TOWARDS ROBUST REPRESENTATION LEARNING AND BEYOND

by
Cihang Xie

A dissertation submitted to Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy

Baltimore, Maryland
October, 2020

© 2020 Cihang Xie
All Rights Reserved

Abstract

Deep networks have reshaped the computer vision research in recent years. As fueled by powerful computational resources and massive amount of data, deep networks now dominate a wide range of visual benchmarks. Nonetheless, these success stories come with bitterness—an increasing amount of studies has shown the limitations of deep networks on certain testing conditions like small input changes or occlusion. These failures not only raise safety and reliability concerns on the applicability of deep networks in the real world, but also demonstrate the computations performed by the current deep networks are dramatically different from those by human brains.

In this dissertation, we focus on investigating and tackling a particular yet challenging weakness of deep networks—their vulnerability to adversarial examples. The first part of this thesis argues that such vulnerability is a much more severe issue than we thought—the threats from adversarial examples are ubiquitous and catastrophic. We then discuss how to equip deep networks with robust representations for defending against adversarial examples. We approach the solution from the perspective of neural architecture design, and show incorporating architectural elements like feature-level denoisers or smooth activation functions can effectively boost model robustness. The last part of this thesis focuses on rethinking the value of adversarial examples. Rather than treating adversarial examples as a threat to deep networks, we take a further step on uncovering adversarial examples can help deep networks improve the generalization ability, if feature representations are properly disentangled during learning.

Thesis Readers

Dr. Alan L. Yuille (Primary Advisor)
Bloomberg Distinguished Professor
Department of Computer Science
Johns Hopkins University

Dr. Gregory D. Hager
Mandell Bellmore Professor
Department of Computer Science
Johns Hopkins University

Dr. Quoc V. Le
Principal Research Scientist
Google Brain

Acknowledgements

First of all, I would like to thank my Ph.D. advisor, Prof. Alan Yuille, for his insightful guidance and continuous support over the last five years. I am deeply appreciated that Alan provided me with hands-on mentoring when I was a newbie computer vision researcher. Under his supervision and encouragement, I began to gain interests in exploring the limitations of deep networks, which later inspired me to explore a relatively new research direction—adversarial learning. This direction became the core path throughout my Ph.D. career, during which I have published a series of solid research works. Without Alan's mentorship, I could have never got a chance to make such achievements. Besides, I would like to thank Wei Shen, Yinzhi Cao, Vishal Patel, Gregory Hager, Alex Szalay and Rama Chellappa for serving on my GBO/thesis committee and providing valuable suggestions.

Next, I feel extremely fortunate to have two amazing research internships, one at Facebook AI Research with Kaiming He, Laurens van der Maaten and Yuxin Wu, and the other at Google with Quoc Le, Mingxing Tan, Boqing Gong and Jiang Wang. These two internships significantly broadened my research horizons and enabled me to establish a more mature and profound perspective regarding this area. Specifically, I would like to thank Kaiming He and Quoc Le for teaching me how to think critically, conduct high impact research and improve writing skills, and keeping supporting my career even after my internships.

I would also like to thank all people who have helped me build my academic career, including Ming-Hsuan Yang, Yong Jae Lee, Tengyu Ma, Fei-Fei Li, Nuno

Vasconcelos and Ian Goodfellow for providing useful suggestions to my academic job search preparation, and Yingwei Li, Xinyun Chen, Adam Kortylewski, Song Bai, Zhaowei Cai, Andrei Barbu, Wieland Brendel, Bo Li, Andrea Vedaldi, Luc Van Gool, Philip Torr and Dawn Song for supporting my workshop applications.

I feel so excited to get a chance to work with a lot of wonderful colleagues at CCVL in the last five years, including Zhishuai Zhang, Jianyu Wang, Lingxi Xie, Jun Zhu, Zhou Ren, Siyuan Qiao, Chenglin Yang, Yucheng Han, Hongru Zhu, Yuhui Xu, Jieru Mei, Peng Tang, Yixiao Zhang, Qihang Yu, Yan Wang, Zhuowan Li, Yongyi Lu, Chenxi Liu, Alex Wong, Fengze Liu, Qing Liu, Yingda Xia, Xuan Dong, Yutong Bai, Weichao Qiu, Huiyu Wang, Angtian Wang, Chenxu Luo, Zihao Xiao, Fangting Xia, Xianjie Chen, Vittal Premachandran, Ehsan Jahangiri, Jieneng Chen, Yi Zhang, Xiao Chu, Shuhao Fu, Chen Wei, Junhua Mao, Michelle Shu, Liang-Chieh Chen, Peng Wang, Zhuotun Zhu, Qi Chen, Christian Cosgrove, Minghui Liao, Zefan Li, Jiteng Mu, Xiaochen Lian, Zhe Ren, Yuan Gao and many others.

The last part is dedicated to acknowledging my family members. I am a first-generation graduate student, and I feel so grateful to my parents, Jishan Xie and Chunxiang Zhang, for unconditionally supporting me to pursue my academic dreams. I would also like to thank my girlfriend, Yuyin Zhou, for years of company, in both the academic research and our daily life. Additionally, I want to thank my kittens, Lucky, Ben and Rachel, for decorating my "boring" life during this COVID-19 pandemic.

To all my family members for their unreserved and unconditional support.

Contents

List of Tables

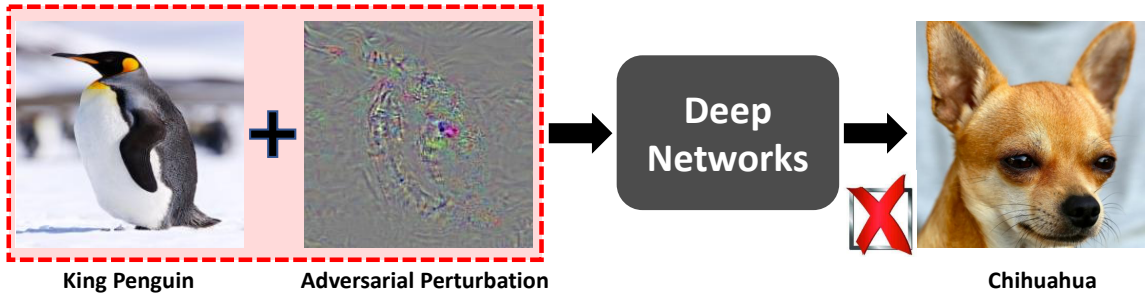
List of Figures

Chapter 1

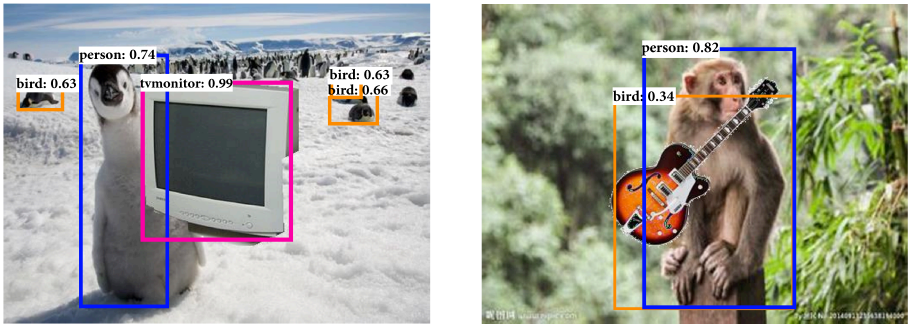
Introduction

Deep networks are highly successful in computer vision. Compared to traditional methods [10], [18], [40], [97], [125], [148], deep networks not only ease human efforts on feature engineering, but also yield significantly better performance on a wide range of visual benchmarks [24], [60], [61], [72], [99], [122], [168], [183]. Even eight years after the seminal work AlexNet [99], our exploration of deep networks has not yet came to an end, *e.g.*, deep networks keep achieving better performance when the advanced neural architectures are applied [118], [167], [197], [240], [250], [251], or the amount of training data is scaled [15], [98], [129], [192], [231].

However, interestingly, deep networks unexpectedly show catastrophic failures when testing beyond the traditional visual benchmarks [59], [74], [78], [79], [135], [170], [196], [207]. For examples, deep networks are sensitive to small changes in the input data (*i.e.*, adversarial examples, Figure 1.1(a)), incapable of generalizing well to previously unseen objects, easily fooled by occlusion (Figure 1.1(b)) or viewpoint variations, *etc.* Note that these scenarios generally cannot confuse human observers. Such failures not only raise profound safety and reliability concerns on the deployment of deep networks based cyber-physical systems, but also suggest that deep networks prone to exploit unintended shortcuts during learning [58], which is fundamentally different from the learning paradigm of the human perception system.



(a) Adversarial examples



(b) Occlusion patterns

Figure 1.1. Two failure cases of deep networks. *Top panel*: by adding adversarial perturbations to a king penguin image, the resulted example can fool deep networks to output the wrong prediction, *i.e.*, chihuahua. *Bottom panel*: The occluding monitor/guitar turns a penguin/monkey be to be wrongly detected as a person (with a high confidence score) by deep networks. While in both scenarios, these images generally cannot confuse human observers.

