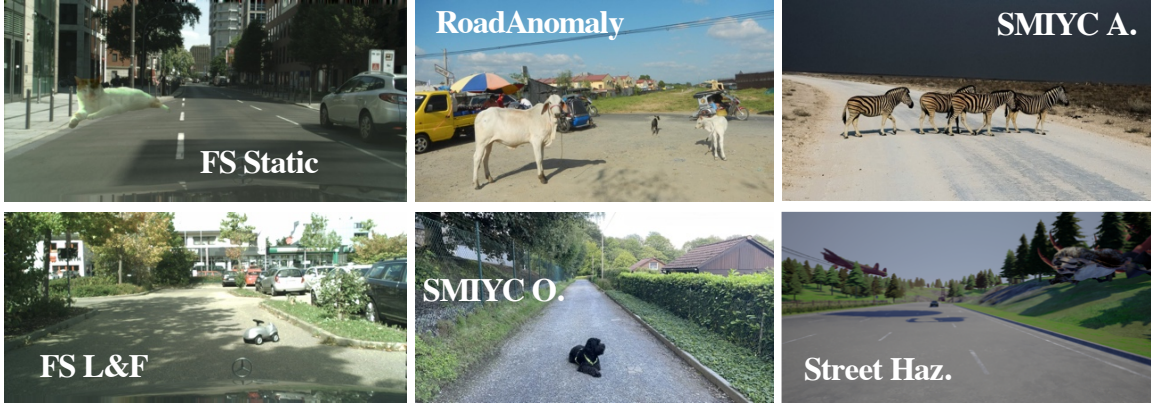


Figure 6.2: Samples from previous OOD datasets to visually support Table 6.1: FS Static has unrealistic OOD objects while RoadAnomaly and SMIYC datasets have strong domain shifts from Cityscapes. FS L&F (which manually inserts OOD objects) and StreetHazards (full simulation) have large set-up costs.

Dataset	No domain shift	OOD realism	Dynamic	Low set-up cost
FS Static (Blum et al., 2019)	✓	✗	(✓)	(✓)
FS L&F (Pinggera et al., 2016)	(✓)	✓	✗	✗
SMIYC O. (Chan et al., 2021a)	✗	✓	✗	✗
SMIYC A. (Chan et al., 2021a)	✗	✓	✗	✗
RoadAnom. (Lis et al., 2019)	✗	✓	✗	✗
StreetHaz. (Hendrycks et al., 2019)	✗	(✓)	(✓)	✗
POC (ours)	✓	(✓)	✓	✓

Table 6.1: Comparison of anomaly test sets. We qualitatively compare datasets on four main axes. We score them into good (✓), medium ((✓)) and bad (✗). Further discussion in Section 6.2.

regions where to place the object, *e.g.* “the road”. We then feed the selected region to an inpainting model Rombach et al. (2022) with a conditioning prompt, *e.g.* “a cat”. After inpainting, we apply again the segmentation model to the modified area to automatically annotate the added object and detect generation failures, *i.e.* when the object was not properly generated.

In our experiments, we show that fine-tuning on POC-generated data can significantly improve the performance of state-of-the-art anomaly segmentation methods—outperforming models fine-tuned via the standard practice of *stitching* COCO objects. We also present three POC-generated *evaluation* sets based on Cityscapes and two other self-driving datasets, and report the performance of anomaly segmentation

methods (see Figure 6.1). Finally, since POC can add arbitrary objects, we show it can be used to learn new classes. For instance, augmenting Cityscapes images with animal classes leads to 93.14 mIoU on Pascal’s test set (animal classes) *without seeing any real animal*, while directly training on Pascal yields 94.75.

We will release our datasets and the POC pipeline to enable the generation of arbitrary datasets in future work.

6.2 Related work

Anomaly segmentation methods. Initial works relied on approximating uncertainty via softmax probabilities Hendrycks and Gimpel (2016); Liang et al. (2017), model ensembles Lakshminarayanan et al. (2017) or dropout Mukhoti and Gal (2018); Gal and Ghahramani (2016). Yet, models tend to be overconfident, resulting in high confidence also for OOD samples Nguyen et al. (2015); Guo et al. (2017); Jiang et al. (2018). Alternative confidence measures have been proposed that either rely on logits Corbière et al. (2019); Hendrycks et al. (2019); Jung et al. (2021); Liu et al. (2020) or density estimators Lee et al. (2018). Another body of works reconstruct images with generative models and detect anomalies as discrepancies between original images and their reconstructions Haldimann et al. (2019); Lis et al. (2019, 2020); Xia et al. (2020). Currently, the most promising methods use OOD data to fine-tune the models Chan et al. (2021b); Tian et al. (2022); Grcić et al. (2022). In particular, they crop OOD objects from COCO Lin et al. (2014) and *stitch* them in Cityscapes images. In this work, we build on three state-of-the-art OOD fine-tuning methods Nayal et al. (2023); Liu et al. (2023b); Rai et al. (2023) and combine them with POC.

Anomaly segmentation datasets. In Table 6.1 we compare existing anomaly segmentation datasets across four axes (*Domain shift*, *OOD realism*, *Dynamism* and *Set up cost*) while in Figure 6.2 we show corresponding sample images.

Fishyscapes Static Blum et al. (2019) randomly stitches OOD objects from Pascal Everingham et al. (2012) to Cityscapes images, thus, there is no *domain shift*, but

stitched OOD objects lack *realism*. If OOD object images and masks are available, datasets can be generated *dynamically*. However, if new objects need to be used, they need to be acquired from additional datasets or from the web with mild *set-up costs*.

At the other end of the spectrum, **RoadAnomaly** Lis et al. (2019) and **Segment-me-if-you-can (SMIYC)** Chan et al. (2021a) download real images with anomalies from the web. This ensures *OOD realism*, but often leads to a large *domain shift*. Moreover, manual labelling of OOD objects has a significant *set-up cost* and is *not dynamic*, *i.e.* new images would need to be acquired and labelled to generate new samples. **Lost & Found** Pinggera et al. (2016) mimicked the Cityscapes set-up to reduce domain shift, but planted OOD objects artificially, which leads to low variability and is even more difficult to scale.

Street Hazards Hendrycks et al. (2019) presents a fully simulated dataset. This allows for *dynamic* generation of images while the simulation engine inserts OOD objects *realistically* in terms of lighting. However, the pose and size of the object are pseudo-random, which is not always realistic and there is a strong *domain shift* from simulation to real images. Moreover, accurate 3D models for all objects are required which has a significant *set-up cost* and is hard to scale.

Our proposed pipeline is *plug & play* and can add objects into images with *no set-up costs*. Since it is built on top of open-vocabulary models, it can *dynamically* insert any object by changing the text prompts. We observe qualitatively and quantitatively that our pipeline leads to more *OOD realism* and applying inpainting also avoids *domain shift*.

Generative models. Diffusion models Sohl-Dickstein et al. (2015) have led to unprecedented quality in image generation Dhariwal and Nichol (2021); Ho et al. (2020); Saharia et al. (2022a); Song and Ermon (2019). In particular, text-to-image models can receive a text prompt to easily condition the image generation process Nichol et al. (2021); Ramesh et al. (2022); Rombach et al. (2022); Saharia et al. (2022b). Besides image generation, there has been a growing interest in image editing.

Particularly relevant to our goal are works that perform image inpainting, *i.e.* modifying only masked parts of an image Rombach et al. (2022); Avrahami et al. (2022); Ramesh et al. (2022). General-purpose editing models that can edit images based on text prompts Brooks et al. (2023); Meng et al. (2021) are also relevant for our goals. In particular, InstructPix2Pix Brooks et al. (2023) has recently shown very realistic results following text instructions, *e.g.* “make it look like it was a sunset”. However, we found that it often failed to add new objects to the scene *e.g.* “add a dog on the street”. A more detailed discussion can be found in Appendix D.3. Rather than training a diffusion model to edit images end-to-end like InstructPix2Pix, we build a pipeline on top of Stable Diffusion Rombach et al. (2022) to automatically add objects into images.

Similar to us, Du *et al.* Du et al. (2023) use text-to-image models to generate OOD images for classification and Karazija *et al.* Karazija et al. (2023) to perform zero-shot semantic segmentation. While the latter Karazija et al. (2023) relies on a frozen feature extractor and focuses on zero-shot segmentation, we *extend* an existing dataset with new classes. Different from the former Du et al. (2023), we target OOD in segmentation and rather than generating fully OOD images, we insert OOD objects realistically.

6.3 Placing Objects in Context (POC)

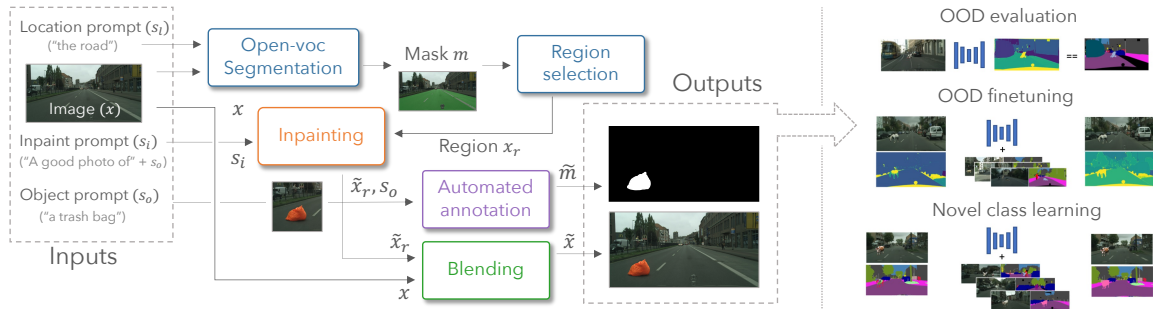


Figure 6.3: Illustration of our POC pipeline and applications. Our pipeline is built on top of inpainting and open-vocabulary segmentation models to insert arbitrary objects into images realistically. The modified image and mask can be used for different tasks.

We now present our proposed pipeline, Placing Objects in Context (POC). To recap, our desiderata to generate OOD datasets are *i*) minimal shift w.r.t. training set, *ii*) realism, *iii*) dynamic generation and *iv*) low set-up costs.

6.3.1 The POC pipeline

Our pipeline is built on top of two open-source models with permissive licenses: an inpainting model from Stable Diffusion (SD Rombach et al. (2022)) and an open-vocabulary segmentation model (GSAM) Grounded-SAM Contributors (2023) based on SAM Kirillov et al. (2023) and GroundingDINO Liu et al. (2023a). In the following, we detail the four main stages that constitute our pipeline, we provide a full overview of our pipeline in Figure 6.3.

Selecting a region to inpaint. In order to insert objects realistically, it is important to find suitable locations for them (*i.e.* a cat should not be levitating). We leverage GSAM Grounded-SAM Contributors (2023) in order to segment a *valid* area to insert the object based on a location prompt s_l , *e.g.* "the road". Within this valid area, we select a region r with random size and location based on user-specified limits to increase diversity.

Object inpainting. After selecting r we crop a square around it x_r and apply SD to obtain $\tilde{x}_r$. The strong vision grounding from SD allows us to add objects more realistically. For instance, we observe that the inpainting model tends to adjust the size of the object (*e.g.* a bird will be much smaller than a garbage bin for the same region) although there is a tendency towards filling all the inpainting area. We also observe that the illumination of the added object changes based on the image. This is particularly noticeable in night images from ACDC (see Figure 6.1).

Automated annotation. In most downstream applications, we need the corresponding mask of the added object. We leverage again the open-vocabulary GSAM, to get $\tilde{m}$ based on an object prompt s_o . We also use this step to reject images with generation failures (*e.g.* the object was not generated or it was very unrealistic). However, we

observed that applying GSAM to the full image often led to false positives and better results were obtained by applying it only to the inpainted region. Also note that using a model like GSAM we can provide more accurate labels than simpler methods like foreground/background segmentation Karazija et al. (2023) that can not distinguish between a bicycle and its rider.

Object blending. While SD tends to preserve the details of the original image, it introduces slight modifications to the textures (especially fine-grained) and some noticeable changes, *e.g.* in lane markings. To reduce undesired edits, we blend the original image x and inpainted one $\tilde{x}$ as:

$$\tilde{x} := (1 - m) \odot x + m \odot \tilde{x}$$

where $:=$ indicates an update operation, $\odot$ the element-wise product and $m = \mathcal{G}(\tilde{m})$ is the object mask convolved with a Gaussian kernel. This allows for a smooth transition between the inpainted object and the rest of the image, while preserving some realistic local modifications, like shadows or reflections. We also tried applying an image2image generative model with an empty prompt after inpainting, but this does not prevent undesired modifications in general and often modifies the image beyond the inpainting region.

6.3.2 Generating datasets with POC

Although our pipeline can be used for multiple applications, we have two main motivations: (i) extending datasets for *anomaly segmentation* and (ii) learning new classes. In both cases we use Cityscapes as the known classes and, regardless of the number of added classes, all POC-datasets have $3\times$ the size of Cityscapes— *i.e.* we augment each image three times with randomly sampled prompts.

Generation prompts. In all datasets we follow a similar approach. Each object class has an object prompt s_o , or several for diversity (*e.g.* “car”, “suv”, “van” all belong

to the Cityscapes class “car”). Then, the inpainting prompt is built as $s_i = \text{“A good photo of”} + s_o$. Given our datasets are all for autonomous driving, we use as location prompt $s_l = \text{“the road”}$ for all objects except the class “bird”, which has unconstrained location. We found this simple approach to yield good results without the need for prompt engineering.

Generating OOD *test* sets. To evaluate anomaly segmentation methods, we generate three new test sets, namely *CS-POC*, *IDD-POC* and *ACDC-POC*. We start from the Cityscapes Cordts et al. (2016b), IDD Varma et al. (2019), and ACDC Sakaridis et al. (2021) validation sets (which have increasing domain shift w.r.t. Cityscapes de Jorge et al. (2023)) and use the same list of OOD objects to augment the images within the validation set. The rationale behind the creation of three sets is to disentangle the difficulty in detecting certain OOD objects (that might be harder to detect as OOD) *vs.* the difficulty caused by a large distribution shift on the ID classes (while the OOD classes remain the same). A second goal is to showcase that POC can be used with different datasets seamlessly. These datasets contain 25 different OOD classes that can be commonly found on the street (*e.g.* wild animals, garbage bags and bins, *etc.*). Given our OOD objects are all *synthetic*, following Blum et al. (2019) we also add ID objects (*e.g.* cars or persons) to ensure the model is not “simply” performing synth *vs.* real discrimination. A full list of all the objects can be found in Appendix D.1.

Generating OOD *fine-tuning* sets. Recent work on OOD fine-tuning have used COCO classes not present in Cityscapes to extract OOD objects. For consistency, we generate *POC-coco* using the names of COCO classes used in previous works as prompts to inpaint them with POC (as opposed to cropping and stitching from COCO images). Additionally, we generate another fine-tuning dataset, *POC-alt*, with the same 25 OOD classes used in our evaluation sets to see the dependence of the fine-tuning methods on the number and type of OOD objects.

Extending datasets to *learn new classes*. Given a dataset $\mathcal{D}$ with set of classes $\mathcal{K}$, we consider the task of generating an *extended* dataset $\tilde{\mathcal{D}}$ with set of classes $\tilde{\mathcal{K}} = \mathcal{K} \cup \mathcal{U}$ such that it can be used to train models that perform well on $\mathcal{D}$ as well as learning the

additional set of classes $\mathcal{U}$. Following the autonomous driving use-case, we extend the Cityscapes dataset with the 6 animal classes present in the PASCAL dataset (which we use to evaluate). Our motivation is that wild animals cause accidents Saad and Alsayyari (2019) and being able to segment them individually (not just as anomalies) might help prevent collisions. We call this dataset *POC-A*, however, we also add the same classes on the CS validation set which we call *CS extended*. Additionally, we generate another dataset where we inpaint the animal classes but also CS classes present in PASCAL to study the generalization capabilities.

6.4 POC for anomaly segmentation

We have already discussed that, in order to reliably evaluate the risk of deploying a model in a certain scenario, we need test images with minimal distribution shift w.r.t. the original distribution and realistic OOD objects. In a similar spirit, we hypothesize that when fine-tuning models for anomaly segmentation, more realistic anomalies will lead to better results. To test this hypothesis, we take three recent methods for OOD fine-tuning that rely on stitching COCO objects into CS images, but instead of COCO, we use our POC pipeline to generate the anomalies for fine-tuning.

6.4.1 Experimental setting

Anomaly segmentation methods. *RPL* Liu et al. (2023b) learns a module to detect anomalies via contrastive learning, on top of a segmentation network that is kept frozen. This allows improving OOD detection with minimal degradation to the closed-set performance. *Mask2Anomaly* (M2A) Rai et al. (2023) and *RbA* Nayal et al. (2023) are both based on the novel Mask2Former architecture Cheng et al. (2022) which performs segmentation at the mask level, *i.e.* by grouping pixels into “masks” and classifying the whole masks into the closed-set categories. This significantly reduces the pixel-level noise on the anomaly scores. *RbA* Nayal et al. (2023) performs OOD fine-tuning with a squared hinge loss while M2A Rai et al. (2023) also uses contrastive

6. Placing Objects in Context via Inpainting for Out-of-distribution Segmentation

learning. We use the original code with default settings for each method and only modify the fine-tuning dataset.

	OOD data	FS Static			FS Lost&Found			SMIYC Anomaly			SMIYC Obstacle		
		F1* ↑	AuPRC ↑	FPR ↓	F1* ↑	AuPRC ↑	FPR ↓	F1* ↑	AuPRC ↑	FPR ↓	F1* ↑	AuPRC ↑	FPR ↓
M2A	No ft.	65.39	60.07	7.41	45.53	36.77	13.87	88.63	92.99	3.94	69.28	73.62	6.91
	COCO	82.73	88.35	<u>2.22</u>	70.92	64.46	13.11	<u>90.52</u>	94.69	<u>3.28</u>	89.97	94.76	0.39
	POC alt.	<u>82.61</u>	87.40	3.15	74.49	68.84	11.35	92.21	93.78	2.05	90.84	95.34	0.26
	POC coco	81.99	87.01	2.12	76.52	72.97	9.21	88.8	92.13	8.35	91.35	96.04	0.13
RPL	No ft.	20.99	13.95	38.54	5.40	1.57	66.28	50.19	52.99	39.55	46.02	45.15	3.19
	COCO	83.83	89.62	1.23	58.02	58.41	3.22	73.4	78.26	18.14	<u>89.74</u>	94.29	0.34
	POC alt.	88.37	94.44	0.71	70.32	74.22	2.67	69.42	78.21	<u>26.89</u>	89.83	<u>93.61</u>	<u>0.88</u>
	POC coco	88.54	95.07	0.49	<u>69.45</u>	<u>69.17</u>	1.64	62.65	68.00	46.78	87.76	92.31	0.92
RbA	No ft.	58.21	59.15	17.68	63.7	60.95	10.63	<u>86.47</u>	86.89	86.44	92.75	95.88	0.16
	COCO	67.04	72.23	3.98	69.72	70.87	8.69	84.71	91.07	5.35	94.85	98.19	0.04
	POC alt.	<u>68.31</u>	<u>77.2</u>	<u>3.4</u>	<u>74.05</u>	<u>76.74</u>	<u>5.36</u>	86.48	<u>90.47</u>	4.36	94.92	<u>98.35</u>	0.04
	POC coco	68.56	77.58	3.32	76.48	78.87	3.32	85.21	89.4	<u>4.99</u>	94.59	98.39	0.04

	OOD data	RoadAnomaly			Cityscapes – POC			IDD – POC			ACDC – POC		
		F1* ↑	AuPRC ↑	FPR ↓	F1* ↑	AuPRC ↑	FPR ↓	F1* ↑	AuPRC ↑	FPR ↓	F1* ↑	AuPRC ↑	FPR ↓
M2A	No ft.	54.75	55.74	52.15	40.95	35.36	13.05	45.57	42.94	11.96	18.58	10.34	26.74
	COCO	77.19	<u>78.89</u>	18.50	88.26	93.88	0.54	80.98	85.54	1.03	70.78	<u>72.83</u>	6.17
	POC alt.	81.45	82.29	36.65	91.39	95.81	<u>0.43</u>	<u>82.82</u>	<u>87.74</u>	<u>1.09</u>	74.02	74.45	<u>7.56</u>
	POC coco	<u>78.92</u>	77.96	<u>24.61</u>	<u>89.09</u>	<u>94.67</u>	0.37	83.05	89.07	1.26	<u>72.77</u>	72.04	8.38
RPL	No ft.	24.4	14.99	70.42	22.74	13.79	44.87	7.95	3.45	61.43	3.31	1.26	79.01
	COCO	60.08	59.20	27.14	82.06	88.41	0.71	72.35	78.68	1.76	63.64	65.42	2.56
	POC alt.	66.39	69.42	<u>21.68</u>	<u>86.1</u>	<u>93.18</u>	<u>0.48</u>	<u>79.66</u>	<u>85.78</u>	<u>0.97</u>	79.42	85.68	0.78
	POC coco	<u>61.38</u>	<u>64.16</u>	21.36	87.58	94.30	0.36	82.8	89.97	0.69	<u>76.0</u>	<u>81.16</u>	<u>1.31</u>
RbA	No ft.	72.78	78.4	11.83	73.49	77.89	3.67	65.54	65.05	78.92	24.25	18.65	90.04
	COCO	<u>78.54</u>	83.4	8.31	87.17	92.92	0.49	79.22	85.25	1.19	33.45	31.96	11.19
	POC alt.	78.32	84.08	8.29	<u>89.34</u>	<u>95.02</u>	<u>0.37</u>	<u>83.69</u>	89.1	0.73	<u>54.99</u>	<u>58.37</u>	8.39
	POC coco	77.34	82.95	8.82	90.48	95.83	0.3	83.77	<u>89.09</u>	<u>0.89</u>	57.59	61.14	<u>9.06</u>

Table 6.2: Anomaly segmentation metrics after OOD finetuning. We use three recent OOD fine-tuning methods and report results prior to fine-tuning (*No ft.*), after fine-tuning with *COCO* objects and using our POC pipeline to inpaint coco objects (*POC coco*) or an alternative set of 25 objects (*POC alt.*) likely to be found on the street. Best and second best numbers for each method are highlighted in **bold** and underlined respectively. The best number over all methods is shaded in **gray**. Using our pipeline improves performance in most cases. Moreover, fine-tuning with COCO also leads to significant improvements in our POC eval sets, consistent with previous benchmarks.

OOD fine-tuning data. For each method, we consider different ways of generating the dataset used to fine-tune the anomaly segmentation modules. First, we consider a baseline where no fine-tuning occurs (*No ft.* in our tables). Then, we consider fine-tuning on *COCO* stitching. When we fine-tune using our POC-generate images, we consider two cases: *POC coco* (inpainting the same classes as COCO stitching) and *POC alt.* (inpainting alternative classes, more likely to be found on the street).

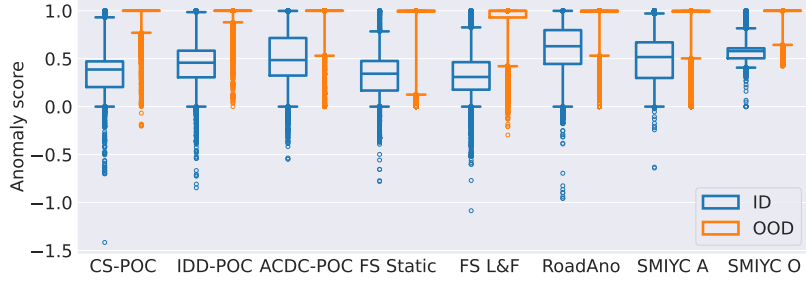


Figure 6.4: Boxplots of anomaly scores. All datasets have consistently very high scores for OOD pixels while ID pixels of datasets with strong distribution shifts also have shifted scores. Thus, distribution shifts may lead to underestimated performance.

Anomaly datasets and metrics. We evaluate on five commonly used datasets, discussed in Section 6.2, and on our POC-generated datasets. Following previous work Chan et al. (2021b); Liu et al. (2023b), we compute three different metrics: maximum F1 score over all thresholds ($F1^*$), Area under the Precision-Recall Curve (AuPRC) and False Positive Rate at 95% recall (FPR).

6.4.2 Results

POC improves OOD detection. In Table 6.2, we show fine-tuning with *POC coco* is remarkably better than the *No ft.* baseline despite only using data with synthetic anomalies. Moreover, *POC coco* also brings important improvements over COCO fine-tuning in some settings. For instance, in FS Static (for RPL or RbA), in SMIYC Obstacle (for M2A or RbA) and in FS Lost & Found (for all methods). In other settings, it is competitive with COCO fine-tuning except in SMIYC Anomaly (for RPL), where we observe a significant drop. One reason may be that the strong domain shift in SMIYC Anomaly limits the benefits of using more realistic data. On the other hand, in FS Lost & Found, which has the closest setting to Cityscapes and real OOD objects, we carry the largest improvements.

Robustness to the choice of OOD classes. We observe that inpainting different OOD classes than COCO, *POC alternative*, also leads to strong anomaly segmentation results, improving over the COCO baseline in several settings and sometimes surpassing *POC coco*. This indicates that fine-tuning methods are somewhat robust to the choice

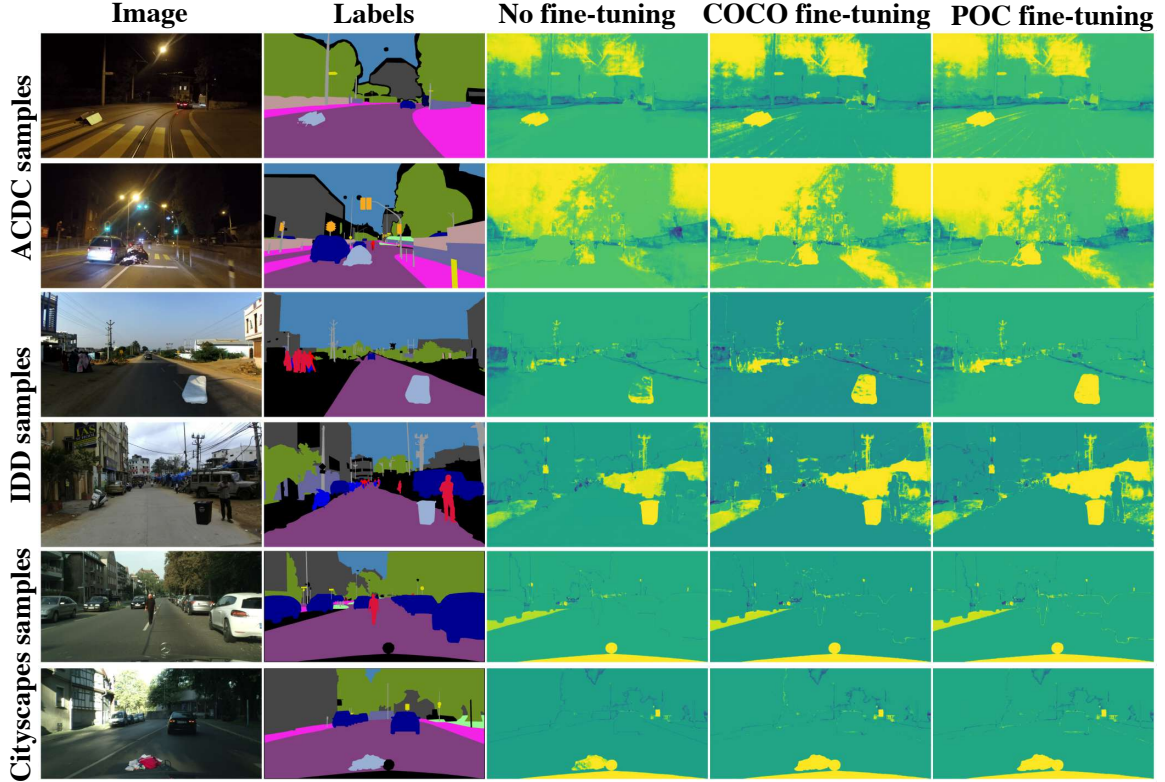


Figure 6.5: Anomaly score maps. Per-pixel anomaly scores on POC-generated images obtained with M2A Rai et al. (2023) before and after fine-tuning with COCO and POC data. COCO and POC fine-tuning have notable improvements over the *No ft.* baseline, *e.g.* note the garbage bag or mattress in second and third images.

of OOD classes. Importantly, the flexibility of the POC pipeline opens the possibility of studying which classes might be best depending on the use-case in future work.

Finally, note that the best score of all methods (highlighted with gray background) corresponds to one of the POC fine-tuning datasets in almost all settings.

Performance on POC evaluation sets. As expected, fine-tuning with our pipeline brings the best results on our POC eval sets. Yet, we also observe that COCO fine-tuning leads to notable improvements, similar to the ones observed in previous datasets. This indicates that POC datasets accurately reflect anomaly segmentation capabilities and can be used to efficiently build OOD detection benchmarks.

Domain shifts hinder anomaly segmentation. Considering our synthetic evaluation datasets, if we move from POC-CS to POC-IDD and POC-ACDC we observe a drop in performance in all methods. Since the sets contain the same anomaly

classes, the drop is due to the domain shift between train and evaluation rather than hard-to-detect anomalies. Additionally, in Figure 6.4 we show boxplots of the predicted anomaly scores (higher meaning higher chance of anomaly) for ID *vs.* OOD pixels. Interestingly, we observe that OOD pixels have very high scores independently of the dataset, while ID scores vary significantly between datasets. Datasets with strong domain shifts (*i.e.* SMIYC Anomaly, RoadAnomaly and POC-ACDC) carry larger ID anomaly scores. While anomaly segmentation under domain shift might also be an interesting task, we argue that, in practice, agents will be carefully deployed in restricted areas (*e.g.* one city). To accurately evaluate the risk represented by anomalies, ideally, there should be no domain shift.

Anomaly segmentation maps. In Figure 6.5 we show per-pixel anomaly scores on POC-generated images computed with Mask2Anomaly prior to fine-tuning and after fine-tuning either with COCO or POC data. Aside from the inpainted anomalies (labelled in gray), we observe that in ACDC (Top) the night sky has a particularly high score and in IDD (Middle) a peculiar instance of a known class (the all-road car) is also highlighted, potentially misleading anomaly segmentation. More anomaly score maps can be found in Appendix D.6.

6.5 POC to learn new classes

Besides anomaly segmentation, another natural application for our POC pipeline is dataset *extension*. In this section we consider extending the dataset in two directions: adding novel objects to learn new classes and adding instances of existing classes to improve generalization.

6.5.1 Experimental settings

Extended datasets. We use Cityscapes as our base dataset. Given our autonomous driving motivation, we are interested in classes that may cause road accidents if

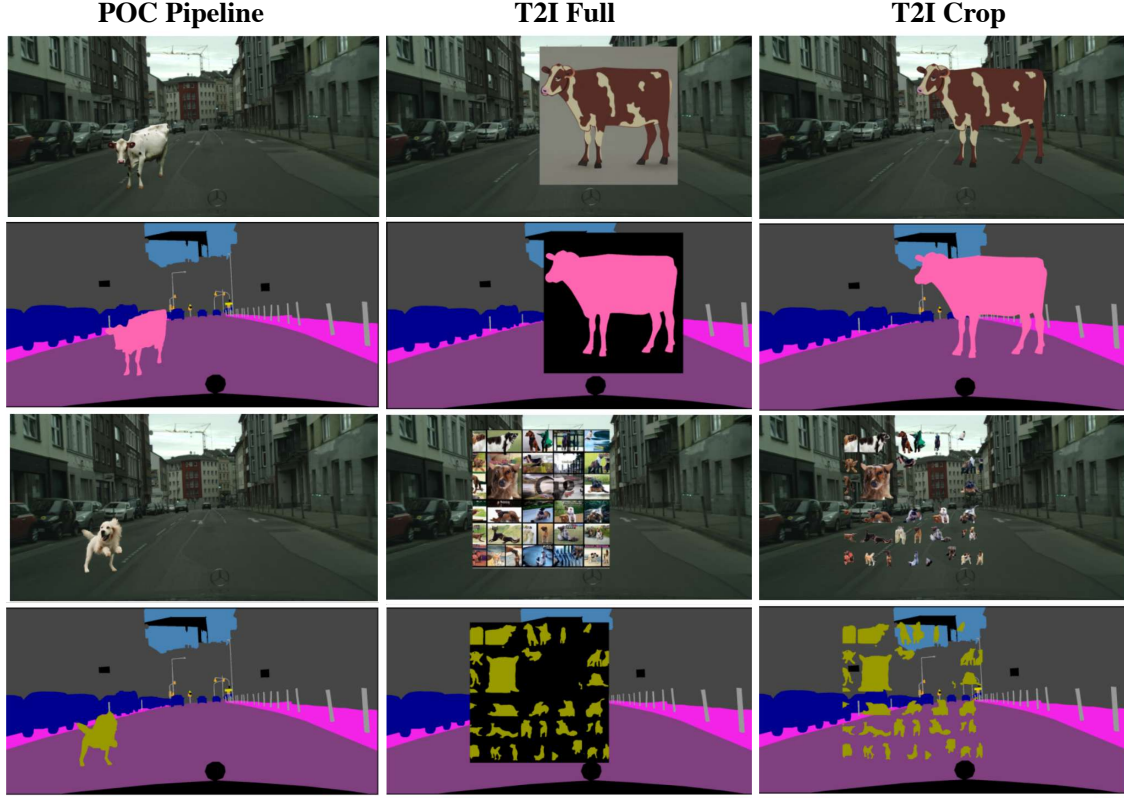


Figure 6.6: Training image samples. Text2Image (T2I) methods often lead to unrealistic objects. Moreover, sometimes T2I outputs are misaligned with caption, *e.g.* “an image of a dog” leading to a *collage* of dog images. More samples in Appendix D.5

undetected. Thus, we add *animal* classes to obtain *POC A* and *POC CS+A* (both animal and cityscapes classes). Similar to OOD detection, finding a test set (*i.e.* Cityscapes images with animals) is challenging. Inspired by Karazija *et al.* Karazija et al. (2023), we use Pascal Everingham et al. (2012) animal classes to assess the synth2real performance of POC-trained models. For completeness, we also evaluate on Cityscapes *extended* with POC.

T2I baseline. Karazija *et al.* Karazija et al. (2023) use text2image diffusion models to cluster the features of a frozen model for zero-shot segmentation. Although it is not the same task (in our case we would not like to use a frozen model but to train one from scratch), we include two augmented datasets in our experiments: *T2I Full* and *T2I Crop* where we take the Full (or cropped) image generated by a T2I model and stitch it into Cityscapes images (we use the text2image model from the same

work used in our POC Rombach et al. (2022)). In Figure 6.6 we show a comparison of the synthetic datasets. Using T2I baselines often leads to unrealistic object size and position. Moreover, perhaps due to the lack of context, the T2I model often generates images misaligned with our goal (*e.g.* a composition of many small dog images). We obtain the segmentation labels with the same open-vocabulary model as our POC pipeline, which is more flexible than foreground/background segmentation suggested in the original work Karazija et al. (2023).

Architectures. We perform experiments using three different models: *DLV3+* Cheng et al. (2020) with ResNet101 He et al. (2016a); *ConvNext*, a recent convolutional architecture Liu et al. (2022); *Segmenter* Strudel et al. (2021), a recent transformer architecture. We use the default training settings for each model. Note our goal is not to compare the architectures, but the extended datasets.

6.5.2 Results

POC performs better than T2I baselines. In Table 6.3 we show the mIoU for each model after training on our different datasets, as well as two baselines trained on Cityscapes and Pascal. In all architectures, we find that training with POC datasets leads to better results than their T2I counterparts on CS extended and Pascal (animal), *i.e.* only animal classes of Pascal. We argue this is due to a higher realism in generated images as observed in Figure 6.6. Interestingly, POC also improves performance on the original Cityscapes. Perhaps extending classes acts as a form of regularization.

Generalization is key to learn from synthetic data. We observe that the performance of DLV3+ trained on *POC A* when evaluated on Pascal (30.43) is much lower than that of Segmenter (92.4), which achieves an *mIoU competitive with the baseline trained directly on Pascal* (94.75). In order to understand that, we evaluate the performance of such models on the Pascal classes also present on Cityscapes (*i.e.* car, motorcycle, bike, person, train and bus). This shows that DLV3+ has much less generalization capability independently of the synthetic classes (*i.e.* DLV3+ trained on

Model	Train set	CS (19 cl.)	CS ext. (19 + 6 cl.)	Pascal (animal)	Pascal (citysc.)
DLV3+	Pascal (baseline)	—	—	80.57	85.81
	CS (baseline)	79.09	—	—	42.22
	T2I Full	77.13	73.2	28.32	23.51
	T2I Crop	78.23	81.49	26.15	25.09
	POC A	79.98	84.09	30.43	35.83
	POC CS+A	79.9	83.8	28.11	53.57
CNXT	Pascal (baseline)	—	—	94.43	93.92
	CS (baseline)	81.57	—	—	70.49
	T2I Full	82.26	82.99	60.41	65.6
	T2I Crop	82.43	86.04	61.09	67.55
	POC A	82.94	86.68	65.46	70.87
	POC CS+A	82.54	86.09	69.75	82.43
Segm.	Pascal (baseline)	—	—	94.75	91.43
	CS (baseline)	76.19	—	—	79.87
	T2I Full	77.19	79.46	81.96	74.32
	T2I Crop	77.62	81.27	75.95	75.44
	POC A	78.48	82.28	92.4	79.1
	POC CS+A	78.39	81.92	93.14	89.55
GSAM	Open Vocab.	42.0	41.06	75.13	76.08

Table 6.3: mIoU evaluation. We train three architectures on Pascal and Cityscapes (as baselines) and compare two Text2Image (T2I) generation methods with our POC. We also add the open vocabulary model used in POC for completeness.

Cityscapes has a much lower mIoU than Segmenter). One could argue that perhaps the low performance of DLV3+ on Pascal is due to the large domain shift, however, in Figure 6.7 we show qualitative results on web images of driving scenes, closer to the Cityscapes domain, and still observe that DLV3+ is notably worse than the other methods.

We hypothesize that, in order to learn transferable features from synthetic data, the generalization capability of the model plays a key role. Indeed, although generative models have improved the realism of generated content remarkably, there is still a synth2real gap. Thus, more robust models (*i.e.* with strong transferability of learned features) may be able to extract more useful features rather than overfitting to brittle patterns in generated data. Pre-training might also play a role: DLV3+ backbone was pretrained on Imagenet-1k, while ConvNeXt and Segmenter on Imagenet-21k.

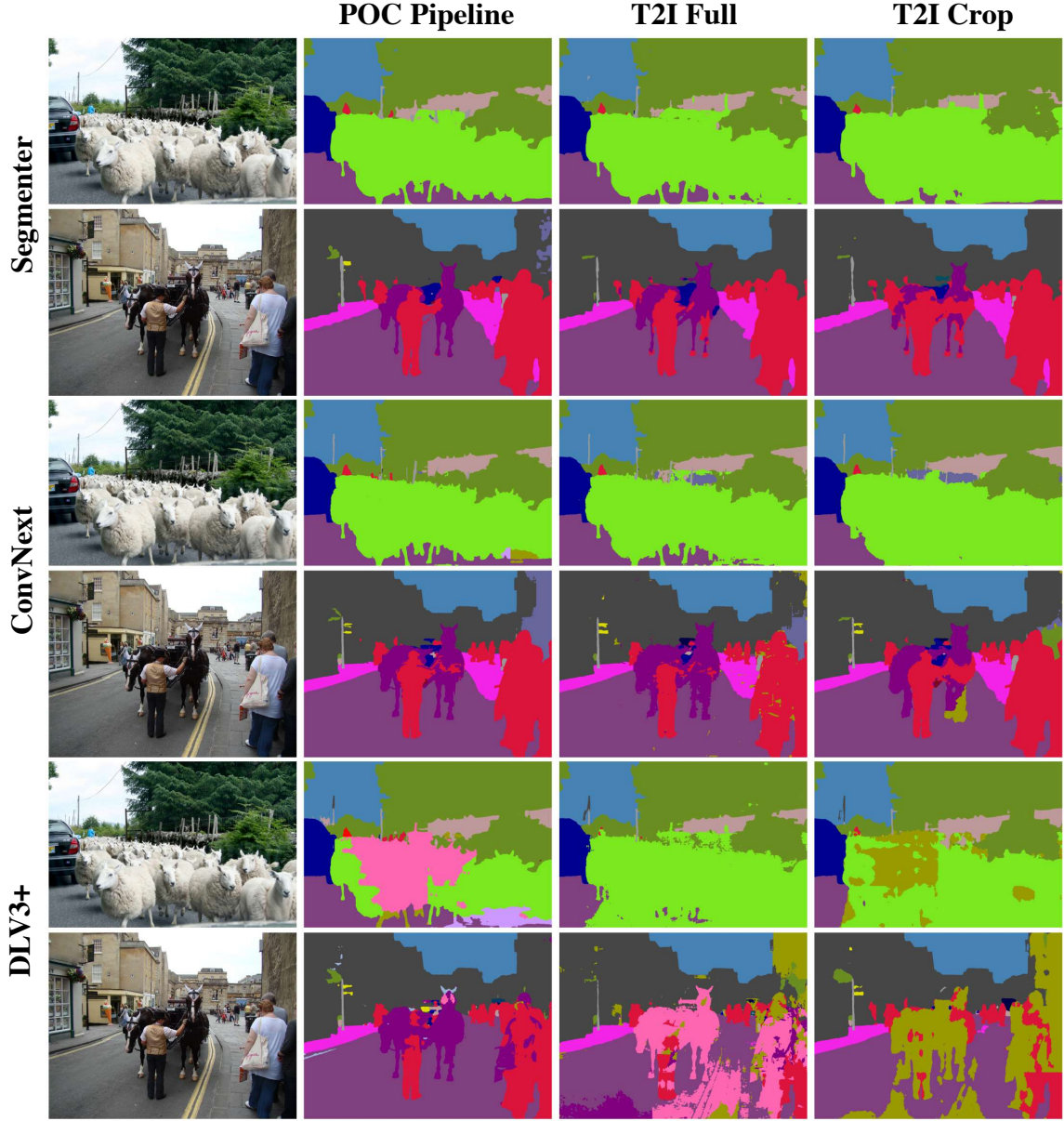


Figure 6.7: Qualitative results. We present results on web images that have a milder domain shift compared to Pascal dataset. We observe less notable differences between CNXT and Segmenter but DLV3+ is still significantly worse. More results in Appendix D.7

Nevertheless, we also observe a gap between ConvNeXt and Segmenter, both in the generalization from CS to Pascal (CS classes) and in Pascal (animals) when trained on *POC A*. Perhaps self-attention or image tokenization play a role, but this would require a more in-depth analysis. Yet, understanding the generalization gap between such models is out of the scope of this work.

POC to augment existing classes. Given the competitive performance of Segmenter with *POC A* on Pascal (animals) compared to Pascal (CS classes), we are compelled to also inpaint Cityscapes classes with POC, obtaining *POC CS+A*. We observe remarkable improvements in Pascal (CS classes) in all three networks, significantly outperforming the baseline that was only trained on Cityscapes. Coupled with the milder but consistent improvement on the original Cityscapes, this indicates that our pipeline could also be helpful in improving generalization.

Open vocabulary baseline. Considering that we use a single open-vocabulary segmentation method (GSAM Grounded-SAM Contributors (2023)) to label all objects inserted with POC, we also evaluate its performance on this task as a baseline. Although its performance on Pascal is reasonable, it significantly underperforms on Cityscapes. We hypothesize this may be due to the one-vs-all nature of open-vocabulary predictors (where each class is predicted individually with a prompt) that does not perform well on complex images with many classes. Interestingly, the Segmenter model trained with POC significantly outperforms GSAM on Pascal. We find this remarkable as we could interpret it as knowledge distillation from a teacher model (GSAM) while significantly improving its performance. On the other hand, note that the setting is very different when we use GSAM in the POC pipeline, as we know which object is inpainted. We also observe a strong improvement when labelling only the cropped region $\tilde{x}_r$ instead of the full image $\tilde{x}$.

6.6 Conclusion

To accurately assess anomaly segmentation capabilities of models deployed in open-world settings, we argue that datasets should be realistic and carry a small domain shift w.r.t. the training distribution, as we show it can hinder OOD detection. Towards this goal, we introduce the Placing Object in Context (POC) pipeline, that allows adding *any* (OOD) object into *any* image based on simple prompting. POC uses diffusion models and open-vocabulary segmentation to achieve high realism and versatility.

We showcase POC’s flexibility by generating three anomaly segmentation test sets: POC-CS, POC-IDD and POC-ACDC. Moreover, we observe that generating data for OOD fine-tuning with POC brings significant improvements in standard anomaly segmentation benchmarks.

Beyond anomaly segmentation, we use POC-generated datasets to learn new classes without any real example. Interestingly, we observe that the combination of more realistic synthetic data with recent segmentation models with strong generalization capabilities can lead to remarkable performances, competitive with training on real data.

In future work, we hope to better understand how to select the optimal list of anomalies for fine-tuning and how can modern architectures make the best use of synthetic data to learn new classes, also in a continual learning setup.

7

Concluding remarks

We close this thesis with a summary of the contributions followed by a discussion on the limitations of the work and potential avenues for future research.

7.1 Summary

Motivated by the issues that arise when deploying deep learning models on autonomous agents in the real world, in this thesis we have worked towards obtaining models that are more efficient as well as reliable. In Chapter 3 propose a method to prune networks at initialization for efficient training and inference on edge devices. To ensure deployed agents are less vulnerable to malicious attacks, in Chapter 4 we present an efficient adversarial training mechanism. Chapter 5 surveys the reliability of semantic segmentation architectures in terms of robustness and uncertainty estimation. In a similar spirit, in Chapter 6 we tackle the need to recognize unknown (OOD) objects for a safe deployment in applications like autonomous driving. In the following, we outline the highlights of each contribution in more detail.

Pruning at initialization – Chapter 3 In this chapter, we explore prior methods to prune at initialization and notice that when removing a large portion of the network

weights, the performance degrades significantly, even below the performance of pruning weights randomly. This is because prior work assumes the gradient of the network after pruning is similar to that of the network before pruning, which, when pruning at high sparsity, would not hold. Instead, we re-formulate the pruning objective function to make explicit that we want to optimize the network after pruning. Then, we propose to remove connections progressively so that the difference in gradients at each step would remain similar.

Efficient adversarial training – Chapter 4 While adversarial training can lead to robust networks, it increases the training cost linearly with the number of gradient steps used to augment the images, thus, to reduce the computational overhead it is desirable to reduce the number of steps. In this chapter we review prior single-step adversarial training methods and propose a simple but effective solution to reduce Catastrophic Overfitting by combining noise augmentations with adversarial attacks. In particular, before computing a single-step adversarial perturbation, we augment the image with random noise. This allows us to match or surpass the robustness of previous state-of-the-art single-step methods at a fraction of the cost.

Reliability in semantic segmentation – Chapter 5 This chapter presents an analysis of recent segmentation models in comparison to older baseline models. Specifically, we are interested in model robustness and uncertainty estimation capabilities under natural distribution shifts. We find that although recent segmentation models seem to be more robust *off-the-shelf*, they are not better calibrated, especially under strong domain shifts. This motivates the need to find methods to improve uncertainty estimation after training. In that regard, we identify adaptive temperature scaling as a promising direction to improve calibration out of domain. Moreover, we observe that improving calibration can also benefit misclassification or OOD detection as well.

Out of distribution segmentation – Chapter 6 In this chapter we tackle the task of anomaly segmentation. That is, detecting pixels that belong to a semantic category not present during training. Given that the distribution of anomaly classes is long tail and annotating data is expensive, we note one of the main challenges in

this field is to produce high-quality anomaly datasets. Leveraging recent advances in generative models, we present a pipeline to add objects into images realistically via inpainting. We leverage our pipeline to generate different anomaly datasets for evaluation and show it can also be used to fine-tune anomaly segmentation methods. Beyond anomaly segmentation, we also show that our pipeline can be used to extend a dataset with new semantic classes.

7.2 Limitations and future work

We outline the limitations for each contribution separately and discuss potential directions for future work.

Pruning at initialization – Chapter 3 Although pruning at initialization can leverage the benefits of pruning during training, the performance of models pruned at initialization is still sub-par to more expensive methods that perform several *train - prune - fine-tune* cycles. This is reasonable since, after training, one might have more information about the importance of each parameter. However, one interesting direction for future work might be to further explore the trade-off between the number of training iterations before pruning and final performance.

Another relevant design choice is to perform unstructured pruning. While unstructured pruning methods can reach higher sparsity levels without loss of performance, standard deep learning frameworks do not benefit from sparse tensors by default and specific libraries need to be used to achieve speed-ups. On the other hand, structured pruning that removes full channels or neurons, can lead to speed-ups directly.

Efficient adversarial training – Chapter 4 Our proposed method for single-step adversarial training can yield robust networks with a smaller computational overhead and it matches state-of-the-art results of single-step methods. Nevertheless, the costly multi-step adversarial training methods can still provide more robustness. We observe in Appendix B that combining our method with additional regularizers can further

boost performance, pushing further in this direction to match the results of multi-step adversarial methods at no extra cost would be a relevant direction for future work.

On the other hand, although our proposed method can prevent Catastrophic Overfitting, it still does not fully explain why are single-step methods prone to Catastrophic Overfitting in the first place. In Appendix E we present a follow-up work in which we delve into the cause of Catastrophic Overfitting. Specifically, we design a dataset in which networks are even more prone to suffer Catastrophic Overfitting and use it to perform controlled studies.

Reliability in semantic segmentation – Chapter 5 Our study uncovers a significant limitation of recent segmentation models. That is, although they seem to enjoy better robustness to domain shifts, their uncertainty estimation capabilities do not improve. We find that content-dependent temperature scaling seems like a promising direction to tackle this but further research in this direction would be warranted given that calibration out of domain is still significantly worse than in domain.

Another interesting direction to investigate is what particular design choice makes recent models more robust. From our experiments, we observe that both recent transformer and convolutional-based architectures are robust and we also ablate the number of training iterations and the choice pre-training datasets in Appendix C with no clear difference.

Out of distribution segmentation – Chapter 6 Although generative models allow us to inpaint potentially any object simply by changing the text prompt, they have notable biases which, if unaddressed, might transfer into the generated datasets. For instance, when inpainting the class “truck” (as an ID object), we noticed most generated trucks were American trucks while the Cityscapes dataset has European trucks. On the bright side, as generative models improve in realism and steerability, so could improve our pipeline.

Another limitation of our pipeline is that the position and size of added objects is still rather coarse, which can hinder realism in some cases. In order to further improve the realism one could think of training a generative model to specifically inpaint objects (instead of simply reconstructing randomly masked parts of images). However, obtaining the ground truth (*i.e.* an image with an object and the same image without it) is not trivial. A first attempt to generate such dataset could be done by removing labelled objects from images with an existing inpainting model.

Bibliography

- N. Ahmed, T. Natarajan, and K. R. Rao. Discrete Cosine Transform. *IEEE Transactions on Computers*, 1974.
- M. Andriushchenko and N. Flammarion. Understanding and improving fast adversarial training. In *Neural Information Processing Systems (NeurIPS)*, 2020.
- D. Arpit, S. Jastrzębski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien. A closer look at memorization in deep networks. In *International Conference on Machine Learning (ICML)*, 2017.
- J. Aspnes, R. Beigel, M. Furst, and S. Rudich. The expressive power of voting polynomials. *Combinatorica*, 1994.
- O. Avrahami, D. Lischinski, and O. Fried. Blended diffusion for text-driven editing of natural images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18208–18218, 2022.
- V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: a Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 39(12):2481–2495, 2017.
- Y. Bai, J. Mei, A. L. Yuille, and C. Xie. Are transformers more robust than CNNs? In *NeurIPS*, 2021.
- G. Bellec, D. Kappel, W. Maass, and R. Legenstein. Deep rewiring: Training very sparse deep networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BJ_wN01C-.

- V. Besnier, A. Bursuc, D. Picard, and A. Briot. Triggering failures: Out-of-distribution detection by learning from local adversarial attacks in semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15701–15710, 2021.
- S. Bhojanapalli, A. Chakrabarti, D. Glasner, D. Li, T. Unterthiner, and A. Veit. Understanding robustness of transformers for image classification. In *ICCV*, 2021.
- B. Biggio and F. Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 2018.
- C. M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.
- H. Blum, P.-E. Sarlin, J. Nieto, R. Siegwart, and C. Cadena. Fishyscapes: A benchmark for safe semantic segmentation in autonomous driving. In *proceedings of the IEEE/CVF international conference on computer vision workshops*, pages 0–0, 2019.
- A. Boloor, X. He, C. D. Gill, Y. Vorobeychik, and X. Zhang. Simple physical adversarial examples against end-to-end autonomous driving models. *arxiv:1903.05157*, 2019.
- T. Brooks, A. Holynski, and A. A. Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023.
- N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.
- N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017.
- M. A. Carreira-Perpiñán and Y. Idelbayev. “learning-compression” algorithms for neural net pruning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

- R. Chan, K. Lis, S. Uhlemeyer, H. Blum, S. Honari, R. Siegwart, P. Fua, M. Salzmann, and M. Rottmann. Segmentmeifyoucan: A benchmark for anomaly segmentation. *arXiv preprint arXiv:2104.14812*, 2021a.
- R. Chan, M. Rottmann, and H. Gottschalk. Entropy maximization and meta classification for out-of-distribution detection in semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5128–5137, 2021b.
- Y. Chauvin. A back-propagation algorithm with optimal use of hidden units. In *Advances in neural information processing systems*, pages 519–526, 1989.
- L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 40(4):834–848, 2017a.
- L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv:1706.05587*, 2017b.
- L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *ECCV*, 2018.
- B. Cheng, M. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L.-C. Chen. Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation. In *CVPR*, 2020.
- B. Cheng, A. Schwing, and A. Kirillov. Per-pixel classification is not all you need for semantic segmentation. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL <https://openreview.net/forum?id=0lz69oI5iZP>.
- B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2022.

- S. Choi, S. Jung, H. Yun, J. Kim, S. Kim, and J. Choo. RobustNet: Improving Domain Generalization in Urban-Scene Segmentation via Instance Selective Whitening. In *CVPR*, 2021.
- M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier. Parseval networks: Improving robustness to adversarial examples. In *International Conference on Machine Learning (ICML)*, 2017.
- M. Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. <https://github.com/open-mmlab/msegmentation>, 2020.
- C. Corbière, N. Thome, A. Bar-Hen, M. Cord, and P. Pérez. Addressing failure prediction by learning model confidence. *Advances in Neural Information Processing Systems*, 32, 2019.
- M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *CVPR*, 2016a.
- M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016b.
- F. Croce and M. Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning (ICML)*, 2020.
- G. Csurka. A Comprehensive Survey on Domain Adaptation for Visual Applications. In G. Csurka, editor, *Domain Adaptation in Computer Vision Applications*, Advances in Computer Vision and Pattern Recognition, pages 1–35. Springer, 2017.
- G. Csurka, R. Volpi, B. Chidlovskii, et al. Semantic image segmentation: Two decades of research. *Foundations and Trends® in Computer Graphics and Vision*, 14(1-2): 1–162, 2022.

- J. Dai, K. He, Y. Li, S. Ren, and J. Sun. Instance-sensitive Fully Convolutional Networks. In *ECCV*, 2016.
- X. Dai, H. Yin, and N. K. Jha. Nest: A neural network synthesis tool based on a grow-and-prune paradigm. *IEEE Transactions on Computers*, 68(10):1487–1497, 2019.
- P. de Jorge, A. Sanyal, H. Behl, P. Torr, G. Rogez, and P. K. Dokania. Progressive skeletonization: Trimming more fat from a network at initialization. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=9GsFOUyUPi>.
- P. de Jorge, A. Bibi, R. Volpi, A. Sanyal, P. H. S. Torr, G. Rogez, and P. K. Dokania. Make some noise: Reliable and efficient single-step adversarial training. *arxiv:2202.01181*, 2022.
- P. de Jorge, R. Volpi, P. H. Torr, and G. Rogez. Reliability in semantic segmentation: Are we on the right track? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7173–7182, 2023.
- T. Dettmers and L. Zettlemoyer. Sparse networks from scratch: Faster training without losing performance, 2020. URL <https://openreview.net/forum?id=ByeSYa4KPS>.
- J. Devlin, M. Changm, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*, 2019.
- P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.
- G. Di Biase, H. Blum, R. Siegwart, and C. Cadena. Pixel-wise anomaly detection in complex driving scenes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16918–16927, 2021.

- Z. Ding, X. Han, P. Liu, and M. Niethammer. Local temperature scaling for probability calibration. In *ICCV*, 2021.
- A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2021.
- X. Du, Y. Sun, X. Zhu, and Y. Li. Dream the impossible: Outlier imagination with diffusion models. *arXiv preprint arXiv:2309.13415*, 2023.
- E. Elsen, M. Dukhan, T. Gale, and K. Simonyan. Fast sparse convnets. *arXiv preprint arXiv:1911.09723*, 2019.
- U. Evci, T. Gale, J. Menick, P. S. Castro, and E. Elsen. Rigging the lottery: Making all tickets winners. *arXiv preprint arXiv:1911.11134*, 2019.
- 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>, 2012.
- A. Fawzi, S.-M. Moosavi-Dezfooli, P. Frossard, and S. Soatto. Empirical study of the topology and geometry of deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=rJl-b3RcF7>.
- J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin. Stabilizing the lottery ticket hypothesis, 2020.
- G. Fumera and F. Roli. Support vector machines with embedded reject option. In *International Workshop on Support Vector Machines*, 2002.

- Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.
- J. Gilmer, N. Ford, N. Carlini, and E. Cubuk. Adversarial examples are a natural consequence of test error in noise. In *International Conference on Machine Learning (ICML)*, 2019.
- Z. Golgooni, M. Saberi, M. Eskandar, and M. H. Rohban. Zeroograd: Mitigating and explaining catastrophic overfitting in fgsm adversarial training. *arXiv:2103.15476*, 2021.
- Y. Gong, X. Lin, Y. Yao, T. G. Dietterich, A. Divakaran, and M. Gervasio. Confidence calibration for domain generalization under covariate shift. In *ICCV*, 2021.
- I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *International Conference on Learning Representations (ICLR)*, 2015.
- J. Grabinski, S. Jung, J. Keuper, and M. Keuper. FrequencyLowCut Pooling - Plug & Play against Catastrophic Overfitting. In *European Conference on Computer Vision (ECCV)*, 2022.
- M. Grcić, P. Bevandić, and S. Šegvić. Densehybrid: Hybrid anomaly detection for dense open-set recognition. In *European Conference on Computer Vision*, pages 500–517. Springer, 2022.
- Grounded-SAM Contributors. Grounded-Segment-Anything. LICENSE Apache-2.0. <https://github.com/IDEA-Research/Grounded-Segment-Anything>, Apr. 2023.
- C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.
- Y. Guo, A. Yao, and Y. Chen. Dynamic network surgery for efficient dnns. In *Advances in neural information processing systems*, pages 1379–1387, 2016.

- K. Gupta, A. Rahimi, T. Ajanthan, T. Mensink, C. Sminchisescu, and R. Hartley. Calibration of neural networks using splines. *arXiv:2006.12800*, 2020.
- D. Haldimann, H. Blum, R. Siegwart, and C. Cadena. This is not what i imagined: Error detection for semantic segmentation through visual dissimilarity. *arXiv preprint arXiv:1909.00676*, 2019.
- S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.
- B. Hassibi, D. G. Stork, and G. J. Wolff. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE, 1993.
- K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.
- K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016a.
- K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)*, 2016b.
- T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li. Bag of tricks for image classification with convolutional neural networks. In *CVPR*, 2019.
- D. Hendrycks and T. Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *ICLR*, 2019.
- D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.
- D. Hendrycks, S. Basart, M. Mazeika, A. Zou, J. Kwon, M. Mostajabi, J. Steinhardt, and D. Song. Scaling out-of-distribution detection for real-world settings. *arXiv preprint arXiv:1911.11132*, 2019.

- D. Hendrycks, K. Zhao, S. Basart, J. Steinhardt, and D. Song. Natural adversarial examples. In *CVPR*, 2021.
- J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry. Adversarial examples are not bugs, they are features. *Neural Information Processing Systems (NeurIPS)*, 2019.
- S. Jetley, N. Lord, and P. Torr. With friends like these, who needs adversaries? *Neural Information Processing Systems (NeurIPS)*, 2018.
- H. Jiang, B. Kim, M. Guan, and M. Gupta. To trust or not to trust a classifier. *Advances in neural information processing systems*, 31, 2018.
- S. Jung, J. Lee, D. Gwak, S. Choi, and J. Choo. Standardized max logits: A simple yet effective approach for identifying unexpected road obstacles in urban-scene segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15425–15434, 2021.
- C. Kamann and C. Rother. Benchmarking the Robustness of Semantic Segmentation Models. In *CVPR*, 2020.
- P. Kang and S.-M. Moosavi-Dezfooli. Understanding catastrophic overfitting in adversarial training. *arXiv:2105.02942*, 2021.
- L. Karazija, I. Laina, A. Vedaldi, and C. Rupprecht. Diffusion models for zero-shot open-vocabulary segmentation. *arXiv preprint arXiv:2306.09316*, 2023.
- H. Kim, W. Lee, and J. Lee. Understanding catastrophic overfitting in single-step adversarial training. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021.
- K. Kireev, M. Andriushchenko, and N. Flammarion. On the effectiveness of adversarial training against common corruptions. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2022.

- A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.
- A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby. Big transfer (bit): General visual representation learning. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pages 491–507. Springer, 2020.
- A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems (NeurIPS)*, 2012.
- A. Krizhevsky et al. Learning multiple layers of features from tiny images, 2009.
- J. Kruger and D. Dunning. Unskilled and unaware of it: how difficulties in recognizing one’s own incompetence lead to inflated self-assessments. *Journal of personality and social psychology*, 77(6):1121, 1999.
- M. Kull, M. Perello Nieto, M. Kängsepp, T. Silva Filho, H. Song, and P. Flach. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. In *NeurIPS*, 2019.
- A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations (ICLR)*, 2017a.
- A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *International Conference on Learning Representations (ICLR), Workshops*, 2017b.
- A. Kusupati, V. Ramanujan, R. Somani, M. Wortsman, P. Jain, S. M. Kakade, and A. Farhadi. Soft threshold weight reparameterization for learnable sparsity. *ArXiv*, abs/2002.03231, 2020.
- B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

- Y. LeCun, J. S. Denker, and S. A. Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990.
- Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature Cell Biology*, 521(7553): 436–444, May 2015. ISSN 1465-7392. doi: 10.1038/nature14539.
- K. Lee, K. Lee, H. Lee, and J. Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018.
- N. Lee, T. Ajanthan, and P. Torr. SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=B1VZqjAcYX>.
- N. Lee, T. Ajanthan, S. Gould, and P. H. S. Torr. A signal propagation perspective for pruning neural networks at initialization. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=HJeTo2VFwH>.
- B. Li, S. Wang, S. Jana, and L. Carin. Towards understanding fast adversarial training. *arXiv:2006.03089*, 2020.
- C. Liang, W. Wang, J. Miao, and Y. Yang. Gmmseg: Gaussian mixture based generative semantic segmentation models. *Advances in Neural Information Processing Systems*, 35:31360–31375, 2022.
- S. Liang, Y. Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.
- T. Lin, S. U. Stich, L. Barba, D. Dmitriev, and M. Jaggi. Dynamic model pruning with feedback. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=SJem8lSFwB>.
- T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

- K. Lis, K. Nakka, P. Fua, and M. Salzmann. Detecting the unexpected via image resynthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2152–2161, 2019.
- K. Lis, S. Honari, P. Fua, and M. Salzmann. Detecting road obstacles by erasing them. *arXiv preprint arXiv:2012.13633*, 2020.
- S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023a.
- W. Liu, X. Wang, J. Owens, and Y. Li. Energy-based out-of-distribution detection. *Advances in neural information processing systems*, 33:21464–21475, 2020.
- Y. Liu, C. Ding, Y. Tian, G. Pang, V. Belagiannis, I. Reid, and G. Carneiro. Residual pattern learning for pixel-wise out-of-distribution detection in semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1151–1161, 2023b.
- Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2018.
- Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *ICCV*, 2021.
- Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A ConvNet for the 2020s. In *CVPR*, 2022.
- C. Louizos, M. Welling, and D. P. Kingma. Learning sparse neural networks through l0 regularization. In *International Conference on Learning Representations*, 2018. URL <https://openreview.net/forum?id=H1Y8hhg0b>.
- A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

- A. Malinin, B. Mlodozienec, and M. Gales. Ensemble distribution distillation. arXiv:1905.00076, 2019.
- A. Mallya, D. Davis, and S. Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018.
- X. Mao, G. Qi, Y. Chen, X. Li, R. Duan, S. Ye, Y. He, and H. Xue. Towards robust vision transformer. In *CVPR*, 2022.
- C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.
- M. Minderer, J. Djolonga, R. Romijnders, F. Hubis, X. Zhai, N. Houlsby, D. Tran, and M. Lucic. Revisiting the calibration of modern neural networks. *NeurIPS*, 2021.
- D. C. Mocanu, E. Mocanu, P. Stone, P. H. Nguyen, M. Gibescu, and A. Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12, 2018.
- P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations*, 2017.
- S.-M. Moosavi-Dezfooli, A. Fawzi, J. Uesato, and P. Frossard. Robustness via curvature regularization, and vice versa. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- H. Mostafa and X. Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization, 2019. URL <https://openreview.net/forum?id=S1xBioR5KX>.
- M. C. Mozer and P. Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Advances in Neural Information Processing Systems*, 1989.

- J. Mukhoti and Y. Gal. Evaluating bayesian deep learning methods for semantic segmentation. *arXiv preprint arXiv:1811.12709*, 2018.
- K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.
- M. P. Naeini, G. Cooper, and M. Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *AAAI*, 2015.
- M. M. Naseer, K. Ranasinghe, S. H. Khan, M. Hayat, F. Shahbaz Khan, and M.-H. Yang. Intriguing properties of vision transformers. *NeurIPS*, 2021.
- N. Nayal, M. Yavuz, J. F. Henriques, and F. Güney. Rba: Segmenting unknown regions rejected by all. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 711–722, 2023.
- Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *Neural Information Processing Systems (NeurIPS), Workshops*, 2011.
- A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.
- K. Nguyen and B. O’Connor. Posterior calibration and exploratory analysis for natural language processing models. *arXiv:1508.05154*, 2015.
- A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.
- J. Nixon, M. W. Dusenberry, L. Zhang, G. Jerfel, and D. Tran. Measuring calibration in deep learning. In *CVPR Workshops*, 2019.
- H. Noh, S. Hong, and B. Han. Learning Deconvolution Network for Semantic Segmentation. In *ICCV*, 2015.

- G. Ortiz-Jimenez, A. Modas, S.-M. Moosavi, and P. Frossard. Neural anisotropy directions. *Neural Information Processing Systems (NeurIPS)*, 2020a.
- G. Ortiz-Jimenez, A. Modas, S.-M. Moosavi, and P. Frossard. Hold me tight! influence of discriminative features on deep network boundaries. *Neural Information Processing Systems (NeurIPS)*, 2020b.
- G. Ortiz-Jimenez, P. de Jorge, A. Sanyal, A. Bibi, P. K. Dokania, P. Frossard, G. Rogez, and P. Torr. Catastrophic overfitting can be induced with discriminative non-robust features. *Transactions on Machine Learning Research*, 2023.
- G. Ortiz-Jiménez, A. Modas, S.-M. Moosavi-Dezfooli, and P. Frossard. Optimism in the face of adversity: Understanding and improving deep learning through adversarial robustness. *Proceedings of the IEEE*, 2021.
- Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek. Can you trust your model’s uncertainty? evaluating predictive uncertainty under dataset shift. *NeurIPS*, 2019.
- A. Pampari and S. Ermon. Unsupervised calibration under covariate shift. arXiv:2006.16405, 2020.
- N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE symposium on security and privacy (SP)*, 2016.
- G. Y. Park and S. W. Lee. Reliably fast adversarial training via latent adversarial perturbation. In *International Conference on Learning Representations (ICLR), Workshops*, 2021.
- S. Park, O. Bastani, J. Weimer, and I. Lee. Calibrated prediction with covariate shift via unsupervised domain adaptation. In *AISTATS*, 2020.
- A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch, 2017.
- S. Paul and P.-Y. Chen. Vision transformers are robust learners. In *AAAI*, 2022.

- P. Pinggera, S. Ramos, S. Gehrig, U. Franke, C. Rother, and R. Mester. Lost and found: detecting small road hazards for self-driving vehicles. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1099–1106. IEEE, 2016.
- F. Pinto, P. H. Torr, and P. K. Dokania. An impartial take to the cnn vs transformer robustness contest. In *ECCV*, 2022.
- C. Qin, J. Martens, S. Gowal, D. Krishnan, K. Dvijotham, A. Fawzi, S. De, R. Stanforth, and P. Kohli. Adversarial robustness through local linearization. In *Neural Information Processing Systems (NeurIPS)*, 2019.
- S. N. Rai, F. Cermelli, D. Fontanel, C. Masone, and B. Caputo. Unmasking anomalies in road-scene segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4037–4046, 2023.
- A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.
- B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. Do imagenet classifiers generalize to imagenet? In *ICML*, 2019.
- L. Rice, E. Wong, and Z. Kolter. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning (ICML)*, 2020.
- R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.
- O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI*, 2015.
- O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3): 211–252, 2015. doi: 10.1007/s11263-015-0816-y.

- W. Saad and A. Alsayyari. Loose animal-vehicle accidents mitigation: Vision and challenges. In *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*, pages 359–364. IEEE, 2019.
- C. Saharia, W. Chan, H. Chang, C. Lee, J. Ho, T. Salimans, D. Fleet, and M. Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022a.
- C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022b.
- C. Sakaridis, D. Dai, and L. Van Gool. ACDC: The Adverse Conditions Dataset with Correspondences for Semantic Driving Scene Understanding. In *ICCV*, 2021.
- M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. doi: 10.1109/CVPR.2018.00474.
- A. Sanyal, V. Kanade, and P. H. S. Torr. Robustness via deep low-rank representations. *arxiv:1804.07090*, 2018.
- A. Sanyal, P. K. Dokania, V. Kanade, and P. Torr. How benign is benign overfitting ? In *International Conference on Learning Representations (ICLR)*, 2021.
- C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022.
- A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein. Adversarial training for free! *Neural Information Processing Systems (NeurIPS)*, 2019.

- H. Shah, K. Tamuly, A. Raghunathan, P. Jain, and P. Netrapalli. The pitfalls of simplicity bias in neural networks. *Neural Information Processing Systems (NeurIPS)*, 2020.
- Y. Shi, I. Daunhawer, J. E. Vogt, P. H. Torr, and A. Sanyal. How robust are pre-trained models to distribution shift? *arXiv preprint arXiv:2206.08871*, 2022.
- D. Silver et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.
- K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.
- J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.
- Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.
- G. Sriramanan, S. Addepalli, A. Baburaj, et al. Guided adversarial attack for evaluating and enhancing adversarial defenses. *Advances in Neural Information Processing Systems*, 33:20297–20308, 2020.
- G. Sriramanan, S. Addepalli, A. Baburaj, et al. Towards efficient and effective adversarial training. *Advances in Neural Information Processing Systems*, 34:11821–11833, 2021.
- R. Strudel, R. Garcia, I. Laptev, and C. Schmid. Segmenter: Transformer for Semantic Segmentation. In *ICCV*, 2021.
- C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

- C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.
- H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6377–6389. Curran Associates, Inc., 2020. URL <https://proceedings.neurips.cc/paper/2020/file/46a4378f835dc8040c8057beb6a2da52-Paper.pdf>.
- R. Taori, A. Dave, V. Shankar, N. Carlini, B. Recht, and L. Schmidt. Measuring robustness to natural distribution shifts in image classification. *NeurIPS*, 2020.
- N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso. The computational limits of deep learning. *arXiv preprint arXiv:2007.05558*, 2020.
- Y. Tian, Y. Liu, G. Pang, F. Liu, Y. Chen, and G. Carneiro. Pixel-wise energy-biased abstention learning for anomaly segmentation on complex urban driving scenes. In *European Conference on Computer Vision*, pages 246–263. Springer, 2022.
- M. Toldo, A. Maracani, U. Michieli, and P. Zanuttigh. Unsupervised Domain Adaptation in Semantic Segmentation: a Review. *arXiv:2005.10876*, 2020.
- H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021.
- F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations (ICLR)*, 2018.
- G. Varma, A. Subramanian, A. Namboodiri, M. Chandraker, and C. Jawahar. IDD: A Dataset for Exploring Problems of Autonomous Navigation in Unconstrained Environments. In *WACV*, 2019.

- S. Verdenius, M. Stol, and P. Forré. Pruning via iterative ranking of sensitivity statistics. *arXiv preprint arXiv:2006.00896*, 2020.
- B. Vivek and R. V. Babu. Single-step adversarial training with dropout scheduling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- R. Volpi, H. Namkoong, O. Sener, J. C. Duchi, V. Murino, and S. Savarese. Generalizing to Unseen Domains via Adversarial Data Augmentation. In *NeurIPS*, 2019.
- R. Volpi, P. De Jorge, D. Larlus, and G. Csurka. On the Road to Online Adaptation for Semantic Image Segmentation. In *CVPR*, 2022.
- C. Wang, G. Zhang, and R. Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020a. URL <https://openreview.net/forum?id=SkgsACVKPH>.
- X. Wang, M. Long, J. Wang, and M. Jordan. Transferable calibration with lower bias and variance in domain adaptation. *NeurIPS*, 2020b.
- L. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, L. Daniel, D. Boning, and I. Dhillon. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning (ICML)*, 2018.
- E. Wong, L. Rice, and J. Z. Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations (ICLR)*, 2020.
- Y. Xia, Y. Zhang, F. Liu, W. Shen, and A. L. Yuille. Synthesize then compare: Detecting failures and anomalies for semantic segmentation. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 145–161. Springer, 2020.
- T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun. Unified Perceptual Parsing for Scene Understanding. In *ECCV*, 2018.
- E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo. SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers. In *NeurIPS*, 2021.

- X. Yue, Y. Zhang, S. Zhao, A. Sangiovanni-Vincentelli, K. Keutzer, and B. Gong. Domain Randomization and Pyramid Consistency: Simulation-to-Real Generalization without Accessing Target Domain Data. In *ICCV*, 2019.
- S. Zagoruyko and N. Komodakis. Wide residual networks. In *bmvc*, 2016.
- H. Zhang, Y. Yu, J. Jiao, E. Xing, L. E. Ghaoui, and M. Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning (ICML)*, 2019.
- R. Zhang. Making convolutionall neural networks shift-invariant again. In *International Conference on Machine Learning (ICML)*, 2019.
- W. Zhang, J. Pang, K. Chen, and C. C. Loy. K-net: Towards unified image segmentation. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL <https://openreview.net/forum?id=uDeDDoF0Epj>.
- S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. Torr, and L. Zhang. Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers. In *CVPR*, 2021.
- D. Zhou, Z. Yu, E. Xie, C. Xiao, A. Anandkumar, J. Feng, and J. M. Alvarez. Understanding the robustness in vision transformers. In *ICML*, 2022.
- B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

Appendices



Appendix of Progressive Skeletonization: Trimming more fat from a network at initialization

A.1 Pruning implementation details

We present experiments on CIFAR-10/100 (Krizhevsky et al., 2009), which consists of 60k 32×32 colour images divided into 10/100 classes, and also on Imagenet challenge ILSVRC-2012 (Russakovsky et al., 2015) and its smaller version Tiny-ImageNet, which respectively consist of 1.2M/1k and 100k/200 images/classes. Networks are initialized using the Kaiming normal initialization (Dai et al., 2016). For CIFAR datasets, we train Resnet50¹ and VGG19² architectures during 350 epochs with a batch size of 128. We start with a learning rate of 0.1 and divide it by 10 at 150 and 250 epochs. As optimizer, we use SGD with momentum 0.9 and weight decay 5×10^{-4} . We separate 10% of the training data for validation and report results on the test set. We perform mean and std normalization and augment the data with random crops and horizontal flips. For Tiny-Imagenet, we use the same architectures. We train during 300 epochs and

¹<https://github.com/kuangliu/pytorch-cifar/blob/master/models/resnet.py>

²<https://github.com/alecwangcq/GraSP/blob/master/models/base/vgg.py>

divide the learning rate by 10 at 1/2 and 3/4 of the training. Other hyper-parameters remain the same. For ImageNet training, we adapt the official code³ of Paszke et al. (2017) and we use the default settings. In this case, we use the Resnet50 and VGG19 with batch normalization architectures as implemented in Paszke et al. (2017).

In the case of FORCE and Iter SNIP, we adapt the same public implementation⁴ of SNIP as Wang et al. (2020a). Instead of defining an auxiliary mask to compute the saliencies, we compute the product of the weight times the gradient, which was shown to be equivalent in Lee et al. (2020). As for GRASP, we use their public code.⁵ After pruning, we implement pruned connections by setting the corresponding weight to 0 and forcing the gradient to be 0. This way, a pruned weight will remain 0 during training.

An important difference between SNIP and GRASP implementations is in the way they select the mini-batch to compute the saliency. SNIP implementation simply loads a batch from the dataloader. In contrast, in GRASP implementation they keep loading batches of data until they obtain exactly 10 examples of each class, discarding redundant samples. In order to compare the methods in equal conditions, we decided to use the way SNIP collects the data since it is simpler to implement and does not require extra memory. This might cause small discrepancies between our results and the ones reported in Wang et al. (2020a).

A meaningful design choice regarding SNIP and GRASP implementations is that they only prune convolutional and fully connected layers. These layers constitute the vast majority of parameters in most networks, however, as we move to high sparsity regimes, batch norm layers constitute a non-negligible amount. For CIFAR10, batch norm plus biases constitute 0.2% and 0.05% of the parameters of Resnet50 and VGG19 networks respectively. For consistency, we have as well restricted pruning to convolutional and fully connected layers and reported percentage sparsity with respect to the prunable parameters, as is also done in Lee et al. (2019) and Wang et al. (2020a) to the best

³<https://github.com/pytorch/examples/tree/master/imagenet>

⁴<https://github.com/mi-lad/snip>

⁵<https://github.com/alecwangcq/GraSP>

Table A.1: Percentage of weights per layer for each network and dataset.

Layer type	Conv	Fully connected	BatchNorm	Bias	Prunable	Total
CIFAR10						
Resnet50	99.69	0.09	0.11	0.11	99.78	23.52M
VGG19	99.92	0.03	0.03	0.03	99.95	20.04M
CIFAR100						
Resnet50	98.91	0.86	0.11	0.11	99.78	23.71M
VGG19	99.69	0.25	0.03	0.03	99.94	20.08M
TinyImagenet						
Resnet50	98.06	1.71	0.11	0.11	99.78	23.91M
VGG19	99.44	0.51	0.03	0.03	99.94	20.13M
Imagenet						
Resnet50	91.77	8.01	0.10	0.11	99.79	25.56M
VGG19	13.93	86.05	0.01	0.01	99.98	143.68M

of our knowledge. In Table A.1 we show the percentage of prunable weights for each network and dataset we use. In future experiments we will explore the performance of pruning at initialization when including batch norm layers and biases as well.

A.2 Additional accuracy-sparsity plots

In the main text we show the complete range of the accuracy-sparsity curves for the different methods so it is clear why more robust methods are needed. However, it makes it more difficult to appreciate the smaller differences at lower sparsities. In Fig A.1 we show the accuracy-sparsity curves where we cut the y axis to show only the higher accuracies.

A.3 Further analysis of pruning at initialization

A.3.1 Saliency vs T

In Fig 3.4 (left) we have seen that for higher sparsity levels, FORCE obtains a higher saliency when we increase the number of iterations. In Fig A.2 we compare the relative saliencies as we increase the number of iterations for FORCE and Iterative SNIP. As can be seen, both have a similar behaviour.

A. Appendix of Progressive Skeletonization: Trimming more fat from a network at initialization

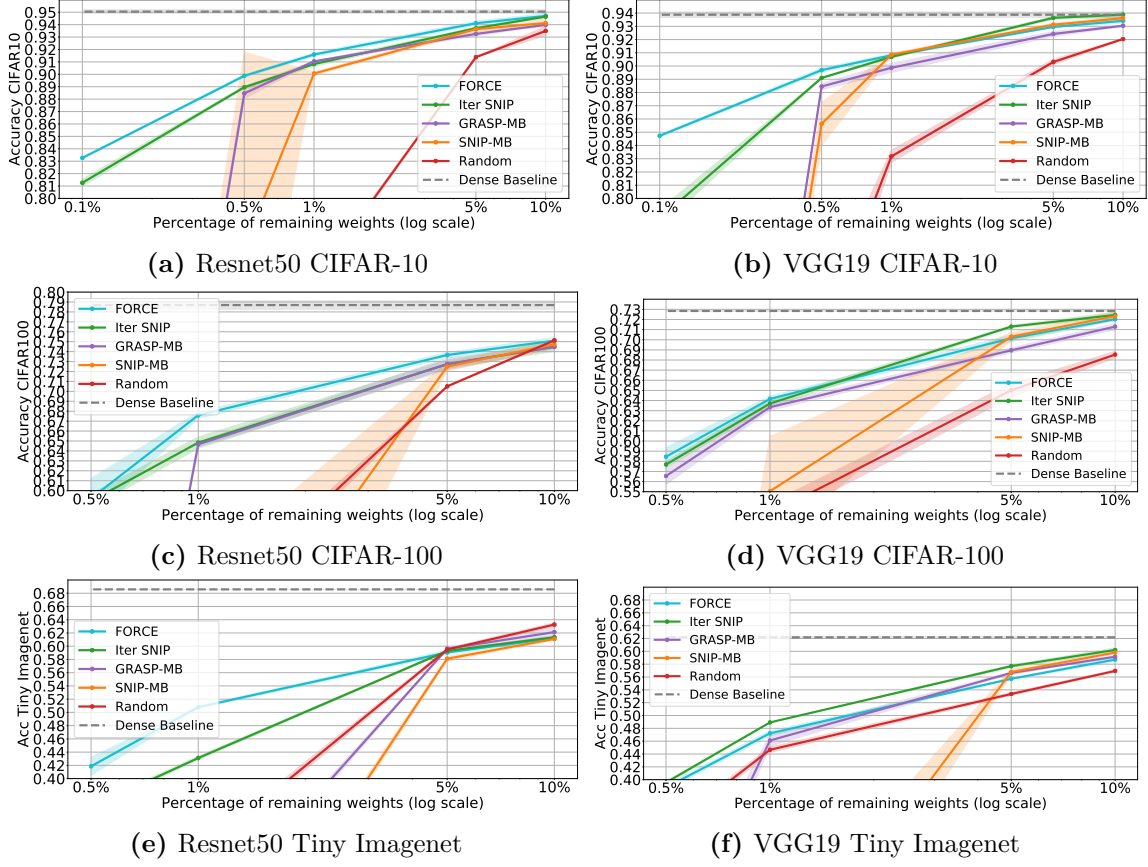


Figure A.1: Test accuracies on CIFAR-10/100 and Tiny Imagenet for different pruning methods. Each point is the average over 3 runs of prune-train-test. The shaded areas denote the standard deviation of the runs (too small to be visible in some cases).

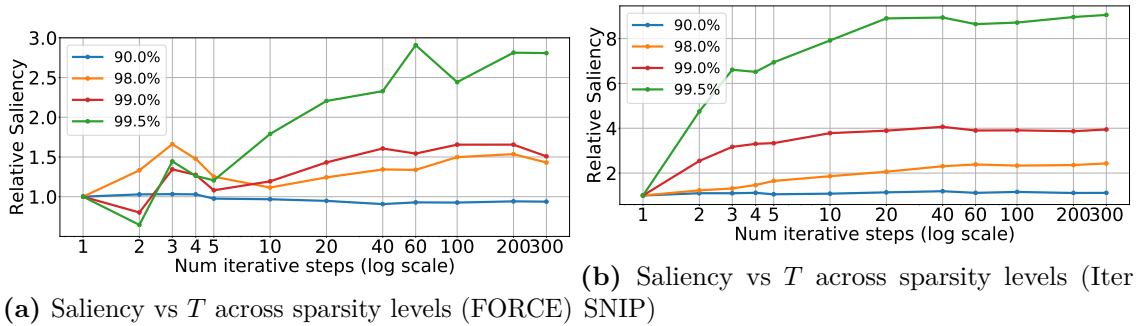


Figure A.2: FORCE saliency (3.6) obtained with iterative pruning normalized by the saliency obtained with one-shot SNIP, $T = 1$. (a) Applying the FORCE algorithm (b) Using Iterative SNIP. Note how both methods have similar behaviour.

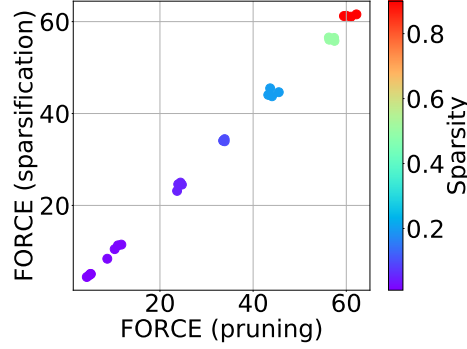


Figure A.3: FORCE saliency computed for masks as we vary sparsity. FORCE (sparsification) refers to measuring FORCE when we allow the gradients of *zeroed* connections to be non-zero, while FORCE (pruning) cuts all backward signal of any *removed* connection. As can be seen on the plot, they are strongly correlated.

A.3.2 Pruning vs Sparsification

FORCE algorithm is able to recover pruned weights in later iterations of pruning. In order to do that, we do not consider the intermediate masks as *pruning masks* but rather as *sparsification masks*, where connections are set to 0 but not their gradients. In order to understand how does computing the FORCE (3.5) on a sparsified vs pruned network affect the saliency, we prune several masks with FORCE algorithm at varying sparsity levels. For each mask, we then compute their FORCE saliency either considering the pruned network (gradients of pruned connections will be set to 0 during the backward pass) or the sparsified network (we only set to 0 the connections, but let the gradient signal flow through all the connections). Results are presented in Fig A.3. We observe that the two methods to compute the saliency are strongly correlated, thus, we can assume that when we use the FORCE algorithm that maximizes the saliency of sparsified networks we will also maximize the saliency of the corresponding pruned networks.

A.3.3 Network structure after pruning

In Fig A.4 we visualize the structure of the networks after pruning 99.9% of the parameters. We show the fraction of remaining weights and the total number of

A. Appendix of Progressive Skeletonization: Trimming more fat from a network at initialization

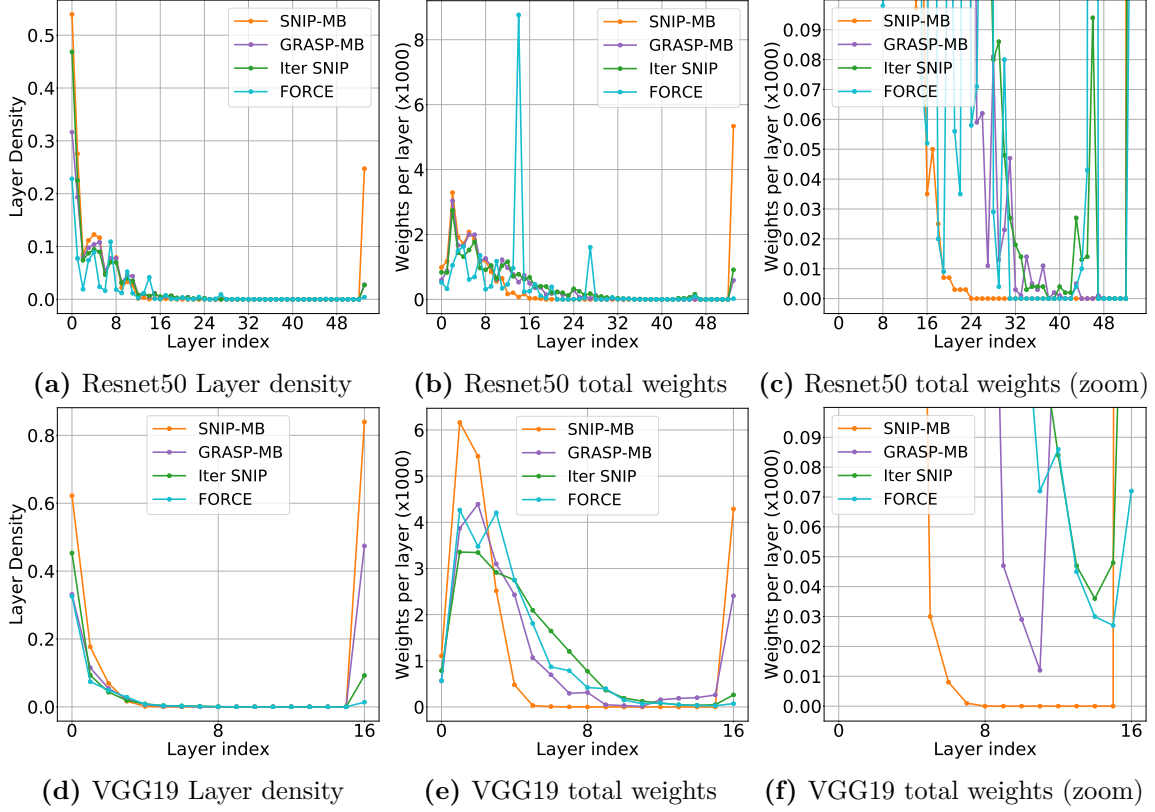


Figure A.4: Visualization of remaining weights after pruning 99.9% of the weights of Resnet-50 and VGG-19 for CIFAR-10. (a) and (d) show the fraction of remaining weights for each prunable layer. (b) and (e) show the actual number of remaining weights and (c) and (f) zoom to the bottom part of the plot. Observe in (c) and (f) that some layers have exactly 0 weights left, so they are removed entirely.

remaining weights per layer after pruning. As seen in (a) and (d), all analysed methods show a tendency to preserve the initial and final layers and to prune more heavily the deep convolutional layers, this is consistent with results reported in Wang et al. (2020a). In (b) and (e), we note that FORCE has a structure that stands out compared to other methods that are more similar. This is reasonable since, it is the only method that allows pruned weights to recover. In the zoomed plots (c) and (f) we would like to point out that FORCE and Iterative SNIP preserve more weights on the deeper layers than GRASP and SNIP for VGG19 while we observe the opposite behaviour for Resnet50.

In Fig 3.2, we observe that gradual pruning allows to prune the Resnet50 network up to 99.99% sparsity without falling to random accuracy. In contrast, with VGG19 we

observe Iterative SNIP is not able to prune more than 99.9%. In Fig A.4 we observe that for Resnet50, all methods prune some layers completely.

However, in the case of ResNets, even if a convolutional layer is entirely pruned, skip connections still allow the flow of forward and backward signal. On the other hand, architectures without skip connections, such as VGG, require non-empty layers to keep the flow of information. Interestingly, in (c) we observe how FORCE and Iter SNIP have a larger amount of completely pruned layers than GRASP, however, there are a few deep layers with a significantly larger amount of unpruned weights. This seems to indicate that when a high sparsity is required, it is more efficient to have fewer layers with more weights than several extremely sparse layers.

A.3.4 Evolution of pruning masks

As discussed in the main text, FORCE allows weights that have been pruned at earlier iterations to become non-zero again, we argue this might be beneficial compared to Iterative SNIP which will not be able to correct any possible "mistakes" made in earlier iterations. In particular, it seems to give certain advantage to prune VGG to high sparsities without breaking the flow of information (pruning a layer entirely) as can be seen in Fig 3.2 (b). To gain a better intuition of how does the amount of pruned/recovered weights ratio evolve during pruning, in Fig A.5 we plot the normalized amount of pruned and recovered weights (globally on the whole network) at each iteration of FORCE and also for Iter SNIP as a sanity check. Note that Iterative SNIP does not recover weights and the amount of weights pruned at each step decays exponentially (this is expected since we derived Iterative SNIP as a constrained optimization of FORCE where each network needs to be a sub-network of the previous iteration. On the other hand, FORCE does recover weights. Moreover, the amount of pruned/recovered weights does not decay monotonically but has a clear peak, indicating there are two phases during pruning: While the amount of pruned weights increases, the algorithm explores masks which are quite far away from each other, although this might be harmful for the *gradient approximation* (refer to section 3.4), we argue that

A. Appendix of Progressive Skeletonization: Trimming more fat from a network at initialization

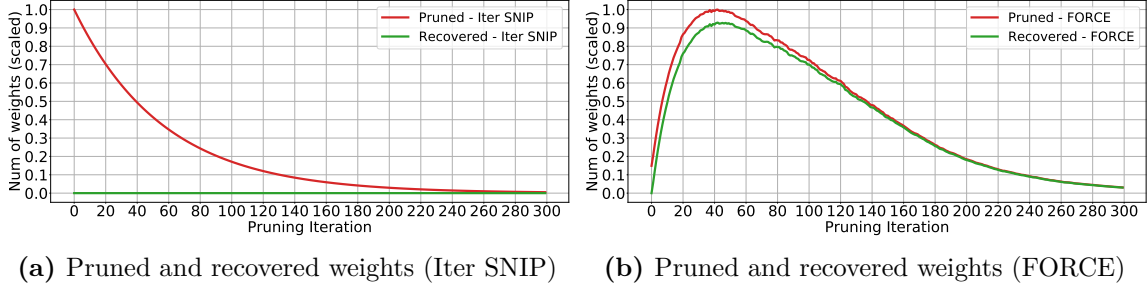


Figure A.5: Normalized amount of globally pruned and recovered weights at each pruning iteration for Resnet50, CIFAR-10 when pruned with (a) Iterative SNIP and (b) FORCE. As expected, the amount of recovered weights for Iterative SNIP is constantly zero, since this is by design. Moreover, the amount of pruned weights decays exponentially as expected from our pruning schedule. On the other hand, we see the amount of recovered weights is non-zero with FORCE, interestingly the amount of pruned/recovered weights does not decay monotonically but has a clear peak, indicating there is an "exploration" and a "convergence" phase during pruning.

during the initial pruning iterations the network is still quite over-parametrized. After reaching a peak, both the pruning and recovery rapidly decay, thus the masks converge to a more constrained subset.

A.3.5 Comparison with early pruning

For fair comparison, we provided the same amount of data to SNIP and GRASP as was used by our approach and call this variant SNIP-MB and GRASP-MB. Similarly, under this new baseline which we call *early pruning*, we train the network on 1 epoch of

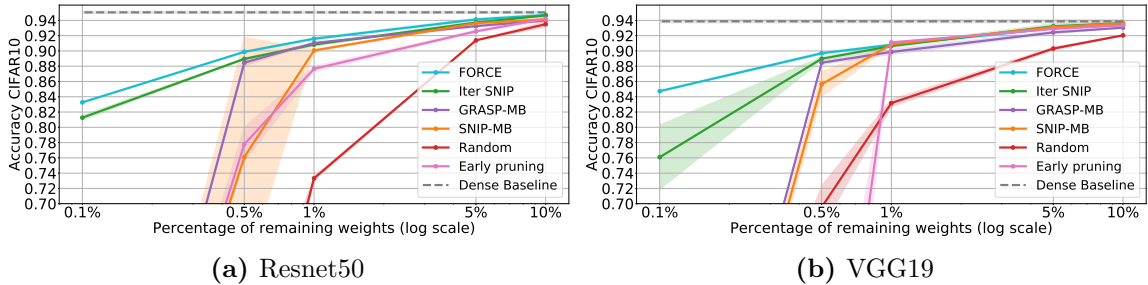


Figure A.6: Test accuracies on CIFAR-10 for different pruning methods. Each point is the average over 3 runs of prune-train-test. The shaded areas denote the standard deviation of the runs (too small to be visible in some cases). Early pruning is at most on par with gradual pruning at initialization methods and has a strong drop in performance as we go to higher sparsities.

data of CIFAR-10, which uses slightly more samples than our pruning at initialization methods (see section 3.5). After training for 1 epoch we perform magnitude pruning which requires no extra cost (results presented in Fig A.6). Although early pruning yields competitive results for VGG at moderate sparsity levels, it soon degrades its performance as we prune more weights. On the other hand, for Resnet architecture it is sub-optimal at all evaluated sparsity levels. Note, this result does not mean that any early pruning strategy would be sub-optimal compared to pruning at initialization, however exploring this further is out of the scope of this work.

A.3.6 Mobilenet experiments

All our experiments were on overparameterized architectures such as Resnet and VGG. To test the wider usability of our methods, in this section we prune the Mobilenet-v2 architecture⁶ (Sandler et al., 2018) which is much more "slim" than Resnet and VGG (Mobilenet has 2.3M params compared to 20.03M and 23.5M of VGG and Resnet respectively). Results are provided in Fig A.7. Similarly to Resnet and VGG architectures, we see that gradual pruning tends to better preserve accuracy at higher sparsity levels than one-shot methods. Moreover, both FORCE and Iter SNIP improve over SNIP at moderate sparsity levels as well. FORCE and Iter SNIP have comparable accuracies except for high sparsity where Iter SNIP surpasses FORCE. We hypothesize for such a slim architecture ($\approx 10 \times$ fewer parameters than Resnet and VGG) the gradient approximation becomes even more sensitive to the distance between iterative masks and perhaps the exploration of FORCE is harmful in this case. As discussed in the main paper, we believe that further research to understand the exploration/exploitation trade-off when pruning might yield to even more efficient pruning schemes, especially for very high sparsity levels. We train using the same settings as described in Appendix A.1 except for the weight decay which is set to 4×10^{-5} , following the settings of the original Mobilenet paper.

⁶<https://github.com/kuangliu/pytorch-cifar/blob/master/models/mobilenetv2.py>

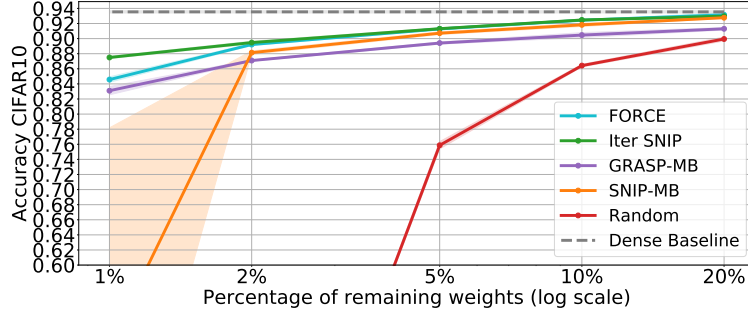


Figure A.7: Test accuracies on CIFAR-10 for different pruning methods on Mobilenet architecture. Each point is the average over 3 runs of prune-train-test. The shaded areas denote the standard deviation of the runs (too small to be visible in some cases). When pruning Mobilenet-v2 architecture, which has roughly $10\times$ less parameters than Resnet or VGG, we observe a similar pattern: Gradual pruning tends to be able to preserve accuracy for higher sparsities better than one-shot methods. Moreover, it tends to improve slightly over SNIP at moderate sparsities as well. Although GRASP does not show an acute drop in performance like SNIP, it seems to be sub-optimal with respect to other methods across all sparsities.

A.4 Local optimal masks

Definition 1 ((p, ϵ) -local optimal mask). Consider any two sets⁷ $\mathbf{c}_t \subseteq \{1, \dots, m\}$ and $\mathbf{c}_{t+1} \subset \mathbf{c}_t$. For any $\epsilon > 0$ and $0 \leq p \leq |\mathbf{c}_t \setminus \mathbf{c}_{t+1}|$, $\mathbf{c}_{t+1}$ is a (p, ϵ) local optimal with respect to $\mathbf{c}_t$ if the following holds

$$S(\theta, \mathbf{c}_{t+1}) \geq S(\theta, (\mathbf{c}_{t+1} \setminus S_-) \cup S_+) - \epsilon \quad (\text{A.1})$$

for all $S_- \subset \mathbf{c}_{t+1}, |S_-| = p$ and $S_+ \subset (\mathbf{c}_t \setminus \mathbf{c}_{t+1}), |S_+| = p$.

Definition 2 (CRS; Coordinate-Restricted-Smoothness). Given a function $\mathcal{L} : \mathbb{R}^m \rightarrow \mathbb{R}$ (which encodes both the network architecture and the dataset), $\mathcal{L}$ is said to be λ_c -Coordinated Restricted Smooth with respect to $\mathbf{c} \subseteq \{1, \dots, m\}$ if there exists a real number λ_c such that

$$\|\mathbf{c} \odot \nabla \mathcal{L}(\mathbf{w} \odot \mathbf{c}) - \mathbf{c} \odot \nabla \mathcal{L}(\mathbf{w} \odot \hat{\mathbf{c}})\|_\infty \leq \lambda_c \|\mathbf{w} \odot \mathbf{c} - \mathbf{w} \odot \hat{\mathbf{c}}\|_1 \quad (\text{A.2})$$

for all $\mathbf{w} \in \mathbb{R}^m$ and $\hat{\mathbf{c}} \subset \mathbf{c}$. When $s = |\mathbf{c} \setminus \hat{\mathbf{c}}|$, an application of Holder's inequality shows

$$\lambda_c \|\mathbf{w} \odot \mathbf{c} - \mathbf{w} \odot \hat{\mathbf{c}}\|_1 \leq \lambda_c \|\mathbf{w}\|_\infty \|\mathbf{c} - \hat{\mathbf{c}}\|_1 = \lambda_c s \|\mathbf{w}\|_\infty$$

⁷For ease of notation, we will use this representation interchangeably with its binary encoding i.e. a m -dimensional binary vector with its support equal to $\mathbf{c}$

We define $\mathcal{L}$ to be Λ -total CRS if there exists a function $\Lambda : \{0, 1\}^m \rightarrow \mathbb{R}$ such that for all $\mathbf{c} \in \{0, 1\}^m$ $\mathcal{L}$ is $\Lambda(\mathbf{c})$ -Coordinate-Restricted-Smooth with respect to $\mathbf{c}$ (for ease of notation we use $\Lambda(\mathbf{c}) = \lambda_{\mathbf{c}}$).

Theorem 1 (Informal). The mask $\mathbf{c}_{t+1}$ produced from $\mathbf{c}_t$ by FORCE is $(p, 2\lambda p \|\theta\|_\infty^2 |\mathbf{c}_t|)$ -local optimal if the $\mathcal{L}$ is Λ -CRS..

Proof. Consider the masks $\mathbf{c}_t$ and $\mathbf{c}_{t+1}$ where the latter is obtained by one step of FORCE on the former. Let S_- and S_+ be any set of size p such that $S_- \subset \mathbf{c}_{t+1}$ and $S_+ \subset (\mathbf{c}_t \setminus \mathbf{c}_{t+1})$. Finally, for ease of notation we define $\zeta = (\mathbf{c}_{t+1} \setminus S_-) \cup S_+$

$$\begin{aligned}
 S(\theta, \zeta) - S(\theta, \mathbf{c}_{t+1}) &= \sum_{i \in \zeta} |\theta_i \cdot \nabla \mathcal{L}(\theta \odot \zeta)_i| - \sum_{i \in \mathbf{c}_{t+1}} |\theta_i \cdot \nabla \mathcal{L}(\theta \odot \mathbf{c}_{t+1})_i| \\
 &= \underbrace{\sum_{i \in \mathbf{c}_{t+1}} |\theta_i \cdot \nabla \mathcal{L}(\theta \odot \zeta)_i| - \sum_{i \in \mathbf{c}_{t+1}} |\theta_i \cdot \nabla \mathcal{L}(\theta \odot \mathbf{c}_{t+1})_i|}_{\Gamma_1} \\
 &\quad + \underbrace{\sum_{i \in S_+} |\theta_i \cdot \nabla \mathcal{L}(\theta \odot \zeta)_i|}_{\Gamma_2} - \underbrace{\sum_{i \in S_-} |\theta_i \cdot \nabla \mathcal{L}(\theta \odot \zeta)_i|}_{\Gamma_3} \quad (\text{A.3})
 \end{aligned}$$

Let us look at the three terms individually. We assume that $\mathcal{L}$ is Λ -CRS.

$$\begin{aligned}
 \Gamma_1 &= \sum_{i \in \mathbf{c}_{t+1}} |\theta_i \cdot \nabla \mathcal{L}(\theta \odot \zeta)_i| - |\theta_i \cdot \nabla \mathcal{L}(\theta \odot \mathbf{c}_{t+1})_i| \\
 &\leq \|\mathbf{c}_{t+1} \odot \theta \odot \nabla \mathcal{L}(\theta \odot \mathbf{c}_{t+1}) - \mathbf{c}_{t+1} \odot \theta \odot \nabla \mathcal{L}(\theta \odot \zeta)\|_1 \quad \text{Triangle Ineq.} \\
 &\leq \|\mathbf{c}_{t+1} \odot \theta\|_1 \|\mathbf{c}_{t+1} \odot \nabla \mathcal{L}(\theta \odot \mathbf{c}_{t+1}) - \mathbf{c}_{t+1} \odot \nabla \mathcal{L}(\theta \odot \zeta)\|_\infty \quad \text{Holder's Ineq} \\
 &\leq 2\lambda_{\mathbf{c}_{t+1}} |\mathbf{c}_{t+1}| p \|\theta\|_\infty^2 \quad \because \mathcal{L} \text{ is } \Lambda\text{-CRS} \quad (\text{A.4})
 \end{aligned}$$

$$\begin{aligned}
 \Gamma_2 &= \sum_{i \in S_+} |\theta_i \cdot \nabla \mathcal{L}(\theta \odot \zeta)_i| - |\theta_i \cdot \nabla \mathcal{L}(\theta \odot \mathbf{c}_t)_i| + |\theta_i \cdot \nabla \mathcal{L}(\theta \odot \mathbf{c}_t)_i| \\
 &\leq \sum_{i \in S_+} |\theta_i \cdot \nabla \mathcal{L}(\theta \odot \mathbf{c}_t)_i| + \lambda_{\mathbf{c}_t} p \|\theta\|_\infty^2 (|\mathbf{c}_t| - |\mathbf{c}_{t+1}|) \quad \because |\mathbf{c}_t \setminus \zeta| = |\mathbf{c}_t| - |\mathbf{c}_{t+1}|, \\
 &\quad (\text{A.5})
 \end{aligned}$$

$$\begin{aligned}
 \Gamma_3 &= - \sum_{i \in S_-} |\theta_i \cdot \nabla \mathcal{L}(\theta \odot \zeta)_i| + |\theta_i \cdot \nabla \mathcal{L}(\theta \odot \mathbf{c}_t)_i| - |\theta_i \cdot \nabla \mathcal{L}(\theta \odot \mathbf{c}_t)_i| \\
 &\leq - \sum_{i \in S_-} |\theta_i \cdot \nabla \mathcal{L}(\theta \odot \mathbf{c}_t)_i| + \lambda_{\mathbf{c}_t} p \|\theta\|_\infty^2 (|\mathbf{c}_t| - |\mathbf{c}_{t+1}|), \quad (\text{A.6})
 \end{aligned}$$

Adding eqs. (A.5) and (A.6), we get

$$\begin{aligned}\Gamma_2 + \Gamma_3 &\leq \sum_{i \in S_+} |\theta_i \cdot \nabla \mathcal{L}(\theta \odot \mathbf{c}_t)_i| - \sum_{i \in S_-} |\theta_i \cdot \nabla \mathcal{L}(\theta \odot \mathbf{c}_t)_i| + 2\lambda_{\mathbf{c}_t} p \|\theta\|_\infty^2 (|\mathbf{c}_t| - |\mathbf{c}_{t+1}|) \\ &\leq 2\lambda_{\mathbf{c}_t} p \|\theta\|_\infty^2 (|\mathbf{c}_t| - |\mathbf{c}_{t+1}|) - \gamma p\end{aligned}\tag{A.7}$$

where $\gamma = \min_{i \in \mathbf{c}_{t+1}, j \in (\mathbf{c}_t \setminus \mathbf{c}_{t+1})} |\theta_i \nabla \mathcal{L}(\theta \odot \mathbf{c}_t)_i| - |\theta_j \nabla \mathcal{L}(\theta \odot \mathbf{c}_t)_j| \geq 0$

Substituting eqs. (A.4) and (A.7) into (A.3) we get

$$\begin{aligned}S(\theta, \zeta) - S(\theta, \mathbf{c}_{t+1}) &\leq 2\lambda_{\mathbf{c}_{t+1}} |\mathbf{c}_{t+1}| p \|\theta\|_\infty^2 + 2\lambda_{\mathbf{c}_t} p \|\theta\|_\infty^2 (|\mathbf{c}_t| - |\mathbf{c}_{t+1}|) - \gamma p \\ &= 2\lambda_{\mathbf{c}_{t+1}} p \|\theta\|_\infty^2 (|\mathbf{c}_{t+1}| + (|\mathbf{c}_t| - |\mathbf{c}_{t+1}|)) - \gamma p \\ S(\theta, \mathbf{c}_{t+1}) &\geq S(\theta, \zeta) - 2\lambda_{\mathbf{c}_{t+1}} p \|\theta\|_\infty^2 |\mathbf{c}_t| + \gamma p \\ S(\theta, \mathbf{c}_{t+1}) &\geq S(\theta, \zeta) - 2\lambda_{\mathbf{c}_{t+1}} p \|\theta\|_\infty^2 |\mathbf{c}_t|\end{aligned}$$

□

A.5 Iterative pruning to maximize the gradient norm

A.5.1 Maximizing the gradient norm using the gradient approximation

In order to maximize the gradient norm after pruning, the authors in Wang et al. (2020a) use the first order Taylor's approximation. While this seems to be better suited than SNIP for higher levels of sparsity, it assumes that pruning is a small perturbation on the weight matrix. We argue that this approximation will not be valid as we push towards extreme sparsity values. Our *gradient approximation* (refer to section 3.4) can also be applied to maximize the gradient norm after pruning. In this case, we have

$$G(\theta, \mathbf{c}) := \Delta \mathcal{L}(\theta \odot \mathbf{c}) - \Delta \mathcal{L}(\theta) \approx \sum_{\{i: c_i=0\}} -[\nabla \mathcal{L}(\theta)_i]^2, \tag{A.8}$$

where we assume pruned connections have null gradients (this is equivalent to the restriction used for Iterative SNIP) and we assume gradients remain unchanged for unpruned weights (gradient approximation). Combining this approximation with Eq. (3.7), we obtain a new pruning method we name Iterative GRASP, *although it is not the same as applying GRASP iteratively*. Unlike FORCE, Iterative GRASP does not recover GRASP when $T = 1$.

In Fig A.8 we compare Iterative GRASP to other pruning methods. We use the same settings as described in section 3.5. We observe that Iterative GRASP outperforms GRASP in the high sparsity region. Moreover, for VGG19 architecture Iterative GRASP achieves comparable performance to Iterative SNIP. Nevertheless, for Resnet50 we see that Iterative GRASP performance falls below that of FORCE and Iterative SNIP as we prune more weights. FORCE saliency takes into account both the gradient and the magnitude of the weights when computing the saliency, on the other hand, the Gradient Norm only takes gradients into account, therefore it is using less information. We hypothesize this might be the reason why Iterative GRASP does not match Iterative SNIP.

A.5.2 Iterative GRASP (Pruning consistency)

As explained in the main text, we tried to apply GRASP iteratively with the Taylor approximation described in Wang et al. (2020a). Unfortunately, we found that all resulting masks yield networks unable to train. In light of this result, we performed some analysis of the behaviour of GRASP compared to that of SNIP and, in the following, we provide some insights as to why we can not use GRASP’s approximation iteratively.

In Liu et al. (2018), the authors show that applying a pruning method to the same architecture with different random initializations would yield consistent pruning masks. Specifically, they find that the percentage of pruned weights in each layer had very low variance. We reproduce the same experiment and additionally explore another

dimension, the (global) sparsity level. Given an architecture, we prune it at varying levels of sparsity and extract the percentage of remaining weights at each layer. For each level of sparsity, we average the results over 3 trials of initialize-prune. As shown in Fig 3.2, both SNIP and GRASP have very low variance across initializations, on the other hand, as we vary the global sparsity with GRASP, the percentage of remaining weights for each layer is inconsistent. The layers that are most preserved at high levels of sparsity, such as the initial and last layers, are the most heavily pruned at low sparsity levels.

The authors in Liu et al. (2018), reason that manually designed networks have layers which are more redundant than others. Therefore, pruning methods even this redundancies by pruning layers with different percentages. We extend this reasoning, and hypothesize that pruning algorithms should always have preference for pruning the same (redundant) layers across all levels of sparsity. We denote this as *pruning consistency*. We observe that when applying iterative pruning to GRASP, the resulting masks tend to prune almost all of the weights at the initial and final layers, producing networks that are unable to converge. When using iterative pruning, we prune a small portion of remaining weights at each step. Thus, we are always in the low sparsity regime, where the GRASP behaviour is reversed. Conversely, when we use SNIP the behaviour changes completely. In this case, the preserved layers are consistent across sparsity levels, and when we use iterative pruning we obtain networks that reach high accuracy values.

A. Appendix of Progressive Skeletonization: Trimming more fat from a network at initialization

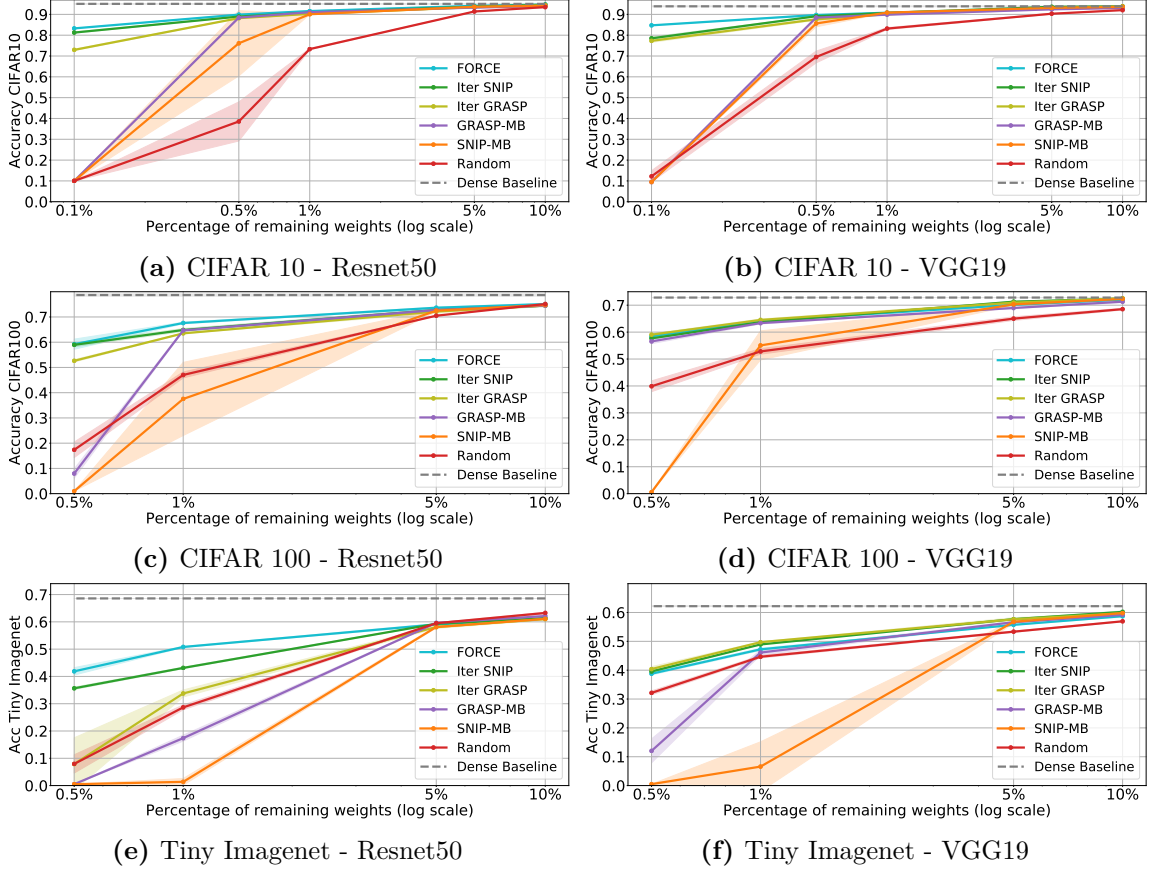


Figure A.8: Test accuracies for different datasets and networks when pruned with different methods. Each point is the average over 3 runs of prune-train-test. The shaded areas denote the standard deviation of the runs (sometimes too small to be visible).