Towards the goal of developing human-level recognition systems, in this thesis, we focus on *identifying, understanding and eliminating* a particular yet challenging weakness of deep networks—their vulnerability to *adversarial examples*. Specifically, three aspects are explored: (1) We first investigate how severe that deep networks are exposed to adversarial examples, including the existence of adversarial examples beyond the image classification task and the transferability of adversarial examples across different network architectures; (2) We next discuss how deep networks, with different neural architecture designs, can effectively learn robust feature representations for defending against adversarial examples; (3) Lastly a showcase is provided to demonstrate the positive effects of adversarial examples on the representation learning of deep networks. We will elaborate these three aspects in the following sections.

1.1 Exploring Adversarial Vulnerability

The vulnerability of deep networks to adversarial examples is first discovered in [196]. To craft adversarial examples, we need to optimize the following objective:

$$\max_{\epsilon \in \mathbb{S}} L(f(\theta, x+\epsilon), y), \tag{1.1}$$

where $L(\cdot)$ is the loss function, $f(\cdot)$ is a deep network with parameter θ, x is an image with the ground-truth label y, ϵ is the adversarial perturbation, and $\mathbb{S}$ is the allowed perturbation range. With a successful optimization of this objective, the deep network will be fooled, *i.e.*, the network prediction $f(\theta, x+\epsilon)$ will be different from the ground-truth label y. To ensure the generated adversarial example (*i.e.*, $x+\epsilon$) is human-imperceptible, we usually restrict the perturbation range $\mathbb{S}$ to be small.

Though the discovery of deep networks' adversarial vulnerability can be dated back as early as 2013 [196], the severity of such issue is just extensively explored and discussed until most recently. For examples, early works in this direction study adversarial examples only in the image classification setting, and generally assume adversarial examples cannot effectively transfer across different network architectures. In this thesis, we challenge these traditional beliefs based on the following findings:

Adversarial examples for detection and segmentation. In Chapter 2, we present the first work to demonstrate the adversarial vulnerability of deep networks beyond the image classification task—adversarial examples can also exist in the complex and realistic computer vision scenarios like object detection and semantic segmentation [227]. Specifically, we first to define a dense set of targets inside each image (*e.g.*, targets are object proposals in detection and image pixels in segmentation), and then to optimize the loss function to produce incorrect predictions on all the targets simultaneously. We also extend this work to the real-world setting [86], where we create the first universal adversarial pattern for attacking advanced object detectors in the physical world.

Transferable adversarial examples. One intriguing property of adversarial examples is the transferability—adversarial examples generated on one specific model can remain malicious to others with certain probability. The existence of transferable adversarial examples indicates that networks (though with distinct architecture designs) tend to make similar mistakes. Nonetheless, such transferability is relatively weak as demonstrated in previous literature [64], [141], [196]. In Chapter 3, we focus on improving the transferability of adversarial examples. By noticing the overfitting of attackers to a specific deep network is the main factor that hampers transferability, we present a data augmentation based method to alleviate this optimization issue for boosting transferability [230]. We also briefly discuss the other explorations (*i.e.*, model augmentation [113] or enforcing locally smoothed adversarial perturbations [112]) to further the transferability of adversarial examples.

1.2 Exploring Adversarial Defenses

The aforementioned studies on deep networks' vulnerability in turn urge the progress on developing robust models against adversarial examples. Studying model robustness not only helps to alleviate the security concerns on deep networks based cyber-physical systems, but also encourages researchers to explore the missing recipes in the current deep network framework. To secure robustness, our countermeasures for adversarial examples are listed as the following.

To defend against adversarial examples, we mainly explore the effects of different architectures designs on helping networks gain robust feature representations. In Chapter 4, we notice adversarial perturbations on images lead to noise in the features constructed by these networks, therefore explore feature denoising approaches to improve the robustness of deep networks against adversarial examples [228]. Specifically, our networks contain blocks that denoise the features using non-local means or other filters; the entire networks are trained end-to-end. When combined with adversarial

training, our feature denoising network achieved the **state-of-the-art** robustness against adversarial examples on ImageNet in 2019.

In Chapter 5, we next explore the effects of activation functions in the context of adversarial training. We note that the widely-used ReLU activation function significantly weakens adversarial training due to its non-smooth nature, therefore develop smooth adversarial training (SAT), in which we replace ReLU with its smooth approximations (e.g., SILU, softplus, SmoothReLU) to strengthen adversarial training [225]. The purpose of smooth activation functions in SAT is to allow it to find harder adversarial examples and compute better gradient updates during adversarial training. Compared to previous adversarial defense works, SAT is the first work that can improve adversarial robustness for "free".

Though adversarial training offers models with strong robustness, our understanding to adversarial training is still limited, especially on large scale datasets, *e.g.*, previous works show adversarial training may cannot secure model robustness on ImageNet. In Chapter 6, we present the first work that delves into the key ingredients of training deep networks adversarially at scale, and reveals two intriguing properties [229]: (1) conducting normalization in the right manner is essential for training robust models; and (2) our so-called "deep" networks are still too shallow for adversarial learning. These findings not only serve as a guideline for instructing later works on training robust models at scale, but also provides empirical supports on the theoretical study of the relationship between model complexity and robust optimization.

1.3 Exploring Benefits of Robust Learning

Traditional studies generally consider adversarial examples as a threat to deep networks, and massive efforts have been made to increasing model robustness. In Chapter 7, we stand on an opposite perspective—adversarial examples can help deep networks to learn

more generalizable feature presentations. Specifically, we propose an enhanced training scheme, named AdvProp, which treats adversarial examples as additional examples (to clean images), to prevent overfitting. Key to AdvProp is the usage of a separate auxiliary batch norm for adversarial examples, as they have different underlying distributions to normal examples. AdvProp is the first work that demonstrates adversarial examples can improve (rather than degrade) image recognition on large scale datasets. The best model with AdvProp reports the **state-of-the-art** 85.5% ImageNet top-1 accuracy **without** extra data. This result even surpasses the work (with 84.4% ImageNet top-1 accuracy) [129] which is trained with 3.5B Instagram images (~3000× more than ImageNet) and ~9.4× more parameters.

This work successfully demonstrating the value of studying adversarial examples beyond the traditional model security domain—adversarial examples indeed can help recognition models. Besides, we would like to highlight that adversarial example has the potential to be a more general way for augmenting data—it can be useful everywhere (*e.g.*, language, structured data), not just for image recognition.

The following publications compose the main ideas in this dissertation.

1. Chapter 2 - **Cihang Xie**, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, Alan Yuille, "Adversarial Examples for Semantic Segmentation and Object Detection," in *ICCV*, 2017.

2. Chapter 3 - **Cihang Xie**, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, Alan Yuille, "Improving Transferability of Adversarial Examples with Input Diversity," in *CVPR*, 2019.

3. Chapter 4 - **Cihang Xie**, Yuxin Wu, Laurens van der Maaten, Alan Yuille, Kaiming He, "Feature Denoising for Improving Adversarial Robustness," in *CVPR*, 2019.

4. Chapter 5 - **Cihang Xie**, Mingxing Tan, Boqing Gong, Alan Yuille, Quoc Le, "Smooth Adversarial Training," in *Arxiv*, 2020

5. Chapter 6 - **Cihang Xie**, Alan Yuille, "Intriguing Properties of Adversarial Training at Scale," in *ICLR*, 2020.

6. Chapter 7 - **Cihang Xie**, Mingxing Tan, Boqing Gong, Jiang Wang, Alan Yuille, Quoc Le, "Adversarial Examples Improve Image Recognition," in *CVPR*, 2020.

Part I

Adversarial Vulnerability of Deep Neural Networks

Chapter 2

Adversarial Examples for Semantic Segmentation and Object Detection

It has been well demonstrated that adversarial examples, *i.e.*, natural images with visually imperceptible perturbations added, cause deep networks to fail on image classification. In this chapter, we extend adversarial examples to semantic segmentation and object detection, which are much more challenging than image classification. Our observation is that both segmentation and detection are based on classifying multiple targets on an image (*e.g.*, the target is a pixel or a receptive field in segmentation, and an object proposal in detection). This inspires us to optimize a loss function over a set of pixels/proposals for generating adversarial perturbations. Based on this idea, we propose a novel algorithm, Dense Adversary Generation (DAG), which generates a large family of adversarial examples, and applies to a wide range of state-of-the-art deep networks for segmentation and detection. We also find the generated adversarial perturbations can be transferred across networks with different training data, based on different architectures, and even for different recognition tasks. In particular, adversarial examples transfer the best across networks with the same architecture than in other cases. Besides, summing up heterogeneous perturbations leads to better transferability, which provides an effective solution of black-box adversarial attack.

2.1 Introduction

Deep networks [72], [84], [99], [183], [194] have become the de facto solution for a wide range of visual recognition problems. Based on a large-scale labeled dataset such as ImageNet [173] and powerful computational resources like modern GPUs, it is possible to train a hierarchical deep network to capture different levels of visual patterns. Deep networks are also capable of generating transferable features for different tasks such as image classification [45] and instance retrieval [166], or being fine-tuned to deal with a wide range of vision tasks, including object detection [39], [60], [61], [117], [168], semantic segmentation [24], [122], [247], boundary detection [179], [233], *etc*.

Despite their success in visual recognition and feature representation, deep networks are often sensitive to small perturbations to the input image. Szegedy *et al*. [196] showed that adding visually imperceptible perturbations can result in failures for image classification. These perturbed images, often called *adversarial examples*, are considered to fall on some areas in the large, high-dimensional feature space which are not explored in the training process. Investigating adversarial examples not only helps us understand the working mechanism of deep networks, but also provides opportunities to improve the robustness of network training.