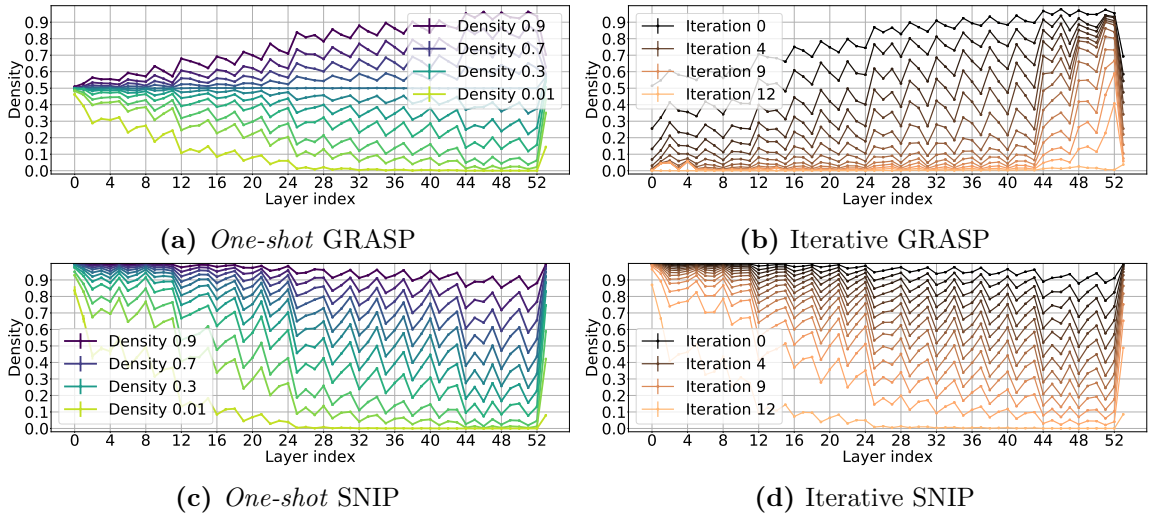


Figure A.9: (a) - (c) Portion of remaining weights for each layer. Each point is an average of 3 runs and the error bars (hardly visible) denote standard deviation. (b) - (d) Portion of remaining weights for each intermediate mask that is computed during iterative pruning with global portion of remaining weights of 0.01.

B

Appendix of Make some noise: Reliable and efficient single-step adversarial training

B.1 Additional plots for PreActResNet18 experiments

In the main paper we compare N-FGSM with other single-step methods and multi-step methods separately and remove clean accuracies for better visualization. In this section we present the curves for all methods with both the clean and robust accuracy. The tendency in the three datasets is for N-FGSM PGD-50-10 accuracy to be slightly above that of GradAlign, while the opposite happens to the clean accuracy. We also observe that clean accuracy becomes significantly more noisy when CO happens. Exact numbers for all the curves are in Appendix B.19.

B.2 Experiments with WideResNet28-10 architecture

In this section we present the plots of our experiments with WideResNet28-10. We report the results in two figures. In Figure B.2 we compare all single-step methods

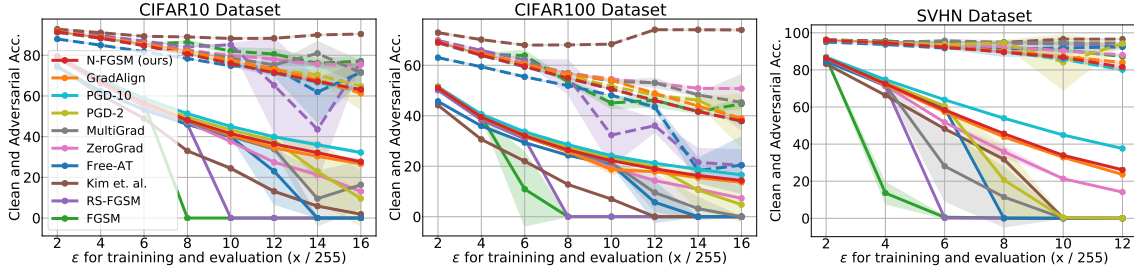


Figure B.1: Comparison of all methods on CIFAR-10, CIFAR-100 and SVHN with PreActResNet18 over different perturbation radius (ϵ is divided by 255). We plot both the robust (solid line) and the clean (dashed line) accuracy for each method. Our method, N-FGSM, is able to match or surpass the state-of-the-art single-step method GradAlign while *reducing the cost by a $3\times$ factor*. Adversarial accuracy is based on PGD-50-10 and experiments are averaged over 3 seeds. Legend is shared among all plots.

and we do not plot the clean accuracy for better visualization. In Figure B.3 we plot all methods, including multi-step methods, and report the clean accuracy as well with dashed lines. Since we observed that our baseline, RandAlpha, outperformed Kim et al. (2021) in all settings for PreActResNet18, we only report RandAlpha for WideResNet. As mentioned in the main paper, we observe that CO seems to be more difficult to prevent for WideResNet. In particular, for GradAlign we observed the regularizer hyperparameter settings proposed by Andriushchenko and Flammarion (2020) for CIFAR-10 (searched for a PreActResNet18) worked well. However, those parameters led to CO for $6 \leq \epsilon \leq 12$ in CIFAR-100. Since $\epsilon = 14, 16$ did not show CO, we increased the GradAlign regularizer hyperparameter λ for CIFAR-100 so that each $6 \leq \epsilon \leq 12$ would have the default value corresponding to $\epsilon + 2$, for instance, λ for $\epsilon = 6$ would be the default λ in Andriushchenko and Flammarion (2020) for $\epsilon = 8$.

For SVHN we observed that the default values for λ led to models close to a constant classifier for $\epsilon \geq 6$. We tried to increase the lambda for those ϵ values to 1.25λ but observed the same result. Since the model did not show typical CO but rather it seemed as it was underfitting, we tried to reduce the step-size to $\alpha = 0.75\epsilon$ and also both decreasing α and increasing λ . When reducing the step size we obtain accuracies above those of a constant classifier for some radii, however, some or all seeds converge to a constant classifier for each setting, hence the large standard deviations. For

N-FGSM, the default configuration of N-FGSM ($\alpha = \epsilon$, $k = 2\epsilon$) works well in all settings except for $\epsilon = 16$ on CIFAR-10 and $\epsilon = 10, 12$ on SVHN. For CIFAR-10, we increase the noise magnitude to $k = 4\epsilon$. For SVHN we find that decreasing α as we tried for GradAlign works better than increasing the noise. We use $\alpha = 8$ for both ϵ radii. Exact numbers for all the curves are in Appendix B.19

B.3 Increasing adversarial perturbations during training

As mentioned in the main paper, N-FGSM perturbations have ℓ_∞ -norm larger than ϵ , see Appendix B.14. In Section 4.6 we have seen that the benefits of N-FGSM can not be reproduced by simply increasing α without increasing the noise. However, for the sake of completeness, we also ablate other single-step baselines by using a larger ϵ during training *i.e.* $\{\epsilon = 8/255, \epsilon = 12/255, \epsilon = 16/255\}$ while testing with a fixed $\epsilon = 8/255$ on CIFAR-10. Results are presented in Table B.1. We observe that increasing ϵ_{train} seems to lead to a decrease in robustness for most methods, *e.g.* PGD-50-10 accuracy for RS-FGSM goes from 46.08 ± 0.18 when training with $\epsilon = 8/255$ to 0.0 ± 0.0 with $\epsilon = 12/255$. In two cases (GradAlign and MultiGrad) we observe a small increase, highest increase is for GradAlign which goes from 48.14 ± 0.15 to 50.6 ± 0.45 , however, the clean accuracy drops from 81.9 ± 0.22 to 73.29 ± 0.23 .

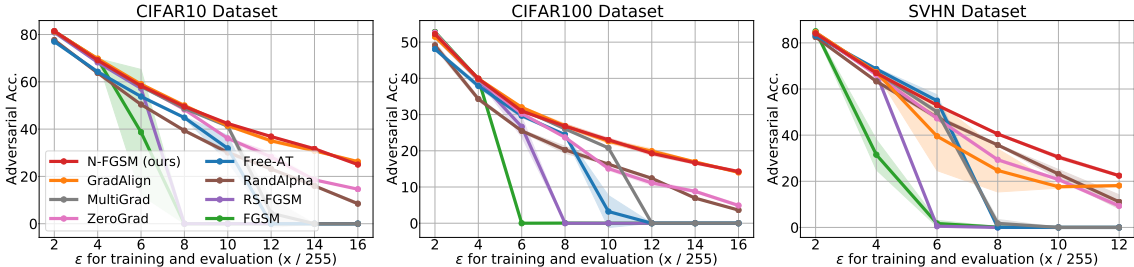


Figure B.2: Comparison of single-step methods on CIFAR-10, CIFAR-100 and SVHN with WideResNet28-10 over different perturbation radius (ϵ is divided by 255). Our method, N-FGSM, is able to match or surpass the state-of-the-art single-step method GradAlign while *reducing the cost by a $3\times$ factor*. Moreover, we could not find any competitive hyperparameter setting for GradAlign for $\epsilon \geq 6$ in SVHN dataset. Adversarial accuracy is based on PGD-50-10 and experiments are averaged over 3 seeds. Legend is shared among all plots.

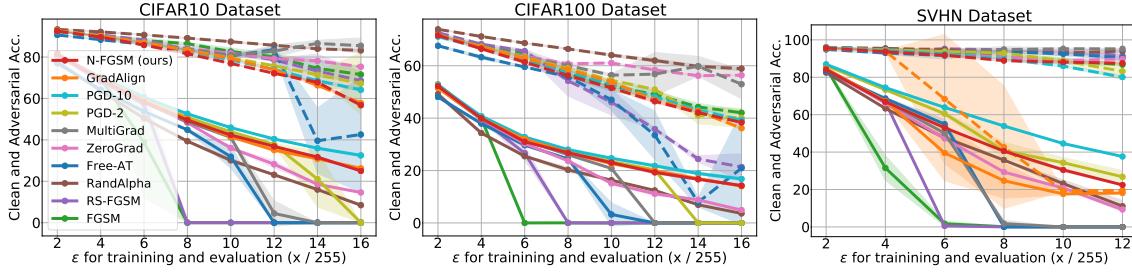


Figure B.3: Comparison of all methods on CIFAR-10, CIFAR-100 and SVHN with WideResNet28-10 over different perturbation radius (ϵ is divided by 255). We plot both the robust (solid line) and the clean (dashed line) accuracy for each method. Legend is shared among all plots.

This is similar to increasing α for N-FGSM (see Figure 4.6 (C)). However, this is tied to a significant degradation of clean accuracy. All in all, taking into account both clean and robust accuracy we conclude all baselines perform best without increasing the training ϵ . All ablation results are presented in Table B.1.

Table B.1: Ablation of the PGD-50-10 accuracy for single-step methods when increasing the ϵ_{train} . All models are evaluated with PGD-50-10 attack and $\epsilon_{\text{test}} = 8/255$. Note that considering the trade-off between clean and robust accuracy, all methods perform best when training with the same epsilon to be applied at test time.

Method	$\epsilon_{\text{train}} = 1\epsilon_{\text{test}}$		$\epsilon_{\text{train}} = 1.5\epsilon_{\text{test}}$		$\epsilon_{\text{train}} = 2\epsilon_{\text{test}}$		Rel. Cost
	Clean acc.	PGD acc.	Clean acc.	PGD acc.	Clean acc.	PGD acc.	
GradAlign	81.9 ± 0.22	48.14 ± 0.15	73.29 ± 0.23	50.6 ± 0.45	61.3 ± 0.15	46.67 ± 0.29	3
MultiGrad	82.33 ± 0.14	47.29 ± 0.07	75.28 ± 0.2	50.0 ± 0.79	71.42 ± 5.63	0.0 ± 0.0	2
AT Free	78.41 ± 0.18	46.03 ± 0.36	73.91 ± 4.19	32.4 ± 22.91	71.64 ± 3.89	0.0 ± 0.0	1.6
Kim <i>et al.</i>	89.02 ± 0.1	33.01 ± 0.09	88.35 ± 0.31	27.36 ± 0.31	90.45 ± 0.08	9.28 ± 0.12	1.5
FGSM	86.41 ± 0.7	0.0 ± 0.0	80.6 ± 2.59	0.0 ± 0.0	77.14 ± 2.46	0.0 ± 0.0	1
RS-FGSM	84.05 ± 0.13	46.08 ± 0.18	65.22 ± 23.23	0.0 ± 0.0	76.66 ± 0.38	0.0 ± 0.0	1
ZeroGrad	82.62 ± 0.05	47.08 ± 0.1	78.11 ± 0.2	46.43 ± 0.37	75.42 ± 0.13	45.63 ± 0.39	1
N-FGSM	80.58 ± 0.22	48.12 ± 0.07	71.46 ± 0.14	50.23 ± 0.31	63.18 ± 0.49	46.46 ± 0.1	1

B.4 Longer training schedule

In our experiments, we have followed the “fast” training schedule introduced by Wong et al. (2020). However, Rice et al. (2020) suggest that a longer training schedule coupled with early stopping may lead to a boost in performance. We also use the long training schedule for N-FGSM and observe that it does not lead to CO. In Table B.2

we compare the performance of N-FGSM and GradAlign for the long training schedule. We observe that GradAlign does not seem to benefit from the long training schedule. On the other hand, although N-FGSM seems to obtain a slight increase in performance, the “fast” schedule provides comparable performance. It is worth mentioning that for GradAlign, the default regularizer hyperparameter for $\epsilon = 8/255$ and CIFAR-10 ($\lambda = 0.2$) does not prevent CO. We do a hyperparameter search and keep the value with the largest final robust accuracy ($\lambda = 0.632$).

Table B.2: Comparison of “long” (Rice et al., 2020) and “fast” (Wong et al., 2020) training schedules for N-FGSM and GradAlign. GradAlign does not seem to benefit from the long training schedule. Although N-FGSM seems to obtain a slight increase in performance, the “fast” schedule provides comparable performance.

N-FGSM		Grad Align	
Clean Acc	Robust Acc	Clean Acc	Robust Acc
Long schedule: Final model			
83.18 $\pm$ 0.11	36.56 $\pm$ 0.26	84.13 $\pm$ 0.24	36.17 $\pm$ 0.19
Long schedule: Best model			
80.8 $\pm$ 0.36	48.48 $\pm$ 0.27	81.57 $\pm$ 0.44	47.86 $\pm$ 0.1
fast schedule: Final model			
80.58 $\pm$ 0.22	48.12 $\pm$ 0.07	81.9 $\pm$ 0.22	48.14 $\pm$ 0.15

In Table B.2 we observe that the performance of the final model is lower than that of an early stopped method. This could be expected due to the phenomena of robust overfitting described in Rice et al. (2020). However, as a sanity check to make sure that this is not due to a hidden CO during the long schedule which somehow the model recovers from we plot the full training history in Figure B.4. There we can observe that *for N-FGSM there is no CO during training*. We also show FGSM (which is well known has CO for $\epsilon = 8/255$) for comparison.

B.5 Randomized Alpha

Kim et al. (2021) evaluate intermediate points along the RS-FGSM direction in order

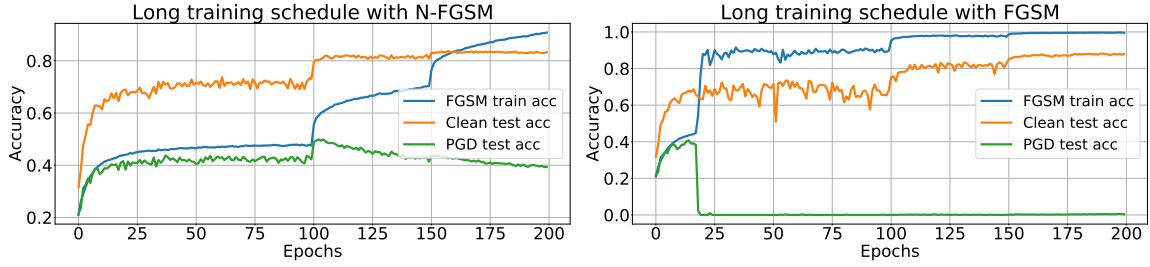


Figure B.4: Training and test accuracy during the long training schedule proposed in Rice et al. (2020). We observe that N-FGSM (left) does not present CO at any point during training, however suffers from robust overfitting as described in Rice et al. (2020) which suggested selecting the best validated model as a simple and yet effective way to improve robustness. On the other hand FGSM (right) suffers from CO where the robustness drops suddenly to 0 and does not recover.

to pick the “optimal” perturbation size. However, we find that increasing the number of intermediate evaluated points does not necessarily lead to increased adversarial accuracy. Moreover, for large perturbations we could not prevent CO even with twice the number of evaluations tested by Kim et al. (2021). This motivates us to test a very simple baseline where instead of evaluating intermediate steps, the RS-FGSM perturbation size is randomly selected as: $\delta = t \cdot \delta_{\text{RS-FGSM}}$ where $t \sim \mathcal{U}[0, 1]^d$. Interestingly, as reported in Figure B.5, we find that this very simple baseline, dubbed *RandAlpha*, is able to avoid CO for all values of ϵ and outperforms Kim et al. (2021) on CIFAR-10, CIFAR-100 and SVHN. This is aligned with our main finding that combining noise with adversarial attacks is indeed a powerful tool that should be explored more thoroughly before developing more expensive solutions.

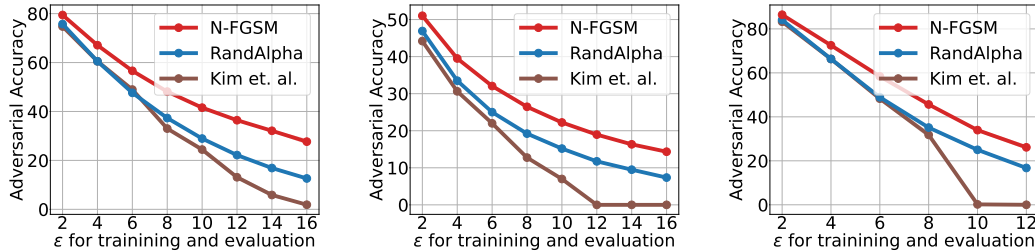


Figure B.5: Comparison of Kim et al. (2021) with RandomAlpha, our baseline where we multiply the RS-FGSM perturbation by a scalar uniformly sampled in $[0, 1]$. We present results on CIFAR-10 (Left), CIFAR-100 (Middle) and SVHN (Right) with PreActResNet18.

B.6 Further visualizations of adversarial perturbations and gradients

In this section we present an extension of Figure 4.5 with further examples. As observed in the main paper, early in training adversarial perturbations (δ) and gradients are consistent across epochs, however, after CO they become hard to interpret. Note that although we label rows as either pre-CO or post-CO we only observe CO for FGSM and RS-FGSM. Both PGD-10 and N-FGSM obtain robust models as shown in detail in the paper.

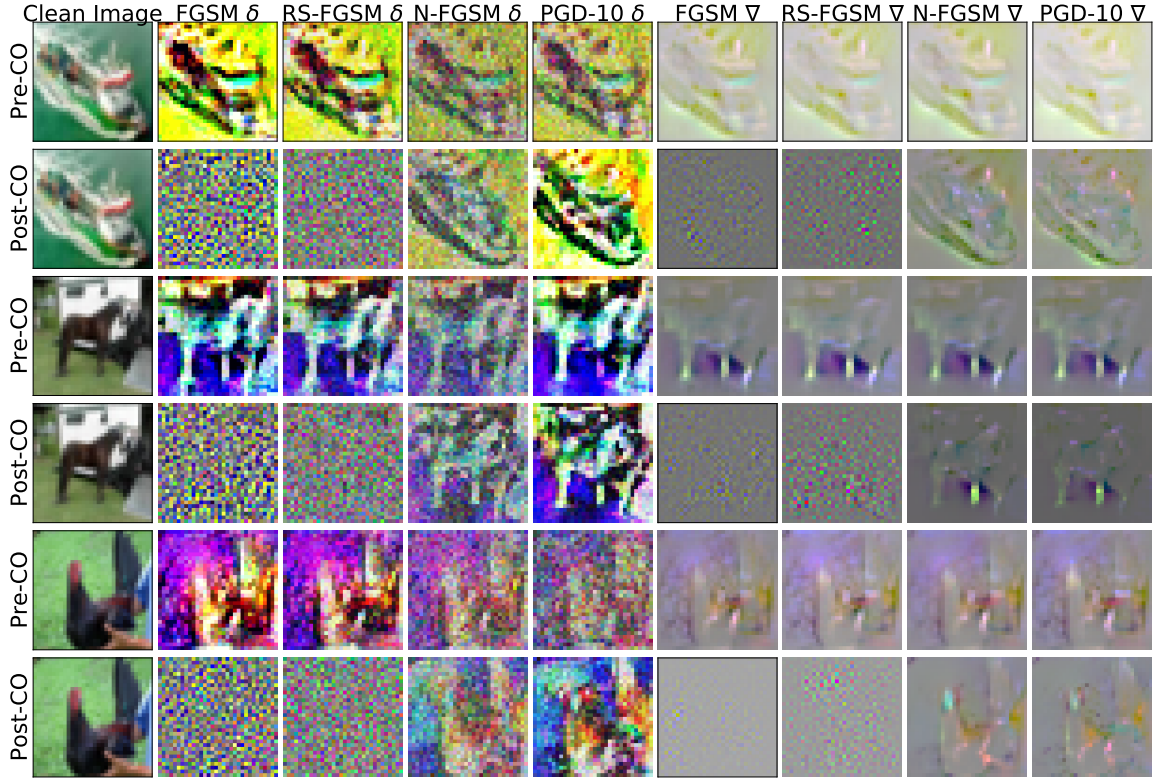


Figure B.6: Visualization of adversarial perturbations (δ 's) and gradients averaged across several epochs before CO (pre-CO) and after (post-CO). Note that only FGSM and RS-FGSM present CO, PGD-10 and N-FGSM do not. Post-CO, FGSM and RS-FGSM obtain δ 's that are hard to interpret, idem for their gradients.

B.7 Robust evaluations with autoattack

Table B.3: Clean (top) and robust accuracy (bottom) for CIFAR-10 and PreacResNet18 evaluated with autoattack (AA) Croce and Hein (2020). We find the same trend as with PGD50-10.

	$\epsilon = 2/255$	$\epsilon = 4/255$	$\epsilon = 6/255$	$\epsilon = 8/255$	$\epsilon = 10/255$	$\epsilon = 12/255$	$\epsilon = 14/255$	$\epsilon = 16/255$
FGSM	91.52 ± 0.08	88.59 ± 0.08	85.17 ± 0.03	86.62 ± 0.08	83.35 ± 2.03	78.51 ± 3.3	77.31 ± 1.9	75.88 ± 1.49
	78.99 ± 0.19	65.99 ± 0.24	54.0 ± 0.32	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
GradAlign	91.48 ± 0.08	88.55 ± 0.18	85.23 ± 0.22	81.69 ± 0.1	77.73 ± 0.18	73.46 ± 0.16	67.87 ± 0.5	61.66 ± 0.32
	79.09 ± 0.21	65.65 ± 0.13	53.99 ± 0.2	44.11 ± 0.34	35.72 ± 0.34	28.66 ± 0.15	22.93 ± 0.33	18.4 ± 0.28
N-FGSM	91.44 ± 0.09	88.36 ± 0.04	84.56 ± 0.12	80.36 ± 0.03	75.81 ± 0.22	71.03 ± 0.16	66.49 ± 0.36	62.86 ± 0.88
	78.99 ± 0.17	66.06 ± 0.25	53.94 ± 0.3	44.36 ± 0.26	36.73 ± 0.27	30.45 ± 0.2	25.08 ± 0.15	19.0 ± 1.08

Following previous work, Andriushchenko and Flammarion (2020); Goodfellow et al. (2015) we have evaluated robustness with PGD50-10, i.e. PGD with 50 iterations and 10 restarts. However, for the sake of completeness, we also present results of robust accuracy evaluated with autoattack Croce and Hein (2020). In Table B.3 we evaluate models adversarially trained with our proposed method N-FGSM, the baseline FGSM and GradAlign. We observe the same pattern as with the PGD50-10 attack, therefore we are convinced that our results are general.

B.8 Catastrophic Overfitting outside the ResNet family

Previous work focusing on CO has only used architectures from the ResNet family. In Table B.4 we present results for adversarial training with a VGG-16 architecture Simonyan and Zisserman (2015). Similarly to other studied models we observe that FGSM leads to CO while N-FGSM is able to prevent it. However, it seems that FGSM presents CO for slightly larger ϵ radii, indicating that the architecture might play a role in CO. We consider investigating this further a promising direction of future work.

Table B.4: Clean (top) and robust accuracy (bottom) for CIFAR-10 and VGG-16 Simonyan and Zisserman (2015) evaluated with PGD50-10. We also observe CO for VGG architecture when trained with FGSM, moreover, N-FGSM is able to prevent CO. Interestingly, for VGG CO happens for slightly large ϵ values indicating that the architecture might play a role in CO.

	$\epsilon = 4/255$	$\epsilon = 6/255$	$\epsilon = 8/255$	$\epsilon = 10/255$	$\epsilon = 12/255$	$\epsilon = 14/255$	$\epsilon = 16/255$
FGSM	85.04 ± 0.1	79.34 ± 0.11	73.39 ± 0.0	82.6 ± 0.0	83.04 ± 0.0	81.4 ± 0.0	80.41 ± 0.21
	62.94 ± 0.07	52.72 ± 0.12	44.0 ± 0.02	0.07 ± 0.0	0.8 ± 0.0	0.25 ± 0.0	0.31 ± 0.15
N-FGSM	84.53 ± 0.0	79.42 ± 0.0	72.01 ± 0.28	66.81 ± 0.54	61.19 ± 0.0	56.97 ± 0.0	53.1 ± 1.19
	63.32 ± 0.0	53.0 ± 0.0	44.3 ± 0.09	38.25 ± 0.1	33.36 ± 0.0	29.23 ± 0.0	25.72 ± 0.22

B.9 Further increasing the attack radii

Following previous work Andriushchenko and Flammarion (2020) we have studied ϵ attack radii up to $\epsilon = 16/255$. Indeed, the performance at these radius is already significantly degraded and thus it would not be very practical for most applications. However, to show that N-FGSM can prevent CO at even larger radii we test two additional radii, $\epsilon = 20/255$ and $\epsilon = 24/255$. In both cases N-FGSM is able to prevent CO. For $\epsilon = 20/255$ we obtain a clean accuracy of 51.63 ± 0.38 and robust of 20.62 ± 0.37 while for $\epsilon = 24/255$ we obtain a clean accuracy of 40.16 ± 0.96 and robust of 15.3 ± 1.49 . We argue that it is of little interest to try even larger perturbations unless more effective methods to improve both the clean and robust performance are found.

B.10 Testing other norms

Following previous work, we have focused on the ℓ_∞ threat model. Although this is where works studying CO have mainly focused, we observe that CO is also present in other norms such as ℓ_1 and ℓ_2 . Moreover, in both cases we observe that N-FGSM is able to prevent CO. Interestingly, the range of norms in which we observe CO is usually much higher than normally tested for these norms which would explain why the ℓ_∞ norm has been the main focus of study in related works.

B.11 Combining N-FGSM with additional regularizers

In this section we present the results from Section 4.7 where we combine N-FGSM with additional regularizers (Sriramanan et al., 2020, 2021) that were proposed for single-step adversarial training to boost the performance. First, we try the proposed methods with the default settings (which use a version of RS-FGSM with Bernoulli noise) and observe they lead to CO for larger ϵ . Then we compare them with N-FGSM

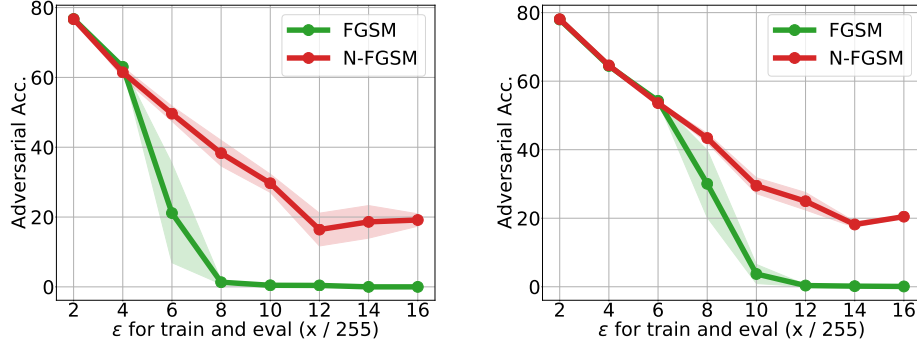


Figure B.7: Robust accuracy after training with FGSM or N-FGSM using ℓ_1 (left) and ℓ_2 (right) perturbations. As observed for ℓ_∞ perturbations FGSM leads to CO, while N-FGSM is able to prevent it. Note that the strength of the perturbations is indicated to be equivalent to ℓ_∞ perturbations where all pixels have maximum magnitude i.e. $\epsilon = 8/255$ indicates perturbations were restricted to an ℓ_p norm of a vector where all components are in $\{-\epsilon, +\epsilon\}$. Which would correspond to an ℓ_1 norm of $n\epsilon$ and an ℓ_2 norm of $\epsilon\sqrt{n}$ where n indicates the dimensionality of the input.

+ Regularizer where we apply their proposed regularizers to N-FGSM. If we apply GAT or NuAT regularizers to N-FGSM then we do not observe CO and usually a boost in performance. Results are presented in Table B.5.

Table B.5: Clean accuracy (top) and PGD50-10 accuracy (bottom) of N-FGSM with additional regularizers introduced in GAT (Sriramanan et al., 2020) and NuAT (Sriramanan et al., 2021). Both GAT and NuAT present CO with their default training method. If we apply their proposed regularizers to N-FGSM we can avoid CO while achieving a boost in performance.

	$\epsilon = 2/255$	$\epsilon = 4/255$	$\epsilon = 6/255$	$\epsilon = 8/255$	$\epsilon = 10/255$	$\epsilon = 12/255$	$\epsilon = 14/255$	$\epsilon = 16/255$
GAT	88.79 ± 0.15	84.35 ± 0.11	80.16 ± 0.15	76.75 ± 0.38	73.71 ± 0.12	80.44 ± 5.08	83.9 ± 1.0	82.17 ± 2.47
	80.04 ± 0.06	68.51 ± 0.08	59.16 ± 0.24	50.98 ± 0.12	43.34 ± 0.23	14.93 ± 9.26	2.33 ± 0.58	1.25 ± 0.51
N-FGSM+GAT	89.1 ± 0.08	84.84 ± 0.05	81.38 ± 0.07	78.28 ± 0.04	75.66 ± 0.35	73.56 ± 0.23	70.84 ± 0.51	65.48 ± 0.96
	79.96 ± 0.21	69.5 ± 0.18	60.06 ± 0.09	51.8 ± 0.34	44.97 ± 0.07	38.71 ± 0.16	32.71 ± 0.11	27.87 ± 0.35
NuAT	87.81 ± 0.24	82.9 ± 0.18	78.06 ± 0.2	73.22 ± 0.34	71.08 ± 4.87	74.38 ± 7.32	78.5 ± 1.54	80.1 ± 1.08
	79.49 ± 0.03	67.77 ± 0.13	57.93 ± 0.17	50.1 ± 0.33	34.35 ± 9.0	17.54 ± 8.82	6.6 ± 0.77	3.29 ± 0.87
NGFSM+NuAT	87.92 ± 0.0	83.54 ± 0.0	78.86 ± 0.25	74.61 ± 0.34	70.37 ± 0.12	65.56 ± 0.19	60.76 ± 0.74	52.79 ± 0.66
	79.52 ± 0.0	68.36 ± 0.0	58.88 ± 0.16	51.12 ± 0.2	44.62 ± 0.38	38.24 ± 0.38	32.85 ± 0.58	29.19 ± 0.35
N-FGSM	91.48 ± 0.17	88.44 ± 0.09	84.72 ± 0.04	80.58 ± 0.22	75.98 ± 0.1	71.46 ± 0.14	67.11 ± 0.37	63.18 ± 0.49
	79.43 ± 0.21	67.09 ± 0.31	56.62 ± 0.26	48.12 ± 0.07	41.56 ± 0.16	36.43 ± 0.16	32.11 ± 0.2	27.67 ± 0.93

B.12 Imagenet experimental details

For our experiments on Imagenet we mainly follow the settings from Wong et al. (2020). However, for simplicity we did not do image resizing which requires storing two additional Imagenet datasets. More importantly, we found that the learning rate schedule suggested in Wong et al. (2020) was not optimal for N-FGSM. The schedule suggested in Wong et al. (2020) follows three different stages in which the learning increases or decreases linearly for some iterations. In particular in the first stage, the learning rate has an initial warm-up where it increases linearly from 0.0 to 0.4 during the first epoch and then decreases linearly to 0.04 during the next 5 epochs. As a lucky coincidence when debugging, we modified this initial stage such that we preserved the initial increase to 0.4 for the first epoch, but then we directly jumped to a learning rate of 0.04 which remained constant for the next 5 epochs. For phase 2 and 3 both schedules remained the same. First decreasing from 0.04 to 0.004 for epoch 6 to 12 and finally from 0.004 to 0.0004 for epoch 12 to 15. This small change made N-FGSM improve both in clean and robust accuracy for $\epsilon = 4/255$, $6/255$ and these are the numbers reported. This indicates that further tuning the learning rate schedule might be an effective way to improve performance and even help prevent CO, however, due to the computational demands of ImageNet adversarial training we leave it for future work. To be thorough we also trained RS-FGSM and FGSM with the modified schedule and found that neither of them benefit from it. Regarding N-FGSM hyperparameters, for $\epsilon = 2/255$ we used $\alpha = 2/255$ and $k = 1$; for $\epsilon = 4/255$ we used $\alpha = 4/255$ and $k = 1$; and for $\epsilon = 6/255$ we also used $\alpha = 4/255$ and $k = 1$.

B.13 Visualization of the loss surface

In this section we present a visualization of the loss surface. We adapted the code from Kim et al. (2021) to analyse the shape of the loss surface at the end of training for different methods. Kim et al. (2021) reported that after adversarial training CO, the loss surface would become non-linear. In particular, they found that the FGSM

perturbation seems to be misguided by local maxima very close to the clean image that result in ineffective attacks. We note this was already reported by Tramèr et al. (2018) which proposed to perform a random step to *escape* those maxima. We argue that adding noise to the random step, when properly implemented, actually prevents those maxima to appear in the first place.

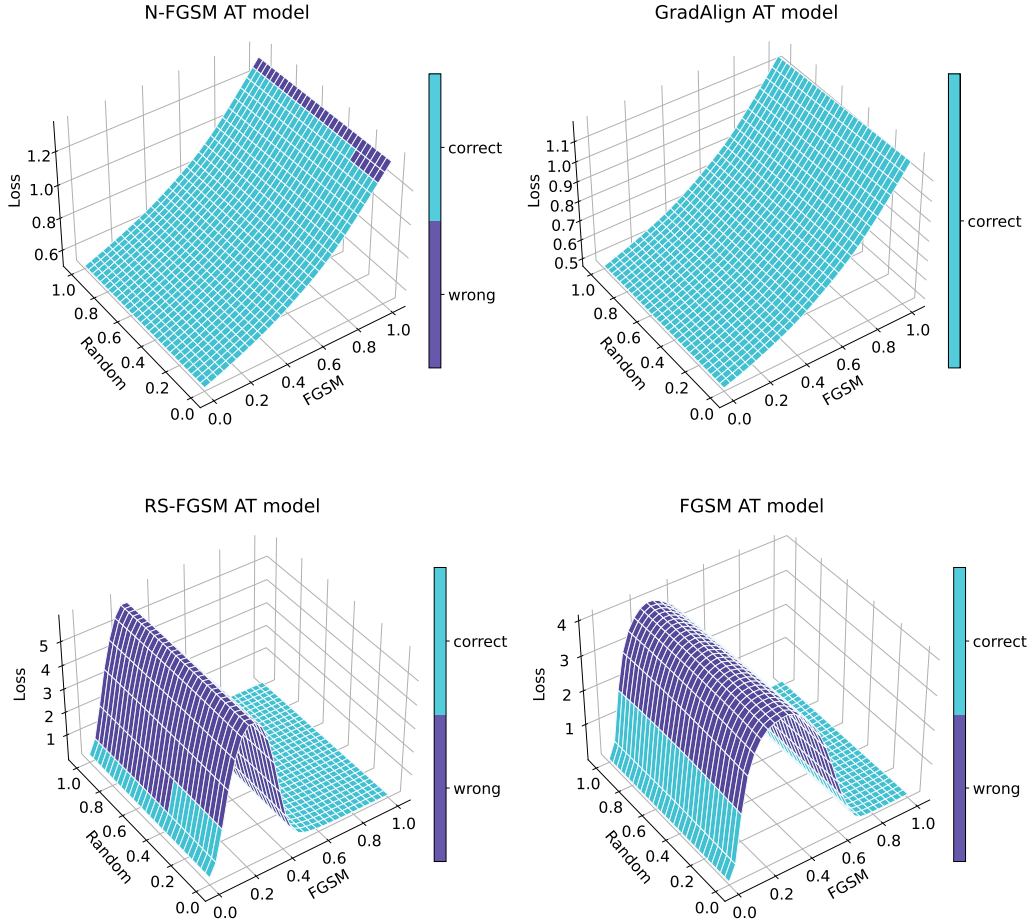


Figure B.8: Visualization of the loss surface for models trained using different methods. Given a clean sample from the test set in coordinate $(0,0)$, we compute the FGSM perturbation and evaluate the loss on the subspace generated by the FGSM perturbation direction and a random direction. That is, we evaluate $x_{\text{clean}} + t_1 \cdot \delta_{\text{FGSM}} + t_2 \cdot \delta_{\text{random}}$, where $t_1, t_2 \in [0, 1]$. Note that FGSM and RS-FGSM both have CO and the final models present a highly non-linear loss surface, on the other hand, both N-FGSM and GradAlign produce final models with a very linear loss surface which is key to obtain meaningful perturbations.

B.14 Magnitude of N-FGSM perturbations

Lemma B.14.1 (Expected perturbation). *Consider the N-FGSM perturbation as defined in (4.3)*

$$\delta_{N-FGSM} = \eta + \alpha \cdot \text{sign}(\nabla_x \ell(f(x + \eta), y)), \quad \text{where } \eta \sim \Omega.$$

Let the distribution Ω be the uniform distribution $\mathcal{U}([-k\epsilon, k\epsilon]^d)$ and $\alpha > 0$. Then,

$$\mathbb{E}_\eta [\|\delta_{N-FGSM}\|_2^2] = d \left(\frac{k^2 \epsilon^2}{3} + \alpha^2 \right) \quad \text{and} \quad \mathbb{E}_\eta [\|\delta_{N-FGSM}\|_2] \leq \sqrt{d \left(\frac{k^2 \epsilon^2}{3} + \alpha^2 \right)}$$

Proof. By Jensen's inequality, we have

$$\mathbb{E}_\eta [\|\delta_{N-FGSM}\|_2] \leq \sqrt{\mathbb{E}_\eta [\|\delta_{N-FGSM}\|_2^2]}$$

Then let us consider the term $\mathbb{E}_\eta [\|\delta_{N-FGSM}\|_2^2]$ and use the shorthand $\nabla(\eta)_i = (\nabla_x \ell(f(x + \eta), y))_i$.

$$\begin{aligned}
\mathbb{E}_\eta \left[\|\delta_{\text{N-FGSM}}\|_2^2 \right] &= \mathbb{E}_\eta \|\eta + \alpha \cdot \text{sign}(\nabla_x \ell(f(x + \eta), y))\|_2^2 \\
&= \mathbb{E}_\eta \left[\sum_{i=1}^d (\eta_i + \alpha \cdot \text{sign}(\nabla(\eta)_i))^2 \right] \\
&= \sum_{i=1}^d \mathbb{E}_\eta \left[(\eta_i + \alpha \cdot \text{sign}(\nabla(\eta)_i))^2 \right] \\
&= \sum_{i=1}^d \mathbb{E}_\eta \left[(\eta_i + \alpha \cdot \text{sign}(\nabla(\eta)_i))^2 \mid \text{sign}(\nabla(\eta)_i) = 1 \right] \mathbb{P}_\eta [\text{sign}(\nabla(\eta)_i) = 1] \\
&\quad + \sum_{i=1}^d \mathbb{E}_\eta \left[(\eta_i + \alpha \cdot \text{sign}(\nabla(\eta)_i))^2 \mid \text{sign}(\nabla(\eta)_i) = -1 \right] \mathbb{P}_\eta [\text{sign}(\nabla(\eta)_i) = -1] \\
&= \sum_{i=1}^d \frac{1}{2k\epsilon} \int_{-k\epsilon}^{k\epsilon} (\eta_i + \alpha)^2 d\eta_i \cdot \mathbb{P}_\eta [\text{sign}(\nabla(\eta)_i) = 1] \\
&\quad + \frac{1}{2k\epsilon} \sum_{i=1}^d \int_{-k\epsilon}^{k\epsilon} (\eta_i - \alpha)^2 d\eta_i \cdot \mathbb{P}_\eta [\text{sign}(\nabla(\eta)_i) = -1] \\
&= \sum_{i=1}^d \frac{1}{2k\epsilon} \int_{\alpha-k\epsilon}^{\alpha+k\epsilon} z^2 dz \cdot \mathbb{P}_\eta [\text{sign}(\nabla(\eta)_i) = 1] \\
&\quad + \frac{1}{2k\epsilon} \sum_{i=1}^d \int_{-\alpha-k\epsilon}^{-\alpha+k\epsilon} z^2 dz \cdot \mathbb{P}_\eta [\text{sign}(\nabla(\eta)_i) = -1] \\
&= \sum_{i=1}^d \frac{1}{2k\epsilon} \int_{\alpha-k\epsilon}^{\alpha+k\epsilon} z^2 dz \cdot \mathbb{P}_\eta [\text{sign}(\nabla(\eta)_i) = 1] \\
&\quad + \frac{1}{2k\epsilon} \sum_{i=1}^d \int_{\alpha-k\epsilon}^{\alpha+k\epsilon} z^2 dz \cdot \mathbb{P}_\eta [\text{sign}(\nabla(\eta)_i) = -1] \\
&= \frac{1}{2k\epsilon} \int_{\alpha-k\epsilon}^{\alpha+k\epsilon} z^2 dz \sum_{i=1}^d (\mathbb{P}_\eta [\text{sign}(\nabla(\eta)_i) = 1] + \mathbb{P}_\eta [\text{sign}(\nabla(\eta)_i) = -1]) \\
&= \frac{d}{6k\epsilon} \left[(\alpha + k\epsilon)^3 - (\alpha - k\epsilon)^3 \right] = \frac{dk^2\epsilon^2}{3} + d\alpha^2
\end{aligned}$$

Therefore,

$$\mathbb{E}_\eta [\|\delta_{\text{N-FGSM}}\|_2] \leq \sqrt{d \left(\frac{k^2 \epsilon^2}{3} + \alpha^2 \right)}.$$

□

Theorem B.14.2. *Let $\delta_{\text{N-FGSM}}$ be our proposed single-step method defined by (4.3), δ_{FGSM} be the FGSM method (Goodfellow et al., 2015) and $\delta_{\text{RS-FGSM}}$ be the RS-FGSM method (Wong et al., 2020). Then, with default hyperparameter values and for any $\epsilon > 0$, we have that*

$$\mathbb{E}_\eta [\|\delta_{\text{N-FGSM}}\|_2^2] > \mathbb{E}_\eta [\|\delta_{\text{FGSM}}\|_2^2] > \mathbb{E}_\eta [\|\delta_{\text{RS-FGSM}}\|_2^2].$$

Proof. From Lemma B.14.1 we have that

$$\mathbb{E}_\eta [\|\delta_{\text{N-FGSM}}\|_2^2] = d \left(\frac{k^2 \epsilon^2}{3} + \alpha^2 \right).$$

On the other hand, Andriushchenko and Flammarion (2020) showed that

$$\mathbb{E}_\eta [\|\delta_{\text{RS-FGSM}}\|_2^2] = d \left(-\frac{1}{6\epsilon} \alpha^3 + \frac{1}{2} \alpha^2 + \frac{1}{3} \epsilon^2 \right).$$

Finally, we note that

$$\mathbb{E}_\eta [\|\delta_{\text{FGSM}}\|_2^2] = \|\delta_{\text{FGSM}}\|_2^2 = d\epsilon^2.$$

The default hyperparameters for N-FGSM are $k = 2$, $\alpha = \epsilon$ and RS-FGSM uses $\alpha = 5\epsilon/4$. With these hyperparameters and any $\epsilon > 0$ we have

$$\mathbb{E}_\eta [\|\delta_{\text{N-FGSM}}\|_2^2] = \frac{7}{3} d\epsilon^2 > \mathbb{E}_\eta [\|\delta_{\text{FGSM}}\|_2^2] = d\epsilon^2 > \mathbb{E}_\eta [\|\delta_{\text{RS-FGSM}}\|_2^2] = \frac{101}{128} d\epsilon^2$$

□

In Lemma B.14.1 we compute the expected value of the squared ℓ_2 norm of N-FGSM perturbations and by Jensen's inequality we obtain an upper bound for the expected ℓ_2 norm of N-FGSM perturbations. However, obtaining the exact expected magnitude is more complex. To compliment our analytic results, we approximate the ℓ_2 norm

of FGSM, RS-FGSM and N-FGSM via Monte Carlo sampling. Results are presented in Figure B.9. We observe that the empirical estimations are very close to the analytical upper bounds and that indeed, N-FGSM has a magnitude significantly above that of FGSM or RS-FGSM.

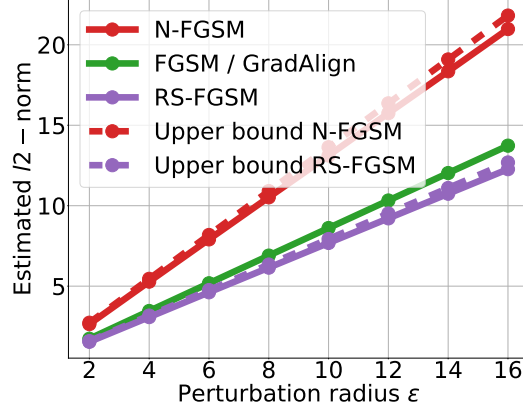


Figure B.9: Monte Carlo estimations of the expected l_2 -norm of perturbations from different methods and corresponding analytical upper bounds. As mentioned in Andriushchenko and Flammarion (2020), we observe that RS-FGSM perturbations have lower l_2 norm than FGSM. However, N-FGSM perturbations have a significantly higher l_2 -norm than both RS-FGSM and FGSM. This seems to indicate that the role of random step is not simply to lower the l_2 norm as previously suggested (Andriushchenko and Flammarion, 2020).

B.15 N-FGSM with Gaussian noise

In the main paper we have only explored noise sources coming from a Uniform distribution. Since we are measuring robustness against l_∞ -attacks, the Uniform distribution is a natural choice because the random perturbations will be bounded to the l_∞ ball defined by the span of the distribution. However, for the sake of completeness, we also explore the performance of augmenting the samples from a Gaussian distribution where we choose its standard deviation to match that of the uniform distribution. In Table B.6 we present a comparison of the clean (top) and PGD-50-10 (bottom) accuracy for different values of α and noise magnitude with $\epsilon = 8/255$. Recall that by default we use Uniform distribution $\mathcal{U}[-k, k]$, therefore hyperparameter k sets the noise magnitude.

Table B.6: Comparison of the clean (top) and PGD-50-10 (bottom) accuracy across different values of step-size α and noise magnitude for the Uniform and Gaussian distributions with $\epsilon = 8/255$. For every value of k , we use a Gaussian with matching standard deviation. We observe that when we match the standard deviation, both distribution perform similarly.

Uniform Noise				Gaussian Noise		
	$\alpha = 6/255 \ (0.75\epsilon)$	$\alpha = 8/255 \ (1\epsilon)$	$\alpha = 10/255 \ (1.25\epsilon)$	$\alpha = 6/255 \ (0.75\epsilon)$	$\alpha = 8/255 \ (1\epsilon)$	$\alpha = 10/255 \ (1.25\epsilon)$
$k = 0.5\epsilon$	85.52 ± 0.23	81.54 ± 0.19	82.81 ± 1.11	85.27 ± 0.11	81.71 ± 0.27	83.34 ± 1.48
	44.14 ± 0.24	47.93 ± 0.11	0.0 ± 0.0	44.23 ± 0.17	47.98 ± 0.14	0.0 ± 0.0
$k = 1\epsilon$	85.03 ± 0.09	81.57 ± 0.07	77.32 ± 0.14	85.01 ± 0.17	81.35 ± 0.14	77.22 ± 0.32
	44.44 ± 0.13	48.16 ± 0.21	49.68 ± 0.25	44.41 ± 0.04	48.21 ± 0.11	49.83 ± 0.1
$k = 2\epsilon$	84.49 ± 0.1	80.58 ± 0.22	76.49 ± 0.14	84.35 ± 0.24	80.44 ± 0.31	76.33 ± 0.37
	44.44 ± 0.15	48.12 ± 0.07	49.77 ± 0.37	44.59 ± 0.22	48.34 ± 0.1	49.77 ± 0.23

Increasing the FGSM step size without increasing the amount of noise leads to CO. Note results for $k = 0.5\epsilon$. More importantly, results are very similar when the two noise distributions share the same standard deviation. Thus, using Gaussian instead of Uniform noise does not seem to alter the results. Although this might be expected, we remark that the Gaussian is an unbounded noise distribution and the common practice in adversarial training is to always restrict the norm of the perturbations.

B.16 Training with noise augmented samples

Gilmer et al. (2019) and Fawzi et al. (2018) report a close link between robustness to adversarial attacks and robustness to random noise. Actually, Gilmer et al. (2019) report that training with noise-augmented samples can improve adversarial accuracy and vice-versa. We note that N-FGSM can actually be seen as a combination of noise-augmentation and adversarial attacks. Here we perform an ablation where we train models with samples augmented with uniform noise $\mathcal{U}[-k, k]$ and then test the PGD-50-10 accuracy. We observe, that indeed random noise can increase the robustness to worst-case perturbations for small $\epsilon - l_\infty$ balls. However, as we increase ϵ , noise augmentation is no longer very effective. With N-FGSM, we apply a weak attack to these noise-augmented samples and this seems to be enough to make them effective for adversarial training.

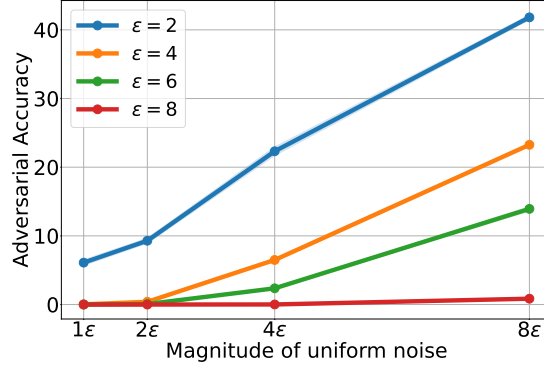


Figure B.10: Training with uniform noise augmented samples improves adversarial accuracy for small perturbations but is not effective to protect against larger l_∞ radius ϵ . This motivates us to further augment the noisy samples with FGSM. All experiments are averaged over 3 runs.

B.17 Comparison of adversarial training cost

In this section we describe how we compute the relative training cost for single-step methods shown in Figure 4.1 (right). We approximate the cost based on the number of forward/backward passes each method uses, disregarding the cost of other additional operations such as adding a random step for RS-FGSM or N-FGSM. We understand these operations have a negligible cost compared to a full forward or backward pass.

FGSM: FGSM is the cheapest of all methods since it only uses one forward/backward to compute the attack and an additional forward/backward to compute the weight update. Hence, $\text{Cost FGSM} = 2 \text{ F/B}$.

RS-FGSM: As previously mentioned, we do not take into account the cost of random steps or clipping, hence we consider RS-FGSM to have the same cost as standard FGSM. $\text{Cost RS-FGSM} = 2 \text{ F/B}$.

N-FGSM: Idem as before, cost of N-FGSM = 2 F/B.

ZeroGrad: For ZeroGrad they need to do an additional sorting operation to find the smallest gradient components. This could be potentially expensive, however, since the size of the input image is several orders of magnitude smaller than that of the network, we also ignore this cost. $\text{Cost ZeroGrad} = 2 \text{ F/B}$.

MultiGrad: MultiGrad computes 3 random steps and evaluates the gradient in all of them. Therefore, it needs to do 3 F/B to compute the attack and an additional one to update the weights. Cost MultiGrad = 4 F/B.

Kim et al. (2021): Kim et al. (2021) compute the RS-FGSM perturbation and evaluate the model on c points along this direction. Therefore, they will spend 1F/B on the RS-FGSM attack, $c - 1$ F on the evaluations since the clean image has already been evaluated; and 1 F/B for the weight update. In our plot, we used $c = 3$ since it was the most chosen setting. Kim et al. (2021) assume the cost of a forward is similar to that of a backward pass, following this assumption, cost of Kim et al. (2021) is $1 \text{ F/B} + 2 \text{ F} + 1 \text{ F/B} = 3 \text{ F/B}$

Free-AT: Shafahi et al. (2019) re-use the gradient from the previous backward pass to compute the FGSM perturbation of the current iteration. Hence, the cost of their training is only 1 F/B per iteration. However, Wong et al. (2020) observed they needed a longer training schedule to produce comparable results. Therefore, the total training cost per iteration (1 F/B) is scaled by 96 in the case of Free-AT, while it is only scaled by 30 for other methods. Relative cost Free = $(96 \cdot 1 \text{ F/B}) / (30 \cdot 2 \text{ F/B})$.

GradAlign: Finally, GradAlign uses FGSM with a regularizer. However, this regularizer needs to compute second-order derivatives via double backpropagation, which does not have the same cost as regular backpropagation. Andriushchenko and Flammarion (2020) report that the cost of using GradAlign regularizer increased the cost of FGSM by 3.

B.18 Infrastructure details and GPU hours

All our training runs have been conducted on either NVIDIA GPU V-100 or P-100 from an internal cluster. The total compute for the results presented in this work is roughly 2500 hours.

B.19 Detailed numerical results

In this section we present the tables with the exact numbers used in plots comparing adversarial training methods in Section 4.5.1 and Section 4.7. For each method and $\epsilon - l_\infty$ radius, the top number is the clean accuracy while the bottom number is the PGD-50-10 accuracy. We separate single-step from multi-step methods with a double line.