In this chapter, we go one step further by generating adversarial examples for semantic segmentation and object detection, and showing the transferability of them. To the best of our knowledge, this topic has not been systematically studied (*e.g.*, on a large dataset) before. Note that these tasks are much more challenging, as we need to consider orders of magnitude more targets (*e.g.*, pixels or proposals). Motivated by the fact that each target undergoes a separate classification process, we propose the Dense Adversary Generation (DAG) algorithm, which considers all the targets simultaneously and optimizes the overall loss function. The implementation of DAG is simple, as it only involves specifying an adversarial label for each target and performing iterative

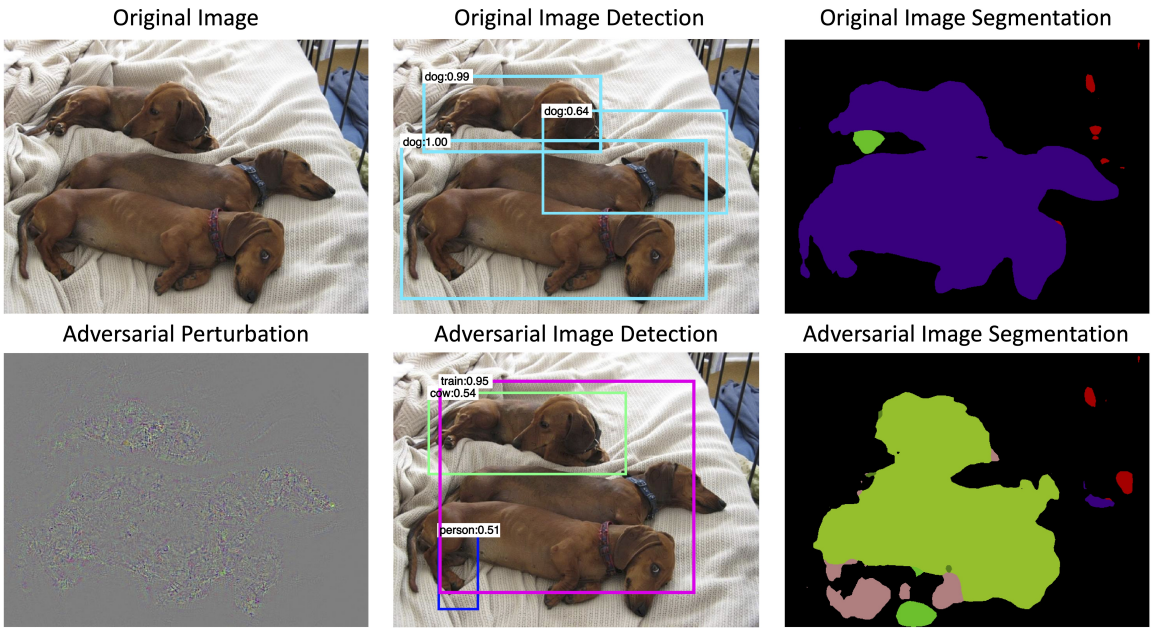


Figure 2.1. An Adversarial example for semantic segmentation and object detection. FCN [122] is used for segmentation, and Faster-RCNN [168] is used for detection. *Top row*: the original image with the normal segmentation (the purple region is predicted as *dog*) and detection results. *Bottom row*: after the adversarial perturbation (magnified by **10**) is added to the original image, both segmentation (the light green region as *train* and the pink region as *person*) and detection results are completely wrong. Note that, though the added perturbation can confuse both networks, it is visually imperceptible (the maximal absolute intensity in each channel is less than 10).

gradient back-propagation. In practice, the algorithm often comes to an end after a reasonable number of, say, 150 to 200, iterations. Figure 2.1 shows an adversarial example which can confuse both deep segmentation and detection networks.

We point out that generating an adversarial example is much more challenging in detection than in segmentation, as the number of targets is orders of magnitude larger in the former case, *e.g.*, for an image with K pixels, the number of possible proposals is $O(K^2)$ while the number of pixels is only $O(K)$, where $O(\cdot)$ is the big-O notation. In addition, if only a subset of proposals are considered, the perturbed image may still be correctly recognized after a new set of proposals are extracted (as DAG only aims to fail the model's recognition on the originally selected proposals). To increase the robustness of DAG, we change the intersection-over-union (IOU) rate to preserve an

increased but still reasonable number of proposals as the attack targets. We empirically verify that when enough proposals are considered in DAG, it is highly likely incorrect recognition results can also be produced on the newly generated proposals of the perturbed image. We also study the effectiveness and efficiency of the algorithm with respect to the *denseness* of the considered proposals.

Following [196], we investigate the transferability of the generated perturbations. Specifically, we use the adversarial perturbation computed on one network to attack another network, and three situations are considered: (1) networks with the same architecture but trained with different data; (2) networks with different architectures but trained for the same task; and (3) networks for different tasks. Although the difficulty increases as the difference goes more significant, the perturbations generated by DAG is able to transfer to some extent. Interestingly, adding two or more heterogeneous perturbations significantly increases the transferability, which provides an effective way of performing black-box adversarial attack [155] to networks with unknown structures and/or properties.

The remainder of this chapter is organized as follows. Section 2.2 briefly introduces prior work related to our research. Section 2.3 describes our algorithm for generating adversarial perturbations, and Section 2.4 investigates the transferability of the perturbations. Section 2.5 discusses the follow-up on attacking object detectors in the physical world. Conclusions are drawn in Section 2.6.

2.2 Related Work

Deep networks for detection and segmentation. Deep networks are very successful in object detection [39], [168] and semantic segmentation [24], [122] tasks. Currently, one of the most popular object detection pipelines [39], [168] involves first generating a number of proposals at different scales and positions, classifying each of

them, and performing post-processing such as non-maximal suppression (NMS). On the other hand, the dominating segmentation pipeline [122] works by first predicting a class-dependent score map at a reduced resolution, and performing up-sampling to obtain high-resolution segmentation. Chen *et al.* [24] incorporates the "atrous" algorithm and the conditional random field to further segmentation performance.

Adversarial attacks. Generating adversarial examples for image classification has been extensively studied recently. Szegedy *et al.* [196] first showed adversarial examples, computed by adding visually imperceptible perturbations to the original images, make deep networks predict a wrong label with high confidence. Goodfellow *et al.* [64] developed the fast gradient sign method to generate adversarial examples based on the linear nature of deep networks. Moosavi-Dezfooli *et al.* [141] proposed a simple algorithm to compute the minimal adversarial perturbation by assuming that the loss function can be linearized around the current data point at each iteration. This algorithm is later extended to find universal (image-agnostic) adversarial perturbations [140]. Unlike adversarial examples which can be recognized by human, Nguyen *et al.* [147] generated fooling images that are different from natural images and difficult for human to recognize, but deep networks classify these images with high confidence.

There are two concurrent works [55], [73] that studied adversarial examples in semantic segmentation on the Cityscapes dataset [34], where [55] showed the existence of adversarial examples, and [73] showed the existence of universal perturbations. We refer interested readers to their papers for details.

2.3 Generating Adversarial Examples

In this section, we introduce DAG algorithm. Given an image and the recognition targets (proposals and/or pixels), DAG generates an adversarial perturbation which is aimed to confuse as many targets as possible.

2.3.1 Dense Adversary Generation

Let $\mathbf{X}$ be an image which contains N recognition targets $\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$. Each target t_n, $n = 1, 2, \ldots, N$, is assigned a ground-truth class label $l_n \in \{1, 2, \ldots, C\}$, where C is the number of classes, *e.g.*, $C = 21$ (including the *background* class) in the PascalVOC dataset [51]. Denote $\mathcal{L} = \{l_1, l_2, \ldots, l_n\}$. The detailed form of $\mathcal{T}$ varies among different tasks. In image classification, $\mathcal{T}$ only contains one element, *i.e.*, the entire image. Conversely, $\mathcal{T}$ is composed of all pixels (or the corresponding receptive fields) in semantic segmentation, and all proposals in object detection. We will discuss how to construct $\mathcal{T}$ in Section 2.3.2.

Given a deep network for a specific task, we use $\mathbf{f}(\mathbf{X}, t_n) \in \mathbb{R}^C$ to denote the classification logits (*i.e.*, the outputs before softmax) of the n-th recognition target in the image $\mathbf{X}$. To generate an adversarial example, the goal is to make the predictions of all targets go wrong, *i.e.*, $\forall n$, $\arg\max_c \{f_c(\mathbf{X} + \mathbf{r}, t_n)\} \neq l_n$. Here $\mathbf{r}$ denotes an adversarial perturbation added to $\mathbf{X}$. To this end, we specify an adversarial label l'_n for each target, in which l'_n is randomly sampled from other incorrect classes, *i.e.*, $l'_n \in \{1, 2, \ldots, C\} \setminus \{l_n\}$. Denote $\mathcal{L}' = \{l'_1, l'_2, \ldots, l'_n\}$. In practice, we define a random permutation function $\pi : \{1, 2, \ldots, C\} \to \{1, 2, \ldots, C\}$ for each image independently, in which $\pi(c) \neq c$ for $c = 1, 2, \ldots, C$, and generate $\mathcal{L}'$ by setting $l'_n = \pi(l_n)$ for all n. Under this setting, the loss function of all targets can be written as:

$$L(\mathbf{X}, \mathcal{T}, \mathcal{L}, \mathcal{L}') = \sum_{n=1}^{N} \left[f_{l_n}(\mathbf{X}, t_n) - f_{l'_n}(\mathbf{X}, t_n) \right] \tag{2.1}$$

Minimizing L can be achieved via making every target to be incorrectly predicted, *i.e.*, suppressing the confidence of the original correct class $f_{l_n}(\mathbf{X} + \mathrm{r}, t_n)$, while increasing that of the desired (adversarial) incorrect class $f_{l'_n}(\mathbf{X} + \mathrm{r}, t_n)$.

We apply the gradient descent algorithm for optimization. For the image at m-th iteration (denoted as $\mathbf{X}_m$), we first find its set of correctly predicted targets, named the *active target set*: $\mathcal{T}_m = \{t_n \mid \arg\max_c \{f_c(\mathbf{X}_m, t_n)\} = l_n\}$, and then accumulate

Algorithm 1: Dense Adversary Generation (DAG)

Input : input image $\mathbf{X}$;
the classifier $\mathbf{f}(\cdot,\cdot) \in \mathbb{R}^C$;
the target set $\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$;
the original label set $\mathcal{L} = \{l_1, l_2, \ldots, l_N\}$;
the adversarial label set $\mathcal{L}' = \{l'_1, l'_2, \ldots, l'_N\}$;
the maximal iterations M_0;

Output : the adversarial perturbation $\mathbf{r}$;

$\mathbf{X}_0 \leftarrow \mathbf{X}$, $\mathbf{r} \leftarrow \mathbf{0}$, $m \leftarrow 0$, $\mathcal{T}_0 \leftarrow \mathcal{T}$;
while $m < M_0$ **and** $\mathcal{T}_m \neq \varnothing$ **do**
  $\mathcal{T}_m = \{t_n \mid \arg\max_c \{f_c(\mathbf{X}_m, t_n)\} = l_n\}$;
  $\mathbf{r}_m \leftarrow \sum_{t_n \in \mathcal{T}_m} \left[\nabla_{\mathbf{X}_m} f_{l'_n}(\mathbf{X}_m, t_n) - \nabla_{\mathbf{X}_m} f_{l_n}(\mathbf{X}_m, t_n)\right]$;
  $\mathbf{r}'_m \leftarrow \frac{\gamma}{\|\mathbf{r}_m\|_\infty} \mathbf{r}_m$;
  $\mathbf{r} \leftarrow \mathbf{r} + \mathbf{r}'_m$;
  $\mathbf{X}_{m+1} \leftarrow \mathbf{X}_m + \mathbf{r}'_m$;
  $m \leftarrow m + 1$;
end

Return : $\mathbf{r}$

the corresponding gradients with respect to the input image $\mathbf{X}_m$ as the following:

$$\mathbf{r}_m = \sum_{t_n \in \mathcal{T}_m} \left[\nabla_{\mathbf{X}_m} f_{l'_n}(\mathbf{X}_m, t_n) - \nabla_{\mathbf{X}_m} f_{l_n}(\mathbf{X}_m, t_n)\right] \tag{2.2}$$

Note that $|\mathcal{T}_m| \ll |\mathcal{T}|$ when m gets large, thus this strategy considerably reduces the computational overhead. To avoid numerical instability, we normalize $\mathbf{r}_m$ as

$$\mathbf{r}'_m = \frac{\gamma}{\|\mathbf{r}_m\|_\infty} \cdot \mathbf{r}_m \tag{2.3}$$

where $\gamma = 0.5$ is a fixed hyper-parameter. We then add $\mathbf{r}'_m$ to the current image $\mathbf{X}_m$ and proceed to the next iteration. The algorithm terminates if either all the targets are predicted as desired, *i.e.*, $\mathcal{T}_m = \varnothing$, or it reaches the maximum iteration number, which is set to be 200 in segmentation and 150 in detection.

The final adversarial perturbation is computed as $\mathbf{r} = \sum_m \mathbf{r}'_m$. Note that, in practice, we often obtain the input image $\mathbf{X}$ after subtracting the mean image $\widehat{\mathbf{X}}$. In this case, the adversarial image is $\text{Trunc}\left(\mathbf{X} + \mathbf{r} + \widehat{\mathbf{X}}\right)$, where $\text{Trunc}(\cdot)$ denotes the function that truncates every pixel value within the range $[0, 255]$. Although truncation may harm

the adversarial perturbation, we empirically observed little effects, mainly because the magnitude of the adversarial perturbation $\mathbf{r}$ is very small (see Section 2.3.5). The overall pipeline of DAG algorithm is illustrated in Algorithm 1.

2.3.2 Selecting Input Proposals for Detection

A critical issue in DAG is to select a proper set $\mathcal{T}$ of targets. This is relatively easy in the semantic segmentation task, because the goal is to produce incorrect predictions on all pixels. In practice, we set each pixel as a separate target, *i.e.*, performing dense sampling on the image lattice, and the computational complexity of DAG is proportional to the total number of pixels.

In the scenario of object detection, the selection of the target set $\mathcal{T}$ becomes a lot more challenging, as the total number of possible targets (bounding box proposals) is orders of magnitudes larger than that in semantic segmentation. A straightforward choice is to only consider the proposals generated by a sideway network, *e.g.*, the regional proposal network (RPN) [168]. However, we find that when the adversarial perturbation $\mathbf{r}$ is added back to the original image $\mathbf{X}$, a different set of proposals will be generated according to the new input $\mathbf{X} + \mathbf{r}$, and the network may still be able to correctly classify these new proposals [124]. To overcome this problem, we make the proposals very dense by increasing the threshold of NMS in RPN. In practice, when IOU goes up from 0.70 to 0.90, the average number of proposals on each image increases from ~300 to ~3000. Using this denser target set $\mathcal{T}$, most probable object bounding boxes are only pixels away from at least one of the selected input proposals, and we can expect the classification error transfers among neighboring bounding boxes. We empirically observe that this heuristics works very well, and the effect of adversarial perturbations is positively correlated to the number of proposals considered in DAG.

Technically, given the proposals generated by RPN, we preserve all *positive* proposals and discard the remaining. Here, a positive proposal satisfies the following two

conditions: 1) the IOU with the closest ground-truth object is greater than 0.1, and 2) the confidence score for the corresponding ground-truth class is greater than 0.1. If both conditions hold on multiple ground-truth objects, we select the one with the maximal IOU. The label of the proposal is defined as the corresponding confident class. This strategy aims to select high-quality targets for Algorithm 1.

2.3.3 Quantitative Evaluation

Following [141], [196], we evaluate the effectiveness of DAG by measuring the gap between the model performance on the original test images and the model performance on the adversarially perturbed counterparts[1].

- For semantic segmentation, we study two network architectures based on the FCN [122] framework. One of them is based on the AlexNet [99] and the other one is based on the 16-layer VGGNet [183]. Both networks have two variants. We use FCN-Alex and FCN-VGG, which are publicly available, to denote the networks that are trained on the original FCN training set which has 9610 images, and use FCN-Alex* and FCN-VGG* to denote the networks that are trained on the DeepLab [24] training set which has 10582 images. We use the validation set in [122] (736 images) as our semantic segmentation test set.

- For object detection, based on the Faster-RCNN [168] framework, we study two network architectures, *i.e.*, the ZFNet [237] and the 16-layer VGGNet. Both networks have two variants, which are either trained on the PascalVOC-2007 trainval set, or the combined PascalVOC-2007 and PascalVOC-2012 trainval sets. These four models are publicly available, and are denoted as FR-ZF-07, FR-ZF-0712, FR-VGG-07 and FR-VGG-0712, respectively. We use the PascalVOC-2007 test set which has 4952 images as our object detection test set.

[1]For implementation simplicity, we keep targets with ground-truth class label *background* unchanged when generating adversarial examples.

Network	ORIG	ADVR	PERM
FCN-Alex	48.04	3.98	48.04
FCN-Alex*	48.92	3.98	48.91
FCN-VGG	65.49	4.09	65.47
FCN-VGG*	67.09	4.18	67.08
FR-ZF-07	58.70	3.61	58.33
FR-ZF-0712	61.07	1.95	60.94
FR-VGG-07	69.14	5.92	68.68
FR-VGG-0712	72.07	3.36	71.97

Table 2.1. Semantic segmentation (measured by mIOU, %) and object detection (measured by mAP, %) results of different networks. ORIG is the accuracy on the original image set, ADVR is the accuracy on the adversarially perturbed image set, and PERM is the accuracy on the image set with randomly permuted adversarial perturbations.

Results are summarized in Table 2.1. We can observe that the accuracy (mean IOU (mIOU) for segmentation and mean average precision (mAP) for detection) drops significantly after the adversarial perturbations are added, demonstrating the effectiveness of DAG algorithm. Moreover, for detection, the networks with more training data are often more sensitive to the adversarial perturbation. This is verified by the fact that FR-ZF-07 (from 58.70% to 3.61%) has a smaller performance drop than FR-ZF-0712 (from 61.07% to 1.95%), and that FR-VGG-07 (from 69.14% to 5.92%) has a smaller performance drop than FR-VGG-0712 (from 72.04% to 3.36%).