PreActResNet18 – CIFAR-10 Dataset

	$\epsilon = 2/255$	$\epsilon = 4/255$	$\epsilon = 6/255$	$\epsilon = 8/255$	$\epsilon = 10/255$	$\epsilon = 12/255$	$\epsilon = 14/255$	$\epsilon = 16/255$
N-FGSM	91.48 $\pm$ 0.17 79.43 $\pm$ 0.21	88.44 $\pm$ 0.09 67.09 $\pm$ 0.31	84.72 $\pm$ 0.04 56.62 $\pm$ 0.26	80.58 $\pm$ 0.22 48.12 $\pm$ 0.07	75.98 $\pm$ 0.1 41.56 $\pm$ 0.16	71.46 $\pm$ 0.14 36.43 $\pm$ 0.16	67.11 $\pm$ 0.37 32.11 $\pm$ 0.2	63.18 $\pm$ 0.49 27.67 $\pm$ 0.93
Grad Align	91.73 $\pm$ 0.04 79.16 $\pm$ 0.03	88.76 $\pm$ 0.0 67.13 $\pm$ 0.26	85.67 $\pm$ 0.02 56.27 $\pm$ 0.31	81.9 $\pm$ 0.22 48.14 $\pm$ 0.15	77.54 $\pm$ 0.06 40.75 $\pm$ 0.28	73.29 $\pm$ 0.23 34.51 $\pm$ 0.63	68.01 $\pm$ 0.32 30.36 $\pm$ 0.27	61.3 $\pm$ 0.15 26.64 $\pm$ 0.27
FGSM	91.6 $\pm$ 0.1 79.35 $\pm$ 0.06	88.77 $\pm$ 0.04 67.11 $\pm$ 0.09	85.58 $\pm$ 0.11 56.33 $\pm$ 0.41	86.41 $\pm$ 0.7 0.0 $\pm$ 0.0	82.08 $\pm$ 1.62 0.0 $\pm$ 0.0	80.6 $\pm$ 2.59 0.0 $\pm$ 0.0	76.04 $\pm$ 2.37 0.0 $\pm$ 0.0	77.14 $\pm$ 2.46 0.0 $\pm$ 0.0
RS-FGSM	92.09 $\pm$ 0.05 78.64 $\pm$ 0.08	89.69 $\pm$ 0.01 66.12 $\pm$ 0.22	87.0 $\pm$ 0.12 54.87 $\pm$ 0.22	84.05 $\pm$ 0.13 46.08 $\pm$ 0.18	85.21 $\pm$ 0.51 0.0 $\pm$ 0.0	65.22 $\pm$ 23.23 0.0 $\pm$ 0.0	43.59 $\pm$ 25.01 0.0 $\pm$ 0.0	76.66 $\pm$ 0.38 0.0 $\pm$ 0.0
Kim et. al.	92.85 $\pm$ 0.11 74.74 $\pm$ 0.35	91.1 $\pm$ 0.04 60.51 $\pm$ 0.4	89.34 $\pm$ 0.05 48.95 $\pm$ 0.45	89.02 $\pm$ 0.1 33.01 $\pm$ 0.09	88.27 $\pm$ 0.14 24.43 $\pm$ 0.84	88.35 $\pm$ 0.31 13.11 $\pm$ 0.63	90.01 $\pm$ 0.25 5.86 $\pm$ 0.57	90.45 $\pm$ 0.08 1.88 $\pm$ 0.05
AT Free	87.99 $\pm$ 0.16 74.27 $\pm$ 0.33	84.98 $\pm$ 0.13 62.47 $\pm$ 0.25	81.77 $\pm$ 0.11 53.18 $\pm$ 0.15	78.41 $\pm$ 0.18 46.03 $\pm$ 0.36	74.79 $\pm$ 0.22 39.87 $\pm$ 0.07	73.91 $\pm$ 4.19 22.99 $\pm$ 16.26	61.92 $\pm$ 14.94 0.0 $\pm$ 0.0	71.64 $\pm$ 3.89 0.0 $\pm$ 0.0
ZeroGrad	91.71 $\pm$ 0.08 79.36 $\pm$ 0.05	88.8 $\pm$ 0.11 67.32 $\pm$ 0.02	85.71 $\pm$ 0.1 56.14 $\pm$ 0.21	82.62 $\pm$ 0.05 47.08 $\pm$ 0.1	79.91 $\pm$ 0.12 37.58 $\pm$ 0.2	78.11 $\pm$ 0.2 27.41 $\pm$ 0.27	75.66 $\pm$ 0.46 21.29 $\pm$ 0.97	75.42 $\pm$ 0.13 13.06 $\pm$ 0.22
MultiGrad	91.57 $\pm$ 0.16 79.34 $\pm$ 0.02	88.74 $\pm$ 0.12 66.81 $\pm$ 0.02	85.75 $\pm$ 0.05 56.02 $\pm$ 0.3	82.33 $\pm$ 0.14 47.29 $\pm$ 0.07	78.73 $\pm$ 0.16 40.11 $\pm$ 0.24	75.28 $\pm$ 0.2 33.87 $\pm$ 0.17	80.94 $\pm$ 5.94 9.55 $\pm$ 13.5	71.42 $\pm$ 5.63 16.35 $\pm$ 11.57
PGD-2	91.4 $\pm$ 0.07 79.55 $\pm$ 0.15	88.46 $\pm$ 0.13 67.62 $\pm$ 0.03	85.14 $\pm$ 0.13 57.39 $\pm$ 0.13	81.41 $\pm$ 0.05 49.58 $\pm$ 0.08	77.18 $\pm$ 0.15 43.3 $\pm$ 0.11	72.9 $\pm$ 0.26 38.13 $\pm$ 0.15	70.39 $\pm$ 2.71 22.89 $\pm$ 15.26	64.81 $\pm$ 11.58 9.6 $\pm$ 13.37
PGD-10	91.25 $\pm$ 0.04 79.47 $\pm$ 0.13	88.34 $\pm$ 0.11 68.29 $\pm$ 0.24	84.79 $\pm$ 0.11 58.85 $\pm$ 0.18	80.71 $\pm$ 0.14 51.33 $\pm$ 0.31	76.13 $\pm$ 0.35 45.02 $\pm$ 0.49	71.24 $\pm$ 0.3 39.93 $\pm$ 0.5	66.7 $\pm$ 0.39 36.02 $\pm$ 0.67	62.11 $\pm$ 0.62 32.22 $\pm$ 0.64

PreActResNet18 – CIFAR-100 Dataset

	$\epsilon = 2/255$	$\epsilon = 4/255$	$\epsilon = 6/255$	$\epsilon = 8/255$	$\epsilon = 10/255$	$\epsilon = 12/255$	$\epsilon = 14/255$	$\epsilon = 16/255$
N-FGSM	69.12 $\pm$ 0.27 51.02 $\pm$ 0.34	64.0 $\pm$ 0.06 39.5 $\pm$ 0.12	59.53 $\pm$ 0.02 32.06 $\pm$ 0.37	54.9 $\pm$ 0.2 26.46 $\pm$ 0.22	50.6 $\pm$ 0.16 22.23 $\pm$ 0.17	46.06 $\pm$ 0.14 18.95 $\pm$ 0.15	41.67 $\pm$ 0.25 16.33 $\pm$ 0.15	37.91 $\pm$ 0.11 14.34 $\pm$ 0.07
Grad Align	68.96 $\pm$ 0.15 51.31 $\pm$ 0.12	64.71 $\pm$ 0.16 39.37 $\pm$ 0.25	60.42 $\pm$ 0.23 31.91 $\pm$ 0.28	56.53 $\pm$ 0.31 25.8 $\pm$ 0.14	54.06 $\pm$ 0.44 18.7 $\pm$ 1.92	48.87 $\pm$ 0.32 17.86 $\pm$ 0.04	43.84 $\pm$ 0.14 15.51 $\pm$ 0.16	38.93 $\pm$ 0.21 13.62 $\pm$ 0.19
FGSM	69.01 $\pm$ 0.13 51.3 $\pm$ 0.19	64.47 $\pm$ 0.15 39.7 $\pm$ 0.16	63.85 $\pm$ 2.18 10.93 $\pm$ 14.64	53.42 $\pm$ 0.65 0.0 $\pm$ 0.0	45.06 $\pm$ 2.29 0.0 $\pm$ 0.0	46.14 $\pm$ 2.58 0.0 $\pm$ 0.0	41.66 $\pm$ 0.88 0.0 $\pm$ 0.0	44.68 $\pm$ 1.74 0.0 $\pm$ 0.0
RS-FGSM	69.83 $\pm$ 0.29 50.13 $\pm$ 0.32	65.9 $\pm$ 0.36 38.36 $\pm$ 0.19	62.15 $\pm$ 0.23 30.82 $\pm$ 0.08	55.26 $\pm$ 6.86 0.01 $\pm$ 0.01	32.33 $\pm$ 12.12 0.0 $\pm$ 0.0	36.07 $\pm$ 2.59 0.0 $\pm$ 0.0	21.52 $\pm$ 5.56 0.0 $\pm$ 0.0	20.38 $\pm$ 6.15 0.0 $\pm$ 0.0
Kim et. al.	72.92 $\pm$ 0.41 44.19 $\pm$ 0.25	70.16 $\pm$ 0.07 30.63 $\pm$ 0.28	67.98 $\pm$ 0.19 22.0 $\pm$ 0.02	68.07 $\pm$ 0.1 12.75 $\pm$ 0.21	68.37 $\pm$ 0.21 6.98 $\pm$ 0.23	74.09 $\pm$ 0.06 0.0 $\pm$ 0.0	74.06 $\pm$ 0.34 0.0 $\pm$ 0.0	74.01 $\pm$ 0.36 0.0 $\pm$ 0.0
AT Free	63.01 $\pm$ 0.19 45.7 $\pm$ 0.33	59.41 $\pm$ 0.27 35.95 $\pm$ 0.09	55.43 $\pm$ 0.37 29.37 $\pm$ 0.21	51.91 $\pm$ 0.08 24.32 $\pm$ 0.4	48.11 $\pm$ 0.09 20.64 $\pm$ 0.22	43.48 $\pm$ 1.25 5.71 $\pm$ 8.05	18.33 $\pm$ 4.86 0.0 $\pm$ 0.0	20.43 $\pm$ 11.25 0.0 $\pm$ 0.0
ZeroGrad	69.35 $\pm$ 0.36 51.1 $\pm$ 0.09	64.59 $\pm$ 0.32 39.38 $\pm$ 0.15	60.69 $\pm$ 0.09 31.72 $\pm$ 0.21	56.94 $\pm$ 0.13 25.87 $\pm$ 0.09	54.55 $\pm$ 0.17 19.49 $\pm$ 0.08	52.97 $\pm$ 0.34 14.32 $\pm$ 0.08	50.87 $\pm$ 0.26 10.92 $\pm$ 0.59	50.73 $\pm$ 0.3 7.3 $\pm$ 0.16
MultiGrad	69.01 $\pm$ 0.16 51.15 $\pm$ 0.03	64.44 $\pm$ 0.11 39.16 $\pm$ 0.03	60.65 $\pm$ 0.26 31.73 $\pm$ 0.09	56.84 $\pm$ 0.2 25.96 $\pm$ 0.11	53.62 $\pm$ 0.25 21.37 $\pm$ 0.16	53.05 $\pm$ 1.85 9.57 $\pm$ 7.32	48.28 $\pm$ 0.66 3.2 $\pm$ 4.49	45.28 $\pm$ 11.14 0.0 $\pm$ 0.0
PGD-2	69.18 $\pm$ 0.1 51.36 $\pm$ 0.03	64.32 $\pm$ 0.14 40.06 $\pm$ 0.14	60.21 $\pm$ 0.13 32.99 $\pm$ 0.24	55.8 $\pm$ 0.16 27.38 $\pm$ 0.16	51.68 $\pm$ 0.1 23.39 $\pm$ 0.19	48.2 $\pm$ 0.1 19.83 $\pm$ 0.29	46.14 $\pm$ 1.24 10.55 $\pm$ 7.51	37.97 $\pm$ 10.52 4.79 $\pm$ 6.75
PGD-10	68.83 $\pm$ 0.07 51.51 $\pm$ 0.27	63.87 $\pm$ 0.09 40.59 $\pm$ 0.36	59.37 $\pm$ 0.07 33.65 $\pm$ 0.02	54.79 $\pm$ 0.38 28.55 $\pm$ 0.27	50.53 $\pm$ 0.15 24.17 $\pm$ 0.12	46.05 $\pm$ 0.21 21.2 $\pm$ 0.12	41.76 $\pm$ 0.07 18.72 $\pm$ 0.06	37.81 $\pm$ 0.14 16.59 $\pm$ 0.16

PreActResNet18 – SVHN Dataset

	$\epsilon = 2/255$	$\epsilon = 4/255$	$\epsilon = 6/255$	$\epsilon = 8/255$	$\epsilon = 10/255$	$\epsilon = 12/255$
N-FGSM	96.01 $\pm$ 0.04 86.44 $\pm$ 0.1	94.54 $\pm$ 0.15 72.53 $\pm$ 0.19	92.25 $\pm$ 0.33 58.42 $\pm$ 0.14	89.56 $\pm$ 0.49 45.63 $\pm$ 0.11	86.74 $\pm$ 0.86 33.96 $\pm$ 0.49	81.48 $\pm$ 1.64 26.13 $\pm$ 0.81
Grad Align	96.02 $\pm$ 0.05 86.43 $\pm$ 0.1	94.56 $\pm$ 0.21 72.12 $\pm$ 0.19	92.53 $\pm$ 0.24 57.34 $\pm$ 0.24	90.1 $\pm$ 0.34 43.85 $\pm$ 0.14	87.23 $\pm$ 0.75 32.87 $\pm$ 0.33	84.01 $\pm$ 0.46 23.62 $\pm$ 0.41
FGSM	96.04 $\pm$ 0.07 86.5 $\pm$ 0.05	95.67 $\pm$ 0.07 13.61 $\pm$ 5.83	93.73 $\pm$ 0.68 0.56 $\pm$ 0.72	91.74 $\pm$ 0.86 0.26 $\pm$ 0.36	90.76 $\pm$ 0.63 0.07 $\pm$ 0.1	87.17 $\pm$ 0.43 0.0 $\pm$ 0.0
RS-FGSM	96.18 $\pm$ 0.11 86.16 $\pm$ 0.14	95.09 $\pm$ 0.09 71.28 $\pm$ 0.4	95.11 $\pm$ 0.44 0.11 $\pm$ 0.08	94.46 $\pm$ 0.16 0.0 $\pm$ 0.0	93.88 $\pm$ 0.24 0.0 $\pm$ 0.0	92.74 $\pm$ 0.5 0.0 $\pm$ 0.0
Kim et. al.	96.35 $\pm$ 0.02 83.26 $\pm$ 0.24	95.25 $\pm$ 0.08 66.32 $\pm$ 0.63	94.83 $\pm$ 0.02 48.27 $\pm$ 0.52	94.88 $\pm$ 0.29 31.8 $\pm$ 1.1	96.61 $\pm$ 0.09 0.18 $\pm$ 0.21	96.61 $\pm$ 0.01 0.0 $\pm$ 0.0
AT Free	95.01 $\pm$ 0.09 84.55 $\pm$ 0.27	93.66 $\pm$ 0.12 71.61 $\pm$ 0.75	91.72 $\pm$ 0.29 59.31 $\pm$ 1.0	91.29 $\pm$ 4.07 0.01 $\pm$ 0.0	91.86 $\pm$ 3.66 0.0 $\pm$ 0.0	92.36 $\pm$ 1.0 0.0 $\pm$ 0.0
ZeroGrad	96.06 $\pm$ 0.03 86.43 $\pm$ 0.1	94.81 $\pm$ 0.16 71.59 $\pm$ 0.22	93.53 $\pm$ 0.26 51.72 $\pm$ 0.53	92.42 $\pm$ 1.29 35.93 $\pm$ 2.73	90.34 $\pm$ 0.32 21.34 $\pm$ 0.31	88.09 $\pm$ 0.4 14.14 $\pm$ 0.32
MultiGrad	96.01 $\pm$ 0.08 86.4 $\pm$ 0.08	94.71 $\pm$ 0.17 71.98 $\pm$ 0.26	95.75 $\pm$ 0.58 28.1 $\pm$ 18.85	94.86 $\pm$ 0.97 11.49 $\pm$ 16.19	94.7 $\pm$ 0.12 0.0 $\pm$ 0.0	94.48 $\pm$ 0.19 0.0 $\pm$ 0.0
PGD-2	96.03 $\pm$ 0.14 86.72 $\pm$ 0.06	94.66 $\pm$ 0.1 73.29 $\pm$ 0.29	93.77 $\pm$ 0.61 60.53 $\pm$ 0.73	94.63 $\pm$ 1.29 20.68 $\pm$ 18.56	84.09 $\pm$ 14.99 0.41 $\pm$ 0.29	94.16 $\pm$ 0.54 0.02 $\pm$ 0.03
PGD-10	95.92 $\pm$ 0.08 86.94 $\pm$ 0.14	94.37 $\pm$ 0.13 74.76 $\pm$ 0.19	92.46 $\pm$ 0.25 63.9 $\pm$ 0.48	89.67 $\pm$ 0.34 53.95 $\pm$ 0.55	85.75 $\pm$ 0.65 44.91 $\pm$ 0.45	80.08 $\pm$ 0.93 37.65 $\pm$ 0.53

WideResNet28-10 – CIFAR-10 Dataset

	$\epsilon = 2/255$	$\epsilon = 4/255$	$\epsilon = 6/255$	$\epsilon = 8/255$	$\epsilon = 10/255$	$\epsilon = 12/255$	$\epsilon = 14/255$	$\epsilon = 16/255$
N-FGSM	92.51 $\pm$ 0.11 81.43 $\pm$ 0.3	89.65 $\pm$ 0.09 69.11 $\pm$ 0.24	85.8 $\pm$ 0.23 58.29 $\pm$ 0.14	81.59 $\pm$ 0.32 49.53 $\pm$ 0.25	76.92 $\pm$ 0.04 42.37 $\pm$ 0.36	72.13 $\pm$ 0.15 36.85 $\pm$ 0.2	67.82 $\pm$ 0.43 31.66 $\pm$ 0.6	56.73 $\pm$ 0.42 25.01 $\pm$ 0.23
Grad Align	92.59 $\pm$ 0.05 81.33 $\pm$ 0.4	89.95 $\pm$ 0.3 69.81 $\pm$ 0.47	86.98 $\pm$ 0.06 59.0 $\pm$ 0.13	83.19 $\pm$ 0.26 50.0 $\pm$ 0.05	79.35 $\pm$ 0.26 41.48 $\pm$ 0.51	73.79 $\pm$ 0.72 35.06 $\pm$ 0.74	66.38 $\pm$ 0.53 30.83 $\pm$ 0.39	57.75 $\pm$ 0.75 26.26 $\pm$ 0.13
FGSM	92.65 $\pm$ 0.17 81.38 $\pm$ 0.22	90.06 $\pm$ 0.18 69.59 $\pm$ 0.25	87.99 $\pm$ 1.3 38.69 $\pm$ 26.54	86.46 $\pm$ 0.45 0.0 $\pm$ 0.0	82.67 $\pm$ 1.78 0.0 $\pm$ 0.0	80.14 $\pm$ 1.2 0.0 $\pm$ 0.0	74.54 $\pm$ 4.01 0.0 $\pm$ 0.0	71.56 $\pm$ 3.78 0.0 $\pm$ 0.0
RS-FGSM	92.85 $\pm$ 0.1 80.9 $\pm$ 0.13	90.73 $\pm$ 0.2 68.23 $\pm$ 0.17	88.24 $\pm$ 0.19 57.21 $\pm$ 0.17	83.64 $\pm$ 1.74 0.0 $\pm$ 0.0	82.1 $\pm$ 1.45 0.0 $\pm$ 0.0	78.62 $\pm$ 0.7 0.0 $\pm$ 0.0	73.25 $\pm$ 8.16 0.0 $\pm$ 0.0	68.64 $\pm$ 4.3 0.0 $\pm$ 0.0
RandAlpha	93.37 $\pm$ 0.22 77.67 $\pm$ 0.66	92.17 $\pm$ 0.21 63.73 $\pm$ 0.31	90.71 $\pm$ 0.14 50.4 $\pm$ 0.14	89.16 $\pm$ 0.19 39.37 $\pm$ 0.42	87.44 $\pm$ 0.31 30.13 $\pm$ 0.9	85.69 $\pm$ 0.28 23.13 $\pm$ 0.33	83.98 $\pm$ 0.24 16.0 $\pm$ 0.22	83.23 $\pm$ 0.46 8.47 $\pm$ 0.66
AT Free	90.66 $\pm$ 0.25 77.0 $\pm$ 0.27	88.37 $\pm$ 0.15 64.25 $\pm$ 0.33	86.11 $\pm$ 0.29 53.76 $\pm$ 0.48	83.5 $\pm$ 0.27 44.85 $\pm$ 0.39	80.52 $\pm$ 0.32 31.87 $\pm$ 5.53	83.59 $\pm$ 1.35 0.0 $\pm$ 0.0	39.58 $\pm$ 15.8 0.0 $\pm$ 0.0	42.59 $\pm$ 27.96 0.0 $\pm$ 0.0
ZeroGrad	92.62 $\pm$ 0.11 81.42 $\pm$ 0.28	90.17 $\pm$ 0.05 69.28 $\pm$ 0.29	86.98 $\pm$ 0.28 58.4 $\pm$ 0.14	84.25 $\pm$ 0.28 48.29 $\pm$ 0.16	81.72 $\pm$ 0.29 36.08 $\pm$ 0.29	79.24 $\pm$ 0.82 28.24 $\pm$ 1.79	78.14 $\pm$ 0.46 18.54 $\pm$ 0.31	75.34 $\pm$ 0.12 14.6 $\pm$ 0.12
MultiGrad	92.64 $\pm$ 0.1 81.19 $\pm$ 0.28	90.18 $\pm$ 0.13 69.3 $\pm$ 0.2	87.11 $\pm$ 0.36 57.98 $\pm$ 0.08	83.87 $\pm$ 0.46 48.74 $\pm$ 0.09	80.89 $\pm$ 0.14 41.22 $\pm$ 0.57	82.88 $\pm$ 2.85 4.46 $\pm$ 6.09	86.6 $\pm$ 1.52 0.0 $\pm$ 0.0	85.46 $\pm$ 3.73 0.0 $\pm$ 0.0
PGD-2	92.69 $\pm$ 0.14 81.54 $\pm$ 0.18	90.18 $\pm$ 0.19 69.87 $\pm$ 0.26	86.87 $\pm$ 0.18 59.4 $\pm$ 0.19	83.31 $\pm$ 0.16 50.88 $\pm$ 0.16	79.61 $\pm$ 0.47 43.94 $\pm$ 0.24	75.81 $\pm$ 0.24 37.77 $\pm$ 0.57	71.41 $\pm$ 1.38 21.06 $\pm$ 13.39	67.2 $\pm$ 14.94 0.0 $\pm$ 0.0
PGD-10	92.24 $\pm$ 0.31 81.18 $\pm$ 0.57	89.65 $\pm$ 0.33 70.34 $\pm$ 0.26	86.91 $\pm$ 0.51 60.59 $\pm$ 0.21	82.82 $\pm$ 0.7 52.58 $\pm$ 0.2	78.63 $\pm$ 0.66 45.92 $\pm$ 0.38	74.0 $\pm$ 0.67 40.44 $\pm$ 0.17	68.6 $\pm$ 0.58 35.98 $\pm$ 0.56	64.17 $\pm$ 0.72 32.5 $\pm$ 0.61

WideResNet28-10 – CIFAR-100 Dataset

	$\epsilon = 2/255$	$\epsilon = 5/255$	$\epsilon = 6/255$	$\epsilon = 8/255$	$\epsilon = 10/255$	$\epsilon = 12/255$	$\epsilon = 14/255$	$\epsilon = 16/255$
N-FGSM	71.56 $\pm$ 0.13 52.23 $\pm$ 0.33	66.49 $\pm$ 0.46 39.93 $\pm$ 0.37	61.38 $\pm$ 0.68 30.97 $\pm$ 0.21	56.23 $\pm$ 0.59 26.77 $\pm$ 0.65	51.54 $\pm$ 0.63 23.03 $\pm$ 0.54	46.43 $\pm$ 0.61 19.3 $\pm$ 0.59	42.11 $\pm$ 0.32 16.67 $\pm$ 0.4	38.34 $\pm$ 0.47 14.27 $\pm$ 0.33
Grad Align	71.68 $\pm$ 0.33 51.5 $\pm$ 0.45	67.09 $\pm$ 0.19 39.9 $\pm$ 0.42	62.86 $\pm$ 0.1 32.0 $\pm$ 0.22	58.55 $\pm$ 0.41 26.9 $\pm$ 0.62	53.85 $\pm$ 0.73 22.63 $\pm$ 0.62	46.94 $\pm$ 0.86 19.9 $\pm$ 0.65	42.63 $\pm$ 0.5 16.93 $\pm$ 0.12	36.17 $\pm$ 0.45 14.03 $\pm$ 0.24
FGSM	71.92 $\pm$ 0.33 52.83 $\pm$ 0.37	67.34 $\pm$ 0.36 39.83 $\pm$ 0.31	64.72 $\pm$ 1.12 0.0 $\pm$ 0.0	56.87 $\pm$ 1.24 0.03 $\pm$ 0.05	52.31 $\pm$ 2.11 0.0 $\pm$ 0.0	48.99 $\pm$ 1.17 0.0 $\pm$ 0.0	44.27 $\pm$ 1.4 0.0 $\pm$ 0.0	42.05 $\pm$ 1.03 0.0 $\pm$ 0.0
RS-FGSM	72.65 $\pm$ 0.28 51.63 $\pm$ 0.52	68.26 $\pm$ 0.2 39.57 $\pm$ 0.09	65.58 $\pm$ 0.69 26.63 $\pm$ 2.8	54.25 $\pm$ 5.85 0.0 $\pm$ 0.0	46.08 $\pm$ 4.87 0.0 $\pm$ 0.0	35.84 $\pm$ 0.17 0.0 $\pm$ 0.0	24.4 $\pm$ 1.25 0.0 $\pm$ 0.0	21.37 $\pm$ 5.04 0.0 $\pm$ 0.0
RandAlpha	73.9 $\pm$ 0.15 49.13 $\pm$ 0.91	71.17 $\pm$ 0.12 34.3 $\pm$ 0.54	68.65 $\pm$ 0.22 25.5 $\pm$ 0.33	66.42 $\pm$ 0.13 20.27 $\pm$ 0.98	64.05 $\pm$ 0.5 16.3 $\pm$ 0.14	61.99 $\pm$ 0.6 12.4 $\pm$ 0.29	59.74 $\pm$ 0.57 6.93 $\pm$ 0.19	58.9 $\pm$ 0.78 3.63 $\pm$ 0.12
AT Free	67.62 $\pm$ 0.24 48.07 $\pm$ 0.31	63.27 $\pm$ 0.72 37.93 $\pm$ 0.69	59.53 $\pm$ 0.31 29.7 $\pm$ 0.51	55.77 $\pm$ 0.28 24.43 $\pm$ 0.37	47.02 $\pm$ 3.83 3.23 $\pm$ 4.43	33.52 $\pm$ 9.24 0.0 $\pm$ 0.0	7.87 $\pm$ 1.78 0.0 $\pm$ 0.0	20.92 $\pm$ 21.48 0.0 $\pm$ 0.0
ZeroGrad	71.68 $\pm$ 0.07 52.63 $\pm$ 0.61	67.2 $\pm$ 0.14 39.57 $\pm$ 0.33	63.69 $\pm$ 0.14 30.27 $\pm$ 0.54	60.77 $\pm$ 0.26 23.7 $\pm$ 0.08	61.05 $\pm$ 0.38 15.1 $\pm$ 0.49	58.39 $\pm$ 0.16 11.13 $\pm$ 0.68	56.19 $\pm$ 0.11 8.8 $\pm$ 0.36	56.38 $\pm$ 0.18 4.9 $\pm$ 0.36
MultiGrad	71.8 $\pm$ 0.15 51.9 $\pm$ 0.29	67.73 $\pm$ 0.48 39.7 $\pm$ 0.37	63.24 $\pm$ 0.33 31.5 $\pm$ 0.62	60.05 $\pm$ 0.79 26.03 $\pm$ 0.09	56.39 $\pm$ 0.49 20.8 $\pm$ 0.29	56.79 $\pm$ 8.27 0.0 $\pm$ 0.0	59.8 $\pm$ 3.77 0.0 $\pm$ 0.0	52.96 $\pm$ 5.58 0.0 $\pm$ 0.0
PGD-2	71.62 $\pm$ 0.15 51.73 $\pm$ 0.48	67.25 $\pm$ 0.43 40.27 $\pm$ 0.7	63.18 $\pm$ 0.36 32.23 $\pm$ 0.19	59.02 $\pm$ 0.4 27.13 $\pm$ 0.37	54.47 $\pm$ 0.45 23.43 $\pm$ 0.31	50.91 $\pm$ 0.35 20.23 $\pm$ 0.39	41.03 $\pm$ 3.18 0.03 $\pm$ 0.05	40.13 $\pm$ 3.66 0.0 $\pm$ 0.0
PGD-10	71.11 $\pm$ 0.62 52.5 $\pm$ 0.59	66.9 $\pm$ 0.57 40.73 $\pm$ 0.56	62.05 $\pm$ 0.47 32.8 $\pm$ 0.29	57.64 $\pm$ 0.81 27.97 $\pm$ 0.59	52.84 $\pm$ 0.88 24.7 $\pm$ 0.36	48.14 $\pm$ 0.73 21.8 $\pm$ 0.57	43.14 $\pm$ 0.87 18.87 $\pm$ 0.6	39.2 $\pm$ 0.62 16.8 $\pm$ 0.57

WideResNet28-10 – SVHN Dataset

	$\epsilon = 2/255$	$\epsilon = 4/255$	$\epsilon = 6/255$	$\epsilon = 8/255$	$\epsilon = 10/255$	$\epsilon = 12/255$
N-FGSM	95.64 $\pm$ 0.09 84.1 $\pm$ 0.73	93.66 $\pm$ 0.41 66.9 $\pm$ 0.86	91.77 $\pm$ 0.42 53.0 $\pm$ 0.36	88.89 $\pm$ 0.58 40.5 $\pm$ 0.37	88.07 $\pm$ 0.59 30.47 $\pm$ 0.76	87.52 $\pm$ 0.49 22.43 $\pm$ 0.53
Grad Align	95.41 $\pm$ 0.06 84.57 $\pm$ 0.56	93.9 $\pm$ 0.48 67.27 $\pm$ 0.54	68.36 $\pm$ 34.49 39.53 $\pm$ 14.89	42.62 $\pm$ 32.73 24.7 $\pm$ 9.34	19.3 $\pm$ 0.21 17.63 $\pm$ 0.62	19.53 $\pm$ 0.08 18.13 $\pm$ 0.52
FGSM	95.83 $\pm$ 0.1 85.03 $\pm$ 0.37	95.0 $\pm$ 0.24 31.53 $\pm$ 6.57	94.23 $\pm$ 0.79 1.7 $\pm$ 1.36	91.11 $\pm$ 1.36 0.13 $\pm$ 0.19	88.83 $\pm$ 1.71 0.0 $\pm$ 0.0	86.74 $\pm$ 0.7 0.0 $\pm$ 0.0
RS-FGSM	95.81 $\pm$ 0.25 83.8 $\pm$ 0.43	94.53 $\pm$ 0.4 66.67 $\pm$ 0.65	95.23 $\pm$ 0.26 0.53 $\pm$ 0.26	94.68 $\pm$ 0.62 0.0 $\pm$ 0.0	93.9 $\pm$ 0.52 0.0 $\pm$ 0.0	91.64 $\pm$ 2.98 0.0 $\pm$ 0.0
RandAlpha	96.02 $\pm$ 0.23 82.5 $\pm$ 0.45	95.47 $\pm$ 0.18 63.33 $\pm$ 0.53	94.69 $\pm$ 0.26 47.7 $\pm$ 0.99	93.72 $\pm$ 0.44 35.73 $\pm$ 0.34	93.08 $\pm$ 1.45 23.17 $\pm$ 1.97	93.96 $\pm$ 0.68 11.1 $\pm$ 3.05
AT Free	94.85 $\pm$ 0.39 83.13 $\pm$ 0.17	92.95 $\pm$ 0.65 68.67 $\pm$ 0.53	91.62 $\pm$ 1.93 54.93 $\pm$ 2.58	93.74 $\pm$ 0.69 0.03 $\pm$ 0.05	92.47 $\pm$ 0.97 0.0 $\pm$ 0.0	90.5 $\pm$ 1.41 0.0 $\pm$ 0.0
ZeroGrad	95.78 $\pm$ 0.21 84.47 $\pm$ 0.83	94.06 $\pm$ 0.52 66.1 $\pm$ 0.37	92.13 $\pm$ 0.98 47.3 $\pm$ 0.62	91.04 $\pm$ 0.4 29.33 $\pm$ 0.56	88.85 $\pm$ 0.92 20.77 $\pm$ 0.63	89.8 $\pm$ 1.36 9.33 $\pm$ 0.76
MultiGrad	95.63 $\pm$ 0.16 84.37 $\pm$ 0.59	94.27 $\pm$ 0.38 67.27 $\pm$ 0.31	93.64 $\pm$ 1.21 50.1 $\pm$ 0.9	94.83 $\pm$ 1.55 1.77 $\pm$ 1.72	95.26 $\pm$ 0.34 0.0 $\pm$ 0.0	95.22 $\pm$ 0.15 0.0 $\pm$ 0.0
PGD-2	95.88 $\pm$ 0.35 86.25 $\pm$ 0.7	94.66 $\pm$ 0.1 73.29 $\pm$ 0.25	93.77 $\pm$ 0.61 60.53 $\pm$ 0.72	92.99 $\pm$ 1.11 40.77 $\pm$ 4.39	88.81 $\pm$ 0.93 34.33 $\pm$ 2.76	83.17 $\pm$ 4.78 26.8 $\pm$ 3.31
PGD-10	95.92 $\pm$ 0.08 86.94 $\pm$ 0.13	94.36 $\pm$ 0.13 74.46 $\pm$ 0.54	92.46 $\pm$ 0.25 63.87 $\pm$ 0.49	89.67 $\pm$ 0.34 53.95 $\pm$ 0.55	85.98 $\pm$ 0.59 44.59 $\pm$ 0.14	80.08 $\pm$ 0.93 37.64 $\pm$ 0.49

C

Appendix of Reliability in Semantic Segmentation: Are We on the Right Track?

C.1 Ablation of calibration metrics

In the main paper we present calibration results computing the Expected Calibration Error with equally spaced bins, however, alternative calibration metrics have been suggested. In Figure C.1 we compare the results obtained with: ECE with equally spaced bins (ECE), ECE with equally populated bins (Ada ECE) and the Kolmogorov-Smirnov Error (KS Error). For further details see Section 5.3.3. We observe that the three different metrics yield almost identical results.

C.2 Ablation of number of pixels for calibration

As discussed in Section 5.3.3, in segmentation, the number of samples to be taken into account for calibration scales with the number of pixels in an image. In order to be more cost-effective when testing different calibration metrics and strategies, we use a random subset of pixels within each image rather than the full image. In Figure C.2, we ablate the evolution of the different calibration metrics as we vary

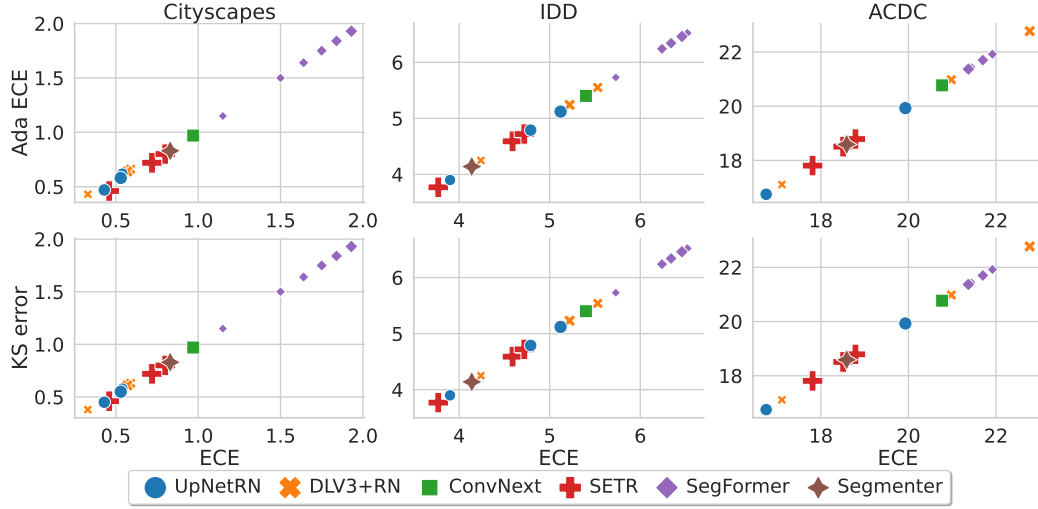


Figure C.1: Comparison of calibration error metrics (↓) Calibration error for different datasets and networks computed with different metrics. All different metrics yield very similar result.

the number of sampled pixels. We can see that from 10k datapoints on, the metrics stabilizes; therefore, we chose to use 20k randomly sampled pixels per image for our experiments.

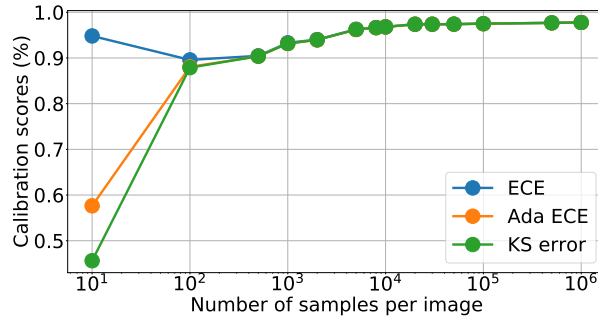


Figure C.2: Pixels-per-image ablation. Evolution of calibration metrics as we vary the number of pixels sampled at random from each image (as opposed to the full image). We observe that when sampling more than 10k pixels all calibration metrics are very similar and the calibration error remains stable. We use 20k random samples in our experiments.

C.3 Ablation of confidence score: max probability vs entropy

Misclassification detection and OOD detection both rely on a metric to evaluate how confident a model is on its predictions. The most straightforward metric would be the pseudo-probability of the predicted class (i.e. the max probability). If the probability is high it is reasonable to assume that the network is confident (this is precisely what we want to impose in the calibration task). Other metrics which involve all the logits have been suggested, negative entropy being the most popular. In Figure C.3 we compare the results obtained with probability and entropy as confidence metrics and observe that there is not a significant difference between the two. Therefore, the simpler confidence metric based on the predicted class probability is used for other experiments by default.

C.4 Ablation number of clusters

One of the main hyperparameters in Gong *et al.* Gong et al. (2021) is the number of clusters. In Figure C.4, we ablate the number of clusters for different test datasets (columns) and calibration datasets (rows). Although not all networks evolve in the same way, we observe that after 16 clusters, performance is more or less stable.

C.5 Visualization of cluster samples

In Section 5.4.3 we observe that adaptive temperature scaling via clustering does not significantly improve calibration under distribution shift – especially when the shift is strong. The method by Gong *et al.* Gong et al. (2021) makes the implicit assumption that the different domains captured in the clusters during calibration will be representative of the domains encountered at test time. In order to have a better intuition, we visualize a few samples randomly picked from each cluster.

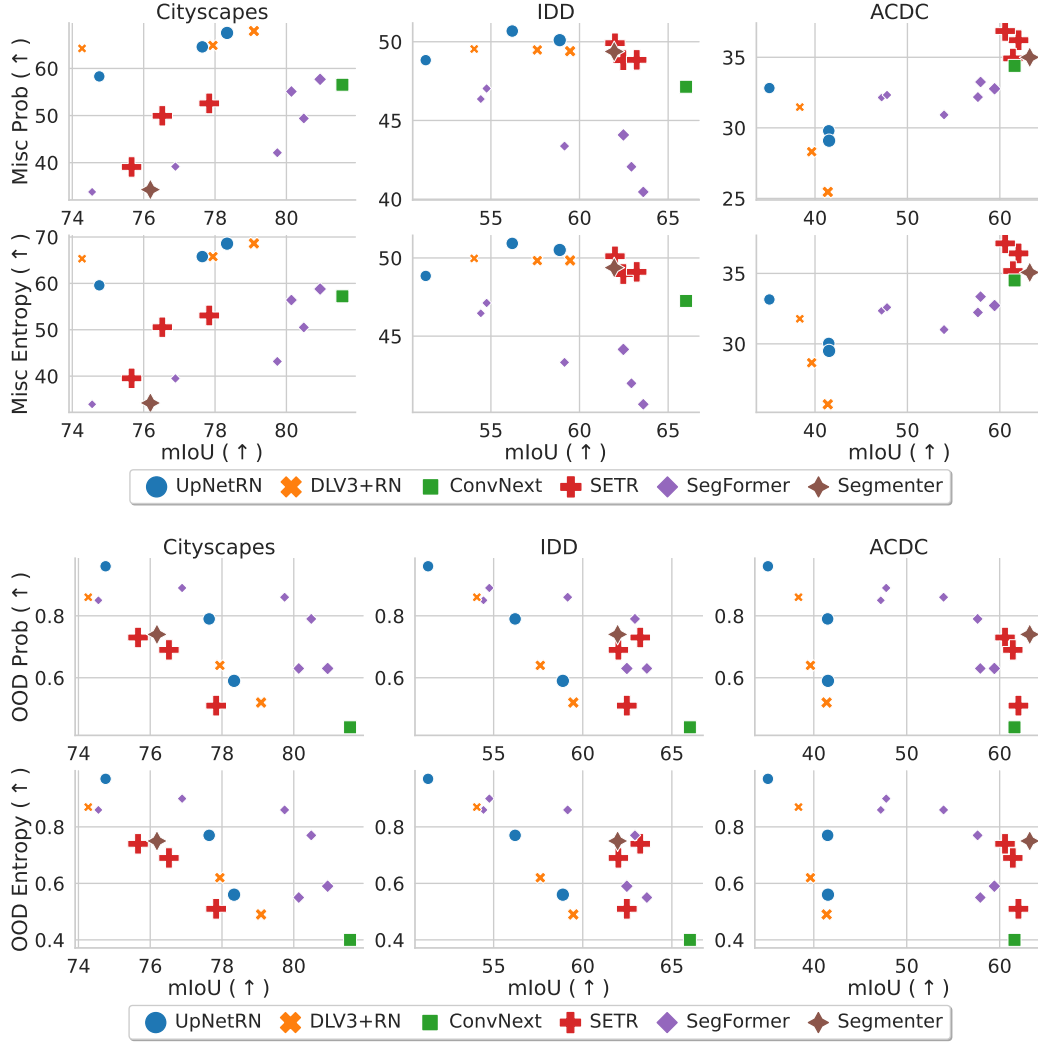


Figure C.3: Ablation of confidence score: max probability vs. entropy Comparison of misclassification (top) and OOD (bottom) detection when using probability or negative entropy as confidence metrics. We observe that there is no significant difference between the two metrics, therefore we use the simpler probability as the default.

We show images from both the calibration set (used to compute the clusters and calibrate the models) and the test set (used to evaluate the calibration error). In our visualizations, the test set comprises images of the three datasets (CS, IDD and ACDC), while the calibration set changes for each Figure. In Figures C.5 to C.7, respectively, we show representatives from clusters in *Clust All* (all datasets used during calibration), *Clust CS* (CS images used) and *Clust CS aug* (augmented CS images used). Qualitatively, when all datasets are used for calibration, the cluster assignments appear quite reasonable (*e.g.* night ACDC images are assigned to night

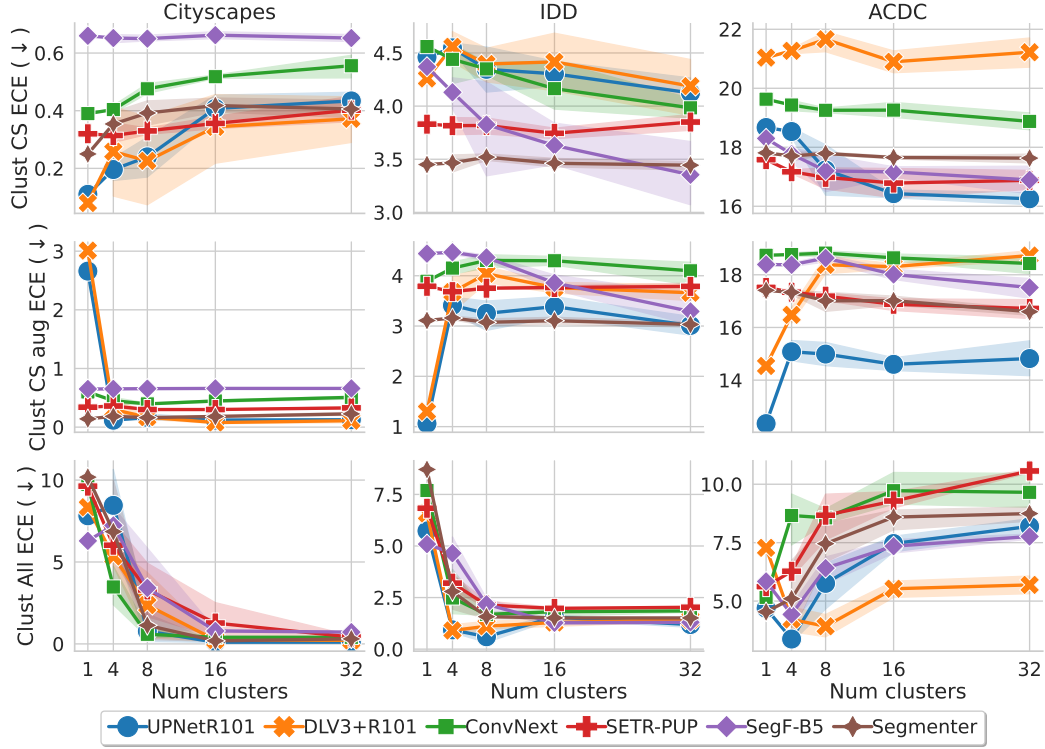


Figure C.4: Ablation number of clusters. ECE (↓) for different models and datasets as we vary the number of clusters computed for calibration. We find 16 clusters to be relatively stable.

images from calibration). However, when calibrating on CS and CS augmented, we observe that the calibration clusters are not diverse enough for the test images and the cluster assignments do not appear so intuitive.

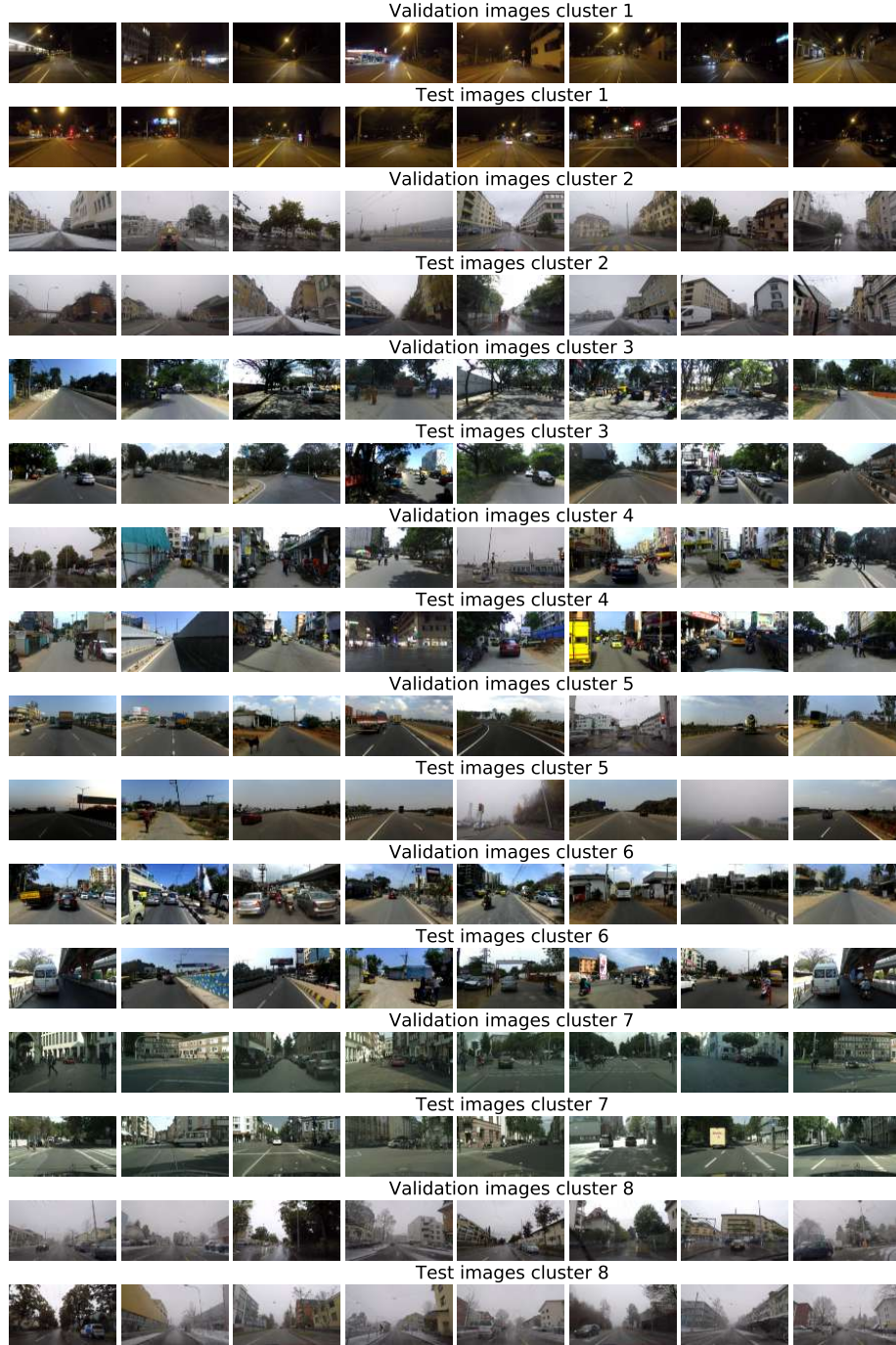


Figure C.5: Visualization of clusters (Clust All). Sample images from clusters computed for Gong et al. (2021). In this case, the calibration set (where clusters are computed) contains imaged from all datasets and we qualitatively observe the cluster assignments to align with human intuition.

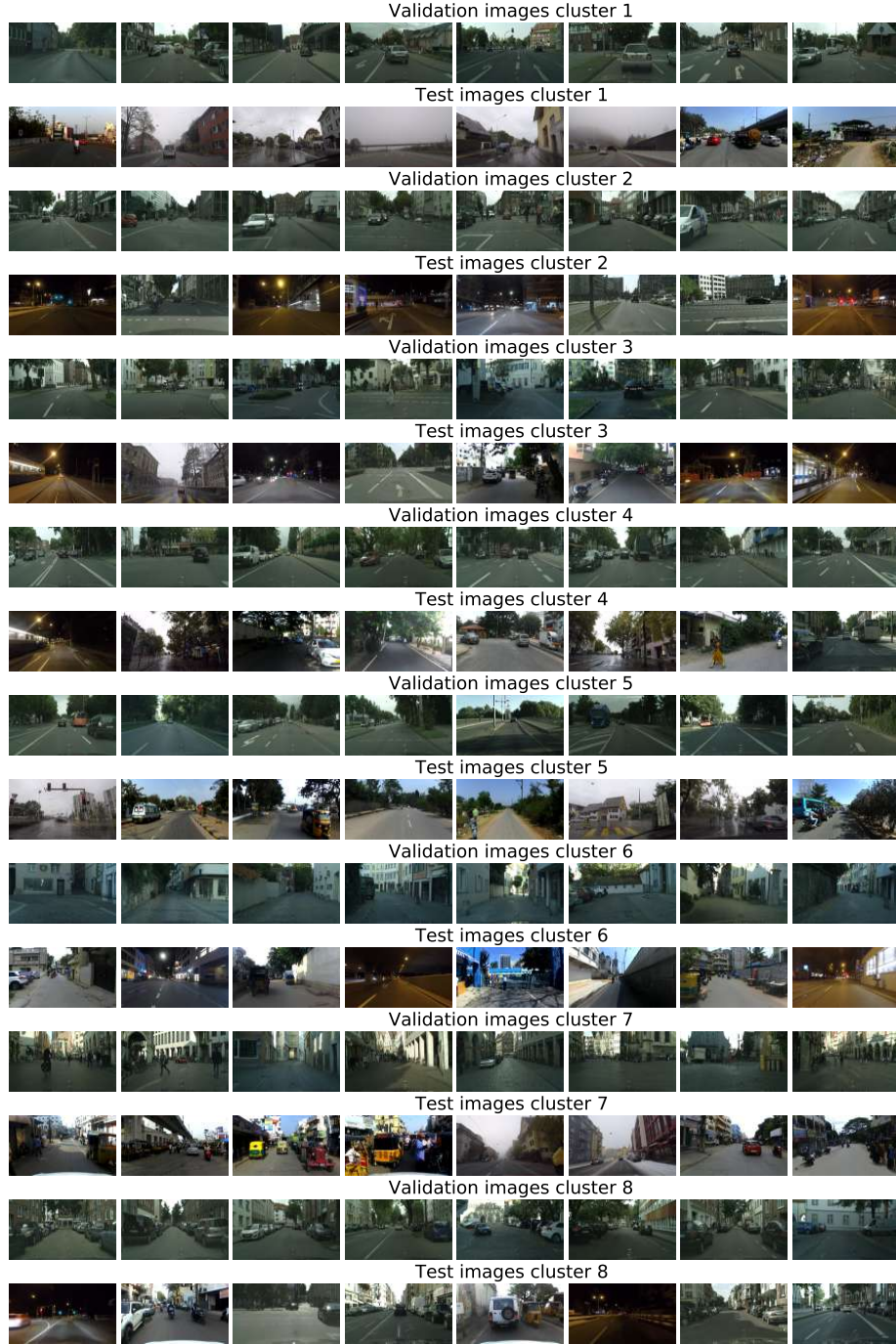


Figure C.6: Visualization of clusters (Clust CS). Sample images from clusters computed for Gong et al. (2021). In this case, the calibration set (where clusters are computed) contains imaged from CS only. We qualitatively observe that the clusters are not representative of the test distribution.

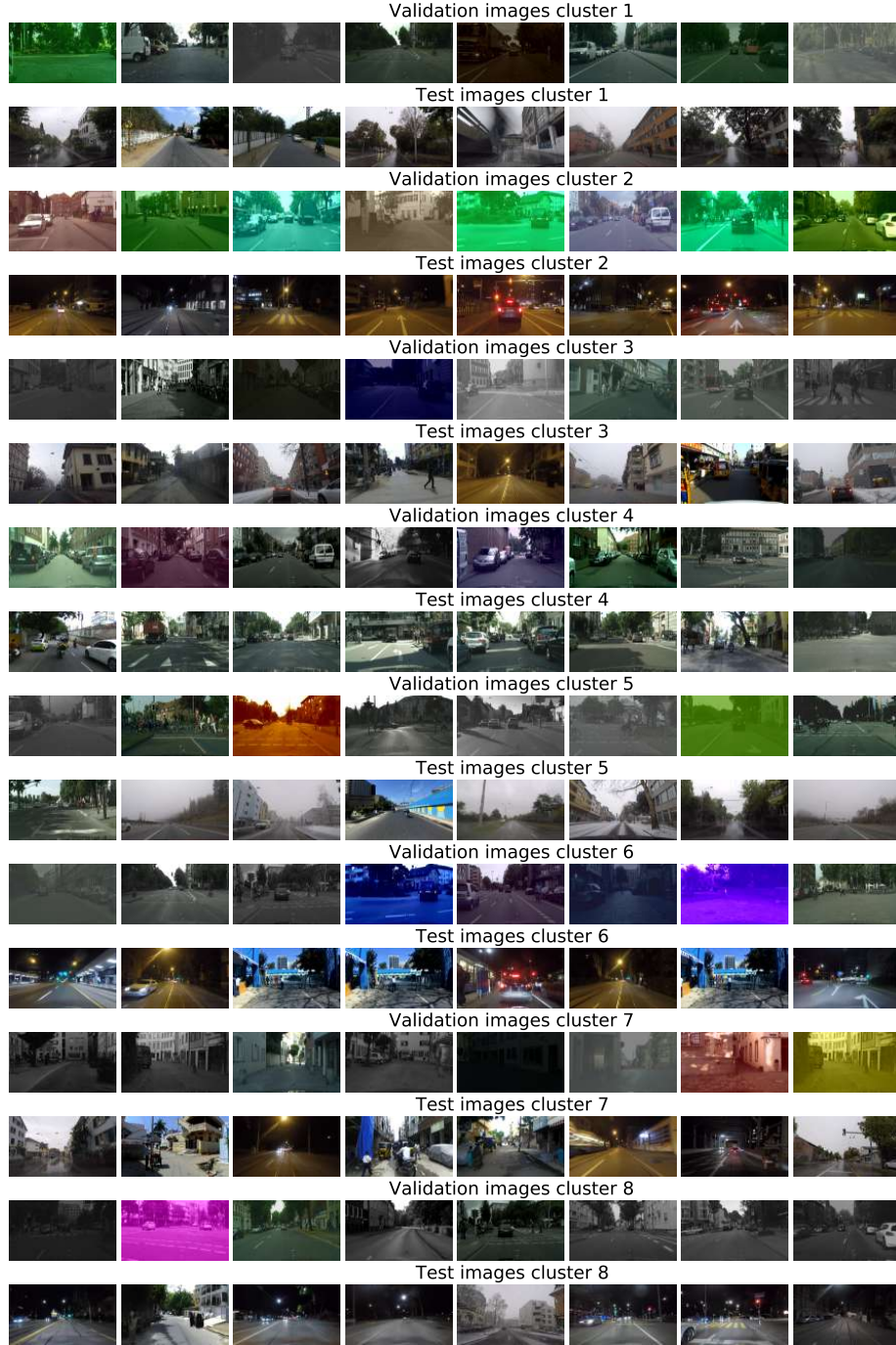


Figure C.7: Visualization of clusters (Clust CS aug). Sample images from clusters computed for Gong et al. (2021). In this case, the calibration set (where clusters are computed) contains imaged from CS augmented only. Even if data augmentations introduce variability to the dataset, it is still not representative of the test distribution.

C.6 Theoretical insights on adaptive temperature via clustering

In Figure 5.4 we observed that even when we evaluate the ECE on the calibration set, the calibration error does not monotonically decrease as we increase the number of clusters. This is somewhat counterintuitive as one would think that, with more clusters, the temperatures can be more fine-grained and evaluating on the calibration set there are no issues with overfitting. However, it is indeed possible since the temperatures of each cluster are optimized independently. In the following we present a theorem and proof to show that decreasing the ECE for several disjoint subsets of images (clusters) independently does not guarantee that the ECE on the union set will decrease.

First, we introduce some preliminaries and notation. Consider a classifier $f : \mathcal{X} \rightarrow \mathcal{Y}$ with $\mathcal{Y} = \{1, 2, 3, \dots, k\}$. We model our classifier as $f = \operatorname{argmax} \hat{p}(\mathbf{y}|\mathbf{x})$ where $\hat{p}(\mathbf{y}|\mathbf{x})$ are the pseudo-probabilities estimated by the model for each class given the input. We say that a model is calibrated when

$$P(\mathbf{y} \mid \hat{p}(\mathbf{y}|\mathbf{x}) = p) = p, \quad \forall \mathbf{x} \sim \mathcal{D} \quad (\text{C.1})$$

where P is the true probability of the classes and $\mathcal{D}$ the data distribution. However, most works focus on a simplification of this problem where only the probability of the predicted class is taken into account, that is:

$$P(y = \operatorname{argmax} \hat{p}(\mathbf{y}|\mathbf{x}) \mid \max \hat{p}(\mathbf{y}|\mathbf{x}) = p^*) = p^* \quad \forall \mathbf{x}, y \sim \mathcal{D}. \quad (\text{C.2})$$

The most common metric to measure calibration is the *Expected Calibration Error* (ECE). Which looks at the expected difference between the predicted and actual probabilities:

$$\mathbb{E} [\mid p^* - \mathbb{E} [P(\operatorname{argmax} \hat{p}(\mathbf{y}|\mathbf{x}) = y) \mid \max \hat{p}(\mathbf{y}|\mathbf{x}) = p^*] \mid]. \quad (\text{C.3})$$

In order to empirically estimate the ECE, it is standard practice to quantize the output probabilities given by the model and compute the mean probability (confidence) and accuracy in each bin. That is,

$$\widehat{\text{ECE}}_f = \sum_{i=1}^m \frac{\#B_i}{n} |\text{accuracy}(B_i) - \text{confidence}(B_i)| \quad (\text{C.4})$$

where $\#B_i$ denotes the number of elements in the i^{th} bin, m denotes the number of bins and $n = \sum_{i=1}^m \#B_i$ the total number of elements used to estimate the ECE. We also use f to indicate the dependency of the ECE on the classifier.

Consider now, that we split the data into two different sets and we quantize it in bins $\mathcal{B} = \{B_i\}$ and $\mathcal{B}' = \{B'_i\}$ with the same boundaries so that for each pair B_i and B'_i the range of confidence values are the same. Moreover, consider now the respective ECE computed for each subset of data independently — denoted $\widehat{\text{ECE}}_f(\mathcal{B})$ and $\widehat{\text{ECE}}_f(\mathcal{B}')$ — and on the full set of points $\widehat{\text{ECE}}_f(\mathcal{B} + \mathcal{B}')$ where $\mathcal{B} + \mathcal{B}'$ is an abuse of notation to indicate the union of elements in the bins for each index i .