To verify the importance of the spatial structure of adversarial perturbations, we evaluate the accuracy after randomly permuting the rows and/or columns of $\mathbf{r}$. In Table 2.1, we find that permuted adversarial perturbations cause negligible accuracy drop, indicating that it is the spatial structure of $\mathbf{r}$, instead of its magnitude, contributes the most to the malicious effects of adversarial examples. For permutation results, we randomly permute $\mathbf{r}$ for three times and report the average.

2.3.4 Fancy Adversarial/Fooling Images

In addition to simply confusing networks to output wrong predictions, DAG is also able to control those wrong predictions in a very fine-grained manner. As shown in

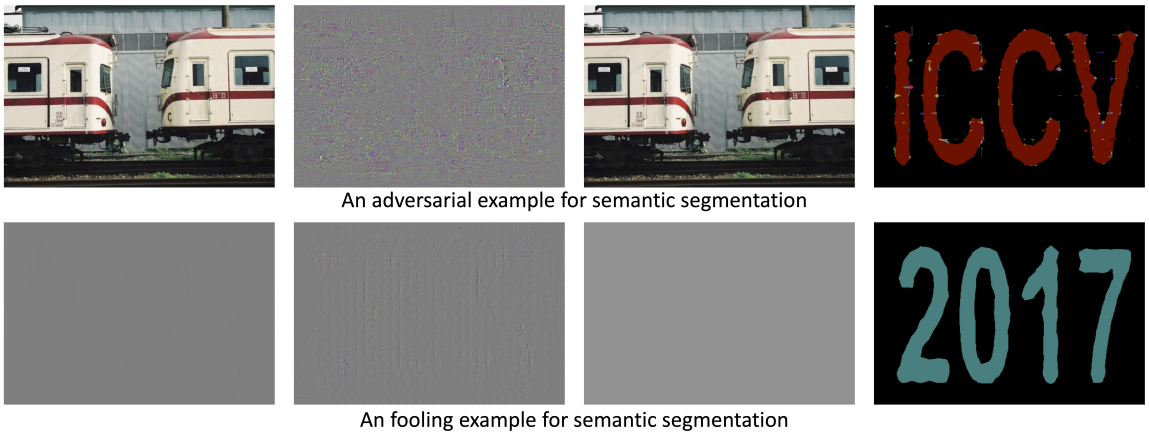


Figure 2.2. Fancy examples generated by DAG for semantic segmentation. An *adversarial* image is shown in the top and an *fooling* image is shown in the bottom. From leftmost to rightmost: the original image, the perturbation (**magnified by 10**), the perturbed image, and the segmentation results. The red, blue and black regions correspond to *airplane*, *bus* and *background*, respectively.

Figure 2.2, DAG successfully generates one *adversarial* image (which humans can recognize but deep networks are failed) with the network predictions of *ICCV* shape, and one *fooling* image [147] (which is completely unrecognizable to humans but deep networks produce false positives) with the network predictions of *2017* shape.

2.3.5 Diagnostics

Denseness of proposals. We first analyze the impact of proposal denseness on DAG. To this end, we use different IOU rates in NMS, which directly affects the number of proposals preserved in Algorithm 1. As we can see in Figure 2.3, the mAP value goes down (*i.e.*, stronger adversarial perturbations are generated) as the IOU rate increases (*i.e.*, fewer proposals are filtered out and thus the set of targets $\mathcal{T}$ becomes larger). This phenomenon is in line of our expectation, since DAG only guarantees misclassification on the targets in $\mathcal{T}$. The denser sampling on proposals allows the recognition error to propagate to other possible object positions better. This observation motivates us empirically to choose a large IOU value (0.90) for strengthening DAG.

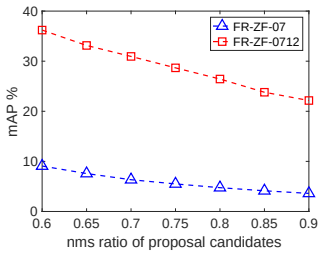


Figure 2.3. The mAP of using adversarial perturbations on FR-ZF-07 to attack FR-ZF-07 and FR-ZF-0712, with respect to the IOU rate. A larger IOU rate leads to a denser set of proposals.

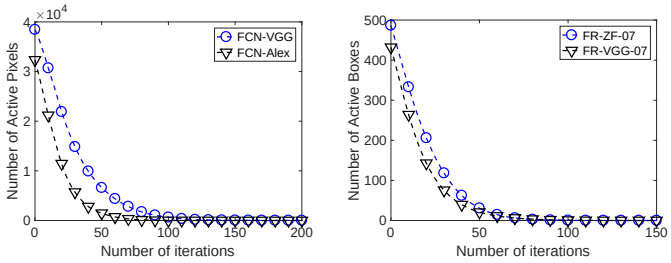


Figure 2.4. The convergence of DAG measured by the number of active targets, *i.e.*, $|\mathcal{T}_m|$, with respect to the number of iterations. Over the entire dataset, the average number of iteration is 31.78 and 54.02 for FCN-Alex and FCN-VGG respectively, and is 47.05 and 41.42 for FR-ZF-07 and FR-VGG-07 respectively.

Convergence. We then investigate the convergence of DAG, *i.e.*, how many iterations are needed to find the desired adversarial perturbation. Figure 2.4 shows the number of active targets, *i.e.*, $|\mathcal{T}_m|$, with respect to the number of iterations m. In general, the attacking process goes smoothly in the early rounds, in which we find that the number of active proposals is significantly reduced. After the algorithm reaches the maximal number of iterations, *i.e.*, 200 in segmentation and 150 in detection, only few (less than 1%) image fail to converge. Note that even on these "uncoverged" cases, DAG is still able to produce reasonable adversarial perturbations.

Another interesting observation is the difficulty in generating adversarial examples. In general, the detection networks are more difficult to attack than the segmentation networks, which is arguably caused by the much larger number of potential targets (recall that the total number of possible bounding boxes is one or two orders of magnitudes larger). Meanwhile, as the IOU rate increases, *i.e.*, a larger set $\mathcal{T}$ of proposals is considered, convergence also becomes slower, implying that more iterations are required to generate stronger adversarial perturbations.

Perceptibility. Following [141], [196], we compute the perceptibility of the adversarial perturbation $\mathbf{r}$ defined by $p = \left(\frac{1}{K}\sum_k \|\mathbf{r}_k\|_2^2\right)^{1/2}$, where K is the number of pixels, and $\mathbf{r}_k$ is the intensity vector (3-dimensional in the RGB color space, $k = 1, 2, 3$) normalized

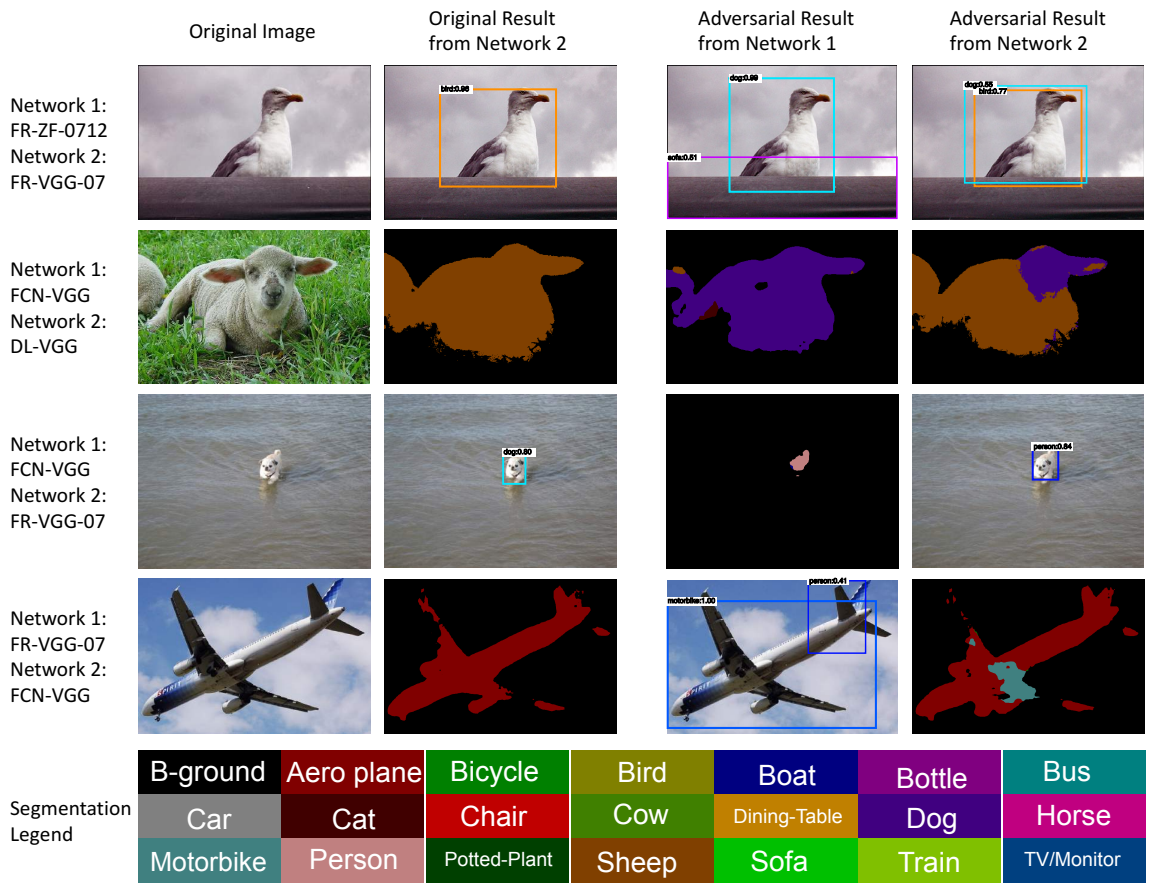


Figure 2.5. Transferable adversarial examples for semantic segmentation and object detection. These four rows, from top to bottom, show the transferability of adversarial examples, from one detection network to another detection network, from one segmentation network to another segmentation network, from one detection network to one segmentation network, from one segmentation network to one detection network, respectively. The segmentation legend are borrowed from [247].

in $[0, 1]$. We average the perceptibility value over the entire test set. In semantic segmentation, these values are 2.6×10^{-3}, 2.5×10^{-3}, 2.9×10^{-3} and 3.0×10^{-3} on FCN-Alex, FCN-Alex*, FCN-VGG and FCN-VGG*, respectively. In object detection, these values are 2.4×10^{-3}, 2.7×10^{-3}, 1.5×10^{-3} and 1.7×10^{-3} on FR-ZF-07, FR-ZF-0712, FR-VGG-07 and FR-VGG-0712, respectively. All these values are very small, which quantitatively guarantees the imperceptibility of the generated adversarial perturbations in DAG. Our visualized examples (Figures 2.1 and 2.2) also qualitatively corroborates this point.

Adversarial Perturbations from	**FR-ZF-07**	**FR-ZF-0712**	**FR-VGG-07**	**FR-VGG-0712**	**R-FCN-RN50**	**R-FCN-RN101**
None	58.70	61.07	69.14	72.07	76.40	78.06
FR-ZF-07 ($\mathbf{r}_1$)	**3.61**	22.15	66.01	69.47	74.01	75.87
FR-ZF-0712 ($\mathbf{r}_2$)	13.14	**1.95**	64.61	68.17	72.29	74.68
FR-VGG-07 ($\mathbf{r}_3$)	56.41	59.31	**5.92**	48.05	72.84	74.79
FR-VGG-0712 ($\mathbf{r}_4$)	56.09	58.58	31.84	**3.36**	70.55	72.78
$\mathbf{r}_1 + \mathbf{r}_3$	**3.98**	21.63	**7.00**	44.14	68.89	71.56
$\mathbf{r}_1 + \mathbf{r}_3$ (**permute**)	58.30	61.08	68.63	71.82	76.34	77.71
$\mathbf{r}_2 + \mathbf{r}_4$	13.15	**2.13**	28.92	**4.28**	63.93	67.25
$\mathbf{r}_2 + \mathbf{r}_4$ (**permute**)	58.51	61.09	68.68	71.78	76.23	77.71

Table 2.2. Transfer results for detection networks. FR-ZF-07, FR-ZF-0712, FR-VGG-07 and FR-VGG-0712 are used as four basic models to generate adversarial perturbations, and R-FCN-RN50 and R-FCN-RN101 are used as black-box models. All models are evaluated on the PascalVOC-2007 test set and its adversarial counterpart.

2.4 Transferring Adversarial Perturbations

We hereby investigate the transferability of the generated adversarial perturbations. Specifically, we add the adversarial perturbation computed on one model to attack other models. The attacked model may be trained based on a different (sometimes unknown) network architecture, or even targeted at a different vision task. Quantitative results are summarized in Tables 2.2 - 2.4, and typical examples are shown in Figure 2.5. In the following parts, we analyze these results by organizing them into three categories, *i.e.*, *cross-training* transfer, *cross-network* transfer and *cross-task* transfer.

2.4.1 Cross-Training Transfer

By *cross-training* transfer, we mean to apply the perturbations learned from one network to another network with the same architecture but trained on a different dataset. We observe that the transferability *largely* exists within the same network structure[2]. For example, using the adversarial perturbations generated by FR-ZF-07 to attack FR-ZF-0712 obtains a 22.15% mAP. This is a dramatic drop from the

[2]We also studied training on strictly non-overlapping datasets, *e.g.*, the model FR-ZF-07 trained on PascalVOC-2007 trainval set and the model FR-ZF-12val trained on PascalVOC-2012 val set. The experiments deliver similar conclusions. For example, using FR-ZF-07 to attack FR-ZF-12val results in a mAP drop from 56.03% to 25.40%, and using FR-ZF-12val to attack FR-ZF-07 results in a mAP drop from 58.70% to 30.41%.

Adversarial Perturbations from	**FCN-Alex**	**FCN-Alex***	**FCN-VGG**	**FCN-VGG***	**DL-VGG**	**DL-RN101**
None	48.04	48.92	65.49	67.09	70.72	76.11
FCN-Alex ($\mathbf{r}_5$)	**3.98**	7.94	64.82	66.54	70.18	75.45
FCN-Alex* ($\mathbf{r}_6$)	5.10	**3.98**	64.60	66.36	69.98	75.52
FCN-VGG ($\mathbf{r}_7$)	46.21	47.38	**4.09**	16.36	45.16	73.98
FCN-VGG* ($\mathbf{r}_8$)	46.10	47.21	12.72	**4.18**	46.33	73.76
$\mathbf{r}_5 + \mathbf{r}_7$	**4.83**	8.55	**4.23**	17.59	43.95	73.26
$\mathbf{r}_5 + \mathbf{r}_7$ (**permute**)	48.03	48.90	65.47	67.09	70.69	76.04
$\mathbf{r}_6 + \mathbf{r}_8$	5.52	**4.23**	13.89	**4.98**	44.18	73.01
$\mathbf{r}_6 + \mathbf{r}_8$ (**permute**)	48.03	48.90	65.47	67.05	70.69	76.05

Table 2.3. Transfer results for segmentation networks. FCN-Alex, FCN-Alex*, FCN-VGG and FCN-VGG* are used as four basic models to generate adversarial perturbations, and DL-VGG and DL-RN101 are used as black-box models. All models are evaluated on validation set in [122] and its adversarial counterpart.

performance (61.07%) reported on the original images, although the drop is less than that observed in attacking FR-ZF-07 itself (from 58.70% to 3.61%). Meanwhile, using the adversarial perturbations generated by FR-ZF-0712 to attack FR-ZF-07 causes the mAP drop from 58.70% to 13.14%, We observe similar phenomena when FR-VGG-07 and FR-VGG-0712, or FCN-Alex and FCN-Alex*, or FCN-VGG and FCN-VGG* are used to attack each other. Detailed results are shown in Tables 2.2 and Table 2.3.

2.4.2 Cross-Network Transfer

We then study the transferability across different network structures. We additionally consider two models, namely DeepLab [24] for semantic segmentation and R-FCN [39] for object detection, for accessing the transferability on unknown networks. For DeepLab, we use DL-VGG to denote the network based on 16-layer VGGNet [183], and use DL-RN101 to denote the network based on 101-layer ResNet. Both networks are trained on original DeepLab training set which has 10582 images. For R-FCN, we use R-FCN-RN50 to denote the network based on 50-layer ResNet, and use R-FCN-RN101 to denote the network based on 101-layer ResNet. Both networks are trained on the combined trainval sets of PascalVOC-2007 and PascalVOC-2012. Note the attacks applied to these four models are considered as in the black-box setting [155], since DAG does not know the structure of these networks beforehand.

Adversarial Perturbations from	**FR-ZF-07**	**FR-VGG-07**	**FCN-Alex**	**FCN-VGG**	**R-FCN-RN101**
None	56.83	68.88	35.73	54.87	80.20
FR-ZF-07 ($\mathbf{r}_1$)	**5.14**	66.63	31.74	51.94	76.00
FR-VGG-07 ($\mathbf{r}_3$)	54.96	**7.17**	34.53	43.06	74.50
FCN-Alex ($\mathbf{r}_5$)	55.61	68.62	**4.04**	54.08	77.09
FCN-VGG ($\mathbf{r}_7$)	55.24	56.33	33.99	**4.10**	73.86
$\mathbf{r}_1 + \mathbf{r}_3 + \mathbf{r}_5$	**5.02**	**8.75**	**4.32**	37.90	69.07
$\mathbf{r}_1 + \mathbf{r}_3 + \mathbf{r}_7$	**5.15**	**5.63**	28.48	**4.81**	65.23
$\mathbf{r}_1 + \mathbf{r}_5 + \mathbf{r}_7$	**5.14**	47.52	**4.37**	**5.20**	68.51
$\mathbf{r}_3 + \mathbf{r}_5 + \mathbf{r}_7$	53.34	**5.94**	**4.41**	**4.68**	67.57
$\mathbf{r}_1 + \mathbf{r}_3 + \mathbf{r}_5 + \mathbf{r}_7$	**5.05**	**5.89**	**4.51**	**5.09**	64.52

Table 2.4. Transfer results between detection networks and segmentation networks. FR-ZF-07, FR-VGG-07, FCN-Alex and FCN-VGG are used as four basic models to generate adversarial perturbations, and R-FCN-RN101 is used as the black-box model. When attacking the first four basic networks, we use a subset of the PascalVOC-2012 segmentation validation set which contains 687 images. In the black-box attack, we evaluate our method on the non-intersecting subset of 110 images.