Theorem C.6.1. *With the notation described above, consider a model f_{oracle} such that an “oracle” splits the input according to whether it belongs to $\mathcal{B}$ or $\mathcal{B}'$. Moreover, f_{oracle} uses two calibration strategies (one for $\mathcal{B}$ and one for $\mathcal{B}'$) in a way that it improves its ECE on each subset $\mathcal{B}$, $\mathcal{B}'$ individually compared to some baseline model f (e.g. by means of temperature scaling with a different temperature for each subset). This does not necessarily imply that the oracle model (f_{oracle}) will be better calibrated on the full set of points $\mathcal{B} + \mathcal{B}'$ than the baseline model f . That is, given that:*

$$(a) \quad \widehat{\text{ECE}}_{f_{\text{oracle}}}(\mathcal{B}) \leq \widehat{\text{ECE}}_f(\mathcal{B}) \quad \text{and} \quad \widehat{\text{ECE}}_{f_{\text{oracle}}}(\mathcal{B}') \leq \widehat{\text{ECE}}_f(\mathcal{B}')$$

Then, condition (a) is not sufficient to claim:

$$(b) \quad \widehat{\text{ECE}}_{f_{\text{oracle}}}(\mathcal{B} + \mathcal{B}') \leq \widehat{\text{ECE}}_f(\mathcal{B} + \mathcal{B}')$$

Proof. In order to proof the theorem we will construct a counter-example where condition (a) is satisfied but condition (b) is not. Consider the accuracy and confidence for the full set of points in a given bin:

$$\begin{aligned}\text{acc}(B_i + B'_i) &= \frac{\#B_i \text{acc}(B_i) + \#B'_i \text{acc}(B'_i)}{\#B_i + \#B'_i} \\ \text{conf}(B_i + B'_i) &= \frac{\#B_i \text{conf}(B_i) + \#B'_i \text{conf}(B'_i)}{\#B_i + \#B'_i}\end{aligned}$$

Then the ECE of the full set of points will be:

$$\begin{aligned}\widehat{\text{ECE}}_f(\mathcal{B} + \mathcal{B}') &= \sum_{i=1}^m \frac{\#B_i + \#B'_i}{n + n'} |\text{acc}_f(B_i + B'_i) - \text{conf}_f(B_i + B'_i)| \\ &= \sum_{i=1}^m \frac{1}{n + n'} |\#B_i(\text{acc}_f(B_i) - \text{conf}_f(B_i)) + \#B'_i(\text{acc}_f(B'_i) - \text{conf}_f(B'_i))|.\end{aligned}$$

To simplify the notation, let us define $r_i(f) = \text{acc}_f(B_i) - \text{conf}_f(B_i)$ and similarly for $r'_i(f)$. Then, we can write:

$$\begin{aligned}\widehat{\text{ECE}}_f(\mathcal{B} + \mathcal{B}') &= \sum_{i=1}^m \frac{1}{n + n'} |\#B_i r_i(f) + \#B'_i r'_i(f)|. \\ \widehat{\text{ECE}}_f(\mathcal{B}) &= \sum_{i=1}^m \frac{\#B_i}{n} |r_i(f)|. \\ \widehat{\text{ECE}}_f(\mathcal{B}') &= \sum_{i=1}^m \frac{\#B'_i}{n'} |r'_i(f)|.\end{aligned}$$

Now let us consider a setting where $r_i(f) = r$ and $r_i(f_{\text{oracle}}) = -0.5r$ for some $r \neq 0$ while $r'_i(f) = -r$ and $r'_i(f_{\text{oracle}}) = -0.5r$. Moreover, consider $\#B_i = \#B'_i$ which implies $n = n'$, then this setting would satisfy condition (a) since

$$\begin{aligned}\widehat{\text{ECE}}_f(\mathcal{B}) &= \sum_{i=1}^m \frac{\#B_i}{n} |r| \geq \sum_{i=1}^m \frac{\#B_i}{n} |-0.5r| = \widehat{\text{ECE}}_{f_{\text{oracle}}}(\mathcal{B}) \quad \text{and} \\ \widehat{\text{ECE}}_f(\mathcal{B}') &= \sum_{i=1}^m \frac{\#B'_i}{n'} |-r| \geq \sum_{i=1}^m \frac{\#B'_i}{n'} |-0.5r| = \widehat{\text{ECE}}_{f_{\text{oracle}}}(\mathcal{B}').\end{aligned}$$

However, this same setting would not satisfy condition (b) since

$$\begin{aligned}\widehat{\text{ECE}}_f(\mathcal{B} + \mathcal{B}') &= \sum_{i=1}^m \frac{\#B_i}{2n} |r - r| = 0 \quad \text{and} \\ \widehat{\text{ECE}}_{f_{\text{oracle}}}(\mathcal{B} + \mathcal{B}') &= \sum_{i=1}^m \frac{\#B_i}{2n} |-0.5r - 0.5r| > 0.\end{aligned}$$

Thus, we have showed that condition (a) does not imply (b). $\square$

This result implies that minimizing the ECE for different subsets of the data independently (e.g. each cluster of images) does not necessarily lead to an overall improvement of the ECE. Moreover, we have assumed only two sets of samples without loss of generalization since if (a) implied (b) for an arbitrary number of data splits it would in particular imply it for two. Finally, note that our result is valid for either image classifiers or segmentors. In the first case we would each prediction would be the class of a whole image while in the second case the each pixel in an image would have a different prediction.

C.7 Per-class clustering

In Figure 5.6, we have observed that adaptive temperature via clustering Gong et al. (2021) does not significantly help improving out-of-domain calibration compared to local temperature scaling (LTS) Ding et al. (2021). One important difference between the methods is that LTS computes a temperature for each pixel in the image while clustering is performed at the image level – using a single temperature per image. This motivates us to perform an ablation where, on top of the image level clustering, pixels in a given image are grouped according to their predicted class. Intuitively, we are looking for a temperature for regions in the image that look alike to the network (since they are assigned to the same class). In Figure C.8 we compare standard per-image clustering (top) with the aforementioned per-class clustering (bottom). Note that per-class clustering always groups pixels according to the predicted class, therefore if $k = 1$ then there are 19 clusters (corresponding to the CS classes). Calibration images are from the CS dataset.

Similarly to per-image clustering, increasing the number of clusters does not seem to always help when using per-class clustering. Moreover, we do not find that per-class clustering significantly improves calibration except for SegFormer architecture. We are not stating here that finer-grained clustering may not yield further improvements (and reach similar performance to LTS). However, given that improving ECE in different subdomains independently is not guaranteed to improve overall calibration (see Appendix C.6), perhaps a different approach to finding the temperatures and clusters taking into account both local and global calibration error would be needed.

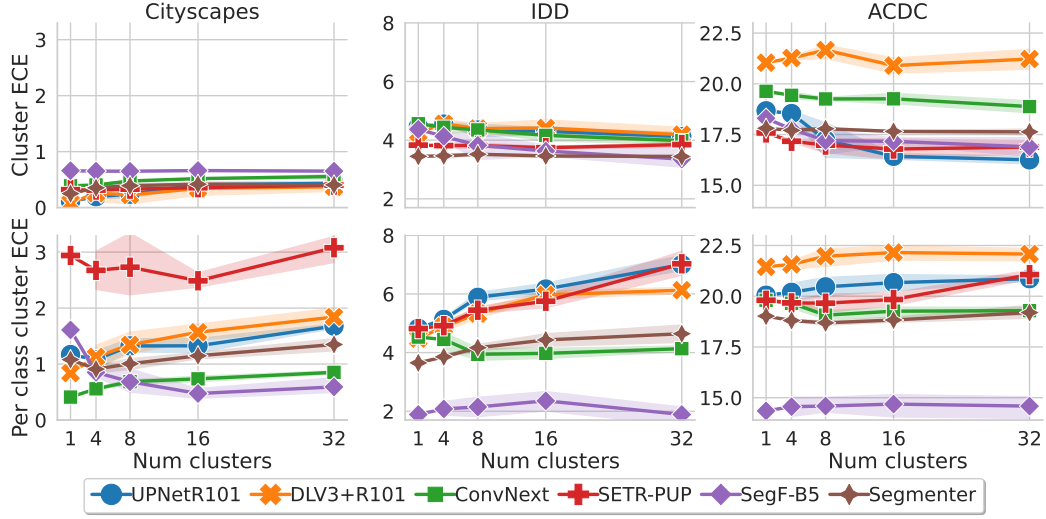


Figure C.8: Per-class clustering ablation. ECE ($\downarrow$) for different models and datasets as we vary the number of clusters computed for calibration. We compare per-image clustering (where all pixels in an image are given the same temperature) *vs.* per-class clustering (where different pixels in an image are given a temperature depending on their predicted class). Overall, we do not observe consistent improvements when using per-class clustering except in SegFormer architecture.

C.8 Ablation LTS: image vs logits

LTS Ding et al. (2021) employs a small-weight calibration network which receives both the image and predicted logits as input and it returns a temperature map to scale the logits (with a different temperature for each pixel in the image). Given its remarkable performance (see Figures 5.5 and 5.6) and, to get further insights into this method, in Figure C.9 we perform an ablation where the calibration network only receives the image or the logits as input. To carry out this experiment, we modify the network in Ding et al. (2021) so that both input branches (logits and image) receive the same input, either both logits or both image. All calibration networks have been trained in CS images only.

Interestingly, we observe that, in distribution (ECE CS), the better performing method for most networks is the LTS variant that uses the logits only. On the other hand, for OOD calibration, the better performing variant in most cases is the one that relies on both logit and image information. Moreover, under strong domain shifts (ECE ACDC), LTS yields better results by using only the image information than by using only the logits. However, this is subject to variability as results vary across different architectures. Further investigations on how logit and image signals are combined may constitute a promising direction to further improve calibration results.

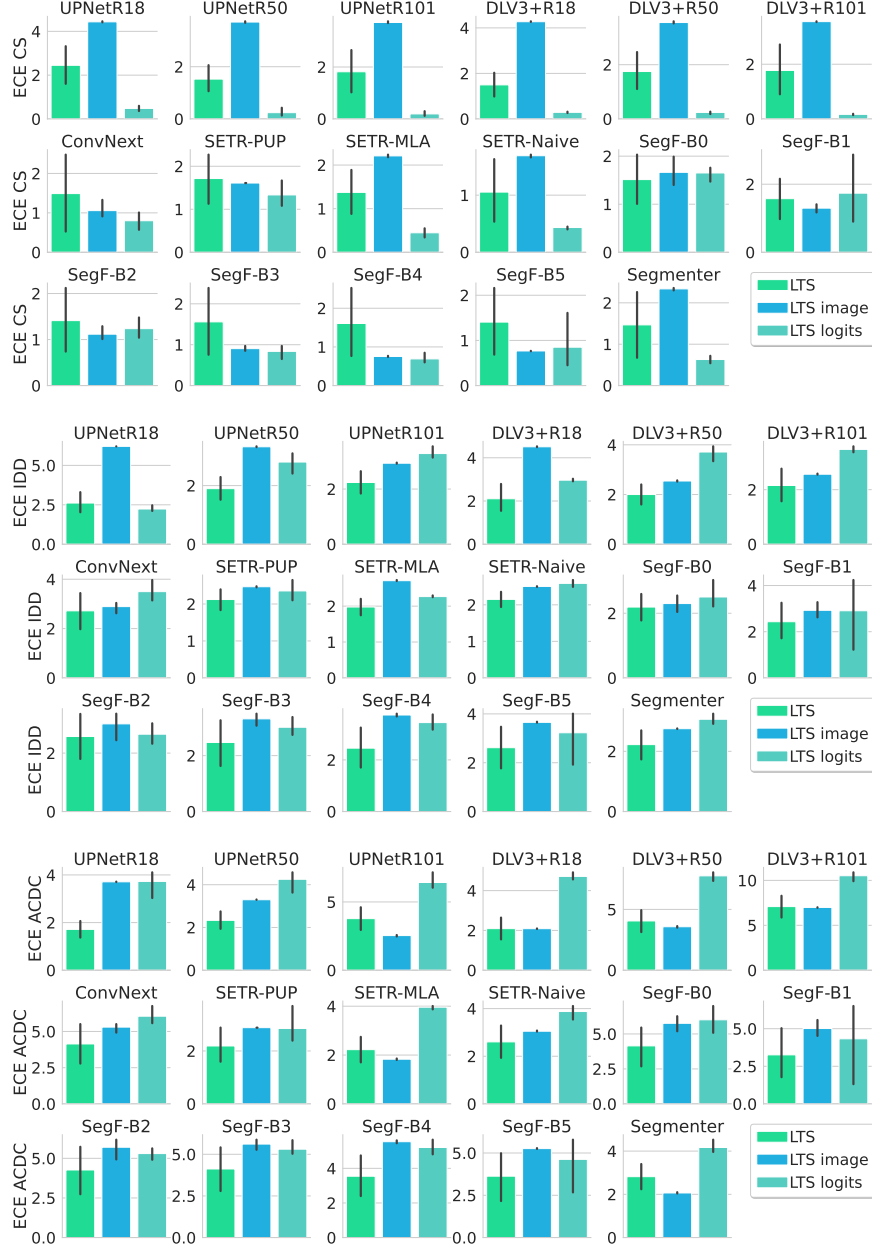


Figure C.9: LTS with image *vs.* logits information ECE ($\downarrow$) after calibration with three LTS variants: the original method combining image and logits (LTS), an LTS version only using logit information (LTS logits) and the complementary version only using image information (LTS image). We observe that in distribution, using only the logits seems to perform better while out of distribution using information from both image and logits works best.

C.9 Ablation of the effect of training settings

Our goal is to assess the reliability of the best available models for segmentation — the ones used by practitioners. Therefore, to test a model at its best, we need to use its most favorable setup (original pre-training, optimizer, etc.), as done in previous work Pinto et al. (2022); Minderer et al. (2021). Nevertheless, in this section we ablate the effects of two different training settings: the dataset used to pre-train the backbone and the number of training iterations. Most recent models were pre-trained on ImageNet 21k, however, ResNet backbones are usually pre-trained on ImageNet 1k. In the top rows of Table C.1 we compare the performance of a BiT-ResNet50 Kolesnikov et al. (2020) and a ConvNext-B Liu et al. (2022) backbones pre-trained with either ImageNet 1k or 21k. We observe that the benefits of a larger pre-training dataset are small in comparison to the gap between architectures.

On the other hand, most models were trained using 80k iterations (as is quite standard for CS dataset, but Segmenter and SegFormer models’ original training schedule uses 160k iterations. In the bottom rows of Table C.1 we compare Segmenter and SegFormer-B5 networks trained with either 160k or 80k iterations, again the differences are minor compared to the gap between architectures.

C.10 Architectures for Universal Image Segmentation

With the appearance of transformers, and in particular motivated by DETR Carion et al. (2020), some architectures have been proposed with the objective to solve the three main Image Segmentation tasks, that is: Semantic Segmentation, Instance Segmentation and Panoptic Segmentation Cheng et al. (2021); Zhang et al. (2021); Cheng et al. (2022). Here we will focus on Mask2Former Cheng et al. (2022) which is based on MaskFormer Cheng et al. (2021) and is the best performing universal architecture to the best of our knowledge. Interestingly, to be able to solve the different

Architecture	mIoU ($\uparrow$)			ECE ($\uparrow$)		
	CS	IDD	ACDC	CS	IDD	ACDC
ConvNext-B (IN-1k)	80.39	64.11	58.77	0.77	5.53	18.95
ConvNext-B (IN-21k)	81.56	65.70	60.59	0.81	5.27	20.07
BiT-RN50 (IN-1k)	76.36	57.46	47.47	0.67	5.18	19.47
BiT-RN50 (IN-21k)	76.49	57.23	46.70	0.65	5.20	19.63
Segmenter (160k)	76.19	61.96	63.24	0.83	4.14	18.59
Segmenter (80k)	76.22	60.74	62.96	0.71	4.28	18.36
SegF-B5 (160k)	80.94	62.47	59.43	1.93	6.46	21.37
SegF-B5 (80k)	80.15	62.95	57.00	1.07	5.32	21.27

Table C.1: Ablations of different training settings. On the top we compare ConvNext and BiT-RN50 architectures with the backbones pre-trained on either ImageNet 1k or 21k datasets. We observe that the benefits of a larger pre-training dataset are small in comparison to the gap between architectures. On the bottom we compare the amount of training iterations, again, we observe that although longer training schedules do improve the performance, changes are also small in comparison to architecture differences.

segmentation tasks jointly, these architectures do not output the standard per-pixel logits when it comes to semantic segmentation. Instead, they predict a set of N object masks (where N is fixed) and the class probabilities for each object. Then, to obtain the per-pixel class probabilities they marginalize over all the possible objects a pixel could belong to, we refer the reader to Cheng et al. (2021, 2022) for further details.

Although this final output can be regarded as per-pixel class probabilities, the way it is obtained differ from the standard *logits + softmax* setting that all calibration methods rely on, therefore it is not straightforward to compare these universal architectures with other models with temperature scaling. Nevertheless, given the good performance and wider applicability of these models, we include them in our study comparing only off-the-shelf performance in terms of mIoU, calibration, OOD detection and misclassification. The best performing model from Cheng et al. (2022) is based on a Swin transformer (Swin Large) Liu et al. (2021), therefore, we also include a Swin transformer model with UpperNet to the comparison for completeness.

C.10.1 Calibration with Mask2Former

In C.10 we present the ECE vs segmentation error ($100 - \text{mIoU}$) for the different models. Interestingly, we observe that Mask2Former seems to be the best-performing model in terms of mIoU in all datasets. In terms of calibration error Mask2Former is poorly calibrated in distribution (CS) but has a milder increase in ECE as the distribution shift becomes stronger.

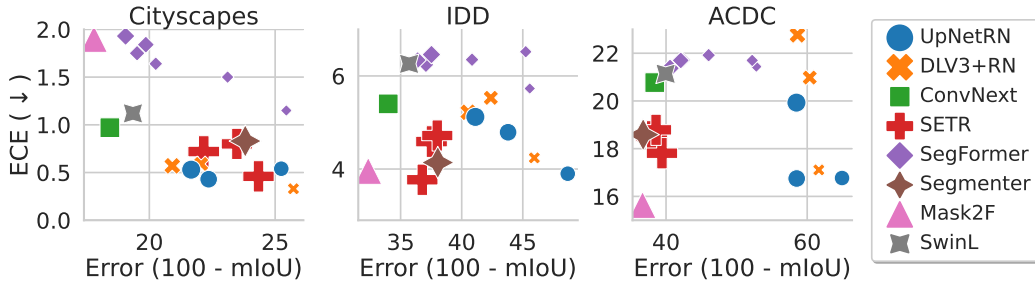


Figure C.10: mIoU error ($\downarrow$) vs. Expected calibration error ($\downarrow$). All models were trained on CS. Markersize proportional to number of parameters. Interestingly, we observe that Mask2Former seems to be the best-performing model in terms of mIoU in all datasets. In terms of calibration error Mask2Former is poorly calibrated in distribution (CS) but has a milder increase in ECE as the distribution shift becomes stronger.

C.10.2 OOD detection with Mask2Former

In C.11 we present the OOD vs mIoU for the different models. Here we observe that Mask2Former and Swin transformer align with the negative trend observed in previous models where models with better mIoU tend to perform worse at OOD detection.

C.10.3 PRR with Mask2Former

In C.12 we present the OOD vs mIoU for the different models. Although Mask2Former is significantly better than other models in terms of misclassification detection in distribution, it seems to perform significantly worse under strong domain shifts (ACDC). This seems to be contrary to the trend followed by other models where robustness seems to be correlated with misclassification under domain shift.

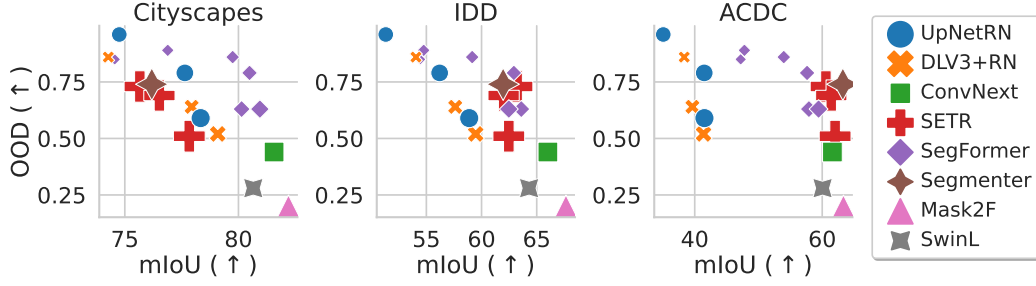


Figure C.11: mIoU ($\uparrow$) vs. OOD - AUROC ($\uparrow$) for different model families. All models trained on CS. Markersize proportional to number of parameters. We observe that Mask2Former and Swin transformer align with the negative trend observed in previous models where models with better mIoU tend to perform worse at OOD detection.

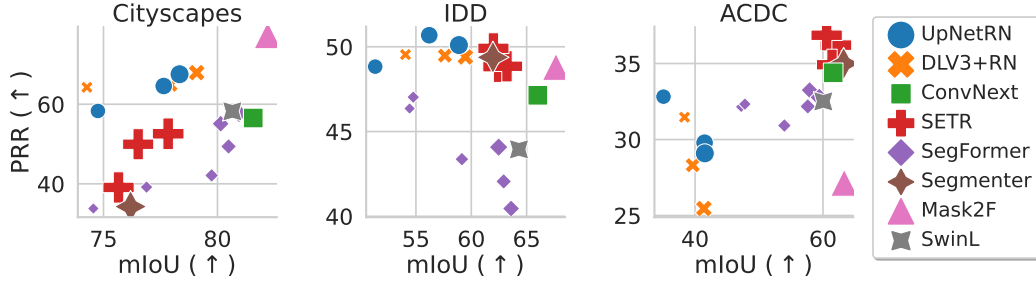


Figure C.12: mIoU ($\uparrow$) vs. prediction rejection ratio ($\uparrow$) for different model families. All models trained on CS. Markersize proportional to number of parameters. Mask2Former is significantly better than other models in terms of misclassification detection in distribution, it seems to perform significantly worse under strong domain shifts (ACDC).

C.11 Visualization of uncertainty and temperature maps

As opposed to classification, semantic segmentation models can assign different confidence to regions of the image. That can allow, among other applications, to detect regions with low confidence that may correspond to novel classes or weird instances of a known class. ACDC shares the same classes as Cityscapes, however, IDD includes some novel classes which are not included in the Cityscapes dataset. In C.13 we illustrate some examples of IDD images with confidence maps before and after LTS, together with the temperature scaling maps predicted by the calibration network. We observe how different OOD classes (*e.g.* autorickshaw, bridge, billboard) or weird instances are highlighted in the calibration maps. In Appendix C.12 we quantify how useful are the calibration maps in order to perform local OOD detection.

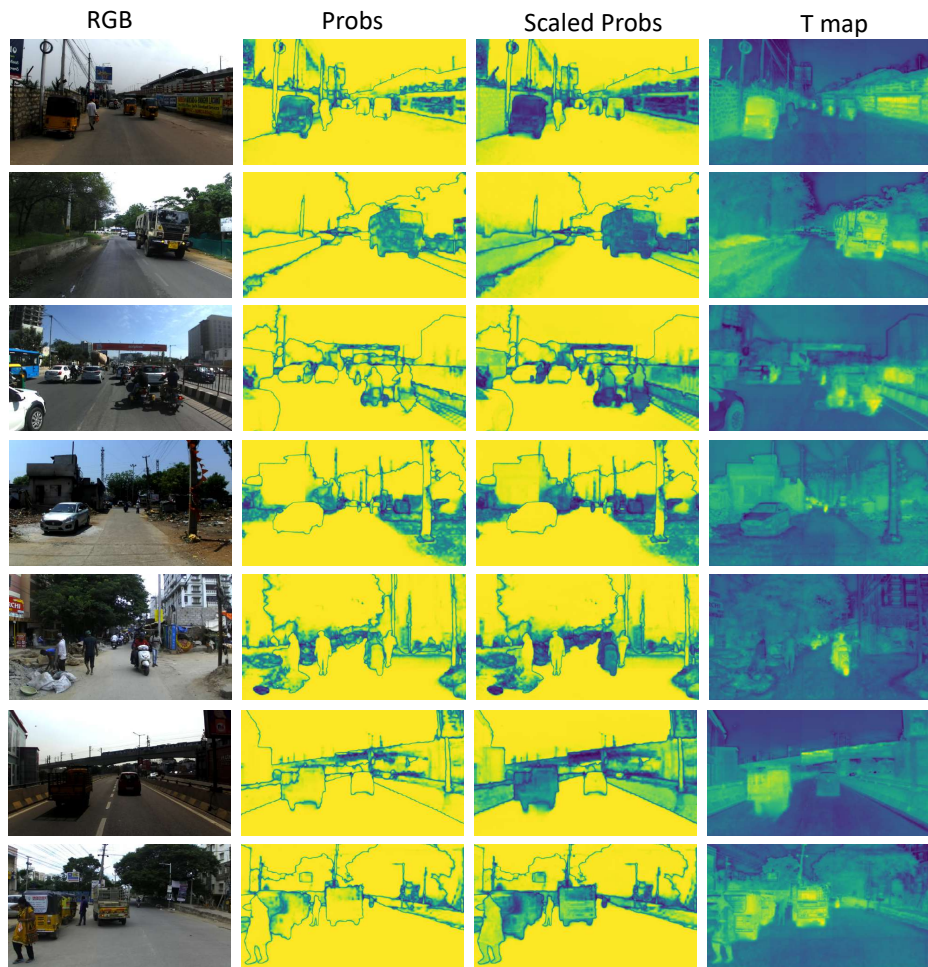


Figure C.13: Calibration and temperature maps for different IDD images. We observe how different OOD classes (*e.g.* autorickshaw, bridge, billboard) or weird instances are highlighted in the calibration maps.

C.12 Local OOD detection

In this section we perform OOD detection at the pixel level to find regions of the image that belong to unknown classes (autorickshaw, guardrail, billboard, bridge). We define pixels of unknown classes as OOD while those corresponding to CS classes are in distribution. In Figure C.14 we present the results of local OOD detection vs mIoU. We make two observations: i) Mask2Former has the best local OOD detection; ii) Differences between models are smaller for local OOD: numbers are roughly in the 0.7 – 0.8 range *vs.* 0.4 – 1.0 range for image-based OOD.

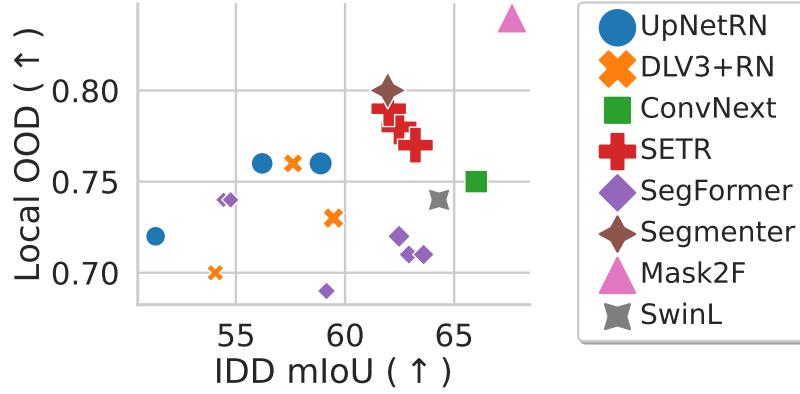


Figure C.14: mIoU (↑) *vs.* local ood (↑) for different model families. All models trained on CS. Markersize proportional to number of parameters. Mask2Former achieves the best local OOD detection performance.

C.13 Additional plots: adaptive temperature via clustering

In Figure 5.3 we analyzed the calibration error after applying the method by Gong *et al.* Gong et al. (2021), which clusters the images in the calibration set and computes a different temperature per cluster. Due to space constraints we only showed the best performing model for each family, in Figure C.15 we show the results for all models.



Figure C.15: ECE ($\downarrow$) after clustering TS Extension of Figure 5.3 in the main paper where we show results for all models.

Complementary to Figure C.15 where we show the different calibration methods for a given architecture with barplots, in Figure C.16 we present the same results but we group them by calibration method (instead of by model). For each test dataset (rows), we plot the ECE *vs.* mIoU after calibrating the models with the corresponding method (columns).

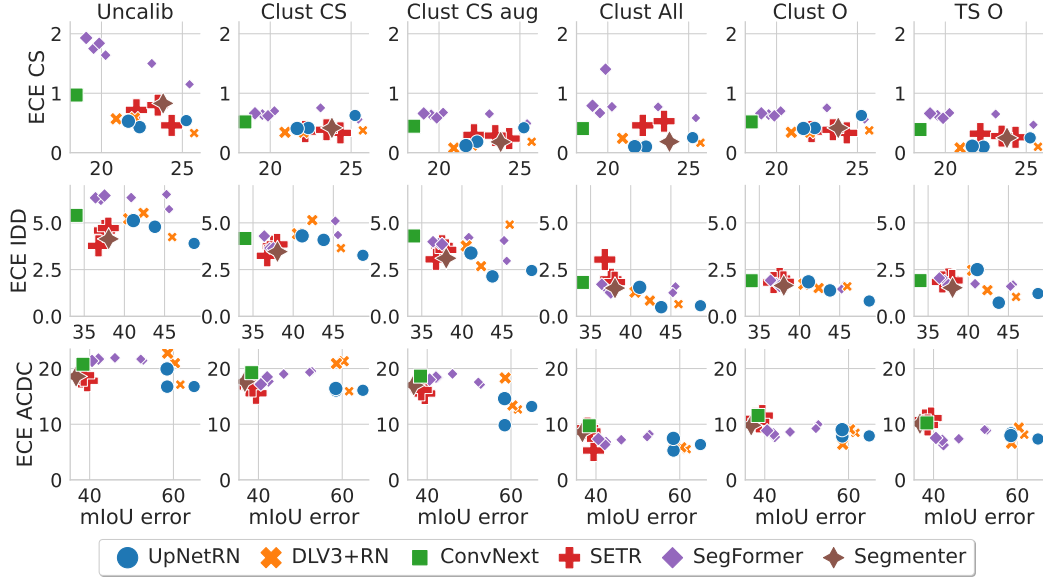


Figure C.16: ECE ($\downarrow$) *vs.* mIoU error ($\downarrow$) after calibration with clustering TS considering different calibration sets. This plot provides a different visualization of the results in Figure C.15.

C.14 Additional plots: local temperature scaling

In Figure 5.5 we analyzed the calibration error after applying the method by Ding *et al.* Ding et al. (2021) which learns a calibration network that predicts the temperature as a function of the image and segmentation model logits. Due to space constraints we only showed the best performing model for each family, in Figure C.17 we show the results for all models.



Figure C.17: ECE (↓) after Local TS Extension of Figure 5.5 in the main paper where we show results for all models.

Complementary to Figure C.17 where we show the different calibration methods for a given architecture with barplots, in Figure C.18 we present the same results but we group them by calibration method (instead of by model). For each test dataset (rows), we plot the ECE *vs.* mIoU after calibrating the models with the corresponding method (columns).

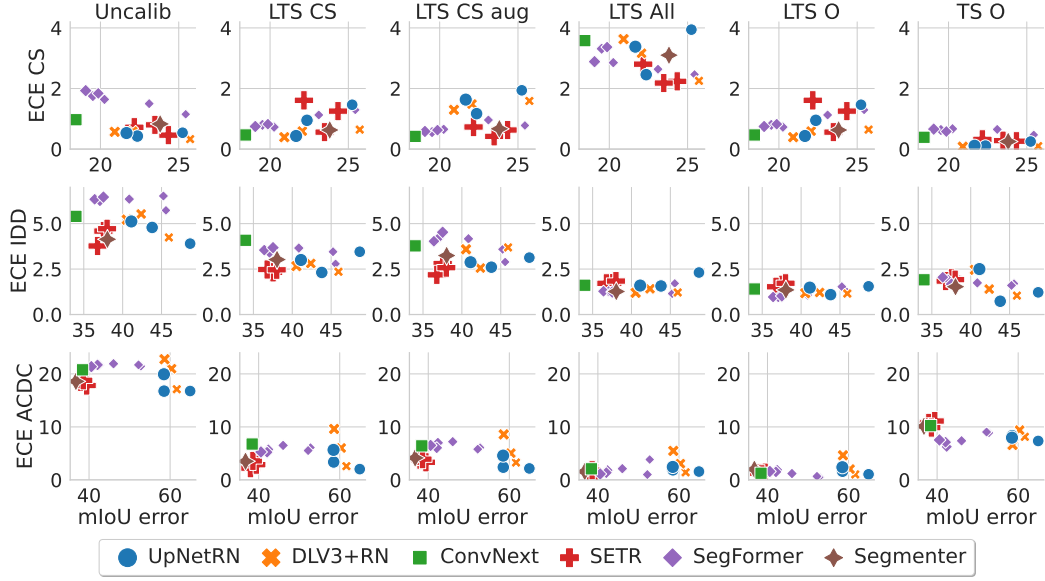


Figure C.18: ECE ($\downarrow$) *vs.* mIoU error ($\downarrow$) after calibration with Local TS considering different calibration sets. This plot is showing the same results as Figure C.17 but in a different visualization.

C.15 Additional plots: Comparison calibration methods

In Figure 5.6 we compared the calibration error after calibrating with TS CS, Clust CS and LTS CS *vs.* the uncalibrated baseline. Due to space constraints we only showed the best performing model for each family, in Figure C.19 we show the results for all models.

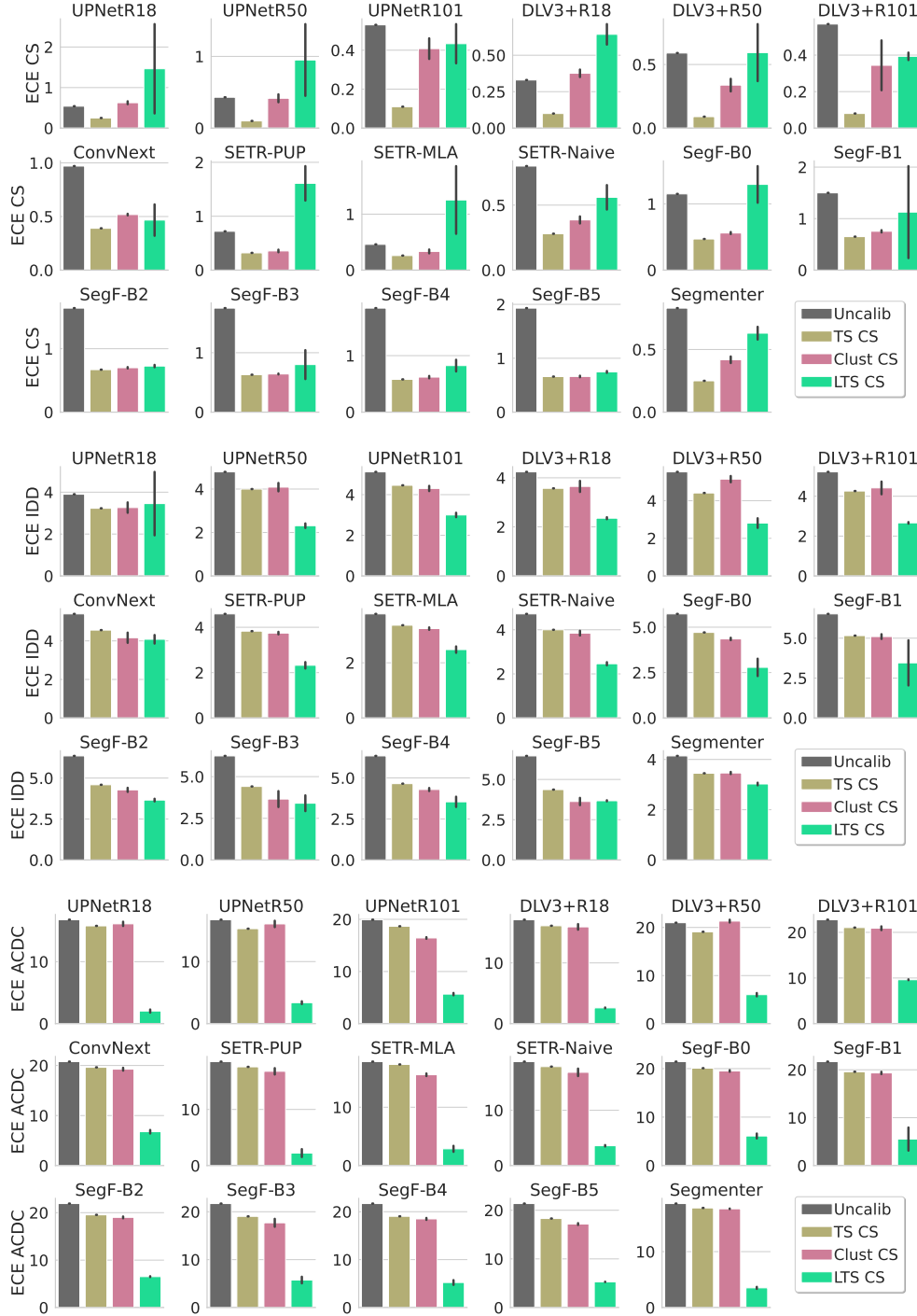


Figure C.19: ECE (↓) after calibration with different methods Extension of Figure 5.6 in the main paper where we show results for all models.

Complementary to Figure C.19 where we show the different calibration methods for a given architecture with barplots, in Figure C.20 we present the same results but we group them by calibration method (instead of by model). For each test dataset (rows), we plot the ECE *vs.* mIoU after calibrating the models with the corresponding method (columns).

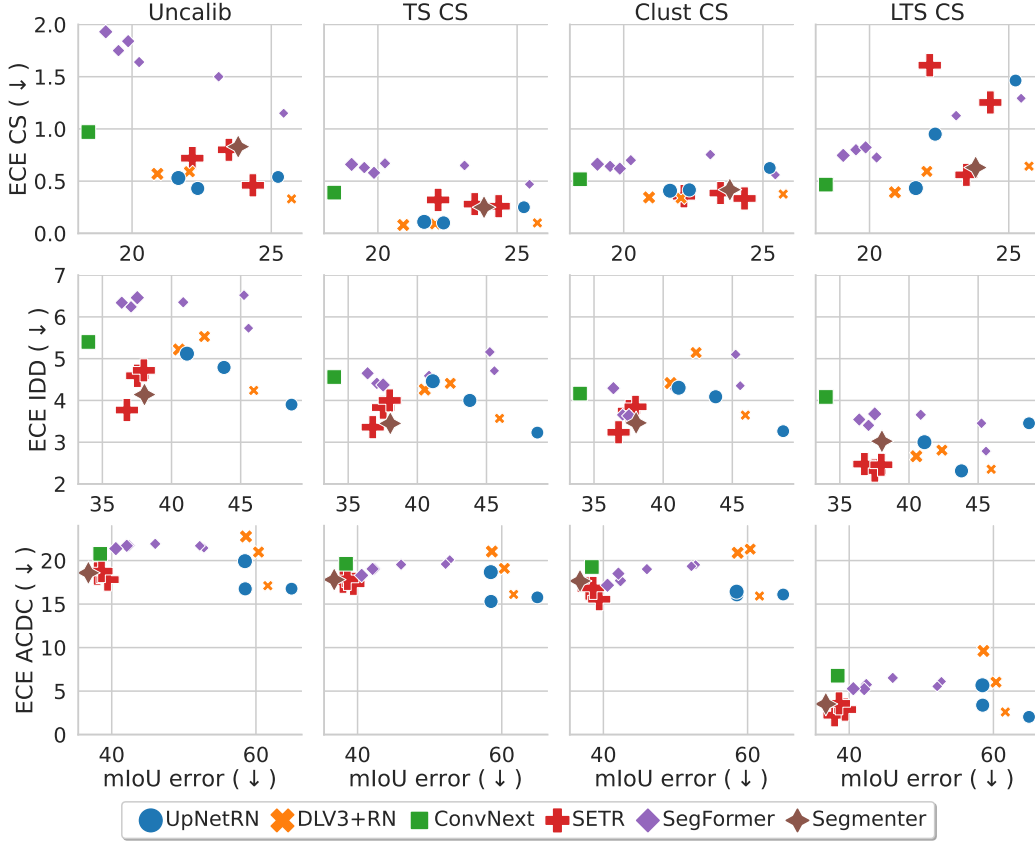


Figure C.20: ECE (↓) *vs.* mIoU error (↓) after calibration with different methods on the CS calibration set. This plot is showing the same results as Figure C.19 but in a different visualization.

C.16 Additional plots: misclassification detection

In Figure 5.8 we compared the misclassification detection and OOD detection performance of the networks after calibrating with TS CS, Clust CS and LTS CS *vs.* the uncalibrated baseline. Due to space constraints we only showed the best performing model for each family, in Figure C.21 we show the misclassification detection results for all models.

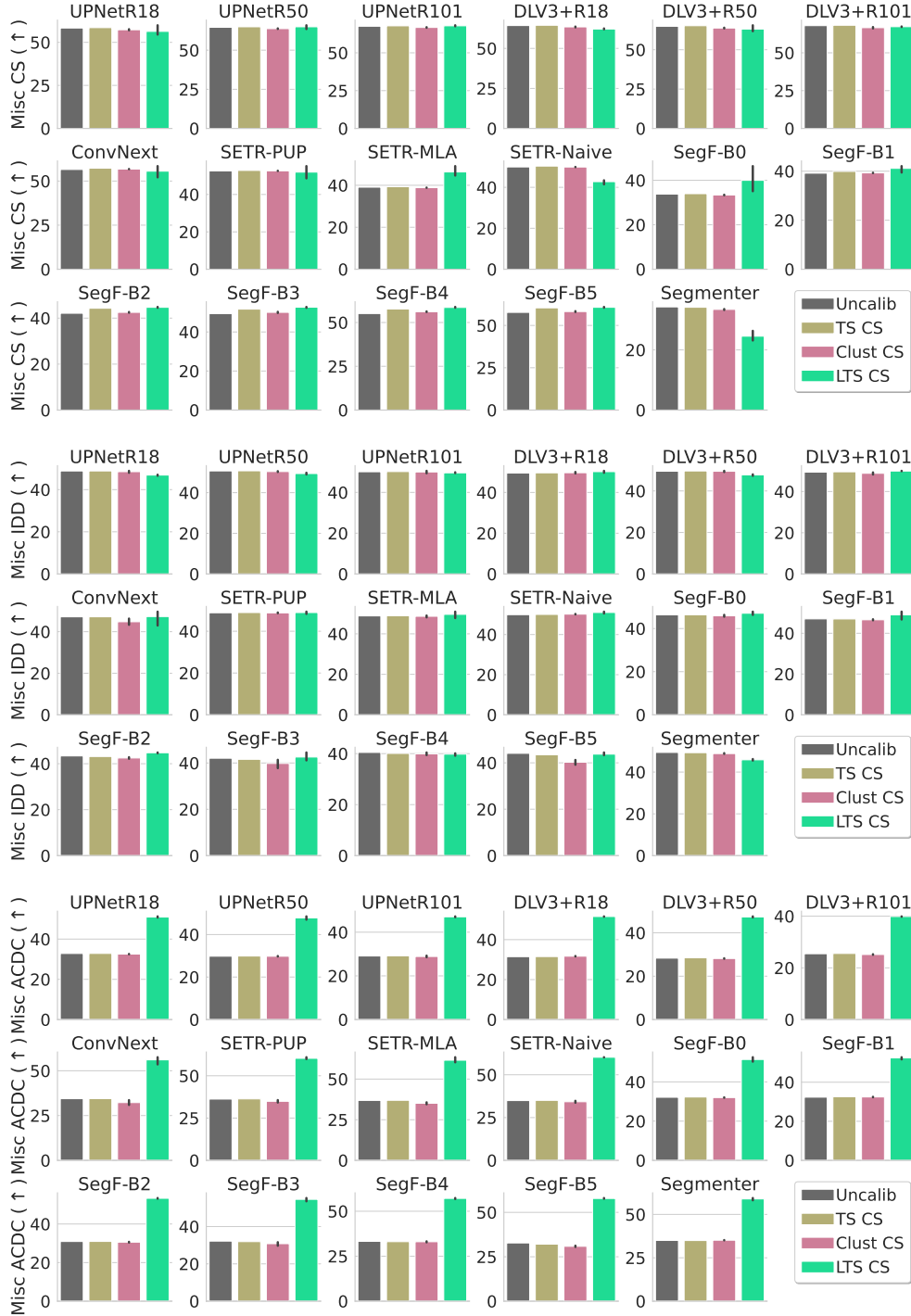


Figure C.21: Misc detection: PRR (↑) after calibration with different methods
 Extension of Figure 5.8 in the main paper where we show misclassification results for all models.

Complementary to Figure C.21 where we show the different calibration methods for a given architecture with barplots, in Figure C.22 we present the same results but we group them by calibration method (instead of by model). For each test dataset (rows), we plot the ECE *vs.* mIoU after calibrating the models with the corresponding method (columns).

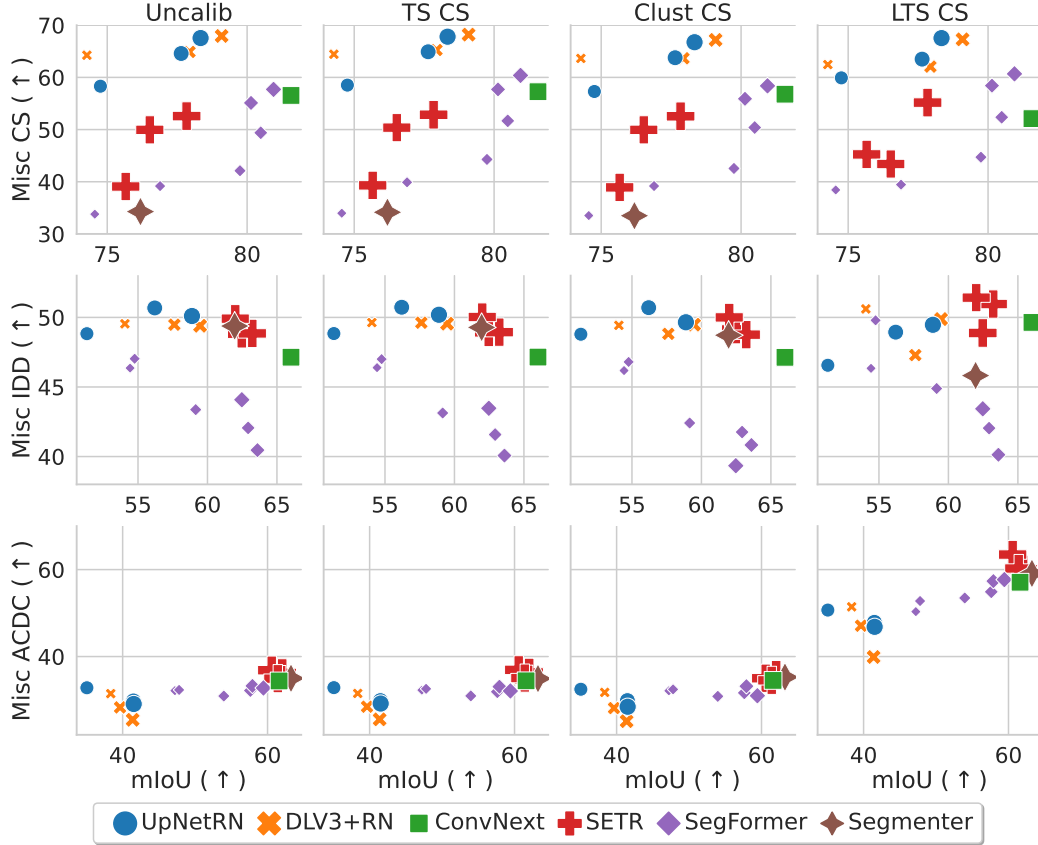


Figure C.22: PRR (↑) *vs.* mIoU (↑) after calibration with different methods on the CS calibration set. This plot is showing the same results as Figure C.21 but in a different visualization.

C.17 Additional plots: out-of-distribution detection

In Figure 5.8 we compared the misclassification detection and ood detection performance of the networks after calibrating with TS CS, Clust CS and LTS CS *vs.* the uncalibrated baseline. Due to space constraints we only showed the best performing model for each family, in Figure C.21 we show the OOD detection results for all models.

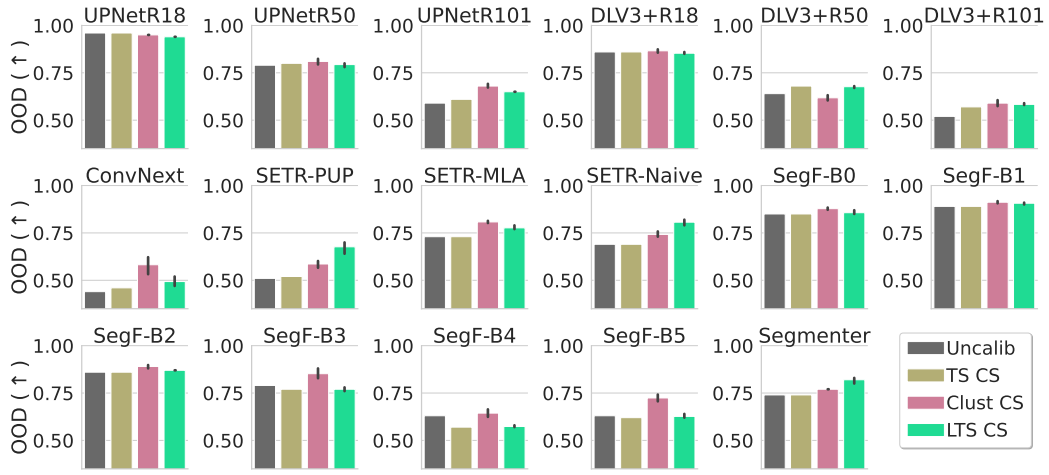


Figure C.23: OOD detection: AUROC (↑) after calibration with different methods
Extension of Figure 5.8 in the main paper where we show OOD detection results for all models.

Complementary to Figure C.23 where we show the different calibration methods for a given architecture with barplots, in Figure C.24 we present the same results but we group them by calibration method (instead of by model). For each test dataset (rows), we plot the ECE *vs.* mIoU after calibrating the models with the corresponding method (columns).

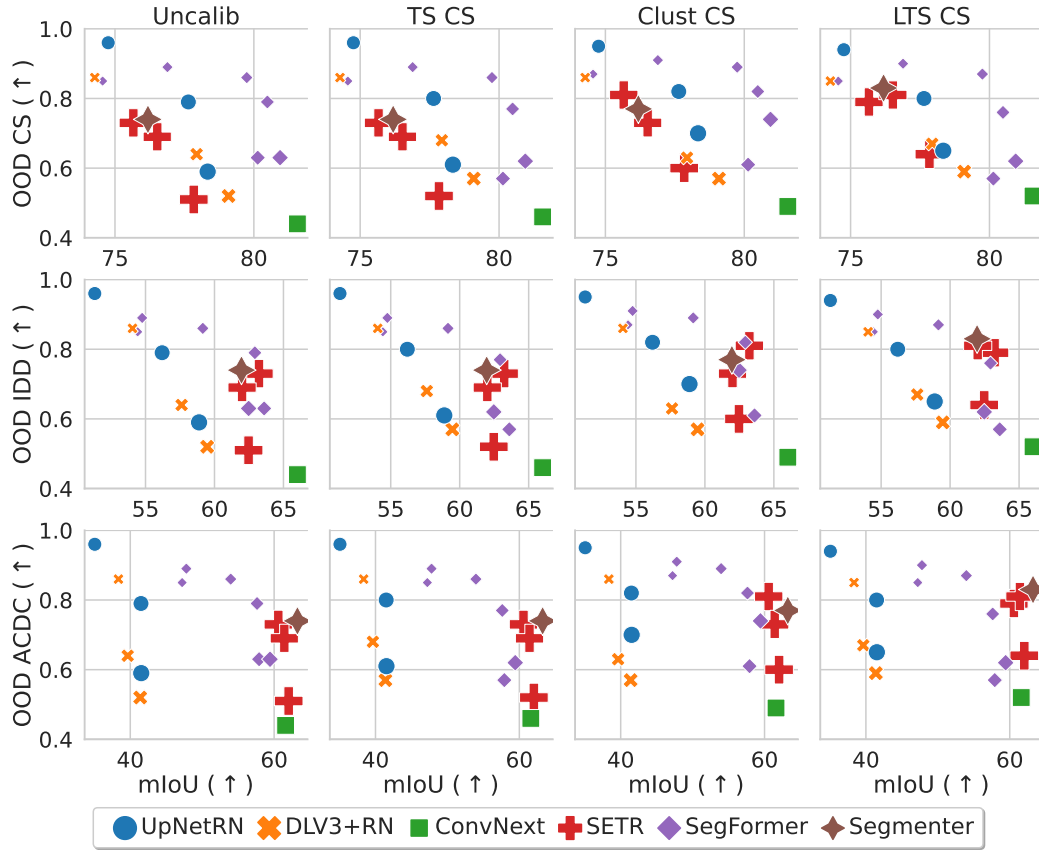


Figure C.24: AUROC (↑) vs. mIoU (↑) after calibration with different methods on the CS calibration set. This plot is showing the same results as Figure C.23 but in a different visualization.



Appendix of Placing Objects in Context via Inpainting for Out-of-distribution Segmentation

D.1 POC datasets object list

Our POC evaluation sets (*i.e.* POC-CS, POC-IDD and POC-ACDC) are obtained by adding objects to different self-driving datasets. Following works like Lost and Found Pinggera et al. (2016), we compiled a list of 25 objects that can be found on the road.

The **anomaly list** is as follows: “stroller”, “trolley”, “garbage bag”, “wheelie bin”, “suitcase”, “skateboard”, “chair dumped on the street”, “sofa dumped on the street”, “furniture dumped on the street”, “mattress dumped on the street”, “garbage dumped on the street”, “clothes dumped on the street”, “cement mixer on the street”, “cat”, “dog”, “bird flying”, “horse”, “skunk”, “sheep”, “crocodile”, “alligator”, “bear”, “llama”, “tiger” and “monkey”.

Additionally, we also add a few classes from Cityscapes to make sure that anomaly segmentation models are indeed detecting *anomalies* and not merely identifying

synthetic objects.

Cityscapes classes included are: “rider”, “bicycle”, “motorcycle”, “bus”, “person” and “car”.

D.2 Anomaly segmentation methods: mIoU on Cityscapes

Although fine-tuning methods with OOD samples can improve anomaly segmentation significantly, they may affect the closed-set performance. In Table D.1 we report the mIoU on Cityscapes of all methods reported in the main paper, showing that fine-tuning with POC data does not negatively impact closed-set performance.

Method	Mask2Anomaly					RPL					RbA				
OOD data	No ft.	COCO	POC	coco	POC alt.	No ft.	COCO	POC	coco	POC alt.	No ft.	COCO	POC	coco	POC alt.
mIoU ↑	78.29	78.34	78.33	78.33	78.49	90.94	90.94	90.94	90.94	90.94	82.25	82.15	82.17	82.16	82.16

Table D.1: mIoU on Cityscapes validation set. We compute the mIoU after fine-tuning with different datasets (complementing results in Table 6.2). We observe that fine-tuning with our POC datasets does not degrade the closed-set performance.

D.3 Adding objects with Instruct Pix2Pix

When building our pipeline, we explored different generative methods. In particular, InstructPix2Pix (IP2P) Brooks et al. (2023) has showed remarkable performance following natural language instructions (*e.g.* “turn the sofa red”). Therefore, it would be natural to have such “general-purpose editing” methods as baselines to add objects to the images. We observed that IP2P seems to be biased towards *modifying* objects in the scene rather than *adding new ones*. In Figure D.1 we show several examples of images generated with IP2P. In our initial experiments, we observe how new objects tend to replace the logo of the ego vehicle (top images). If we remove the bottom of the image (middle section), we then observe that some particular image features (*e.g.* the edge of the sidewalk) tend to be replaced. Moreover, the added objects tend to lack

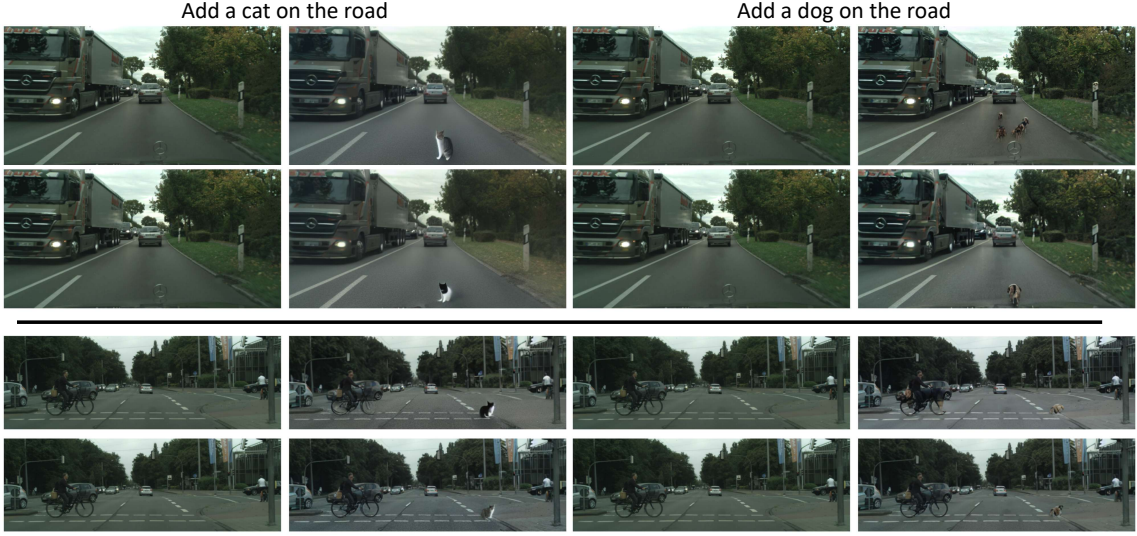


Figure D.1: Sample images from InstructPix2Pix Brooks et al. (2023). We observe that InstructPix2Pix has a bias to replace certain objects or features in the scene. In top images it replaces the mercedes logo of the ego vehicle while in bottom images it replaces the edge of the sidewalk.

realism and, given that the changes are not constrained to a particular region, editing via IP2P usually results in undesired modifications in other image regions.

D.4 AUPRC plots

In Figure D.2 we visualize the AUPRC results for all methods, complementing the visualization in Figure 6.1.

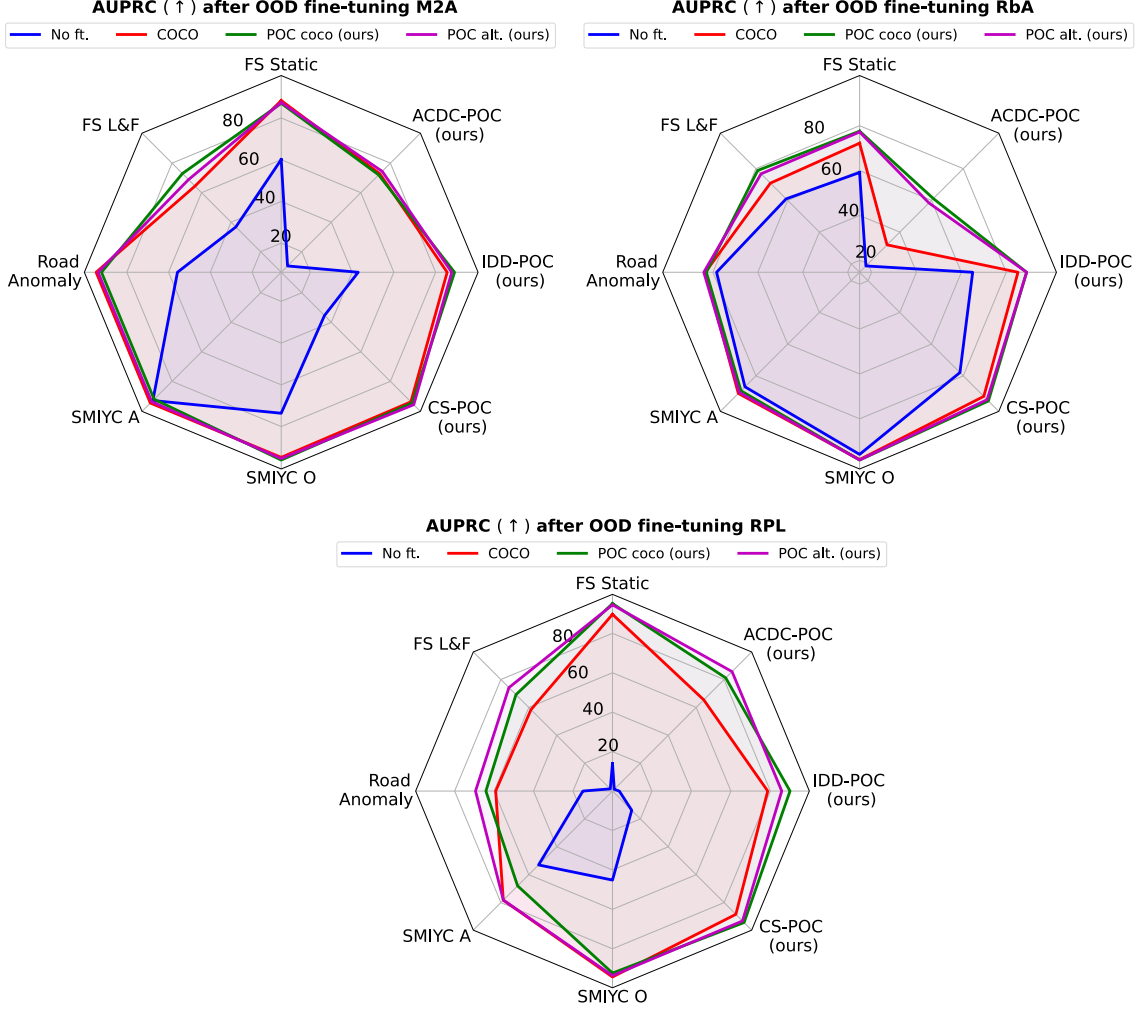


Figure D.2: AUPRC on different anomaly segmentation datasets. We compare three different anomaly segmentation methods, M2ARai et al. (2023), RPL Liu et al. (2023b) and RbANayal et al. (2023) with different fine-tuning datasets. Fine-tuning with POC-generated images tends to bring improvements or match COCO fine-tuning in most settings.

D.5 Additional Pascal training samples

In Figure D.3 we show additional samples generated with our POC pipeline as well as T2I baselines (that use text-to-image models inspired by Karazija et al. (2023)).



Figure D.3: Training image samples. Additional training images to learn new classes, complementing Figure 6.6.

D.6 Additional anomaly score maps

In this appendix section we add more visualizations of anomaly score maps with different methods. We show results from our three POC-generated datasets as well as samples from previous anomaly datasets.

D. Appendix of Placing Objects in Context via Inpainting for Out-of-distribution Segmentation

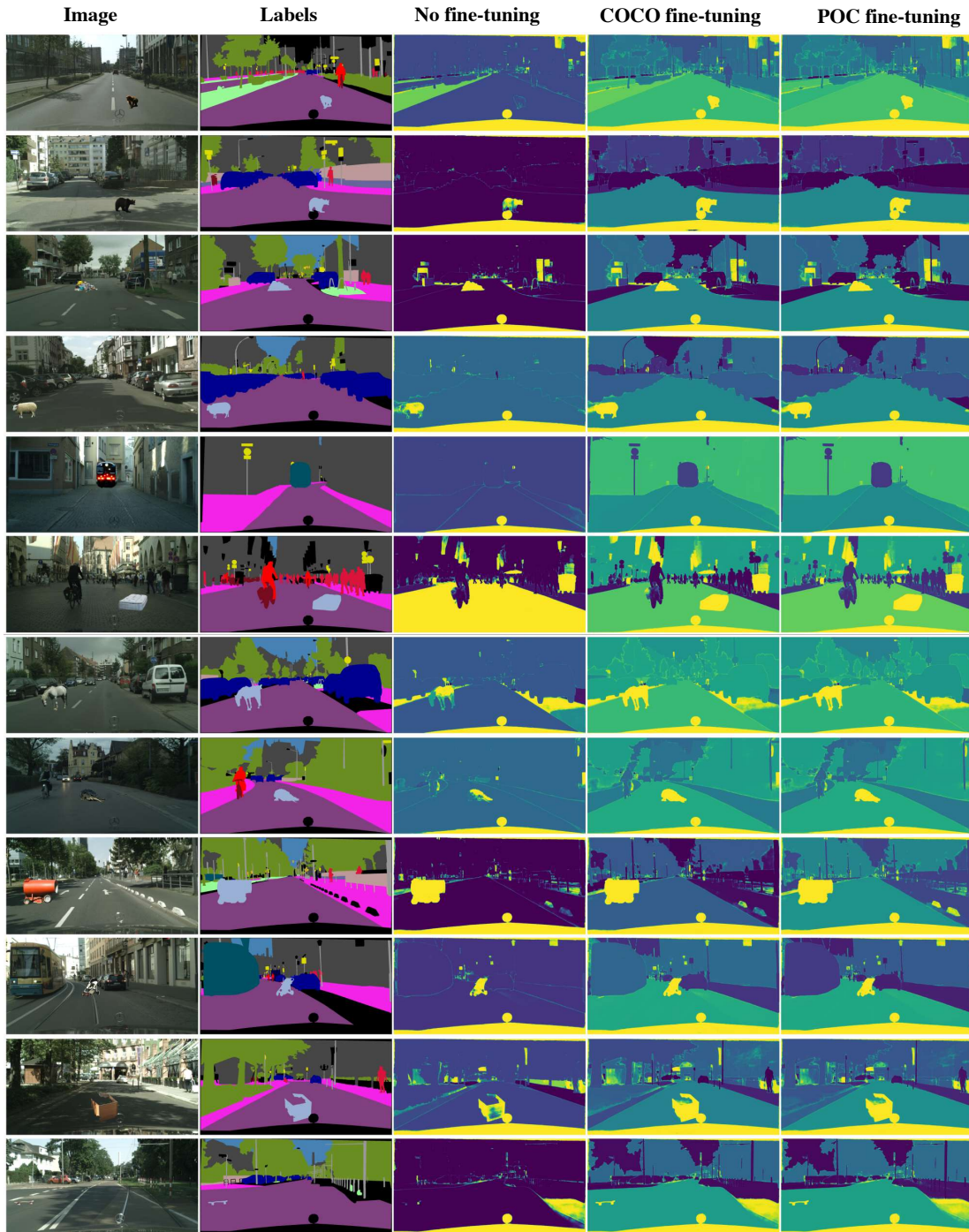


Figure D.4: M2A anomaly scores on CS-POC samples.

D. Appendix of Placing Objects in Context via Inpainting for Out-of-distribution Segmentation

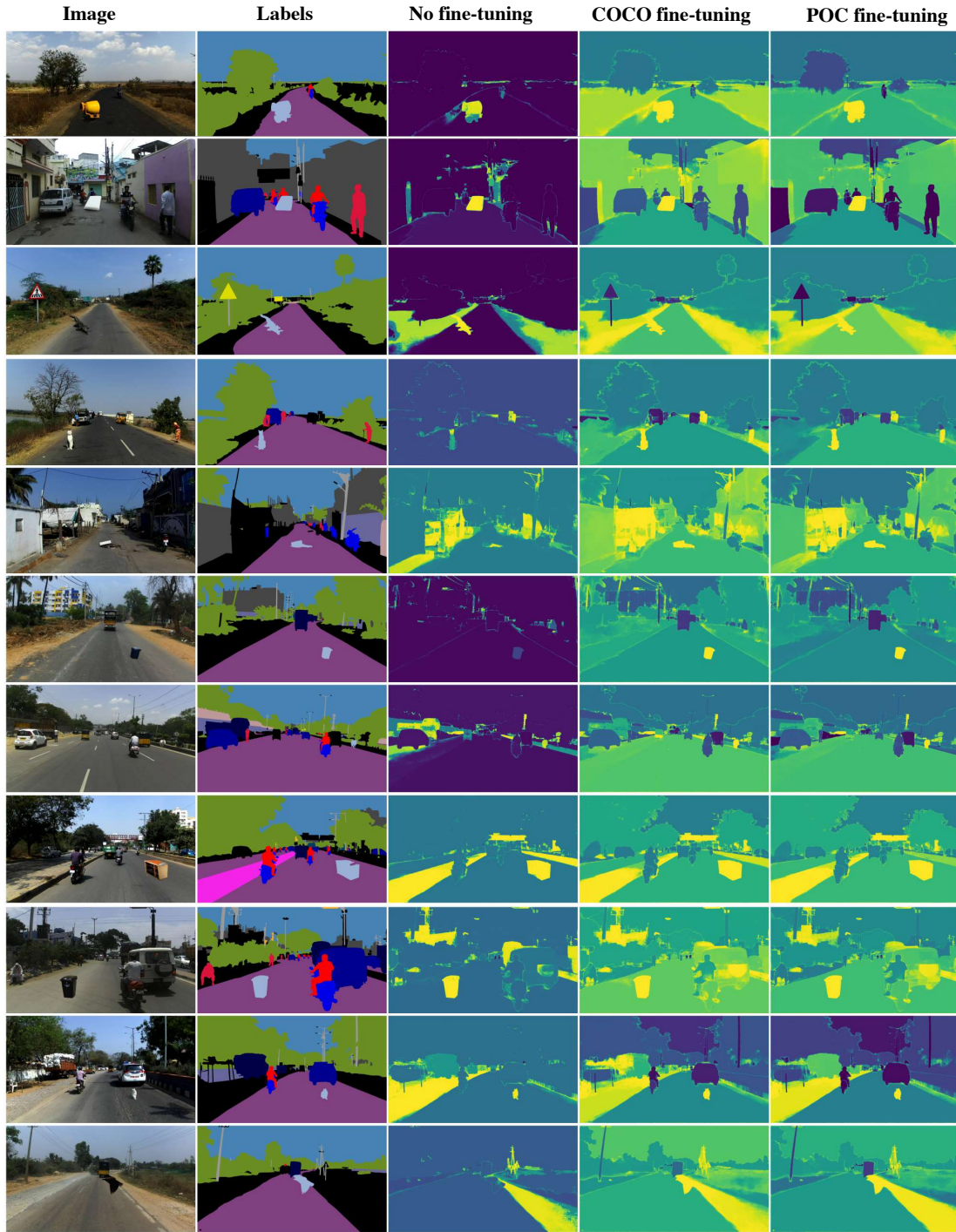


Figure D.5: M2A anomaly scores on IDD-POC samples.

D. Appendix of Placing Objects in Context via Inpainting for Out-of-distribution Segmentation

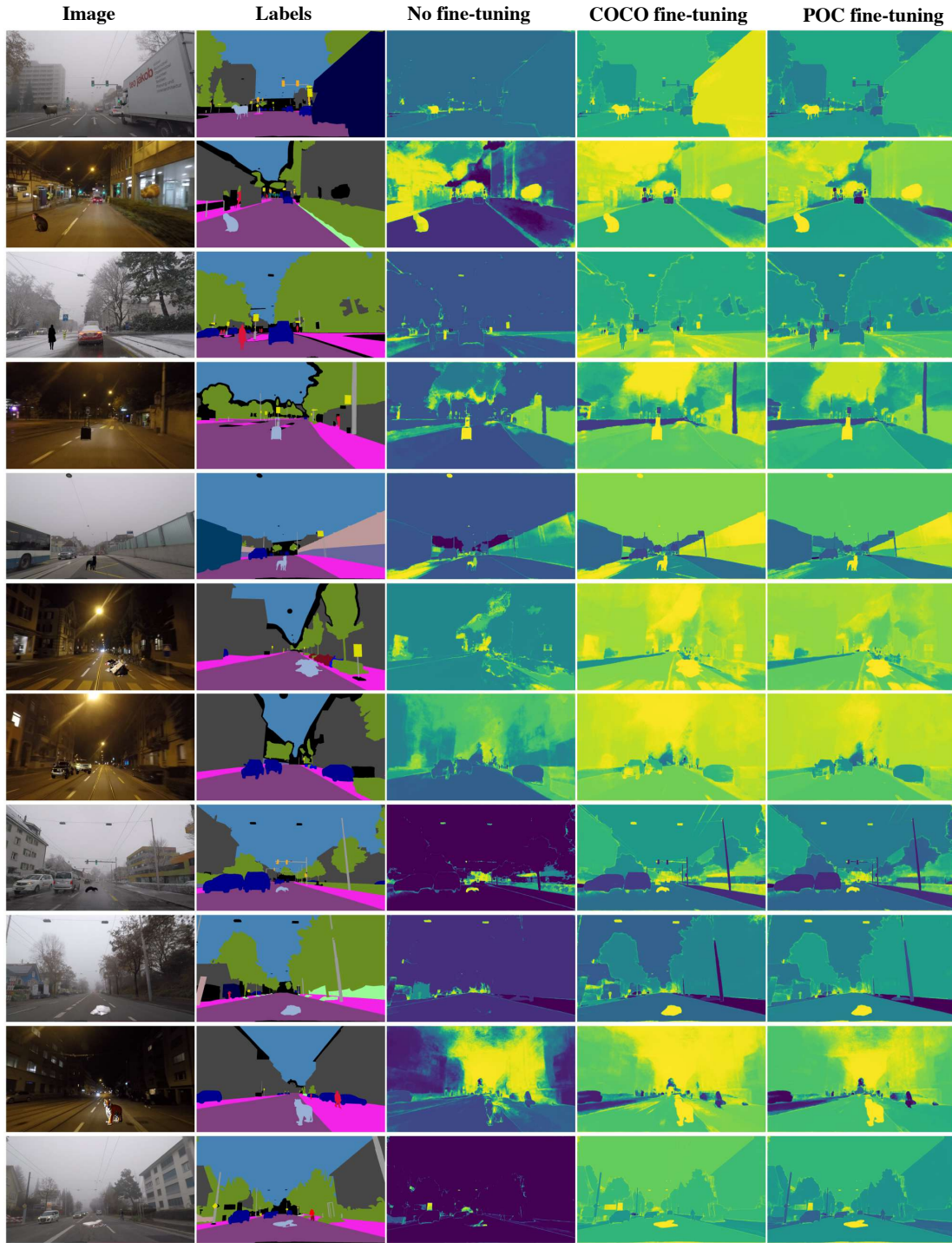
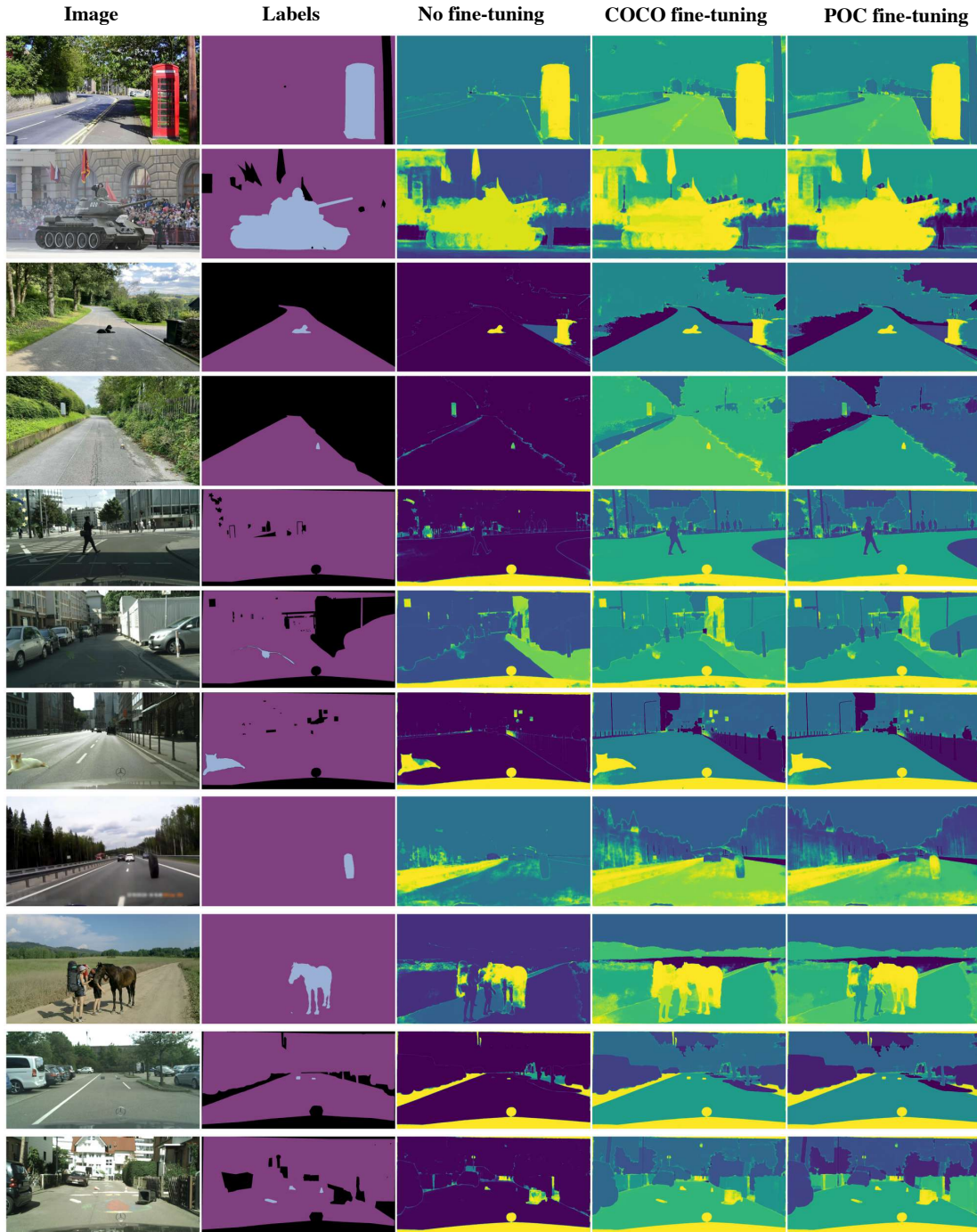


Figure D.6: M2A anomaly scores on ACDC-POC samples.

D. Appendix of Placing Objects in Context via Inpainting for Out-of-distribution Segmentation



D. Appendix of Placing Objects in Context via Inpainting for Out-of-distribution Segmentation

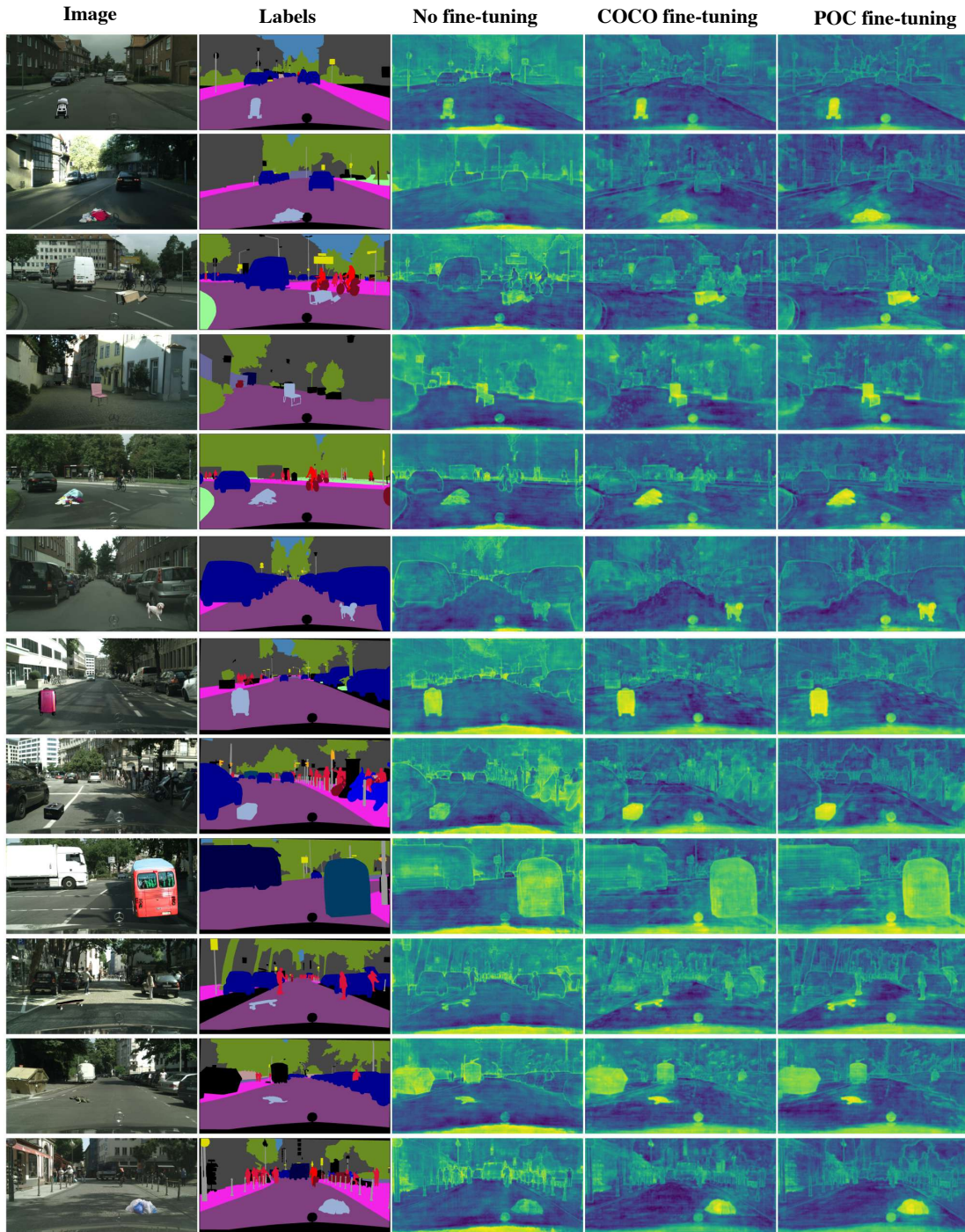


Figure D.8: RPL anomaly scores on CS-POC samples.

D. Appendix of Placing Objects in Context via Inpainting for Out-of-distribution Segmentation

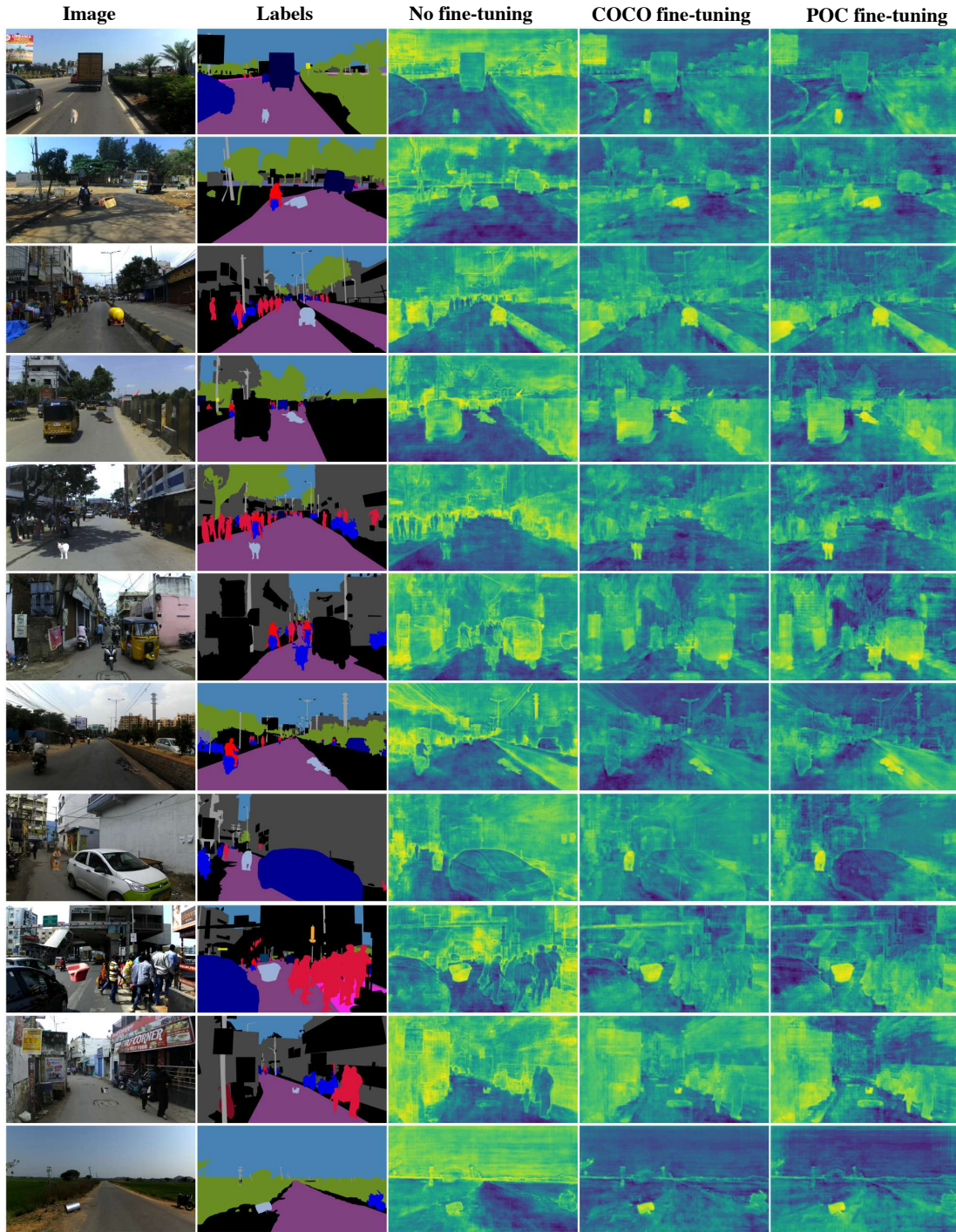


Figure D.9: RPL anomaly scores on IDD-POC samples.

D. Appendix of Placing Objects in Context via Inpainting for Out-of-distribution Segmentation

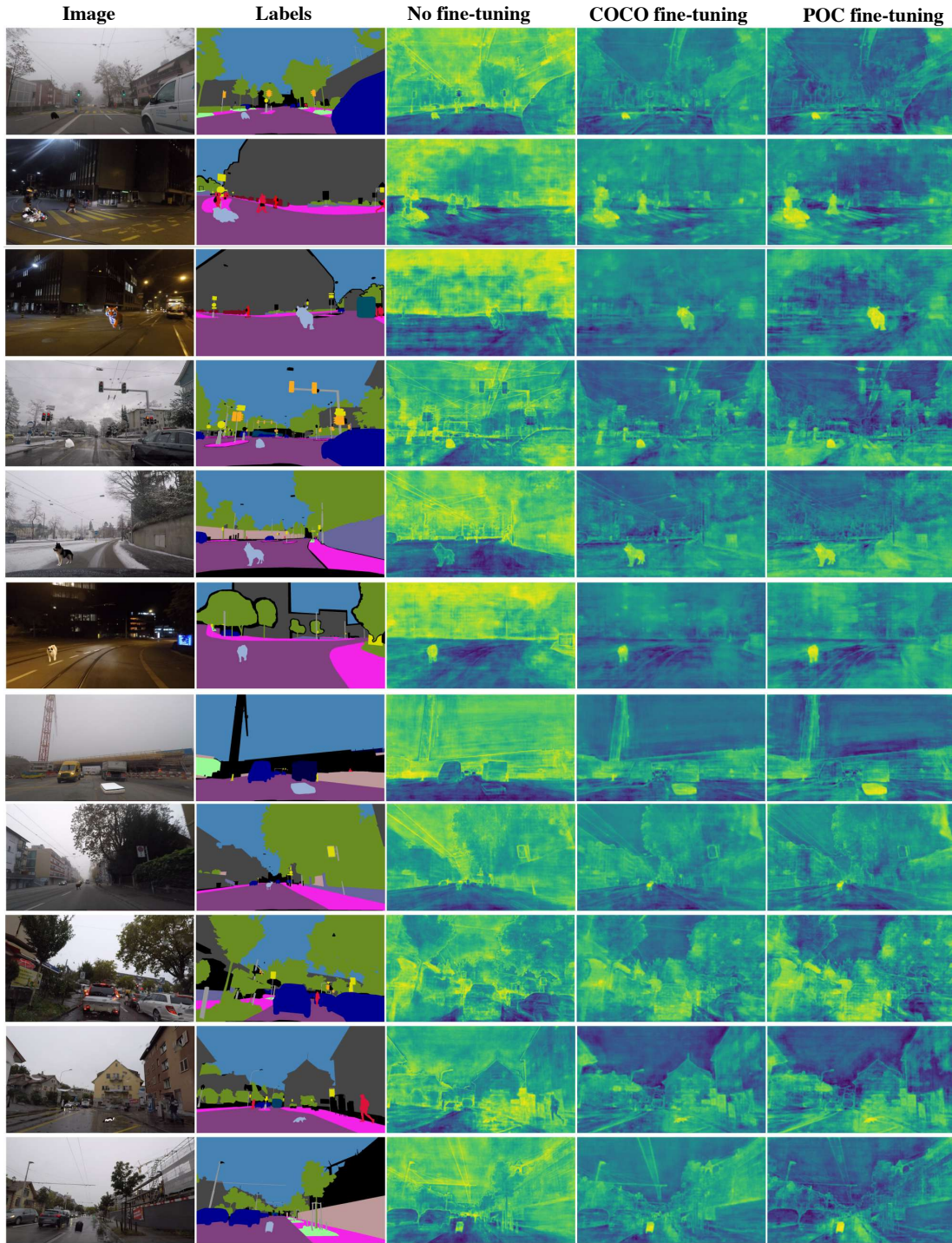


Figure D.10: RPL anomaly scores on ACDC-POC samples.

D. Appendix of Placing Objects in Context via Inpainting for Out-of-distribution Segmentation

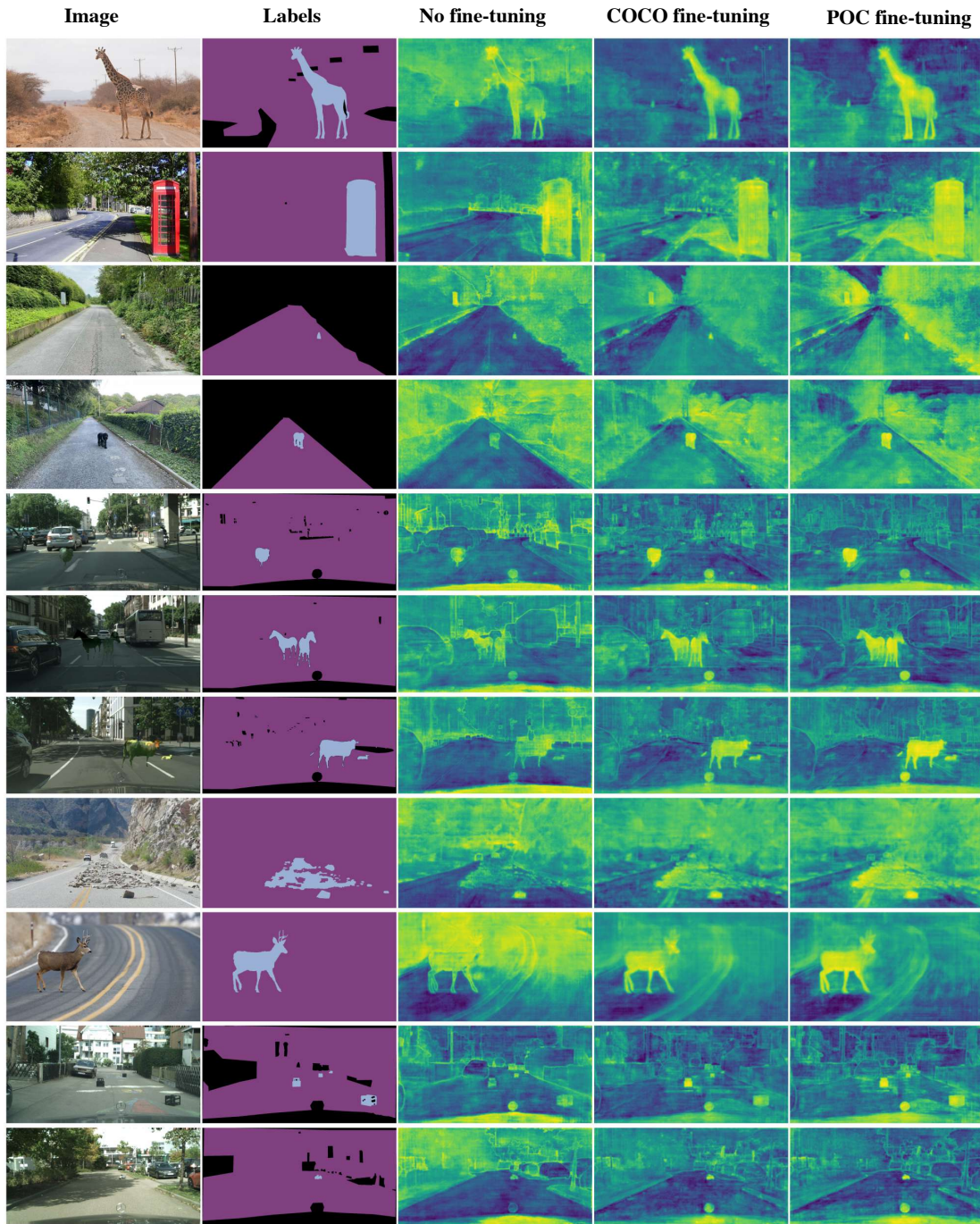


Figure D.11: RPL anomaly scores on samples from related datasets (see Figure 6.2).

D. Appendix of Placing Objects in Context via Inpainting for Out-of-distribution Segmentation

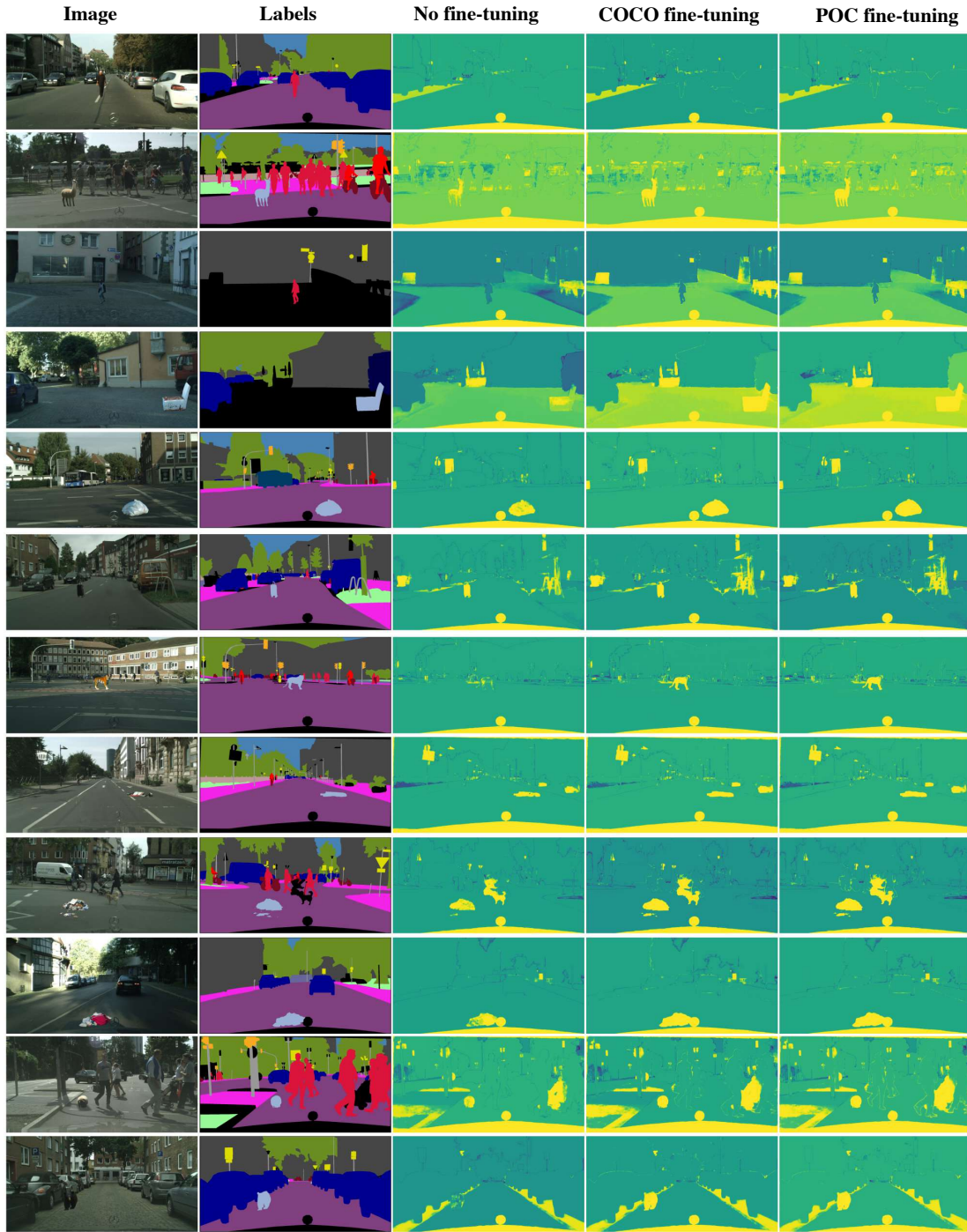


Figure D.12: RbA anomaly scores on CS-POC samples.

D. Appendix of Placing Objects in Context via Inpainting for Out-of-distribution Segmentation

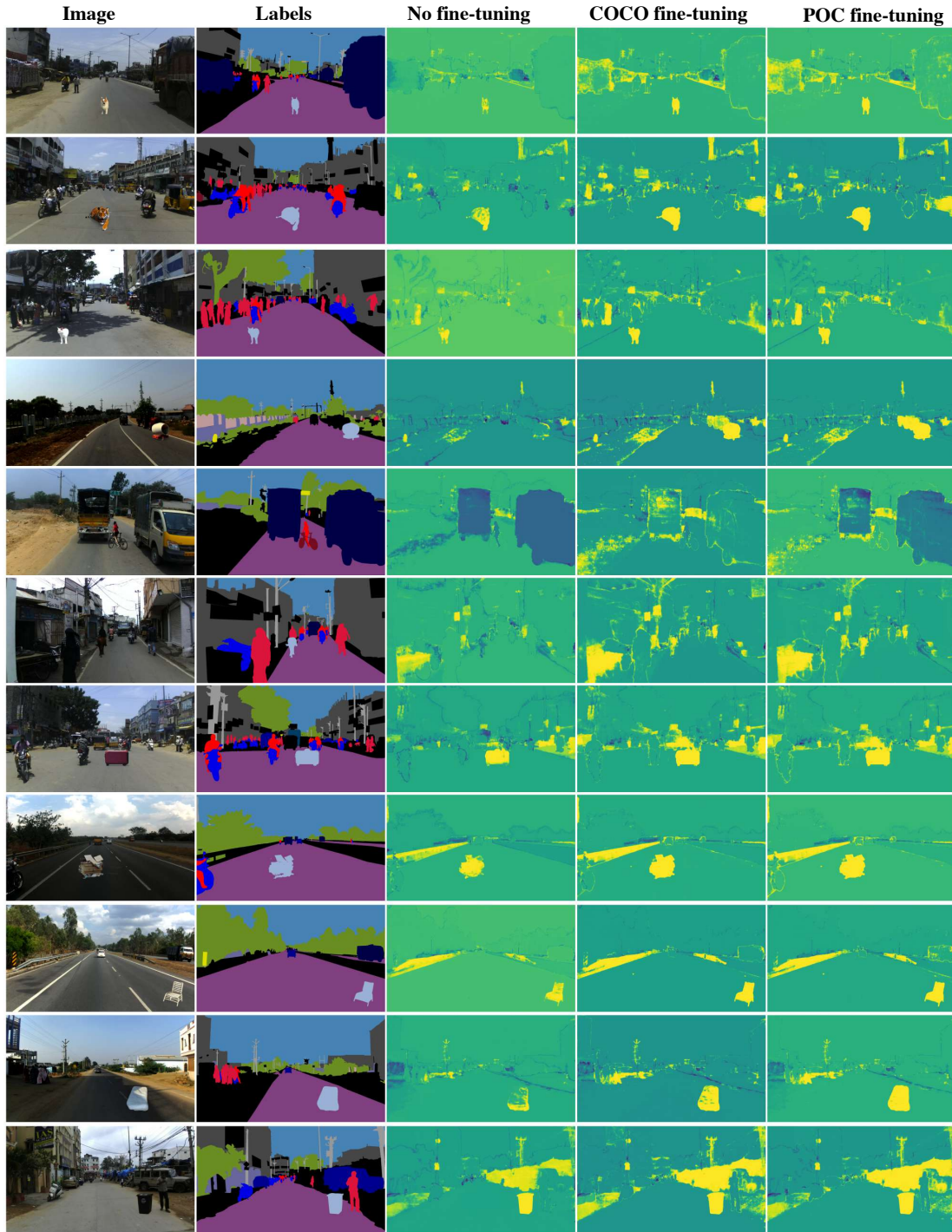


Figure D.13: RbA anomaly scores on IDD-POC samples.

D. Appendix of Placing Objects in Context via Inpainting for Out-of-distribution Segmentation

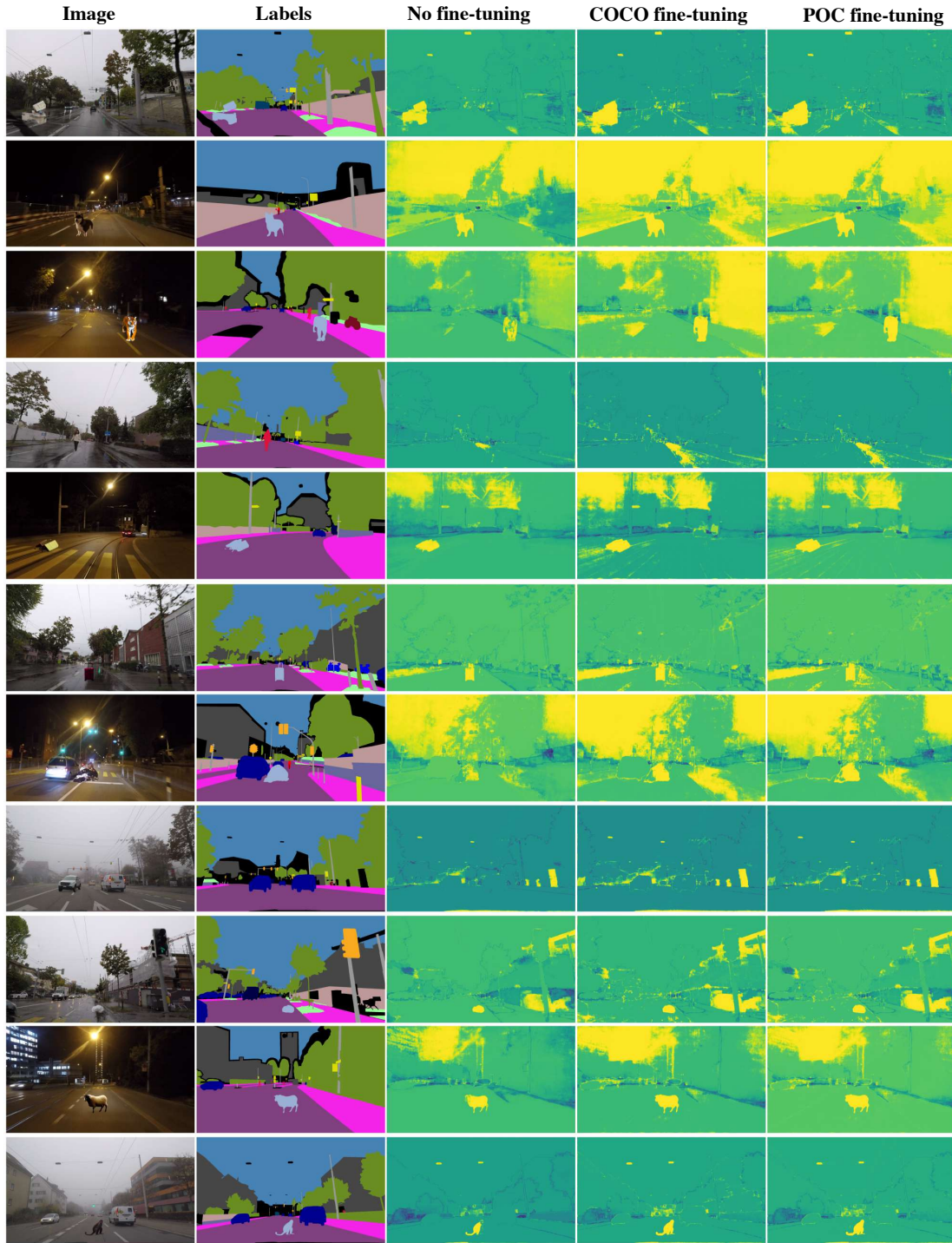


Figure D.14: RbA anomaly scores on ACDC-POC samples.

D. Appendix of Placing Objects in Context via Inpainting for Out-of-distribution Segmentation

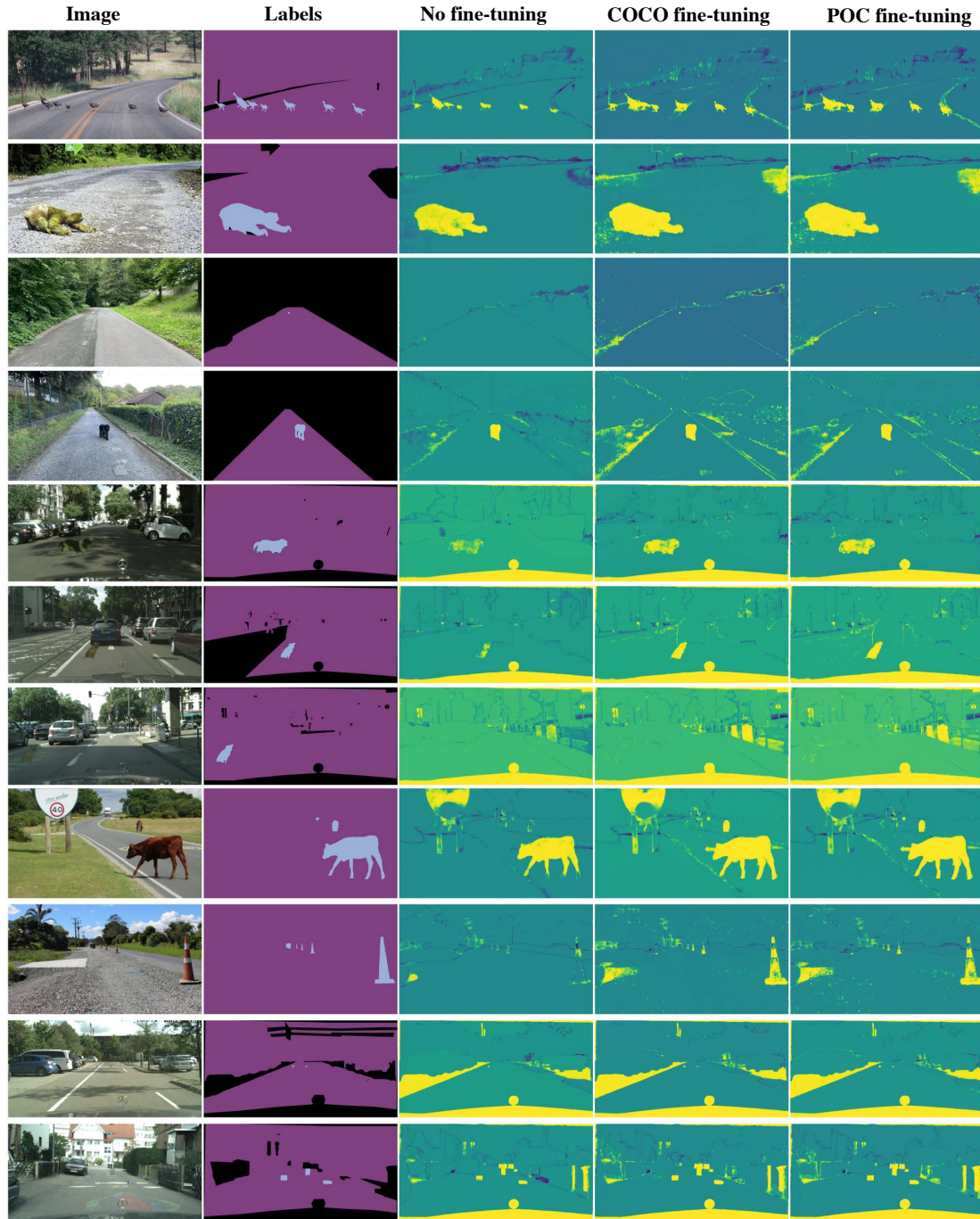


Figure D.15: RbA anomaly scores on samples from related datasets (see Figure 6.2).

D.7 Additional qualitative results

In this section we show additional qualitative results for the dataset extension experiments (*c.f.* Section 6.5). We show predictions on all evaluated datasets for DLV3+, ConvNeXt and Segmenter models.

D. Appendix of Placing Objects in Context via Inpainting for Out-of-distribution Segmentation



Figure D.16: DLV3+ predictions on additional web images.

D. Appendix of Placing Objects in Context via Inpainting for Out-of-distribution Segmentation



Figure D.17: ConvNeXt predictions on additional web images.

D. Appendix of Placing Objects in Context via Inpainting for Out-of-distribution Segmentation

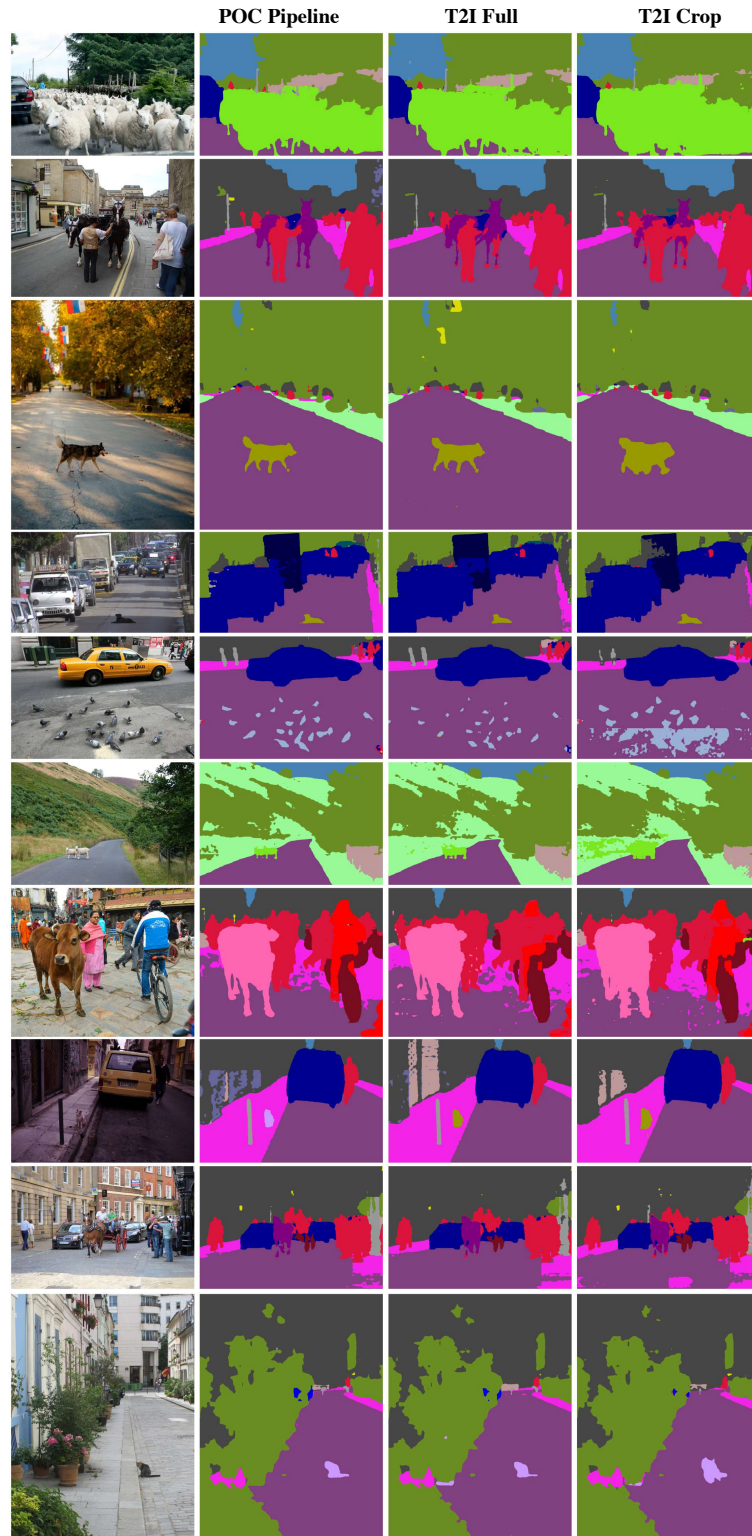


Figure D.18: Segmenter predictions on additional web images.

D. Appendix of Placing Objects in Context via Inpainting for Out-of-distribution Segmentation

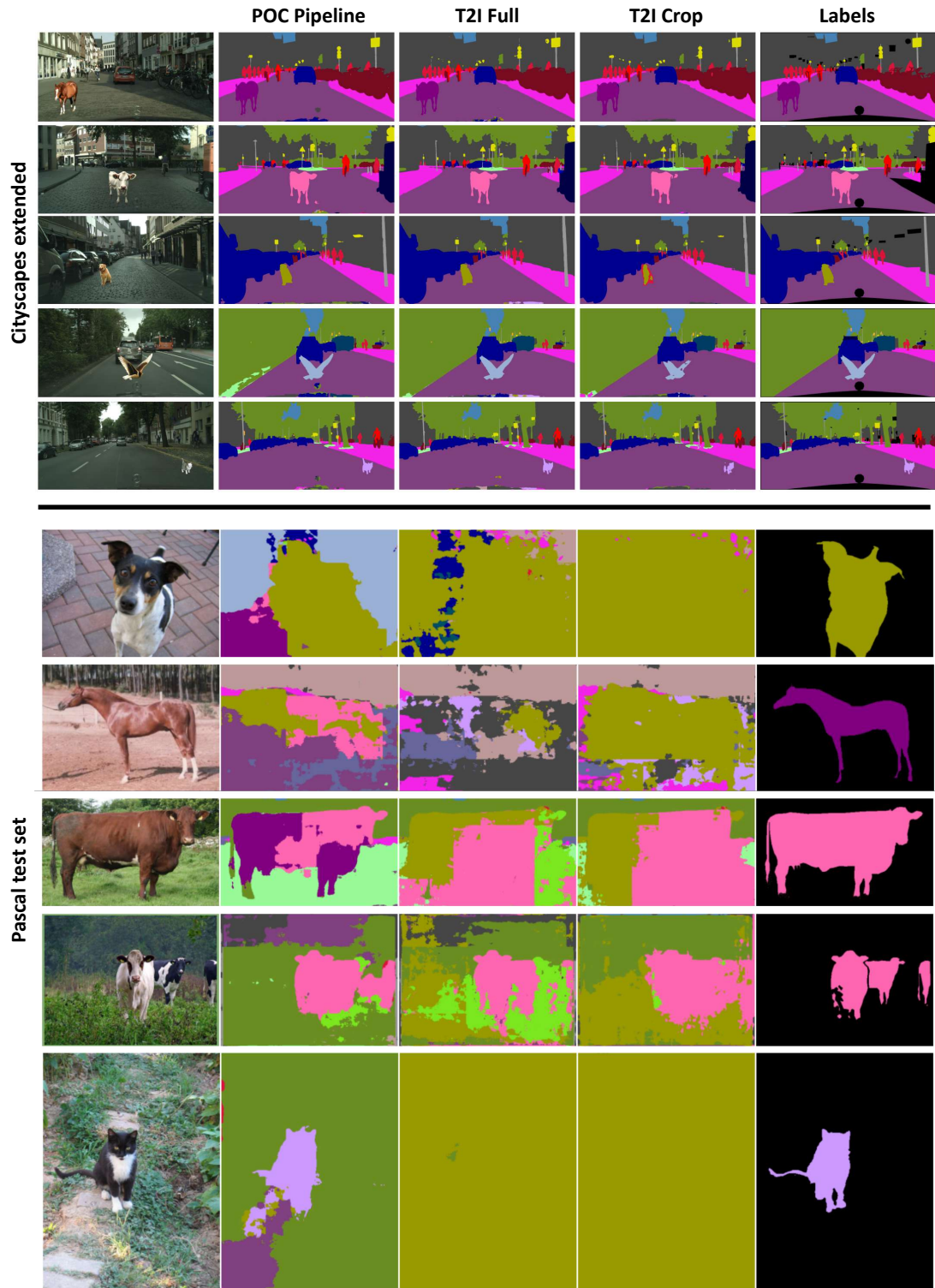


Figure D.19: DLV3+ predictions on extended Cityscapes (*POC A*) and Pascal validation sets.