Detailed results are shown in Tables 2.2 and Table 2.3. Experiments reveal that transferability between different network structures is relatively weak. For example, applying the adversarial examples generated by FR-ZF-07 leads to slight accuracy drop on FR-VGG-07 (from 69.14% to 66.01%), FR-VGG-0712 (from 72.07% to 69.74%), R-FCN-RN50 (from 76.40% to 74.01%) and R-FCN-RN101 (from 78.06% to 75.87%), respectively. Similar phenomena is observed in using different segmentation models to attack each other. The only exception is using FCN-VGG or FCN-VGG* to attack DL-VGG (from 70.72% to 45.16% for FCN-VGG attack, and from 70.72% to 46.33% by FCN-VGG* attack), which results in a significant accuracy drop of DL-VGG. Considering the cues obtained from previous experiments, we conclude that the transferability of adversarial examples is closely related to network architectures.

2.4.3 Cross-Task Transfer

Lastly, we investigate *cross-task* transfer, *i.e.*, using the perturbations generated by a detection network to attack a segmentation network or in the opposite direction. We use a subset of PascalVOC-2012 segmentation validation set as our test set[3]. Results

[3]There are training images of FR-ZF-07, FR-VGG-07, FCN-Alex and FCN-VGG included in the PascalVOC-2012 segmentation validation set, so we evaluate on the non-intersecting 687 images.

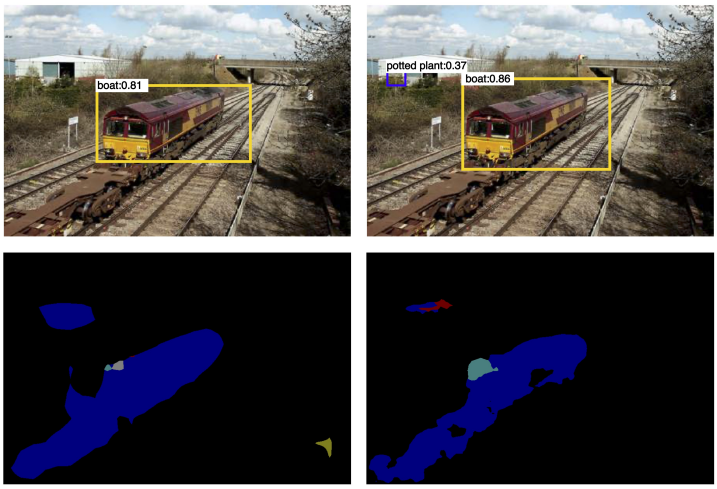


Figure 2.6. The adversarial example (computed by $\mathbf{r}_1 + \mathbf{r}_3 + \mathbf{r}_5 + \mathbf{r}_7$, see Table 2.4) that simultaneously fools four different networks. The top row shows FR-VGG-07 and FR-ZF-07 detection results, and the bottom row shows FCN-Alex and FCN-VGG segmentation results. The blue region in segmentation results corresponds to *boat*.

are summarized in Table 2.4. We note that if the same network structure is used, *e.g.*, using FCN-VGG (segmentation) and FR-VGG-07 (detection) to attack each other, the accuracy drop is significant (the mIOU of FCN-VGG drops from 54.87% to 43.06%, and the mAP of FR-VGG-07 drops from 68.88% to 56.33%). Note that this drop is even larger than the drop observed in *cross-network* transfer on the same task, which again supports our hypothesis that the transferability of adversarial examples is closely related to network architectures.

2.4.4 Combining Heterogeneous Perturbations

Based on the analysis above, we conjecture different network structures generate roughly *orthogonal* perturbations, *e.g.*, if $\mathbf{r}_{\mathbb{A}}$ is generated by one structure $\mathbb{A}$, then adding it to another structure $\mathbb{B}$ merely changes the results, $\mathbf{f}^{\mathbb{B}}(\mathbf{X}, t_n) \approx \mathbf{f}^{\mathbb{B}}(\mathbf{X} + \mathbf{r}_{\mathbb{A}}, t_n)$. This motivates us to combine heterogeneous perturbations to create more transferable adversarial examples. For example, if both $\mathbf{r}_{\mathbb{A}}$ and $\mathbf{r}_{\mathbb{B}}$ are added, we will have $\mathbf{f}^{\mathbb{A}}(\mathbf{X} + \mathbf{r}_{\mathbb{A}} + \mathbf{r}_{\mathbb{B}}, t_n) \approx \mathbf{f}^{\mathbb{A}}(\mathbf{X} + \mathbf{r}_{\mathbb{A}}, t_n)$ and $\mathbf{f}^{\mathbb{B}}(\mathbf{X} + \mathbf{r}_{\mathbb{A}} + \mathbf{r}_{\mathbb{B}}, t_n) \approx \mathbf{f}^{\mathbb{B}}(\mathbf{X} + \mathbf{r}_{\mathbb{B}}, t_n)$, which mean the combined perturbation $\mathbf{r}_{\mathbb{A}} + \mathbf{r}_{\mathbb{B}}$ can confuse both networks.

In Tables 2.2 - 2.4, we show the attack performance of such combined adversarial perturbations. We observe that combining multiple adversarial perturbations significantly boosts transferability. For example, the adversarial perturbation $\mathbf{r}_2 + \mathbf{r}_4$ (combining FR-ZF-0712 and FR-VGG-0712) causes significant mAP drop on all ZFNet-based and VGGNet-based detection networks, and the adversarial perturbation $\mathbf{r}_5 + \mathbf{r}_7$ (combining FCN-Alex* and FCN-VGG*) causes significant mIOU drop on all AlexNet-based and VGGNet-based segmentation networks. We also show such an example in Figure 2.6, where four perturbations (*i.e.*, $\mathbf{r}_1 + \mathbf{r}_3 + \mathbf{r}_5 + \mathbf{r}_7$, see Table 2.4) are added together. Note that, the perceptibility value defined in Section 2.3.5 remains very small for combined perturbation, *e.g.*, it is 4.0×10^{-3} in Figure 2.6.

Additionally, to verify that the spatial structure of combined adversarial perturbations is the key for misleading deep networks, we randomly generate three permuted versions of the combined adversarial perturbations and report the average accuracy. As shown in Table 2.2 and Table 2.3, permutation destroys the spatial structure of the adversarial perturbations, leading to negligible accuracy drops. The same conclusion also holds when the perturbations from different tasks are combined.

2.4.5 Black-Box Attack

Combining heterogeneous perturbations allows us to perform better on the so-called *black-box setting* [155], where we do not know the detailed properties (*e.g.*, architecture, purpose) of the targeted networks. Based on the analysis above, a simple and effective solution is to compute the sum of adversarial perturbations from several of known networks, such as FR-ZF-07, FR-VGG-07 and FCN-Alex, and use it to attack an unknown network. This strategy even works well when the structure of the targeted network is completely not investigated before. As an example shown in Table 2.4, the perturbation $\mathbf{r}_1 + \mathbf{r}_3 + \mathbf{r}_5 + \mathbf{r}_7$ leads to significant accuracy drop (from 80.20% to 64.52%) on R-FCN-RN101, a powerful network based on the deep ResNet.

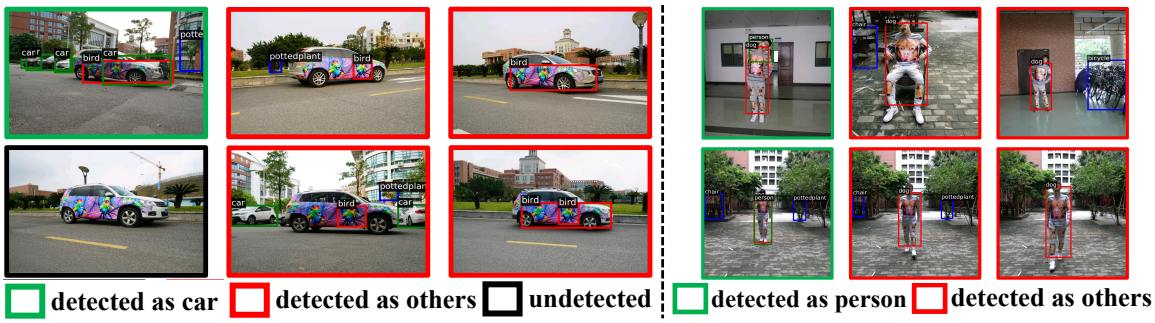


Figure 2.7. Adversarial camouflage patterns for preventing cars/persons from being correctly detected by deep networks in the physical world.

2.5 Universal Physical Camouflage Attacks

We further extend this work to attack object detectors in the real world. Though prior works have revealed the adversarial vulnerability of object detectors under this scenario [26], [52], [245], there are several limitations: (1) focusing on only attacking a specific object (*e.g.* a stop sign [26], [53], commercial logo [185] or car [245]); (2) generating perturbations only for rigid or planar objects (*e.g.*, traffic sign, vehicle body, board [198]), which can be less effective for complex objects (articulated non-rigid or non-planar objects, *e.g.*, human). (3) constructing meaningless which lack semantics and appear unnatural for human observers (*i.e.*, noisy or mosaic-like texture) [26], [198], [245]; and (4) a unified evaluation environment is missing, which makes it difficult to make fair comparisons between different attacks.