D. Appendix of Placing Objects in Context via Inpainting for Out-of-distribution Segmentation

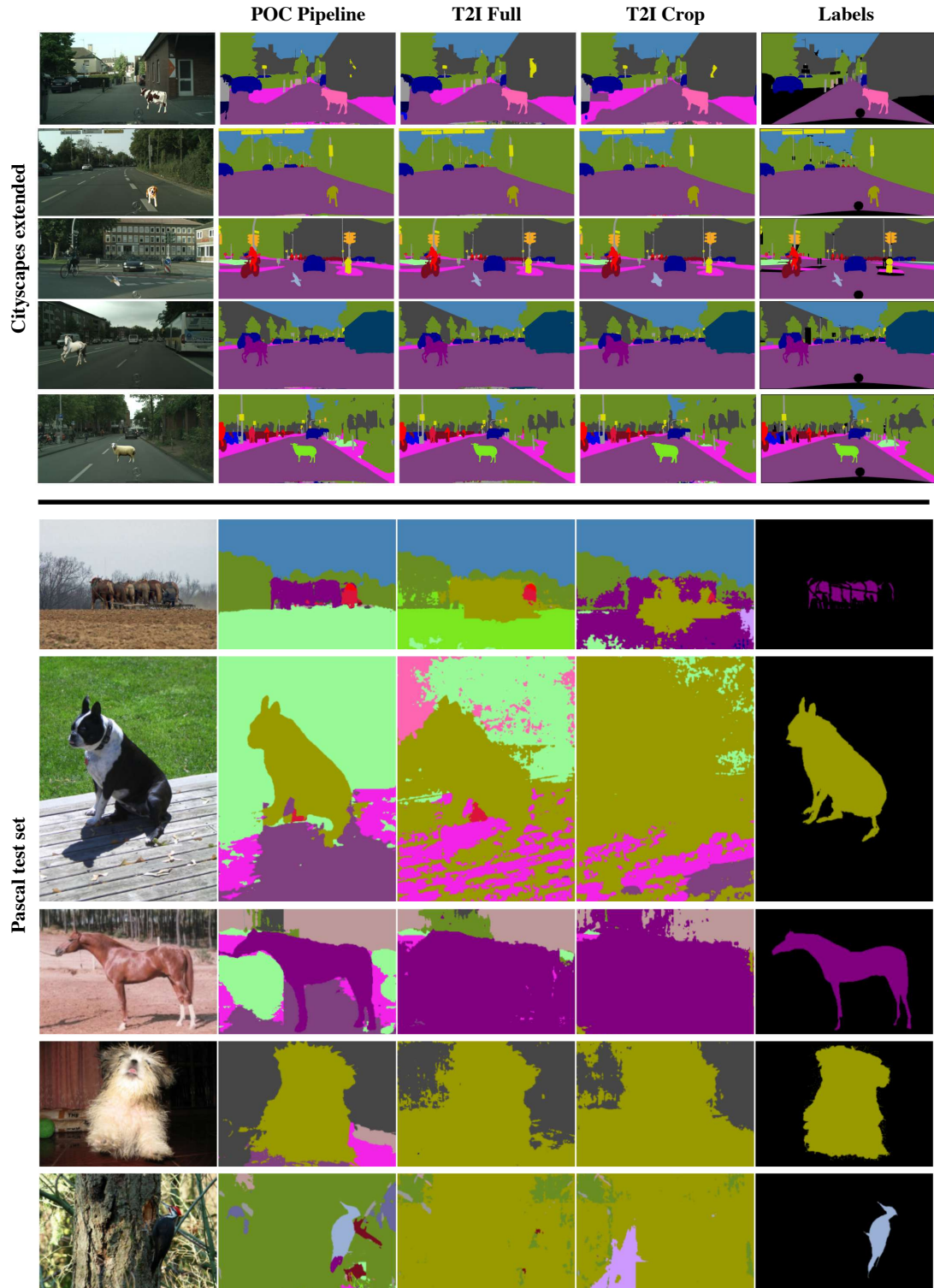


Figure D.20: ConvNeXt predictions on extended Cityscapes (*POC A*) and Pascal validation sets.

D. Appendix of Placing Objects in Context via Inpainting for Out-of-distribution Segmentation

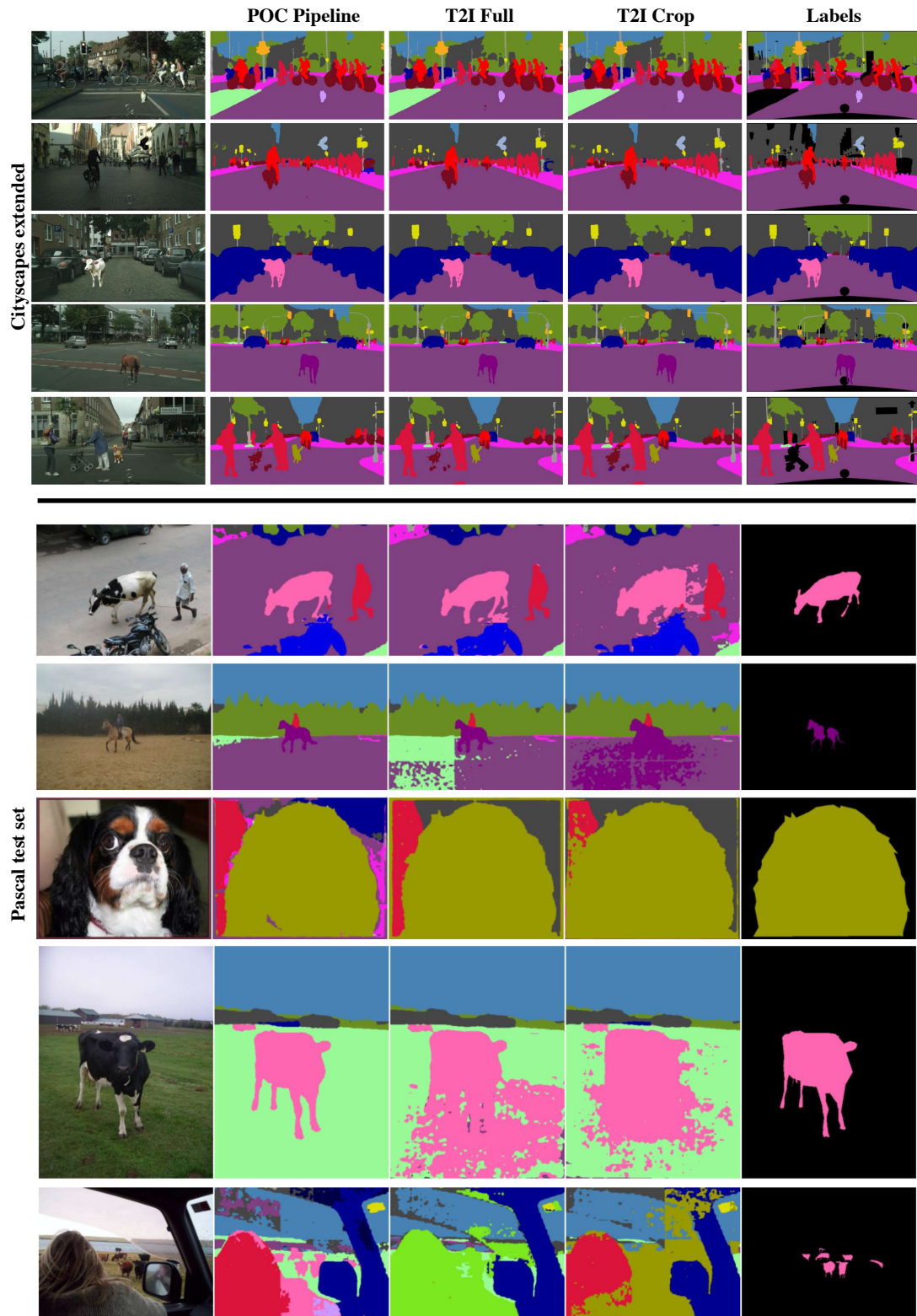


Figure D.21: Segmenter predictions on extended Cityscapes (*POC A*) and Pascal validation sets.

E

Catastrophic overfitting can be induced
with discriminative non-robust features

Abstract

Adversarial training (AT) is the de facto method for building robust neural networks, but it can be computationally expensive. To mitigate this, fast single-step attacks can be used, but this may lead to catastrophic overfitting (CO). This phenomenon appears when networks gain non-trivial robustness during the first stages of AT, but then reach a breaking point where they become vulnerable in just a few iterations. The mechanisms that lead to this failure mode are still poorly understood. In this work, we study the onset of CO in single-step AT methods through controlled modifications of typical datasets of natural images. In particular, we show that CO can be induced at much smaller ϵ values than it was observed before just by injecting images with seemingly innocuous features. These features aid non-robust classification but are not enough to achieve robustness on their own. Through extensive experiments we analyze this novel phenomenon and discover that the presence of these easy features induces a learning shortcut that leads to CO. Our findings provide new insights into the mechanisms of CO and improve our understanding of the dynamics of AT.

E.1 Introduction

Deep neural networks are sensitive to imperceptible worst-case perturbations, also known as adversarial perturbations (Szegedy et al., 2014). As a result, training neural networks that are robust to such perturbations has been an active area of study in recent years (see Ortiz-Jiménez et al. (2021) for a review). The most prominent line of research, referred to as adversarial training (AT), focuses on online data augmentation with adversarial samples during training. However, finding these samples for deep neural networks is an NP-hard problem (Weng et al., 2018). In practice, this is usually overcome with various methods, referred to as *adversarial attacks*, which find approximate solutions to this hard problem. The most popular attacks are based on projected gradient descent (PGD) (Madry et al., 2018), which is still a computationally expensive algorithm that requires multiple steps of gradient *ascent*. This hinders its use in many large-scale applications motivating the use of alternative efficient single-step attacks (Goodfellow et al., 2015).

The use of single-step attacks within AT, however, raises concerns regarding stability. Although single-step AT leads to an initial increase in robustness, the networks often reach a breaking point beyond which they lose all gained robustness in just a few additional training iterations (Wong et al., 2020). This phenomenon is known as catastrophic overfitting (CO) (Wong et al., 2020; Andriushchenko and Flammarion, 2020). Given the clear computational advantage of single-step attacks during AT, a significant body of work has been dedicated to finding ways to circumvent CO via regularization and data augmentation.

However, despite recent methodological advances, the mechanisms that lead to CO remain poorly understood. Nevertheless, due to the complexity of this problem, we argue that identifying such mechanisms cannot be done through observations alone and requires *active interventions* Ilyas et al. (2019). That is, we want to synthetically induce CO in settings where it would not naturally happen in order to explain its causes.

The key contribution of this work is exploring the root causes of CO, leading to the core message:

Catastrophic overfitting is a learning shortcut used by the network to avoid learning complex robust features while achieving high accuracy using easy non-robust ones.

In more detail, our main contributions are:

- (i) We demonstrate that CO can be induced by injecting features that are strong for standard classification but insufficient for robust classification.
- (ii) Through extensive empirical analysis, we find that CO is connected to the network’s preference for certain features in a dataset.
- (iii) We describe and analyse a mechanistic explanation of CO and its different prevention methods.

Our findings improve our understanding of CO by focusing on how data influences AT. They provide insights into the dynamics of AT, where the interaction between robust and non-robust features plays a key role, and open the door to promising future work that manipulates the data to avoid CO.

Outline The rest of the paper is structured as follows: In Appendix E.2, we review related work on CO. Appendix E.3 presents our main observation: CO can be induced by manipulating the data distribution by injecting simple features. In Appendix E.4, we perform an in-depth analysis of this phenomenon by studying the features used by networks trained with different procedures. In Appendix E.5, we link these observations to the evolution of the curvature of the networks and find that CO is a learning shortcut. Appendix E.6 pieces all these evidence together and provides a mechanistic explanation of CO. Finally, in Appendix E.7, we use our new perspective to provide new insights on different ways to prevent CO.

E.2 Preliminaries and related work

Let $f_{\boldsymbol{\theta}} : \mathbb{R}^d \rightarrow \mathcal{Y}$ denote a neural network parameterized by a set of weights $\boldsymbol{\theta} \in \mathbb{R}^n$ which maps input samples $\mathbf{x} \in \mathbb{R}^d$ to $y \in \mathcal{Y} = \{1, \dots, c\}$. The objective of adversarial training (AT) is to find the network parameters $\boldsymbol{\theta} \in \mathbb{R}^n$ that minimize the loss for the worst-case perturbations, that is:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \left[\max_{\|\boldsymbol{\delta}\|_p \leq \epsilon} \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x} + \boldsymbol{\delta}), y) \right], \quad (\text{E.1})$$

where $\mathcal{D}$ is a data distribution, $\boldsymbol{\delta} \in \mathbb{R}^d$ represents an adversarial perturbation, and (p, ϵ) characterize the adversary. This is typically solved by alternately minimizing the outer objective and maximizing the inner one via first-order optimization procedures. The outer minimization is tackled via a standard optimizer, *e.g.* SGD, while the inner maximization problem is approximated with adversarial attacks with one or more steps of *gradient ascent*. Single-step AT methods are built on top of FGSM (Goodfellow et al., 2015). In particular, FGSM solves the linearized version of the inner maximization objective (when $p = \infty$), *i.e.*

$$\boldsymbol{\delta}_{\text{FGSM}} = \underset{\|\boldsymbol{\delta}\|_{\infty} \leq \epsilon}{\operatorname{argmax}} \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}), y) + \boldsymbol{\delta}^{\top} \nabla_{\mathbf{x}} \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}), y) = \epsilon \operatorname{sign}(\nabla_{\mathbf{x}} \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}), y)).$$

FGSM is computationally efficient as it requires only a single forward-backward step. However, FGSM adversarial training (FGSM-AT) often yields networks that are vulnerable to multi-step attacks such as PGD (Madry et al., 2018). In particular, Wong et al. (2020) observed that FGSM-AT presents a failure mode where the robustness of the model increases during the initial training epochs, but then loses all robustness within a few iterations. This is known as catastrophic overfitting (CO). They further observed that augmenting the FGSM attack with random noise seemed to mitigate CO. However, Andriushchenko and Flammarion (2020) showed that this method still leads to CO at larger ϵ . Therefore, they proposed combining FGSM-AT with a smoothness regularizer (GradAlign) that encourages the cross-entropy loss to be locally linear. Although GradAlign successfully avoids CO, the smoothness regularizer adds a significant computational overhead.

Even more expensive, multi-step attacks approximate the inner maximization in (E.1) with several gradient ascent steps (Kurakin et al., 2017a; Madry et al., 2018; Zhang et al., 2019). If a sufficient number of steps are used, these methods do not suffer from CO and achieve better robustness. Due to their superior performance and extensively validated robustness (Madry et al., 2018; Tramèr et al., 2018; Zhang et al., 2019; Rice et al., 2020), multi-step methods such as PGD are considered the reference in AT. However, using multi-step attacks in AT linearly increases the cost of training with the number of steps. Other methods have been proposed to avoid CO while reducing the cost of AT. However, these methods either only move CO to larger ϵ radii (Golgooni et al., 2021), are more computationally expensive (Shafahi et al., 2019; Li et al., 2020), or achieve sub-optimal robustness (Kang and Moosavi-Dezfooli, 2021; Kim et al., 2021). Recently, de Jorge et al. (2022) proposed N-FGSM, which avoids CO without extra cost, however, the phenomena of CO remains poorly understood.

Other works have also analyzed the training dynamics when CO occurs. Initially, it was suggested that CO was a result of the networks overfitting to attacks limited to the corners of the ℓ_∞ ball (Wong et al., 2020). However, this conjecture was later dismissed by Andriushchenko and Flammarion (2020) who showed PGD-AT projected to the corners of the ℓ_∞ ball does not suffer from CO. Alternatively, they suggested that the reason Wong et al. (2020) avoids CO is the reduced radii of their perturbations on expectation, but de Jorge et al. (2022) showed that they could still prevent CO while using larger perturbations. Meanwhile, it has been consistently reported (Andriushchenko and Flammarion, 2020; Kim et al., 2021) that networks suffering from CO exhibit a highly non-linear loss landscape. As FGSM relies on the local linearity of the loss landscape, this sudden increase in non-linearity of the loss renders FGSM ineffective (Kim et al., 2021; Kang and Moosavi-Dezfooli, 2021). However, none of these works have managed to identify what causes the network to become strongly non-linear. In this work, we address this knowledge gap, and explore a plausible mechanism that can cause single-step AT to develop CO.

E.3 Inducing catastrophic overfitting

Our starting point is the observation that robust solutions are not the default outcome of standard training and are avoided unless explicitly enforced through AT. That is, robust solutions are *hard to learn*. We also know that robust classification requires leveraging alternative robust features that are not learned in standard training (Ilyas et al., 2019; Sanyal et al., 2021). Finally, we know that when CO occurs, the robust accuracy drops while the clean and FGSM accuracy increases (Wong et al., 2020; Andriushchenko and Flammarion, 2020). With that in mind, we pose the question: *Can CO be a mechanism to avoid learning complex robust features?* If so, the network may be using CO as a way to favor easy non-robust features over complex robust ones.

Directly testing this hypothesis is difficult as it requires identifying and characterizing robust and non-robust features in a real dataset. This is a challenging task, equivalent to actually solving the problem of adversarial robustness. To overcome this, we follow the standard practice in the field and use controlled modifications of the data (Arpit et al., 2017; Ilyas et al., 2019; Shah et al., 2020; Ortiz-Jimenez et al., 2020a). By conducting experiments on the manipulated data, we can make claims about CO and its causes. Specifically, we find that by injecting a simple discriminative feature on standard vision datasets, we can induce CO at lower values of ϵ (*i.e.* with smaller perturbations) than those at which it naturally occurs. This suggests that the structure of the data plays a significant role in the onset of CO.

Our injected dataset Let $(\mathbf{x}, y) \sim \mathcal{D}$ be an image-label pair sampled from a distribution $\mathcal{D}$. We modify the data by adding a label-dependent feature $\mathbf{v}(y)$, scaled by a parameter $\beta > 0$, and construct a family of *injected datasets* $\tilde{\mathcal{D}}_\beta$:

$$(\tilde{\mathbf{x}}, y) \sim \tilde{\mathcal{D}}_\beta : \quad \tilde{\mathbf{x}} = \mathbf{x} + \beta \mathbf{v}(y) \quad \text{with} \quad (\mathbf{x}, y) \sim \mathcal{D}. \quad (\text{E.2})$$

We design $\mathbf{v}(y)$ to be linearly separable with $\|\mathbf{v}(y)\|_p = 1$ for all $y \in \mathcal{Y}$. We mainly use $p = \infty$ in the paper as CO has been observed more often in this setting (Wong et al.,

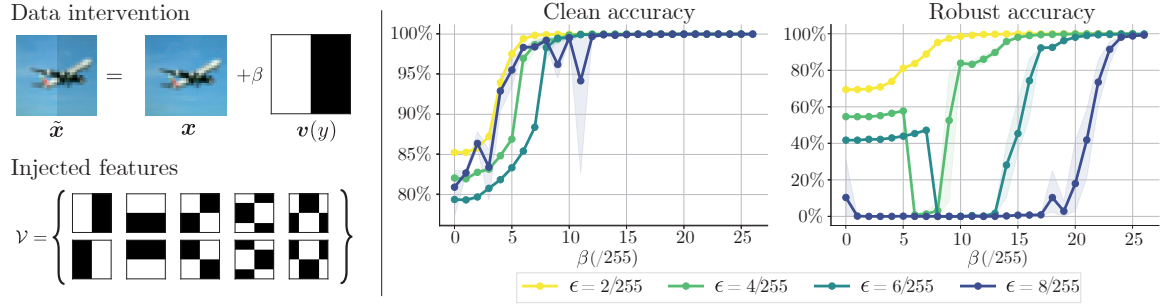


Figure E.1: **Left:** Modified dataset with injected simple and discriminative features. **Right:** Clean and robust performance after FGSM-AT on injected datasets $\tilde{\mathcal{D}}_\beta$. We vary the strength of the synthetic features β ($\beta = 0$ corresponds to the original CIFAR-10) and the robustness budget ϵ . For $\epsilon \in \{4/255, 6/255\}$, our intervention can induce CO when the synthetic features have strength $\beta \approx \epsilon$, even when the original data does not suffer CO. Results averaged over 3 seeds. Shades show min-max values.

2020; Andriushchenko and Flammarion, 2020), but we also present some results with $p = 2$ in Appendix F.4. The set of all injected features is denoted as $\mathcal{V} = \{\mathbf{v}(y) \mid y \in \mathcal{Y}\}$. The scale parameter $\beta > 0$ is fixed for all classes and controls the relative strength of the original and injected features, *i.e.* $\mathbf{x}$ and $\mathbf{v}(y)$, respectively (see Figure E.1 (left)).

Injecting features that are linearly separable and perfectly correlated with the labels induces some interesting properties. In short, although $\mathbf{v}(y)$ is easy-to-learn, the discriminative power and robustness of a classifier that solely relies on the injected features $\mathcal{V}$ depends on the scale parameter β . Indeed, a linear classifier relying only on $\mathcal{V}$ can separate $\tilde{\mathcal{D}}_\beta$ for a large enough β . However, if $\mathbf{x}$ has some components in $\text{span}(\mathcal{V})$, the interaction between $\mathbf{x}$ and $\mathbf{v}(y)$ may decrease the robustness of the classifier for small β . We rigorously illustrate this behavior in Appendix F.1.

To control the interaction between $\mathbf{x}$ and $\mathbf{v}(y)$, we design $\mathcal{V}$ by selecting the first vectors from the low-frequency components of the 2D Discrete Cosine Transform (DCT) (Ahmed et al., 1974), depicted in Figure E.1 (left), which have a large alignment with the space of natural images in our experiments (*e.g.* CIFAR-10). To ensure the norm constraint, we binarize these vectors to have only ± 1 values, resulting in a maximal per-pixel perturbation that satisfies $\|\mathbf{v}(y)\|_\infty = 1$. These design constraints

also make it easy to visually identify the alignment of adversarial perturbations δ with $\mathbf{v}(y)$ as they have distinctive patterns.

Injection strength (β) drives CO We train a PreActResNet18 (He et al., 2016b) on different intervened versions of CIFAR-10 (Krizhevsky et al., 2009) using FGSM-AT for different robustness budgets ϵ and scales β . Figure E.1 (right) shows the results of these experiments in terms of clean accuracy and robustness¹. In terms of *clean accuracy*, Figure E.1 (right) shows two regimes. First, when $\beta < \epsilon$, the network achieves roughly the same accuracy by training and testing on $\tilde{\mathcal{D}}_\beta$ as by training and testing on $\mathcal{D}$ (corresponding to $\beta = 0$), *i.e.* the network ignores the added features $\mathbf{v}(y)$. Meanwhile, when $\beta > \epsilon$, the clean accuracy reaches 100% indicating that the network relies on the injected features and ignores $\mathcal{D}$. This is further shown empirically in Appendix E.4 and Appendix F.6.

The behavior with respect to *robust accuracy* is even more interesting. For small ϵ ($\epsilon = 2/255$), the robust accuracy shows the same trend as the clean accuracy, albeit with lower values. For large ϵ ($\epsilon = 8/255$), the model incurs CO for most values of β . This is not surprising as CO has already been reported for this value of ϵ on the original CIFAR-10 dataset (Wong et al., 2020). However, the interesting setting is for intermediate values of ϵ ($\epsilon \in \{4/255, 6/255\}$). For these settings, Figure E.1 (right) distinguishes three regimes:

- (i) When the injected features are weak ($\beta \ll \epsilon$), the robust accuracy is similar as with the original data.
- (ii) When they are strong ($\beta \gg \epsilon$), robustness is high as using $\mathbf{v}(y)$ is enough to classify $\tilde{\mathbf{x}}$ robustly.
- (iii) When they are mildly robust ($\beta \approx \epsilon$), training suffers from CO and the robust accuracy drops to zero.

¹Robustness measured using PGD with 50 steps and 10 restarts.

This last regime is very significant, as training on the original dataset $\mathcal{D}$ ($\beta = 0$) does not suffer from CO for this value of ϵ . In the following section, we delve deeper into these results to better understand the cause of CO.

Inducing CO in other settings To ensure generality, we replicate our previous experiment using different settings. Specifically, we show in Appendix F.4 that our intervention can induce CO in other datasets such as CIFAR-100 and SVHN (Netzer et al., 2011), and higher-resolution ones such as TinyImageNet (Li et al., 2020) and ImageNet-100 (Kireev et al., 2022). We also reproduce the effect using larger networks like a WideResNet28x10 (Zagoruyko and Komodakis, 2016) and for ℓ_2 perturbations. We, therefore, confidently conclude that there is a link between the structure of the data and CO, independently of actual datasets, and models.

E.4 Networks prefer injected features

Since we now have a method to intervene in the data using (E.2) and induce CO, we can use it to better characterize the mechanisms that lead to CO. The previous experiments showed that when $\beta \ll \epsilon$ or $\beta \gg \epsilon$, our data intervention does not induce CO. However, when $\beta \approx \epsilon$, FGSM-AT consistently experiences CO. This raises the following question: what makes $\beta \approx \epsilon$ special? We now show that for $\beta \approx \epsilon$, a network trained using AT uses information from both the original dataset $\mathcal{D}$ and the injected features in $\mathcal{V}$ to achieve high robust accuracy on the injected dataset $\tilde{\mathcal{D}}_\beta$. However, when trained without any adversarial constraints *i.e.* for standard training, the network only uses the features in $\mathcal{V}$ and achieves close to perfect clean accuracy.

In order to understand what features are learned when training on the injected dataset $\tilde{\mathcal{D}}_\beta$, we evaluate the performance of different models on three distinct test sets: (i) CIFAR-10 test set with injected features ($\tilde{\mathcal{D}}_\beta$), (ii) original CIFAR-10 test set ($\mathcal{D}$), and (iii) CIFAR-10 test set with shuffled injected features ($\tilde{\mathcal{D}}_{\pi(\beta)}$) where the additive signals are correlated with a *permuted* set of labels, *i.e.*

$$(\tilde{\mathbf{x}}^{(\pi)}, y) \sim \tilde{\mathcal{D}}_{\pi(\beta)} : \quad \tilde{\mathbf{x}}^{(\pi)} = \mathbf{x} + \beta \mathbf{v}(\pi(y)), \quad (\text{E.3})$$

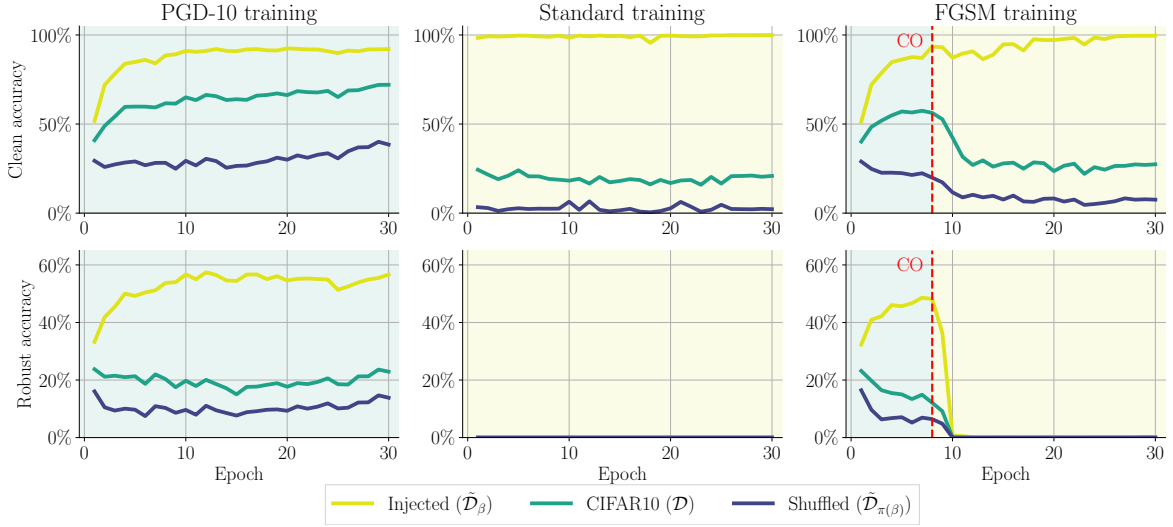


Figure E.2: Clean (**top**) and robust (**bottom**) accuracy on 3 different test sets: (i) original CIFAR-10 ($\mathcal{D}$), (ii) dataset with injected features $\tilde{\mathcal{D}}_\beta$ and (iii) dataset with shuffled injected features $\tilde{\mathcal{D}}_{\pi(\beta)}$. All runs use $\beta = 8/255$ and $\epsilon = 6/255$. Blue shading denotes the use of both $\mathcal{D}$ and $\mathcal{V}$ and yellow only $\mathcal{V}$.

with $(\mathbf{x}, y) \sim \mathcal{D}$ and $\mathbf{v} \in \mathcal{V}$. Here, $\pi : \mathcal{Y} \rightarrow \mathcal{Y}$ is a fixed permutation to shuffle the labels. Evaluating these models on $\tilde{\mathcal{D}}_{\pi(\beta)}$ exposes them to contradictory information, since $\mathbf{x}$ and $\mathbf{v}(\pi(y))$ are correlated with different labels². Thus, if the classifier only relies on $\mathcal{V}$, the performance should be high on $\tilde{\mathcal{D}}_\beta$ and low on $\mathcal{D}$ and $\tilde{\mathcal{D}}_{\pi(\beta)}$, while if it only relies on $\mathcal{D}$, the performance should remain constant for all injected datasets. Figure E.2 shows the results of this evaluation during standard, FGSM-AT and PGD-AT on $\tilde{\mathcal{D}}_\beta$ with $\beta = 8/255$ and $\epsilon = 6/255$.

PGD training: We can conclude that the PGD-AT network achieves a robust solution using both $\mathcal{D}$ and $\mathcal{V}$ from Figure E.2 (left). It uses $\mathcal{D}$ as it achieves better than trivial accuracy on $\mathcal{D}$ and $\tilde{\mathcal{D}}_{\pi(\beta)}$. But it also uses $\mathcal{V}$ as it achieves higher accuracy on $\tilde{\mathcal{D}}_\beta$ than $\mathcal{D}$, and suffers from a drop in performance on $\tilde{\mathcal{D}}_{\pi(\beta)}$. This implies that this network effectively combines information from both the original and injected features for classification.

Standard training: Standard training shows a different behaviour than PGD-AT (see Figure E.2 (center)). In this case, the network achieves excellent clean accuracy on $\tilde{\mathcal{D}}_\beta$, but trivial accuracy on $\mathcal{D}$, indicating that it ignores the information present in $\mathcal{D}$

²We also evaluate on $\mathcal{V}$ alone in Appendix F.5.

and only uses non-robust features from $\mathcal{V}$ for classification. This is further supported by its near-zero accuracy on $\tilde{\mathcal{D}}_{\pi(\beta)}$. We conclude that the injected features are easy to learn *i.e.* they are preferred by standard training.

FGSM training: The behaviour of FGSM-AT in Figure E.2 (right) highlights further the preference for the injected features. FGSM-AT undergoes CO around epoch 8 and its robust accuracy on $\tilde{\mathcal{D}}_\beta$ drops to zero despite its high clean accuracy on $\tilde{\mathcal{D}}_\beta$. It presents two distinct phases:

- (i) Prior to CO the robust accuracy on $\tilde{\mathcal{D}}_\beta$ is non-zero and the network exploits both $\mathcal{D}$ and $\mathcal{V}$, similar to PGD-AT.
- (ii) After CO both the clean and robust accuracy on $\mathcal{D}$ and $\tilde{\mathcal{D}}_{\pi(\beta)}$ drop, exhibiting behavior similar to standard training. This indicates that, after CO, the network forgets $\mathcal{D}$ and solely relies on the features in $\mathcal{V}$.

Why does FGSM change the learned features after CO? From the behaviour of standard training, we concluded that the network has a preference for the injected features $\mathcal{V}$, which are easy to learn. Meanwhile, the behaviour of PGD training suggests that when the easy features are not sufficient for robust classification, the model combines them with other (harder-to-learn) features, such as those in $\mathcal{D}$. FGSM initially learns a robust solution using both $\mathcal{D}$ and $\mathcal{V}$, similar to PGD. However, *if the FGSM attacks are rendered ineffective, the robustness constraints are removed*, allowing the network to revert back to the simple features and the performance on the original dataset $\mathcal{D}$ drops. This is exactly what occurs with the onset of CO around epoch 8, which we will further discuss in Appendix E.5.1. As we will see, the key to understanding why CO happens lies in how learning each type of feature influences the local geometry of the classifier.

E.5 Geometry of CO

Recent works have shown that after the onset of CO, the local geometry around the input $\mathbf{x}$ becomes highly non-linear (Andriushchenko and Flammarion, 2020; Kim et al.,

2021; de Jorge et al., 2022). Motivated by these, and to better understand induced CO, we investigate the evolution of the curvature of the loss landscape when this happens.

E.5.1 Curvature explosion drives CO

To measure curvature, we use the average maximum eigenvalue of the Hessian on $N = 100$ training points $\bar{\lambda}_{\max} = \frac{1}{N} \sum_{n=1}^N \lambda_{\max} \left(\nabla_{\tilde{\mathbf{x}}}^2 \mathcal{L}(f_{\theta}(\tilde{\mathbf{x}}_n), y_n) \right)$ as in Moosavi-Dezfooli et al. (2019)), and record it throughout training. Figure E.3 (left) shows the result of this experiment for FGSM-AT (orange line) and PGD-AT (green line) training on $\tilde{\mathcal{D}}_{\beta}$ where CO with FGSM-AT happens around epoch 8.

Two-phase curvature increase Interestingly, we observe a steep increase in curvature for *both* FGSM-AT and PGD-AT (the y -axis is in logarithmic scale) before CO. However, while there is a large increase in curvature for PGD-AT right before the 8th epoch, it stabilizes soon after – PGD-AT acts as a regularizer on the curvature (Moosavi-Dezfooli et al., 2019; Qin et al., 2019) which explains how it controls the curvature explosion. However, curvature is a second-order property of the loss and unlike PGD, FGSM is based on a coarse linear (first order) approximation of the loss. In this regard, we see that FGSM-AT cannot contain the curvature increase, which eventually explodes around the 8th epoch and saturates at a very large value. The final curvature of the FGSM-AT model is 100 times that of the PGD-AT model.

High curvature leads to meaningless perturbations The fact that the curvature increases rapidly alongside the FGSM accuracy during CO agrees with the findings of Andriushchenko and Flammarion (2020). The loss becomes highly non-linear and thus reduces the success rate of FGSM, which assumes a linear loss. To show that CO indeed occurs due to the increased curvature breaking FGSM, we visualise the adversarial perturbations before and after CO. As observed in Figure E.4, before CO, the adversarial perturbations are non-trivially aligned with $\text{span}(\mathcal{V})$, albeit with some corruptions originating from $\mathbf{x}$. Nonetheless, after CO, the new adversarial

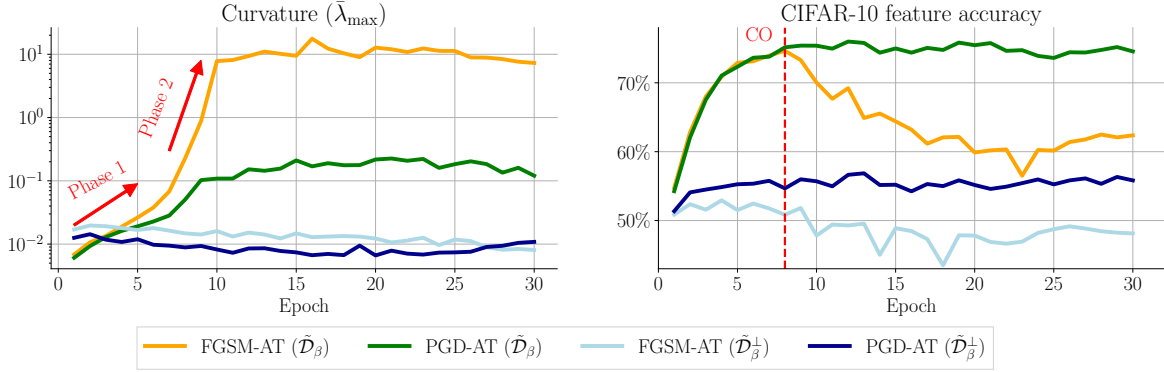


Figure E.3: Evolution of metrics for FGSM-AT and PGD-AT on dataset with injected features ($\tilde{\mathcal{D}}_\beta$) and with orthogonally projected features ($\tilde{\mathcal{D}}_\beta^\perp$), where there is no interaction between features. AT is performed with $\beta = 8/255$ and $\epsilon = 6/255$.



Figure E.4: Samples of the injected dataset $\tilde{\mathcal{D}}_\beta$ (**top**) and FGSM perturbations before (**middle**) and after CO (**bottom**). Before CO, perturbations focus on synthetic features, but after CO they become noisy.

perturbations point towards meaningless directions; they do not align with $\mathcal{V}$ even though the network is heavily reliant on this information for classifying the data³ (cf. Appendix E.4). This reinforces the idea that the increase in curvature causes a breaking point after which FGSM is no longer effective. This behavior of adversarial perturbations after CO is different from the one on standard and PGD-AT networks in which the perturbations align with discriminative directions (Fawzi et al., 2018; Jetley et al., 2018; Ilyas et al., 2019; Ortiz-Jimenez et al., 2020b).

³We also quantify this alignment in Figure E.7.

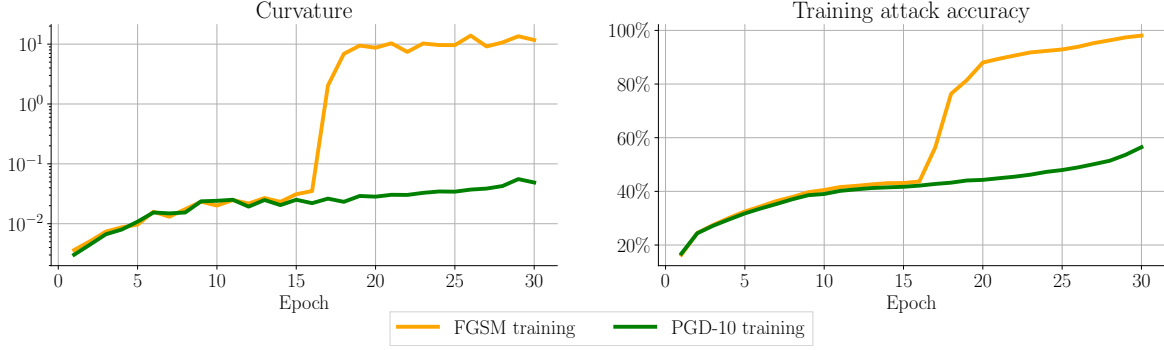


Figure E.5: Evolution of curvature and training attack accuracy of FGSM-AT and PGD-AT trained on the original CIFAR-10 with $\epsilon = 8/255$. When CO happens the curvature explodes.

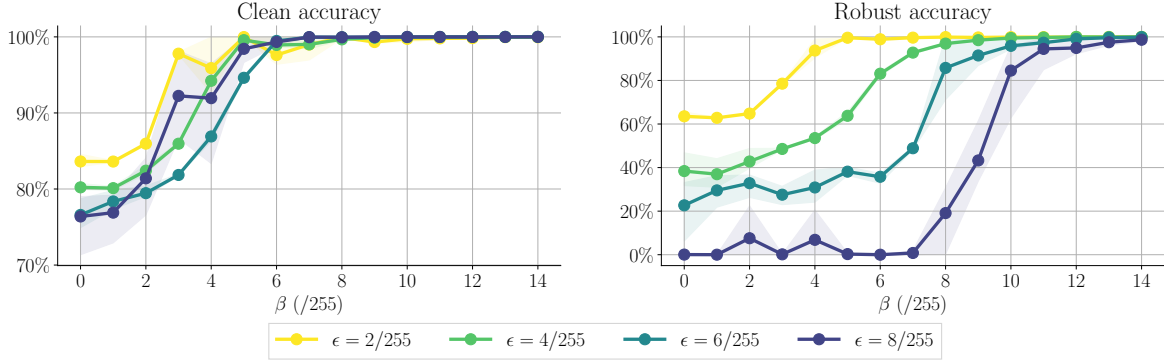


Figure E.6: Clean (**left**) and robust (**right**) accuracy after FGSM-AT on a dataset with orthogonally injected features $\tilde{\mathcal{D}}_{\beta}^{\perp}$ i.e. no interaction between original and injected features. We train while varying the strength of the injected features and the robustness budget ϵ .

Two-phase curvature increase also occurs in naturally occurring CO In Figure E.5 we show the curvature when training on the original CIFAR-10 dataset with $\epsilon = 8/255$ (where CO happens for FGSM-AT). Similarly to our observations on the injected datasets, the curvature during FGSM-AT explodes along with the training accuracy while for PGD-AT the curvature increases at a very similar rate than FGSM-AT during the first epochs and later stabilizes. This indicates that our described mechanisms may as well apply to induce CO on natural image datasets.

E.5.2 Curvature increase is a result of feature interaction

But why does the network increase the curvature in the first place? In Appendix E.4, we observed that this is a shared behavior of PGD-AT and FGSM-AT, at least during the initial stage before CO. Therefore, it should not be a mere “bug”. We conjecture

that the curvature increase is a result of the interaction between features of the dataset which forces the network to increase its non-linearity in order to combine them effectively for robustness.

Curvature does not increase without interaction To demonstrate this, we perform a new experiment in which we modify $\mathcal{D}$ again (as in Appendix E.3). However, this time, we ensure that there is no interaction between the synthetic features $\mathbf{v}(y)$ and the features from $\mathcal{D}$ creating $\tilde{\mathcal{D}}_\beta^\perp$ such that:

$$(\tilde{\mathbf{x}}^\perp, y) \sim \tilde{\mathcal{D}}_\beta^\perp : \quad \tilde{\mathbf{x}}^\perp = \mathcal{P}_{\mathcal{V}^\perp}(\mathbf{x}) + \beta \mathbf{v}(y) \quad (\text{E.4})$$

with $(\mathbf{x}, y) \sim \mathcal{D}$ and $\mathbf{v}(y) \in \mathcal{V}$, where $\mathcal{P}_{\mathcal{V}^\perp}$ denotes the projection operator onto the orthogonal complement of $\mathcal{V}$. Since the synthetic features $\mathbf{v}(y)$ are orthogonal to $\mathcal{D}$, a simple linear classifier relying only on $\mathbf{v}(y)$ can robustly separate the data up to a radius that depends solely on β (see the theoretical construction in Appendix F.1).

Interestingly, we find that, in this dataset, none of the (β, ϵ) configurations used in Figure E.6 induce CO. Here, we observe only two regimes: one that ignores $\mathcal{V}$ (when $\beta < \epsilon$) and one that ignores $\mathcal{D}$ (when $\beta > \epsilon$). This supports our conjecture that the interaction between the features of $\mathbf{x}$ and $\mathbf{v}(y)$ causes CO in $\tilde{\mathcal{D}}_\beta$. Moreover, Figure E.3 (left) shows that, when performing either FGSM-AT (light blue) or PGD-AT (dark blue) on $\tilde{\mathcal{D}}_\beta^\perp$, the curvature is consistently low. This agrees with the fact that in this case there is no need for the network to combine the injected and original features for robustness and hence the network does not need to increase its non-linearity to separate the data.

Non-linear feature extraction Finally, we study the relationship between the features extracted by the network and its curvature. We train multiple logistic classifiers to classify $\mathcal{D}$ based on the feature representations (output of the penultimate layer) of networks trained on $\tilde{\mathcal{D}}_\beta$ as in Shi et al. (2022). The *feature accuracy* of these classifiers will depend on how well the network has learned to extract information from $\mathcal{D}$.

Figure E.3 (right) shows that for the PGD-AT network (green), the feature accuracy on $\mathcal{D}$ progressively increases over time, indicating that the network has learned to

extract meaningful information from $\mathcal{D}$ even though it was trained on $\tilde{\mathcal{D}}_\beta$. The feature accuracy also closely matches the curvature trajectory in Figure E.3 (left). In contrast, the FGSM-AT network (red) exhibits two phases on its feature accuracy: Initially, it grows at a similar rate as the PGD-AT network, but after CO occurs, it starts to decrease. This decrease in feature accuracy does not correspond with a decrease in curvature, which we attribute to the network taking a shortcut and ignoring $\mathcal{D}$ in favor of easy, non-robust features.

When we use features from networks trained on $\tilde{\mathcal{D}}_\beta^\perp$, we find that the accuracy on $\mathcal{D}$ is consistently low, suggesting that the network is increasing its curvature to improve its feature representation as it must combine information from $\mathcal{V}$ with $\mathcal{D}$ to become robust in this case.

E.6 A mechanistic explanation of CO

Based on the previous insights, we can finally summarize the chain of events that leads to CO in our injected datasets:

- (i) To learn a robust solution, the network combine easy non-robust features with complex robust ones. However, without robustness constraints, the network favors learning *only* the non-robust features (see Appendix E.4).
- (ii) When learning both kinds of features simultaneously, the network increases its non-linearity to improve its feature extraction ability (see Appendix E.5.2).
- (iii) This increase in non-linearity provides a shortcut to break FGSM, triggering CO and allowing the network to avoid learning the complex robust features while still achieving a high accuracy using only the easy non-robust ones (see Appendix E.5.1).

How general is this mechanism? So far, all our experiments have dealt with datasets that have been synthetically intervened by injecting artificial signals. However, we advocate that our work provides sufficient and compelling evidence that features are the most likely cause of CO in general. Deep learning is an empirical

discipline, where many insights are gained from experimental studies, and as shown by prior work (Arpit et al., 2017; Ilyas et al., 2019; Shah et al., 2020; Ortiz-Jimenez et al., 2020a), manipulating data in controlled ways is a powerful tool to infer the structure of its features. In this light, we are confident our work provides robust and comprehensive evidence that features are likely the primary catalyst of CO across different datasets.

Our study includes extensive ablations with various types of injected features, such as random directions, and different norms and we have shown these settings also lead to CO (see Appendix F.4). Furthermore, we have observed that the increase in curvature within our manipulated datasets mirrors the patterns found in naturally occurring CO instances, as demonstrated in Figure E.5.

Beyond our empirical investigations, we also provide theoretical support for our conjecture in Appendix F.2. The concept that a robust classifier may need to combine different features can be formally proven in certain scenarios. Specifically, in Appendix F.2, we offer a mathematical proof showing that numerous learning scenarios exist in which learning a robust classifier requires leveraging additional non-linear features on top of the simple ones used for the clean solution.

Finally, in what follows, we will present further evidence that CO is generally induced by features. Specifically, we will show that other feature manipulations besides the injection of features can also influence CO. Additionally, we will show that the primary strategies effective at preventing CO in natural datasets can also mitigate CO in our manipulated datasets. This suggests a commonality between these types of CO, further underscoring the widespread relevance of our mechanism

E.7 Analysis of CO prevention methods

Our proposed dataset intervention, defined in Appendix E.3, has allowed us to better understand the chain of events that lead to CO. In this section, we will focus on methods to prevent CO and analyze them in our framework for further insights.

Table E.1: Clean and robust accuracies of FGSM-AT and PGD-AT trained networks on CIFAR-10 and the low pass version described in Ortiz-Jimenez et al. (2020b) at different ϵ .

Method (ϵ)	Original		Low pass	
	Clean	Robust	Clean	Robust
FGSM ($8/255$)	85.6	0.0	81.1	47.0
PGD ($8/255$)	80.9	50.6	80.3	49.7
FGSM ($16/255$)	76.3	0.0	78.6	0.0
PGD ($16/255$)	68.0	29.2	66.9	28.4

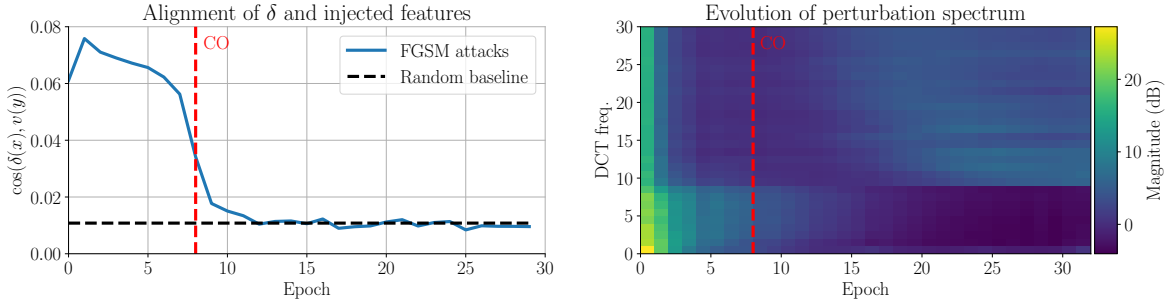


Figure E.7: Quantitative analysis of the directionality of FGSM perturbations before and after CO in CIFAR-10 with $\beta = 8/255$ and $\epsilon = 6/255$. **Left:** Alignment of FGSM perturbations with injected features during training. **Right:** Magnitude of the DCT spectrum of FGSM perturbations during training. The plot shows values for the diagonal components at every epoch.

High-frequency features and CO Recently, Grabinski et al. (2022) suggested that CO could be caused by aliasing effects from downsampling layers and proposed using a low-pass filter before pooling. We take this theory a step further and show that just removing the high-frequency components from CIFAR-10 consistently prevents CO at $\epsilon = 8/255$ (where FGSM-AT fails). Furthermore, in Figure E.7 (right) we observe that after CO on $\tilde{\mathcal{D}}_\beta$, FGSM attacks ignore low-frequencies and become increasingly high-frequency.

Surprisingly, though, applying the same low-pass technique at $\epsilon = 16/255$ does not work and neither does Grabinski et al. (2022) (see Appendix F.10). We conjecture this is because the features robust at $\epsilon = 8/255$ in the low pass version of CIFAR-10 might not be robust at $\epsilon = 16/255$ and although high-frequency features might contribute to CO, the overall set of CO inducing features is more complex. Additionally, we observe that CO can be induced even when the injected features are randomly sampled (see Figure F.6). Although removing low-frequency features does not work in all

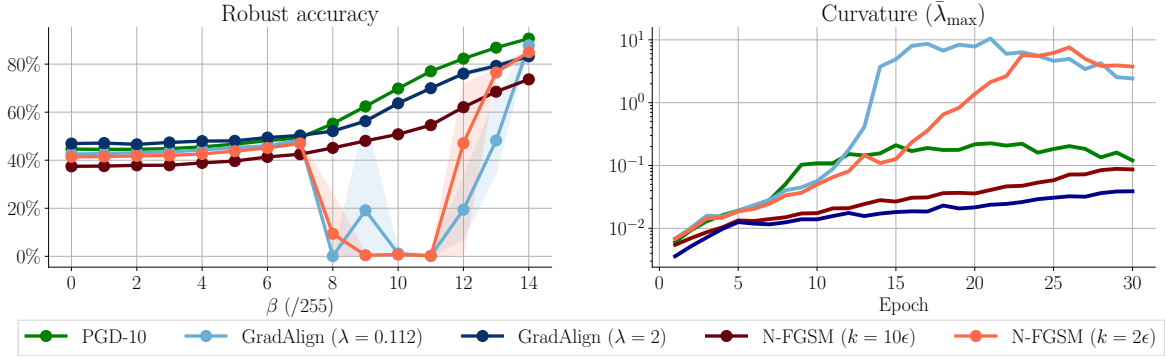


Figure E.8: Left: Clean and robust accuracy of GradAlign, N-FGSM and PGD-AT. **Right:** Curvature evolution of all methods. Training performed on $\tilde{\mathcal{D}}_\beta$ with $\epsilon = 6/255$ and $\beta = 8/255$ and averaged over 3 random seeds.

settings, we see this as a proof of concept that manipulating the data itself may prevent CO in some settings. Generalizing this idea to other datasets is a promising avenue for future work.

Preventing CO in our injected dataset In Figure E.2 (left), we have shown that using PGD-AT prevents CO on our injected dataset, but we now also evaluate the effectiveness of GradAlign and N-FGSM. Our results indicate that these methods can also be successful in preventing CO, but need stronger regularization for certain values of β . As β grows, the injected feature $\mathbf{v}(y)$ becomes more discriminative, creating a stronger incentive for the network to use it. This increases the chances of CO as the network increases its curvature to combine the injected features with others in order to become robust. Figure E.8 shows that the curvature of N-FGSM and GradAlign decreases with stronger regularizations, similar to PGD-AT. This supports the idea that preventing the curvature from exploding can indeed prevent CO.

Stronger CO on high-resolution datasets As discussed in Appendix E.3, we observe CO when injecting discriminative features in several datasets. Interestingly, for ImageNet-100, we have observed that even multi-step attacks like PGD are not able to find adversarial perturbations after CO (see Figure E.9). However, when using AutoAttack (AA) (Croce and Hein, 2020), we find that models are not robust⁴. To the best of our knowledge, this had not been observed before with “vanilla” CO, where

⁴On other datasets, AA and PGD work equally.

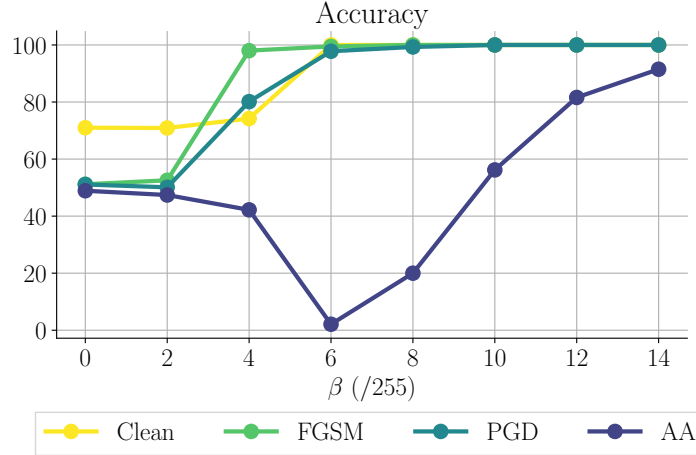


Figure E.9: Clean, FGSM, PGD and AutoAttack accuracy on injected ImageNet-100 with varying β . FGSM-AT performed with $\epsilon = 4/255$.

after CO, models would be vulnerable to very weak PGD attacks, like PGD with 10 steps (Wong et al., 2020). We argue that in this case, the curvature increase that breaks FGSM also hinders PGD optimization. This aligns with our findings with N-FGSM and GradAlign, where stronger regularization is needed to prevent CO with the injected datasets, and suggests that CO is *stronger* in $\tilde{\mathcal{D}}_\beta$.

E.8 Concluding remarks

In this work, we have presented a thorough empirical study to establish a link between the features of the data and the onset of CO in FGSM-AT. Using controlled data interventions, we have shown that CO is a learning shortcut used by the network to avoid learning complex robust features while achieving high accuracy using easy non-robust ones. This new perspective sheds new light on the mechanisms that trigger CO, as it shifts our focus towards studying the way the data structure influences the learning algorithm. We believe this opens the door to promising future work towards understanding the intricacies of these mechanisms, deriving methods for inspecting data and identifying feature interactions, and finding faster ways to make neural networks robust.

F

Appendix of Catastrophic overfitting can be induced with discriminative non-robust features

F.1 Analysis of the separability of the injected datasets

With the aim to illustrate how the interaction between $\mathcal{D}$ and $\mathcal{V}$ can influence the robustness of a classifier trained on $\tilde{\mathcal{D}}_\beta$ we now provide a toy theoretical example in which we discuss this interaction. Specifically, without loss of generality, consider the binary classification setting on the dataset $(\mathbf{x}, y) \sim \mathcal{D}$ where $y \in \{-1, +1\}$ and $\|\mathbf{x}\|_2 = 1$, for ease. Let's now consider the injected dataset $\tilde{\mathcal{D}}_\beta$ and further assume that $\mathbf{v}(+1) = \mathbf{u}$ and $\mathbf{v}(-1) = -\mathbf{u}$ with $\mathbf{u} \in \mathbb{R}^d$ and $\|\mathbf{u}\|_2 = 1$, such that $\tilde{\mathbf{x}} = \mathbf{x} + \beta y \mathbf{u}$. Moreover, let $\gamma \in [0, 1]$ denote the *interaction coefficient* between $\mathcal{D}$ and $\mathcal{V}$, such that $-\gamma \leq \mathbf{x}^\top \mathbf{u} \leq \gamma$.

We are interested in characterizing the robustness of a classifier that only uses information in $\mathcal{V}$ when classifying $\tilde{\mathcal{D}}_\beta$ depending on the strength of the interaction coefficient. In particular, as we are dealing with the binary setting, we will characterize

the robustness of a linear classifier $h : \mathbb{R}^d \rightarrow \{-1, +1\}$ that discriminates the data based only on $\mathcal{V}$, i.e. $h(\tilde{\mathbf{x}}) = \text{sign}(\mathbf{u}^\top \tilde{\mathbf{x}})$. In our setting, we have

$$\begin{aligned}\mathbf{u}^\top \tilde{\mathbf{x}} &= \mathbf{u}^\top \mathbf{x} + \beta \mathbf{u}^\top \mathbf{u} = \mathbf{u}^\top \mathbf{x} + \beta \quad \text{if } y = +1 \\ \mathbf{u}^\top \tilde{\mathbf{x}} &= \mathbf{u}^\top \mathbf{x} - \beta \mathbf{u}^\top \mathbf{u} = \mathbf{u}^\top \mathbf{x} - \beta \quad \text{if } y = -1\end{aligned}$$

Proposition 2 (Clean performance). If $\beta > \gamma$, then h achieves perfect classification accuracy on $\tilde{\mathcal{D}}_\beta$.

Proof. Observe that if $\gamma = 0$, i.e. the features from original dataset $\mathcal{D}$ do not interact with the injected features $\mathcal{V}$, the dataset is perfectly linearly separable. However, if the data $\mathbf{x}$ from $\mathcal{D}$ interacts with the injected signal $\mathbf{u}$, i.e. non zero projection, then the dataset is still perfectly separable but for a sufficiently larger β , such that $\mathbf{u}^\top \mathbf{x} + \beta > 0$ when $y = +1$ and $\mathbf{u}^\top \mathbf{x} + \beta < 0$ when $y = -1$. Because $-\gamma \leq \mathbf{x}^\top \mathbf{u} \leq \gamma$ this is achieved for $\beta > \gamma$. $\square$

Proposition 3 (Robustness). If $\beta > \gamma$, the linear classifier h is perfectly accurate and robust to adversarial perturbations in an ℓ_2 -ball of radius $\epsilon \leq \beta - \gamma$. Or, equivalently, for h to be ϵ -robust, the injected features must have a strength $\beta \geq \epsilon + \gamma$.

Proof. Given $\tilde{\mathbf{x}}$, we seek to find the minimum distance to the decision boundary of such a classifier. A minimum distance problem can be cast as solving the following optimization problem:

$$\epsilon^*(\tilde{\mathbf{x}}) = \min_{\mathbf{r} \in \mathbb{R}^d} \|\mathbf{r} - \tilde{\mathbf{x}}\|_2^2 \quad \text{subject to } \mathbf{r}^\top \mathbf{u} = 0,$$

which can be solved in closed form

$$\epsilon^*(\tilde{\mathbf{x}}) = \frac{|\mathbf{u}^\top \tilde{\mathbf{x}}|}{\|\mathbf{u}\|} = |\mathbf{u}^\top \mathbf{x} + y\beta|.$$

The robustness radius of the classifier h will therefore be $\epsilon = \inf_{\tilde{\mathbf{x}} \in \text{supp}(\tilde{\mathcal{D}}_\beta)} \epsilon^*(\tilde{\mathbf{x}})$, which in our case can be bounded by

$$\epsilon = \inf_{(\tilde{\mathbf{x}}, y) \in \text{supp}(\tilde{\mathcal{D}}_\beta)} \epsilon^*(\tilde{\mathbf{x}}) \leq \min_{|\mathbf{u}^\top \mathbf{x}| \leq \gamma, y = \pm 1} |\mathbf{u}^\top \mathbf{x} + y\beta| = |\mp \gamma \pm \beta| = \beta - \gamma.$$

□

Based on these propositions, we can clearly see that the interaction coefficient γ reduces the robustness of the additive features $\mathcal{V}$. In this regard, if $\epsilon \geq \beta - \gamma$, robust classification at a radius ϵ can only be achieved by also leveraging information within $\mathcal{D}$.

F.2 Robust classification can require non-linear features

We now provide a rigorous theoretical example of a learning problem that provably requires additional complex information for robust classification, even though it can achieve good clean performance using only simple features.

Given some $p \in \mathbb{N}$, let $\mathbb{R}^{p+1}$ be the input domain. A concept class, defined over $\mathbb{R}^{p+1}$ is a set of functions from $\mathbb{R}^{p+1}$ to $\{0, 1\}$. A hypothesis h is s -non-linear if the polynomial with the smallest degree that can represent h has a degree (largest order polynomial term) of s .

Using these concepts we now state the main result.

Theorem 4. For any $p, k \in \mathbb{N}, \epsilon < 0.5$ such that $k < p$, there exists a family of distributions $\mathcal{D}_k$ over $\mathbb{R}^{p+1}$ and a concept class $\mathcal{H}$ defined over $\mathbb{R}^{p+1}$ such that

1. $\mathcal{H}$ is PAC learnable (with respect to the clean error) with a *linear (degree 1)* classifier. However, $\mathcal{H}$ is not robustly learnable with any linear classifier.
2. There exists an efficient learning algorithm, that given a dataset sampled i.i.d. from a distribution $\mathcal{D} \in \mathcal{D}_k$ robustly learns $\mathcal{H}$.

In particular, the algorithm returns a k -non-linear classifier and in addition, the returned classifier also exploits the linear features used by the linear non-robust classifier.

Proof. We now define the construction of the distributions in $\mathcal{D}_k$. Every distribution $\mathcal{D}$ in the family of distribution $\mathcal{D}_k$ is uniquely defined by three parameters: a threshold parameter $\rho \in \{4t\epsilon : t \in \{0, \dots, k\}\}$ (one can think of this as the non-robust, easy-to-learn feature), a p dimensional bit vector $\mathbf{c} \in \{0, 1\}^p$ such that $\|\mathbf{c}\|_1 = k$ (this is the non-linear but robust feature) and ϵ . Therefore, given ρ and $\mathbf{c}$ (and ϵ which we discuss when necessary and ignore from the notation for simplicity), we can define the distribution $\mathcal{D}^{\mathbf{c}, \rho}$. We provide an illustration of this distribution for $p = 2$ in Figure F.1.

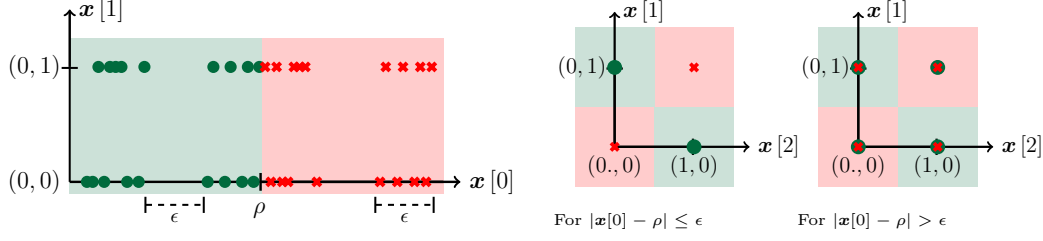


Figure F.1: Illustration of one possible distribution $\mathcal{D}^{c,\rho}$ in three dimensions. The data is linearly separable in the direction $\mathbf{x}[0]$ but has a very small margin in that direction. Leveraging $\mathbf{x}[1]$ and $\mathbf{x}[2]$ additionally, we see how the data can indeed be separated robustly, albeit non-linearly.

Sampling the robust non-linear feature To sample a point $(\mathbf{x}, y) \in \mathbb{R}^{p+1}$ from the distribution $\mathcal{D}^{c,\rho}$, first, sample a random bit vector $\hat{\mathbf{x}} \in \mathbb{R}^p$ from the uniform distribution over the boolean hypercube $\{0, 1\}^p$. Let $\hat{y} = \sum_{i=1}^{p-1} \hat{\mathbf{x}}[i] \cdot \mathbf{c}[i] \pmod{2}$ be the label of the parity function with respect to $\mathbf{c}$ evaluated on $\hat{\mathbf{x}}$. The marginal distribution over $\hat{y}$, if sampled this way, is equivalent to the Bernoulli distribution with parameter $\frac{1}{2}$. To see why, fix all bits in the input except one (chosen arbitrarily from the variables of the parity function), which is distributed uniformly over $\{0, 1\}$. It is easy to see that this forces the output of the parity function to be distributed uniformly over $\{0, 1\}$ as well. Repeating this process for all dichotomies of $p - 1$ variables of the parity function proves the desired result. Intuitively, $\hat{\mathbf{x}}$ constitutes the robust non-linear feature of this distribution.

Sampling the non-robust linear feature To ensure that $\hat{\mathbf{x}}$ is not perfectly correlated with the true label y , we sample the true label y from a Bernoulli distribution with parameter $\frac{1}{2}$. Then we sample the non-robust feature $\mathbf{x}_1$ as follows

$$\mathbf{x}_1 \sim \begin{cases} \text{Unif}(X_1^-) & y = 0 \wedge \hat{y} = 0 \\ \text{Unif}(X_1^+) & y = 1 \wedge \hat{y} = 1 \\ \text{Unif}(X_2^-) & y = 0 \wedge \hat{y} = 1 \\ \text{Unif}(X_2^+) & y = 1 \wedge \hat{y} = 0 \end{cases}$$

where

$$X_1^+ = [\rho, \rho + \epsilon] \text{ and } X_2^+ = [(\rho + 2\epsilon, \rho + 3\epsilon)], X_1^- = [\rho - \epsilon, \rho] \text{ and } X_2^- = [(\rho - 3\epsilon, \rho - 2\epsilon)].$$

Finally, we return $(\mathbf{x}, y)$ where $\mathbf{x} = (\mathbf{x}_1; \hat{\mathbf{x}})$ is the concatenation of $\mathbf{x}_1$ and $\hat{\mathbf{x}}$.

Linear non-robust solution First, we show that there is a linear, accurate, but non-robust solution to this problem. To obtain this solution, sample an m -sized dataset $S_m = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\} \in \mathbb{R}^{p+1} \times \{-1, 1\}$ from the distribution $\mathcal{D}^{c, \rho}$. Ignore all, except the first coordinate, of the covariates of the dataset to create $S_m^0 = \{(\mathbf{x}_1[0], y_1) \dots (\mathbf{x}_m[0], y_m)\}$ where $S_i[j]$ indexes the j^{th} coordinate of the i^{th} element of the dataset. Then, sort S_m^0 on the basis of the covariates (*i.e.* the first coordinate). Let $\hat{\rho}$ be the largest element whose label is 0.

Define $f_{\text{lin}, \hat{\rho}}$ as the linear threshold function on the first coordinate *i.e.* $f_{\text{lin}, \hat{\rho}}(\mathbf{x}) = \mathbb{I}\{\mathbf{x}[0] \geq \hat{\rho}\}$. By construction, $f_{\text{lin}, \hat{\rho}}$ accurately classifies all points in S_m . The VC dimension of a linear threshold function in 1 dimension is 2. Then, using standard VC sample complexity upper bounds¹ for consistent learners, if $m \geq \kappa_0 \left(\frac{1}{\alpha} \log \left(\frac{1}{\beta} \right) + \frac{1}{\alpha} \log \left(\frac{1}{\alpha} \right) \right)$, where κ_0 is some universal constant, we have that with probability at least $1 - \beta$,

$$\text{Err}(f_{\text{lin}, \hat{\rho}}; \mathcal{D}^{c, \rho}) \leq \alpha.$$

Non-linear robust solution Next, we propose an algorithm to find a robust solution and show that this solution has a non-linearity of degree k . First, sample the m -sized dataset S_m and use the method described above to find $\hat{\rho}$. Then, create a modified dataset $\hat{S}$ by first removing all points $\mathbf{x}$ from S_m such that $|\mathbf{x}[0] - \hat{\rho}| \geq \frac{\epsilon}{8}$ and then removing the first coordinate of the remaining points. Thus, each element in $\hat{S}$ belongs to $\mathbb{R}^p \times \{0, 1\}$ dimensional dataset.

Note, that by construction, there is a consistent (*i.e.* an accurate) parity classifier on $\hat{S}$. Let the parity bit vector consistent with $\hat{S}$ be $\hat{\mathbf{c}}$. This can be learned using Gaussian elimination. Consequently, construct the parity classifier $f_{\text{par}, \hat{\mathbf{c}}} = \sum_{i=0}^{p-1} \hat{\mathbf{x}}[i] \cdot \hat{\mathbf{c}}[i] \pmod{2}$.

Finally, the algorithm returns the classifier $g_{\hat{\rho}, \hat{\mathbf{c}}}$, which acts as follows:

$$g_{\hat{\rho}, \hat{\mathbf{c}}}(\mathbf{x}) = \begin{cases} 1 & \mathbb{I}\left\{\mathbf{x}[0] \geq \hat{\rho} + \epsilon + \frac{\epsilon}{8}\right\} \\ 0 & \mathbb{I}\left\{\mathbf{x}[0] \leq \hat{\rho} - \epsilon - \frac{\epsilon}{8}\right\} \\ f_{\text{par}, \hat{\mathbf{c}}}(\tilde{\mathbf{x}}) & \text{o.w.} \end{cases} \quad (\text{F.1})$$

¹<https://www.cs.ox.ac.uk/people/varun.kanade/teaching/CLT-MT2018/lectures/lecture03.pdf>

where $\tilde{\mathbf{x}} = \text{round}(\mathbf{x}[1, \dots, p])$ is obtained by rounding off $\mathbf{x}$ starting from the second index till the last. For example, if $\mathbf{x} = [0.99, 0.4, 0.9, 0.4, 0.8]$, $\epsilon = 0.2$, and $\mathbf{c} = [0, 0, 1, 1]$ then $\tilde{\mathbf{x}} = [0, 1, 0, 1]$ and $g_{0.5, \hat{c}}[\tilde{\mathbf{x}}] = 1$. Finally, it is easy to verify that the classifier $g_{\hat{\rho}, \hat{c}}$ is accurate on all training points and as the number of total parity classifiers is less than 2^p (hence finite VC dimension), as long as $m \geq \kappa_1 \left(\frac{1}{\alpha} \log \left(\frac{1}{\beta} \right) + \frac{p}{\alpha} \log \left(\frac{1}{\alpha} \right) \right)$, where κ_1 is some universal constant, we have that with probability at least $1 - \beta$,

$$\text{Err}(g_{\hat{\rho}, \hat{c}}; \mathcal{D}^{c, \rho}) \leq \alpha.$$

Robustness of $g_{\hat{\rho}, \hat{c}}$ As $\mathbf{x}[0]$ is distributed uniformly in the intervals $[\rho - \epsilon, \rho] \cup [\rho, \rho + \epsilon]$, we have that $|\rho - \hat{\rho}| \leq 4\epsilon \cdot \text{Err}(f_{\text{lin}, \hat{\rho}}; \mathcal{D}^{c, \rho}) \leq 4\epsilon\alpha$. Therefore, when m is large enough ($m = \text{poly}(\frac{1}{\alpha})$) such that $\alpha \leq \frac{1}{32}$, we have that $|\hat{\rho} - \rho| \leq \frac{\epsilon}{8}$. Intuitively, this guarantees that $g_{\hat{\rho}, \hat{c}}$ uses the linear threshold function on $\mathbf{x}[0]$ for classification in the interval $[\rho, \rho + \epsilon] \cup [\rho, \rho - \epsilon]$ and $f_{\text{par}, \hat{c}}$ in the $[\rho + 2\epsilon, \rho + 3\epsilon] \cup [\rho - 2\epsilon, \rho - 3\epsilon]$. A crucial property of $g_{\hat{\rho}, \hat{c}}$ is that for all $x \in \text{Supp}(\mathcal{D}^{c, \rho})$, the classifier $g_{\hat{\rho}, \hat{c}}$ does not alter its prediction in an ℓ_∞ -ball of radius ϵ . We show this by studying four separate cases. First, we prove robustness along all coordinates except the first.

1. When $|\mathbf{x}[0] - \hat{\rho}| \geq \epsilon + \frac{\epsilon}{8}$, as shown above, $g_{\hat{\rho}, \hat{c}}$ is invariant to all $\mathbf{x}[i]$ for all $i > 0$ and is thus robust against all ℓ_∞ perturbations against those coordinates.
2. When $|\mathbf{x}[0] - \hat{\rho}| < \epsilon + \frac{\epsilon}{8}$, due to (F.1), we have that $g_{\hat{\rho}, \hat{c}} = f_{\text{par}, \hat{c}}(\tilde{\mathbf{x}})$ where $\tilde{\mathbf{x}} = \text{round}(\mathbf{x}[1, \dots, p])$ is obtained by rounding off all indices of $\mathbf{x}$ except the first. As the rounding operation on the boolean hypercube is robust to any ℓ_∞ perturbation of radius less than 0.5, we have that $g_{\hat{\rho}, \hat{c}}$ is robust to all ℓ_∞ perturbations of radius less than 0.5 on the support of the distribution $\mathcal{D}^{c, \rho}$.

Next, we prove the robustness along the first coordinate. Let $0 < \delta < \epsilon$ represent an adversarial perturbation. Without loss of generality, assume that $\mathbf{x}[0] > \hat{\rho}$ as similar arguments apply for the other case.

1. Consider the case $\mathbf{x}[0] \leq \hat{\rho} + \epsilon + \frac{\epsilon}{8}$. Then, $|\mathbf{x}[0] - \delta - \hat{\rho}| \leq \left| \epsilon + \frac{\epsilon}{8} - \delta \right| \leq \epsilon + \frac{\epsilon}{8}$ and hence, by construction, $g_{\hat{\rho}, \hat{\epsilon}}(\mathbf{x}) = g_{\hat{\rho}, \hat{\epsilon}}([x[0] - \delta; [x[1], \dots, p]])$. On the other hand, for all δ , we have that $g_{\hat{\rho}, \hat{\epsilon}}([x[0] + \delta; [x[1], \dots, p]]) = 1$ if $g_{\hat{\rho}, \hat{\epsilon}}(x) = 1$.
2. For the case $\mathbf{x}[0] \geq \hat{\rho} + \epsilon + \frac{\epsilon}{8}$, the distribution is supported only on the interval $[\rho + 2\epsilon, \rho + 3\epsilon]$. When a positive δ is added to the first coordinate, the classifier's prediction does not change and it remains 1. For all $\delta \leq \frac{\epsilon}{2}$, when the perturbation is subtracted from the first coordinate, its first coordinate is still larger than $\hat{\rho} + \epsilon + \frac{\epsilon}{8}$ and hence, the prediction is still 1.

This completes the proof of robustness of $g_{\hat{\rho}, \hat{\epsilon}}$ along all dimensions to ℓ_∞ perturbations of radius less than ϵ . Combining this with its error bound, we have that $\text{Adv}_{\epsilon, \infty}(g_{\hat{\rho}, \hat{\epsilon}}; \mathcal{D}^{c, \rho}) \leq \alpha$.

To show that the parity function is non-linear, we use a classical result from Aspnes et al. (1994). Theorem 2.2 in Aspnes et al. (1994) shows that approximating the parity function in k bits using a polynomial of degree ℓ incurs at least $\sum_{i=0}^{k_\ell} \binom{k}{i}$ where $k_\ell = \left\lfloor \frac{k-\ell-1}{2} \right\rfloor$ mistakes. Therefore, the lowest degree polynomial that can do the approximation accurately is at least k .

This completes our proof of showing that the robust classifier is of non-linear degree k while the accurate classifier is linear. Next, we prove that no linear classifier can be robust. We show this by contradiction.

No linear classifier can be robust Construct a set $\mathcal{Z}$ of s (to be defined later) points in $\mathbb{R}^{p+1}$ by sampling the first coordinate from the interval $[\rho, \rho + \epsilon]$ and the remaining p coordinates uniformly from the boolean hypercube. Then, augment the set by subtracting ϵ from the first coordinate while retaining the rest of the coordinates. Note that this set can be generated, along with its labels, by sampling enough points from the original distribution and discarding points that do not fall in this interval. Now construct adversarial examples of each point in the augmented set by either adding or subtracting ϵ from the negatively and the positively labelled examples respectively

and augment the original set with these adversarial points. For a large enough s ,², this augmented set of points can be decomposed into a multiset of points, where all points in any one set has the same value in the first coordinate but nearly half of their label is zero and the other half one.

Now, assume that there is a linear classifier that has a low error on the distribution $\mathcal{D}^{c,\rho}$. Therefore the classifier is also accurate on these sets of points as the classifier is robust, by assumption, and the union of these sets occupy a significant under the distribution $\mathcal{D}^{c,\rho}$. However, as the first coordinate of every point within a set is constant despite half the points having label one and the other half zero, the coefficient of the linear classifier can be set to zero without altering the behavior of the classifier. Then, effectively the linear classifier is representing a parity function on the rest of the p coordinates. However, we have just seen that this is not possible as a linear threshold function cannot represent a parity function on k bits where $k > 1$. This contradicts our initial assumption that there is a robust linear classifier for this problem.

This completes the proof. □

²There is a slight technicality as we might not obtain points that are exact reflections of each other around ρ but that can be overcome by discretising upto a certain precision

F.3 Experimental details

In this section we provide the experimental details for all results presented in the paper. Adversarial training for all methods and datasets follows the fast training schedules with a cyclic learning rate introduced in Wong et al. (2020). We train for 30 epochs on CIFAR Krizhevsky et al. (2009) and 15 epochs for SVHN Netzer et al. (2011) following Andriushchenko and Flammarion (2020). When we perform PGD-AT we use 10 steps and a step size $\alpha = 2/255$; FGSM uses a step size of $\alpha = \epsilon$. Regularization parameters for GradAlign Andriushchenko and Flammarion (2020) and N-FGSM de Jorge et al. (2022) will vary and are stated when relevant in the paper. The architecture employed is a PreactResNet18 He et al. (2016b). Robust accuracy is evaluated by attacking the trained models with PGD-50-10, i.e. PGD with 50 iterations and 10 restarts. In this case we also employ a step size $\alpha = 2/255$ as in Wong et al. (2020). All accuracies are averaged after training and evaluating with 3 random seeds.

The curvature computation is performed following the procedure described in Moosavi-Dezfooli et al. (2019). As they propose, we use finite differences to estimate the directional second derivative of the loss with respect to the input, *i.e.*

$$\mathbf{w}^\top \nabla_x^2 \mathcal{L}(f_\theta(\mathbf{x}), y) \approx \frac{\nabla_x \mathcal{L}(f_\theta(\mathbf{x} + t\mathbf{w}), y) - \nabla_x \mathcal{L}(f_\theta(\mathbf{x} - t\mathbf{w}), y)}{2t},$$

with $t > 0$ and use the Lanczos algorithm to perform a partial eigendecomposition of the Hessian without the need to compute the full matrix. In particular, we pick $t = 0.1$.

All our experiments were performed using a cluster equipped with GPUs of various architectures. The estimated compute budget required to produce all results in this work is around 2,000 GPU hours (in terms of NVIDIA V100 GPUs).

F.4 Inducing catastrophic overfitting with other settings

In Appendix E.3 we have shown that CO can be induced with data interventions for CIFAR-10 and ℓ_∞ perturbations. Here we present similar results when using other datasets (i.e. CIFAR-100 and SVHN) and other types of perturbations (i.e. ℓ_2 attacks). Moreover, we also report results when the injected features $\mathbf{v}(y)$ follow random directions (as opposed to low-frequency DCT components). Overall, we find similar findings to those reported the main text.

F.4.1 Other datasets

Similarly to Appendix E.3 we modify the SVHN, CIFAR-100 and higher resolution Imagenet-100 and TinyImagenet datasets to inject highly discriminative features $\mathbf{v}(y)$. Since SVHN also has 10 classes, we use the exact same settings as in CIFAR-10 and we train and evaluate with $\epsilon = 4$ where training on the original data does not lead to CO (recall $\beta = 0$ corresponds to the unmodified dataset). On the other hand, for CIFAR-100 and Imagenet-100 we select $\mathbf{v}(y)$ to be the 100 DCT components with the lowest frequency and we present results with $\epsilon = 5$ and $\epsilon = 4$ respectively. Similarly, for TinyImagenet (which has 200 classes) we use the first 200 DCT components and present results with $\epsilon = 6$. On ImageNet-100 we evaluate robustness with AutoAttack Croce and Hein (2020), as PGD-50 seems to be broken when CO occurs in this dataset. On the other datasets, we evaluated with AutoAttack at the extrema of our plots and found that the estimated robustness agrees with the one of PGD-50. To save computational complexity, we only used PGD-50 to estimate the robustness on the other datasets.

Regarding the training settings, for CIFAR10/100 and SVHN datasets we use the same settings as Andriushchenko and Flammarion (2020). For ImageNet-100 we follow Kireev et al. (2022) and for TinyImageNet Li et al. (2020).

In all datasets we can observe similar trends as with CIFAR-10: For small values of β the injected features are not very discriminative due to their interaction with the dataset images and the model largely ignores them. As we increase β , there is a range in which they become more discriminative but not yet robust and we observe CO. Finally for large values of β the injected features become robust and the models can achieve very good performance focusing only on those.

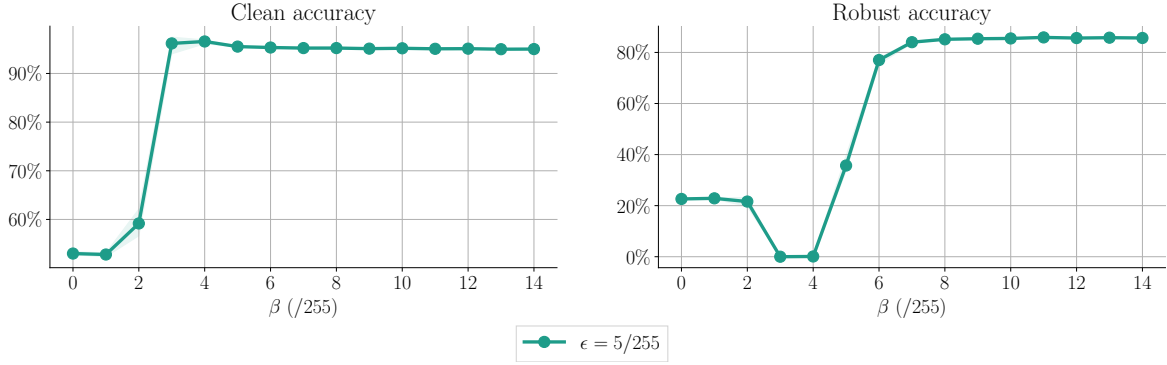


Figure F.2: Clean and robust performance after FGSM-AT on injected datasets $\tilde{\mathcal{D}}_\beta$ constructed from CIFAR-100. As FGSM-AT already suffers CO on CIFAR-100 at $\epsilon = 6/255$ we use $\epsilon = 5/255$ in this experiment where FGSM-AT does not suffer from CO as seen for $\beta = 0$. In this setting, we observe CO happening when β is slightly smaller than ϵ . Results are averaged over 3 seeds and shaded areas report minimum and maximum values.

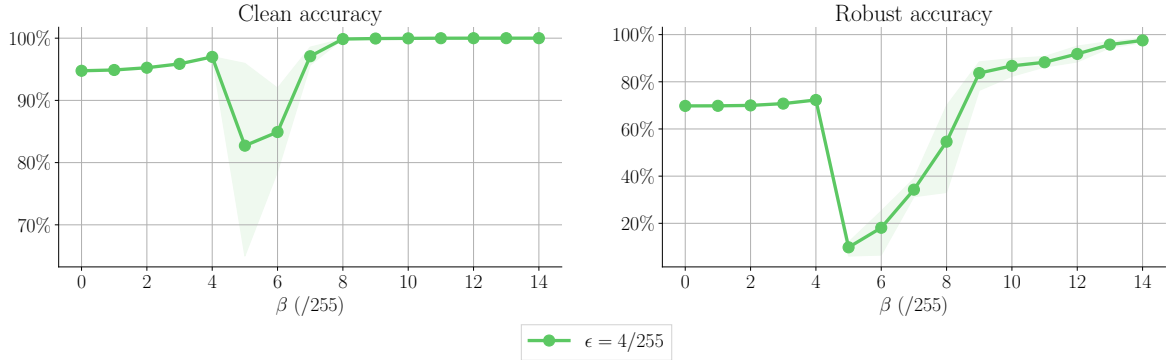


Figure F.3: Clean and robust performance after FGSM-AT on injected datasets $\tilde{\mathcal{D}}_\beta$ constructed from SVHN. As FGSM-AT already suffers CO on SVHN at $\epsilon = 6/255$ we use $\epsilon = 4/255$ in this experiment where FGSM-AT does not suffer from CO as seen for $\beta = 0$. In this setting, we observe CO happening when $\beta \approx \epsilon$. Results are averaged over 3 seeds and shaded areas report minimum and maximum values.

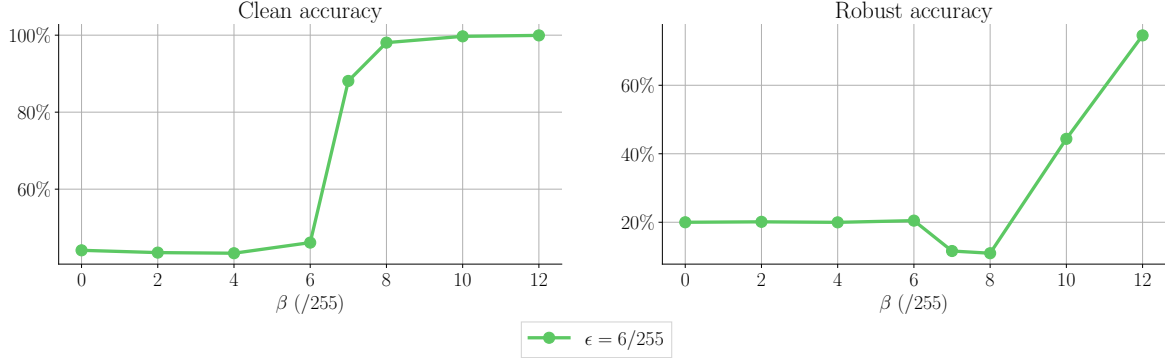


Figure F.4: Clean and robust performance after FGSM-AT on injected datasets $\tilde{\mathcal{D}}_\beta$ constructed from TinyImageNet. We use $\epsilon = 6/255$ in this experiment where FGSM-AT does not suffer from CO as seen for $\beta = 0$. In this setting, we observe CO happening for $\beta \in [7/255, 8/255]$.

F.4.2 Other norms

Catastrophic overfitting has been mainly studied for ℓ_∞ perturbations and thus we presented experiments with ℓ_∞ attacks following related work. However, in this section we also present results where we induce CO with ℓ_2 perturbation which are also widely used in adversarial robustness. In Figure F.5 we show the clean (left) and robust (right) accuracy after FGM-AT³ on our injected dataset from CIFAR-10 ($\tilde{\mathcal{D}}_\beta$). Similarly to our results with ℓ_∞ attacks, we also observe CO as the injected features become more discriminative (increased β). It is worth mentioning that the ℓ_2 norm we use ($\epsilon = 1.5$) is noticeably larger than typically used in the literature, however, it would roughly match the magnitude of an ℓ_∞ perturbation with $\epsilon = 7/255$. Interestingly, we did not observe CO for this range of β with $\epsilon = 1$.

F.4.3 Other injected features

We selected the injected features for our injected dataset from the low frequency components of the DCT to ensure an interaction with the features present on natural images Ahmed et al. (1974). However, this does not mean that other types of features could not induce CO. In order to understand how unique was our choice of features

³FGM is the ℓ_2 version of FGSM where we do not take the sign of the gradient.

F. Appendix of Catastrophic overfitting can be induced with discriminative non-robust features

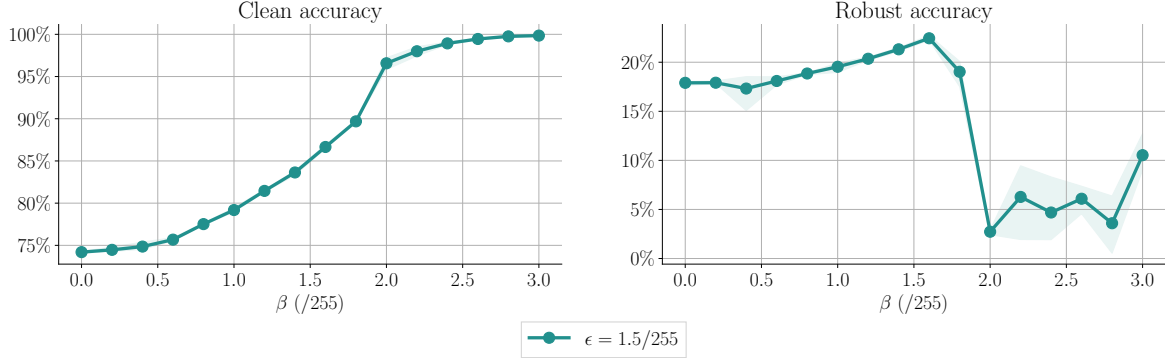


Figure F.5: Clean and ℓ_2 robust performance after FGSM-AT on injected datasets $\tilde{\mathcal{D}}_\beta$ constructed from CIFAR-10. FGM-AT suffers CO on CIFAR-10 around $\epsilon = 2$, so we use $\epsilon = 1.5$ in this experiment where FGM-AT does not suffer from CO as seen for $\beta = 0$. In this setting, we observe CO happening when $\beta \approx \epsilon$. Results are averaged over 3 seeds and shaded areas report minimum and maximum values.

we also created another family of injected datasets but this time using a set of 10 randomly generated vectors as features. As in the main text, we take the sign of each random vector to ensure they take values in $\{-1, +1\}$ and assign one vector per class. In Figure F.6 we observe that using random vectors as injected features we can also induce CO. Note that since our results are averaged over 3 random seeds, each seed uses a different set of random vectors.

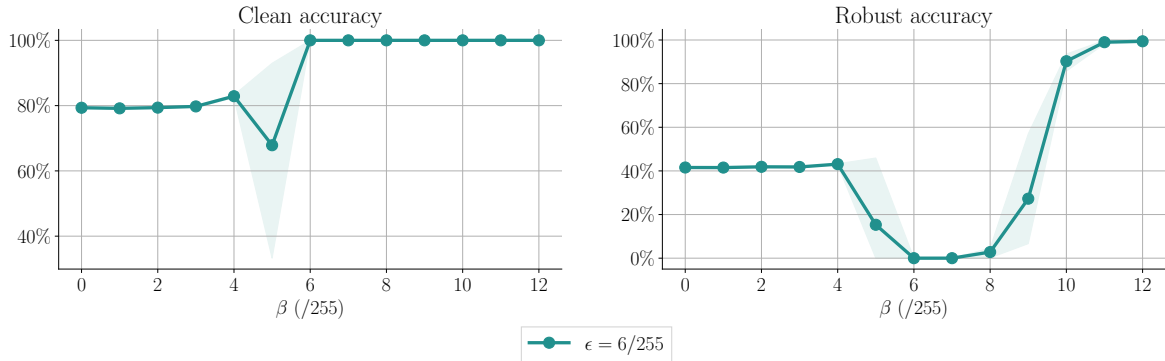


Figure F.6: Clean and robust performance after FGSM-AT on injected datasets $\tilde{\mathcal{D}}_\beta$ constructed from CIFAR-10 using random signals in $\mathcal{V}$. We perform this experiments at $\epsilon = 6/255$ where we saw that injected the dataset with the DCT basis vectors did induce CO. In the random $\mathcal{V}$ setting, we observe the same behaviour, with CO happening when $\beta \approx \epsilon$. Results are averaged over 3 seeds and shaded areas report minimum and maximum values.

F.4.4 Other architectures

In all our previous experiments we trained a PreActResNet18 (He et al., 2016b) as it is the standard architecture used in the literature. However, our observations are also robust to the choice of architecture. As we can see in Figure F.7, we can also induce CO when training a WideResNet28x10 Zagoruyko and Komodakis (2016) on an injected version of CIFAR-10.

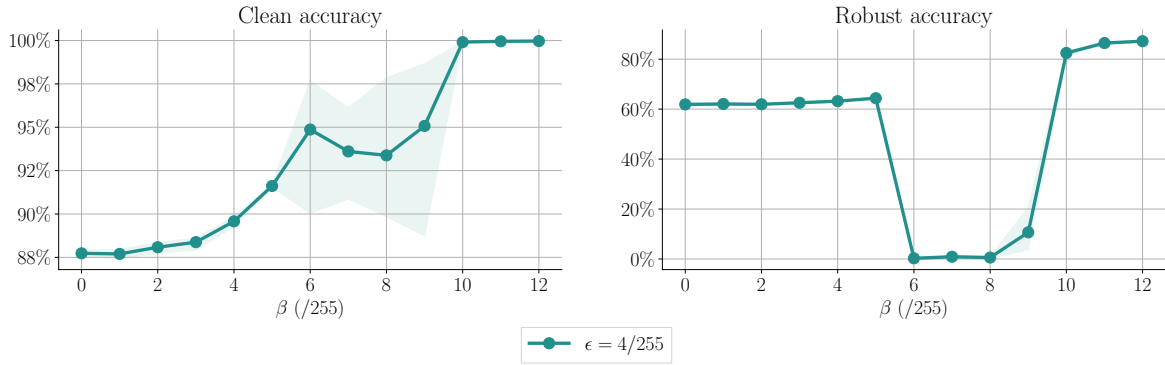


Figure F.7: Clean and robust performance after FGSM-AT on injected datasets $\tilde{\mathcal{D}}_\beta$ constructed from CIFAR-10 when training a WideResNet28x10. We perform this experiments at $\epsilon = 4/255$. Results are averaged over 3 seeds and shaded areas report minimum and maximum values.

F.5 Performance on $\mathcal{V}$

To further understand the behavior of the models described in Appendix E.4, we conduct an additional experiment evaluating the performance of the networks on the set of features $\mathcal{V}$. This experiment corroborates the findings shown in Figure E.2 and reveal more insights into how the networks utilize the features from $\mathcal{V}$ and $\mathcal{D}$.

In Figure F.8, we see that networks trained using standard training demonstrate near-perfect accuracy on $\mathcal{V}$ from the start, indicating that they heavily rely on these features for classification and can correctly classify the data based solely on these. In contrast, networks trained with PGD-AT show a gradual improvement in accuracy on $\mathcal{V}$, as they learn to effectively combine the information from $\mathcal{V}$ with that of $\mathcal{D}$ to achieve robustness. As anticipated, networks trained with FGSM-AT exhibit a similar

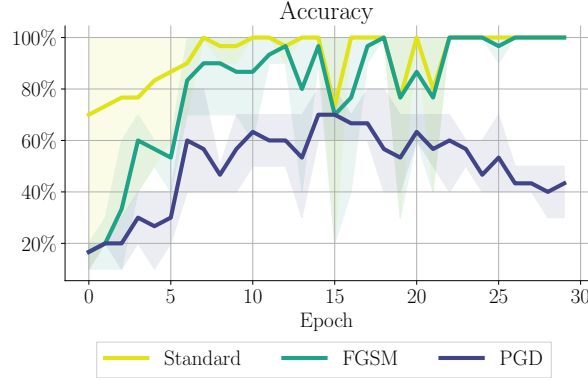


Figure F.8: Accuracy on $\mathcal{V}$ of networks trained on $\tilde{\mathcal{D}}_\beta$ using standard training, FGSM-AT or PGD-AT FGSM-AT. We use $\epsilon = 4/255$ and $\beta = 6/255$. Results averaged over 3 random seeds. Shades show min-max values.

behavior to those trained with PGD-AT initially. However, following the occurrence of catastrophic overfitting (CO), they experience a sudden increase in accuracy on $\mathcal{V}$.

F.6 Learned features at different β

In Appendix E.3 we discussed how, based on the strength of the injected features β , our injected datasets seem to have 3 distinct regimes: (i) When β is small we argued that the network would not use the injected features as these would not be very helpful. (ii) When β would have a very large value then the network would only look at these features since they would be easy-to-learn and provide enough margin to classify robustly. (iii) Finally, there was a middle range of β usually when $\beta \sim \epsilon$ where the injected features would be strongly discriminative but not enough to provide robustness on their own. This regime is where we observe CO.

In this section we present an extension of Figure E.2 where we take FGSM trained models on the injected datasets ($\tilde{\mathcal{D}}_\beta$) and evaluate them on three test sets: (i) The injected test set ($\tilde{\mathcal{D}}_\beta$) with the same features as the training set. (ii) The original dataset ($\mathcal{D}$) where the images are unmodified. (iii) The shuffled dataset ($\tilde{\mathcal{D}}_{\pi(\beta)}$) where the injected features are permuted. That is, the set of injected features is the same

but the class assignments are shuffled. Therefore, the injected features will provide conflicting information with the features present on the original image.

In Figure F.9 we show the performance on the aforementioned datasets for three different values of β . For $\beta = 2/255$ we are in regime (i) : we observe that the tree datasets have the same performance, i.e. the information of the injected features does not seem to alter the result. Therefore, we can conclude the network is mainly using the features from the original dataset $\mathcal{D}$. When $\beta = 20/255$ we are in regime (ii) : the clean and robust performance of the network is almost perfect on the injected test set $\tilde{\mathcal{D}}_\beta$ while it is close to 0% (note this is worse than random classifier) for the shuffled dataset. So when the injected and original features present conflicting information the network aligns with the injected features. Moreover, the performance on the original dataset is also very low. Therefore, the network is mainly using the injected features. Lastly, $\beta = 8/255$ corresponds to regime (iii) : as discussed in Appendix E.4, in this regime the network initially learns to combine information from both the original and injected features. However, after CO, the network seems to focus only on the injected features and discards the information from the original features.

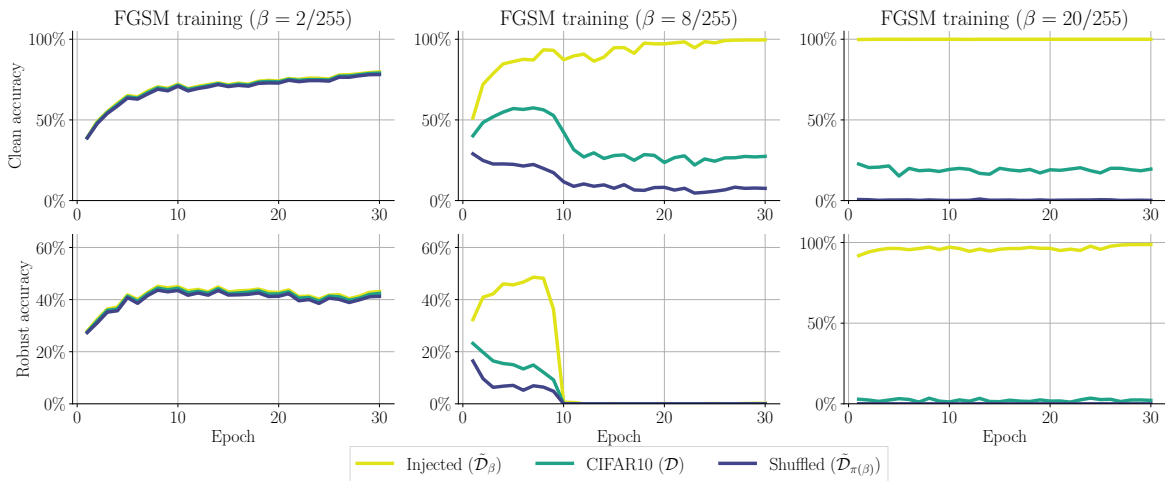


Figure F.9: Clean (**top**) and robust (**bottom**) accuracy of FGSM-AT on $\tilde{\mathcal{D}}_\beta$ at different β values on 3 different test sets: (i) the original CIFAR-10 ($\mathcal{D}$), (ii) the dataset with injected features $\tilde{\mathcal{D}}_\beta$ and (iii) the dataset with shuffled injected features $\tilde{\mathcal{D}}_{\pi(\beta)}$. All training runs use $\epsilon = 6/255$. **Left:** $\beta = 2/255$ **Center:** $\beta = 8/255$ **Right:** $\beta = 20/255$.

F.7 Analysis of curvature in different settings

In Figure E.3 (left) we track the curvature of the loss surface while training on different injected datasets with either PGD-AT or FGSM-AT. We observe that (i) Curvature rapidly increases both for PGD-AT and FGSM-AT during the initial epochs of training. (ii) In those runs that presented CO, the curvature explodes around the 8th epoch along with the training accuracy. (iii) When training with the dataset with orthogonally injected features ($\tilde{\mathcal{D}}_{\beta}^{\perp}$) the curvature does not increase. This is aligned with our proposed mechanisms to induce CO whereby the network increases the curvature in order to combine different features to learn better representations. In this section we extend this analysis to different values of feature strength β on the injected dataset ($\tilde{\mathcal{D}}_{\beta}$). For details on how we estimate the curvature refer to Appendix F.3.

Figure F.10 presents the curvature for different values of feature strength β on the injected dataset ($\tilde{\mathcal{D}}_{\beta}$). We show three different values of β representative of the three regimes discussed in Appendix F.6. Recall that when β is small ($\beta = 2/255$) we observe that the model seems to focus only on CIFAR-10 features. Thus, we observe a curvature increase aligned with (CIFAR-10) feature combination. However, since for the chosen robustness radii $\epsilon = 6/255$ there is no CO, we observe that the curvature increase remains stable. When β is quite large ($\beta = 20/255$) then the model largely ignores CIFAR-10 information and focuses on the easy-to-learn injected features. Since these features are already robust, there is no need to combine them and the curvature does not need to increase. In the middle range when CO happens ($\beta = 8/255$) we observe again the initial curvature increase and then curvature explosion.

F.8 Adversarial perturbations before and after CO

Qualitative analysis In order to further understand the change in behaviour after CO we presented visualizations of the FGSM perturbations before and after CO in Figure E.4. We observed that while prior to CO, the injected feature components $\mathbf{v}(y)$

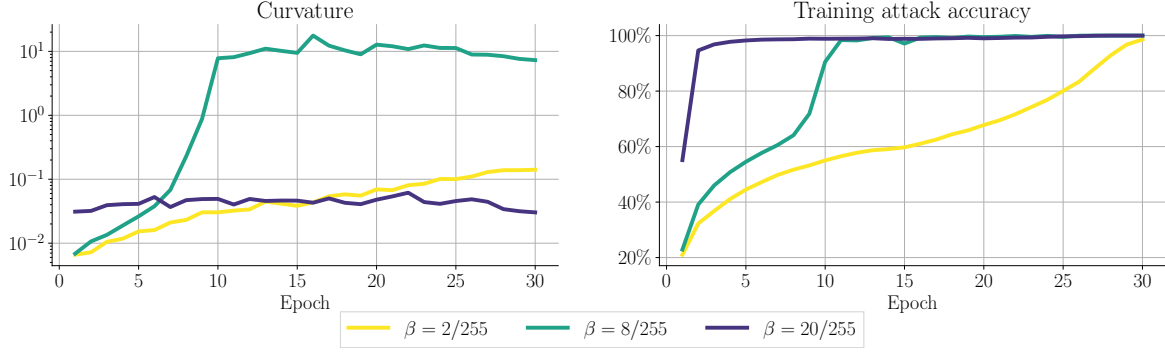


Figure F.10: Evolution of curvature and training attack accuracy of FGSM-AT and PGD-AT trained on $\tilde{\mathcal{D}}_\beta$ at different β and for $\epsilon = 6/255$. Only when CO happens (for $\beta = 8/255$) the curvature explodes. For the other two interventions the curvature does not increase so much. We argue this is because the network does not need to disentangle $\mathcal{D}$ from $\mathcal{V}$, as it ignores either one of them.

were clearly identifiable, after CO the perturbations do not seem to point in those directions although the network is strongly relying on them to classify. In Figure F.11 and Figure F.12 we show further visualizations of the perturbations obtained both with FGSM or PGD attacks on networks trained with either PGD-AT or FGSM-AT respectively.

We observe that when training with PGD-AT, i.e. the training does not suffer from CO, both PGD and FGSM attacks produce qualitatively similar results. In particular, all attacks seem to target the injected features with some noise due to the interaction with the features from CIFAR-10. For FGSM-AT, we observe that at the initial epochs (prior to CO) the perturbations are similar to those of PGD-AT, however, after CO perturbations change dramatically both for FGSM and PGD attacks. This aligns with the fact that the loss landscape of the network has dramatically changed, becoming strongly non-linear. This change yields single-step FGSM ineffective, however, the network remains vulnerable and multi-step attacks such as PGD are still able to find adversarial examples, which in this case do not point in the direction of discriminative features Jetley et al. (2018); Ilyas et al. (2019).

Quantitative analysis Finally, to quantify the radical change of direction of the adversarial perturbations after CO, we compute the evolution of the average alignment (*i.e.* cosine angle) between the FGSM perturbations δ and the injected features, such

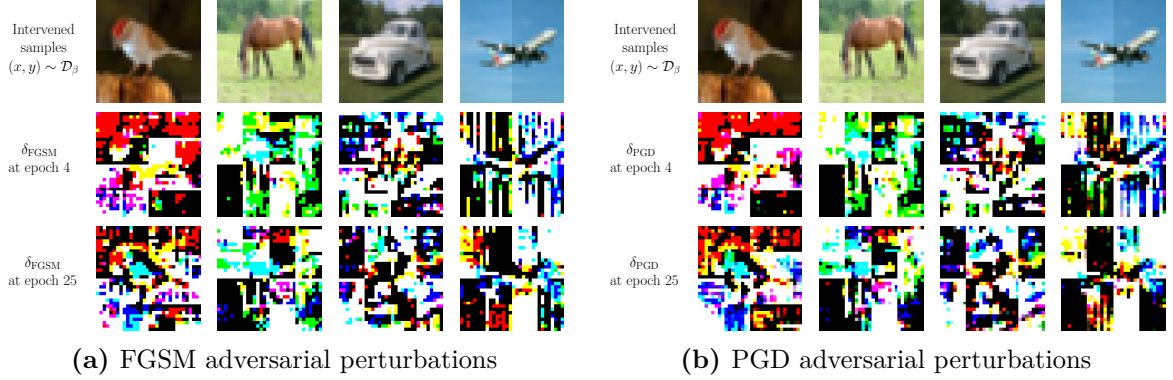


Figure F.11: Different samples of the injected dataset $\tilde{\mathcal{D}}_\beta$, and adversarial perturbations at epoch 4 and 22 of PGD-AT on $\tilde{\mathcal{D}}_\beta$ at $\epsilon = 6/255$ and $\beta = 8/255$ (where FGSM-AT suffers CO). The adversarial perturbations remain qualitatively similar throughout training and align significantly with $\mathcal{V}$.

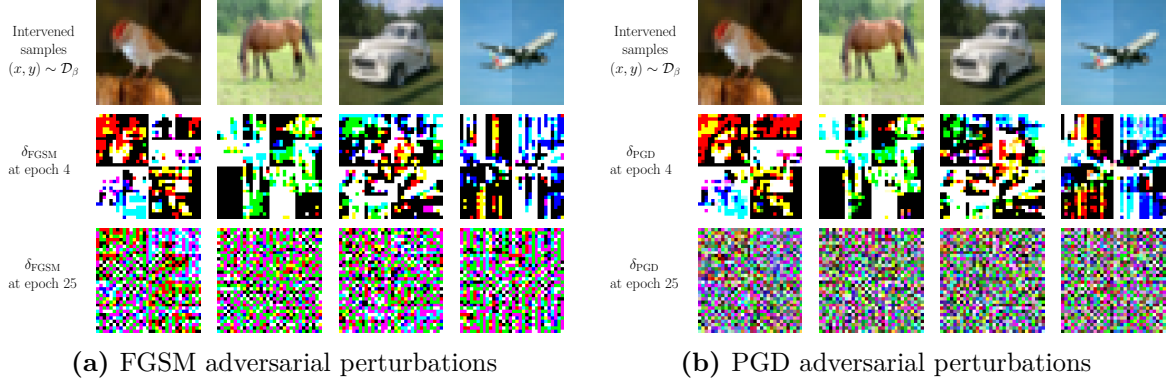


Figure F.12: Different samples of the injected dataset $\tilde{\mathcal{D}}_\beta$, and adversarial perturbations at epoch 4 (before CO) and 22 (after CO) of FGSM-AT on $\tilde{\mathcal{D}}_\beta$ at $\epsilon = 6/255$ and $\beta = 8/255$ (where FGSM-AT suffers CO). The adversarial perturbations change completely before and after CO. Prior to CO, they align significantly with $\mathcal{V}$, but after CO they point to meaningless directions.

that if point $\mathbf{x}$ is associated with class y we compute $\frac{\langle \delta, \mathbf{v}(y) \rangle}{\|\mathbf{v}(y)\|_2 \|\delta\|_2}$. Figure E.7 (left) shows the results of this evaluation, where we can see that before CO there is a non-trivial alignment between the FGSM perturbations and their corresponding injected features, that after CO quickly converges to the same alignment as the one between two random vectors.

To complement this view, we also perform an analysis of the frequency spectrum of the FGSM perturbations. In Figure E.7 (right), we plot the average magnitude of the DCT transform of the FGSM perturbations computed on the test set of an intervened version of CIFAR-10 during training. As we can see, prior to CO, most of the energy

of the perturbations is concentrated around the low frequencies (remember that the injected features are low frequency), but after CO happens, around epoch 8, the energy of the perturbations quickly gets concentrated towards higher frequencies. These two plots corroborate, quantitatively, our previous observations, that before CO, FGSM perturbations are pointing towards meaningful predictive features, while after CO, although we know the network still uses the injected features (see Figure E.2) the FGSM perturbations suddenly point in a different direction.

F.9 Further results with N-FGSM, GradAlign and PGD

In Appendix E.7 we studied different SOTA methods that have been shown to prevent CO. Interestingly, we observed that in order to avoid CO on the injected dataset a stronger level of regularization is needed. Thus, indicating that the intervention is strongly favouring the mechanisms that lead to CO. For completeness, in Figure F.13 we also present results of the clean accuracy (again with the robust accuracy). As expected, for those runs in which we observe CO, clean accuracy quickly saturates. Note that for stronger levels of regularization the clean accuracy is lower. An ablation of the regularization strength might help improve results further, however the purpose of this analysis is not to improve the performance on the injected dataset but rather to show it is indeed possible to prevent CO with the same methods that work for unmodified datasets.

F.10 Further details of low-pass experiment

We expand here over the results in Appendix E.7 and provide further details on the experimental settings of Table E.1. Specifically, we replicate the same experiment, *i.e.* training a low-pass version of CIFAR-10 using FGSM-AT at different filtering bandwidths. As indicated in Appendix E.7 we use the low-pass filter introduced in

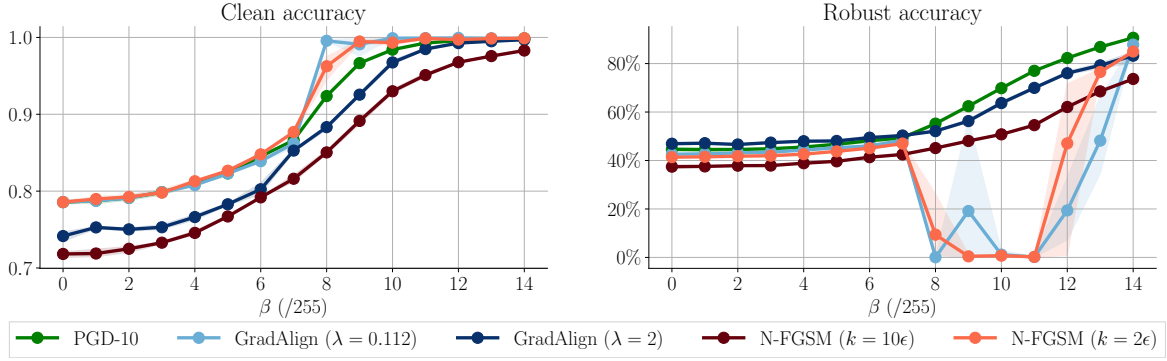


Figure F.13: Clean (**left**) and robust (**right**) accuracy after AT with PGD-10, GradAlign and N-FGSM on $\hat{\mathcal{D}}_\beta$ at $\epsilon = 6/255$. Results averaged over three random seeds and shaded areas report minimum and maximum values.

Ortiz-Jimenez et al. (2020b) which only retains the frequency components in the upper left quadrant of the DCT transform of an image. That is, a low-pass filter of bandwidth W would retain the $W \times W$ upper quadrant of DCT coefficients of all images, setting the rest of the coefficients to 0.

Figure F.14 shows the robust accuracy obtained by FGSM-AT on CIFAR-10 versions that have been pre-filtered using such a low-pass filter. Interestingly, while training on the unfiltered images does induce CO on FGSM-AT, just removing a few high-frequency components is enough to prevent CO $\epsilon = 8255$. However, as described before, it seems that at $\epsilon = 16/255$ no frequency transformation can avoid CO. Clearly, this transformation cannot be used as technique to prevent CO, but it highlights once again that the structure of the data plays a significant role in inducing CO.

Relation with anti-aliasing pooling layers As mentioned in Appendix E.7, our proposed low-passing technique is very similar in spirit to works which propose using anti-aliasing low-pass filters at all pooling layers (Grabinski et al., 2022; Zhang, 2019). Indeed, as shown by Ortiz-Jimenez et al. (2020b), CIFAR10 contains a significant amount of non-robust features on the high-frequency end of the spectrum due to aliasing produced in their down-sampling process. In this regard, it is no surprise that methods like the one proposed in Grabinski et al. (2022) can prevent CO at $\epsilon = 8/255$ using the same training protocol as in our work (robust accuracy is 45.9%). Interestingly, though, repeating the experiments in Grabinski et al. (2022) work using

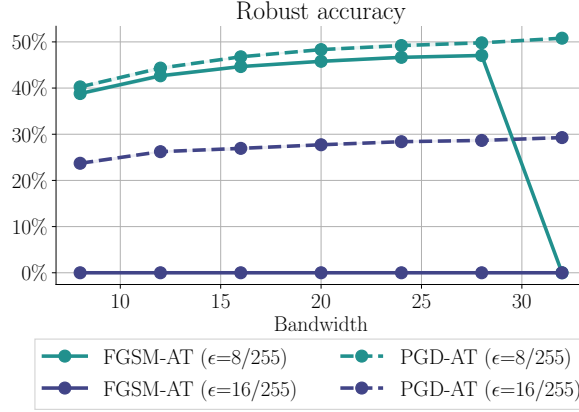


Figure F.14: Robust accuracy of FGSM-AT and PGD-AT on different low-passed versions of CIFAR-10 using the DCT-based low pass filter introduced in Ortiz-Jimenez et al. (2020b). Bandwidth = 32 corresponds to the original CIFAR-10, while smaller bandwidths remove more and more high-frequency components. At $\epsilon = 8/255$ just removing a few high-frequency components is enough to prevent CO, while at $\epsilon = 16/255$ no frequency transformation avoids CO.

$\epsilon = 16/255$ does lead to CO (robust accuracy is 0.0%) in Appendix E.7. This result was not reported in the original paper, but we see it as a corroboration of our observations. Indeed, features play a role in CO, but the problematic features do not always come from excessively high-frequencies or aliasing. However, we still consider that preventing aliasing in the downsampling layers is a promising avenue for future work in adversarial robustness.