To address these issues, we present Universal Physical Camouflage Attack (UPC), which constructs a universal camouflage pattern to hide objects from being detected or to wrongly detect objects as the target label. Unlike former works which generate instance-level perturbations, UPC constructs a universal pattern to attack all instances that belong to the same category (*e.g.*, person, cars) via jointly attacking the region proposal network, the classifier and the regressor. To efficiently handle the deformations of complex objects in the physical world, we propose to model their deformable characteristics as well as external physical environments in UPC. We additionally

impose optimization constraint to make generated patterns look natural to human observers. As shown in FIgure 2.7, these camouflage patterns are visually similar to natural images and thus can be regarded as texture patterns on object surfaces such as car paintings/human accessories. Moreover, to fairly evaluate the effectiveness of different physical-world attacks, we present the first standardized virtual database, ***AttackScenes***, which simulates the real 3D world in a controllable and reproducible environment. Our empirical results demonstrate the superiority of our proposed UPC compared with existing physical adversarial attackers not only in virtual environments, but also in real-world environments. We refer interested readers to [86] for details.

2.6 Summary

In this chapter, we investigate the problem of generating adversarial examples, and extend it from image classification to semantic segmentation and object detection. We propose DAG algorithm for this purpose. The basic idea is to first define a dense set of targets as well as a different set of desired labels, and then optimize a loss function in order to produce incorrect recognition results on all the targets simultaneously. Extensive experimental results verify that DAG is able to fool deep networks in the challenging recognition scenarios like object detection and semantic segmentation, and the generated perturbation are visually imperceptible.

An intriguing property of the perturbation generated by DAG lies in the transferability. The perturbation can be transferred across different training sets, different network architectures and even different tasks. Combining heterogeneous perturbations often leads to more effective adversarial perturbations for black-box attacks. The transferability also suggests that deep networks, though started with different initialization and trained in different ways, share some basic principles such as local linearity, which make them sensitive to a similar source of perturbations. This reveals an interesting topic for future research.

Chapter 3

Improving Transferability of Adversarial Examples with Input Diversity

Though deep networks have achieved the state-of-the-art performance on various visual tasks, they are vulnerable to adversarial examples, which are crafted by adding small perturbations to clean images. However, most of the existing adversarial attacks only achieve relatively low success rates under the challenging black-box setting, where the attackers have no knowledge of the model structure and parameters. In this chapter, we propose to improve the transferability of adversarial examples by creating diverse input patterns. Instead of only using the original images to generate adversarial examples, our method applies random transformations to the input images at each iteration. Extensive experiments on ImageNet show that the proposed attack method can generate adversarial examples that transfer much better to different networks than existing baselines. By evaluating against top defense solutions and official baselines from NIPS 2017 adversarial competition, our method achieves an average success rate of 73.0%, which outperforms the top-1 attack submission in the NIPS competition by a large margin of 6.6%. We hope that our proposed attack strategy can serve as a strong benchmark baseline for evaluating the robustness of networks to adversaries and the effectiveness of different defense methods in the future.

3.1 Introduction

Recent success of deep networks leads to great improvements on a wide range of visual tasks, including image classification [72], [99], [183], object detection [60], [61], [168], [246], and semantic segmentation [24], [122]. However, deep networks are also sensitive to small perturbations to the input images, *i.e.*, human-imperceptible additive perturbations can result in failure predictions of deep networks. These intentionally crafted images are known as adversarial examples [23], [29], [32], [35], [64], [196], [214], [223], [227]. Learning how to generate adversarial examples can not only help us investigate the robustness of different models [2], [191], but also shed lights on understanding the insufficiency of current training algorithms [64], [100], [200].

Several methods [64], [100], [196] have been proposed recently to find adversarial examples. In general, these attacks can be categorized into two types according to the number of steps of gradient computation, *i.e.*, single-step attacks [64] and iterative attacks [100], [196]. On the one hand, iterative attacks can achieve higher success rates than single-step attacks in the white-box setting, where the attackers have a perfect knowledge of the network structure and weights. But on the other hand, if these adversarial examples are tested on a different network (either in terms of network structure, weights or both), *i.e.*, the black-box setting, single-step attacks perform better than iterative attacks. This trade-off is due to the fact that iterative attacks tend to overfit the specific network parameters (*i.e.*, have high white-box success rates) and thus making generated adversarial examples rarely transfer to other networks (*i.e.*, have low black-box success rates), while single-step attacks usually underfit to the network parameters (*i.e.*, have low white-box success rates) thus producing adversarial examples with slightly better transferability. Observing the phenomenon, one interesting question is whether we can generate adversarial examples with high success rates under both white-box and black-box settings.

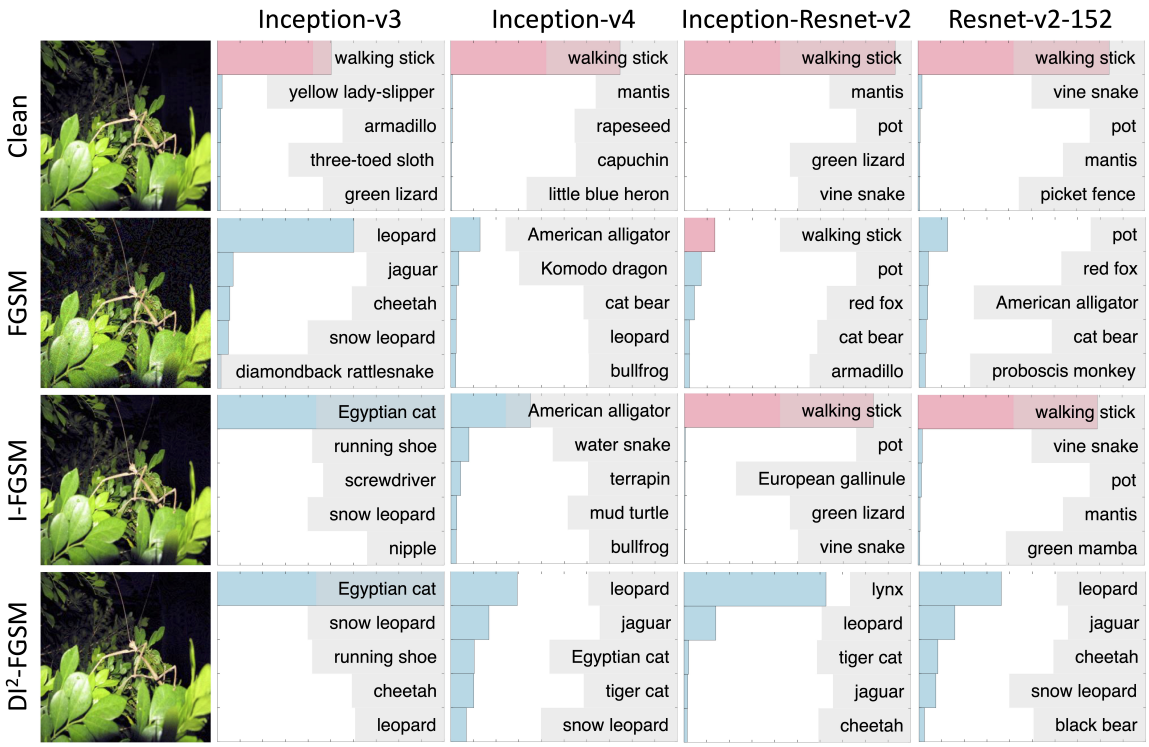


Figure 3.1. The comparison of success rates using three different attacks. The ground-truth "walking stick" is marked as pink in the top-5 confidence distribution plots. The adversarial examples are crafted on Inception-v3 with the maximum perturbation $\epsilon = 15$. From the first row to the the third row, we plot the top-5 confidence distributions of clean images, FGSM and I-FGSM, respectively. The fourth row shows the result of the proposed Diverse Inputs Iterative Fast Gradient Sign Method (DI^2-FGSM), which attacks the white-box model and all black-box models successfully.

In this chapter, we propose to improve the transferability of adversarial examples by creating diverse input patterns. Our work is inspired by the data augmentation strategy [72], [99], [183], which has been proven effective to prevent networks from overfitting by applying a set of label-preserving transformations (*e.g.*, resizing, cropping and rotating) to augment training images. Meanwhile, [69], [226] showed that image transformations can defend against adversarial examples under certain situations, which indicates adversarial examples (generated by existing methods) cannot generalize well under different transformations. These transformed adversarial examples are known as hard examples [180], [182] for attackers, which can then serve as good samples to help attackers to generate more transferable adversarial examples.

We incorporate the proposed input diversity strategy with existing iterative attacks, *e.g.*, I-FGSM [100] and MI-FGSM [46]. At each iteration, unlike the traditional methods which maximize the loss function directly w.r.t. the original inputs, we apply random and differentiable transformations (*e.g.*, random resizing, random padding) to the input images with probability p and maximize the loss function w.r.t. these transformed inputs. Note that these randomized operations were previously used to defend against adversarial examples [226], while here we incorporate them into the attack process to create hard and diverse input patterns. Figure 3.1 shows an adversarial example generated by our method and compares the success rates to other attack methods under both white-box and black-box settings.

We test the proposed input diversity on several networks under both white-box and black-box settings, and single-model and multi-model settings. Compared with traditional iterative attacks, the results on ImageNet (see Sec. 3.4.2) show that our method gets significantly higher success rates for black-box models and maintains similar success rates for white-box models. By evaluating against the top defense solutions and official baselines from NIPS 2017 adversarial competition [102], our method achieves an average success rate of 73.0%, which outperforms the top-1 attack submission in NIPS competition by a large margin of 6.6%. We hope our method can serve as a strong benchmark for evaluating the robustness of networks to adversaries and the effectiveness of different defense methods in the future.

3.2 Related Work

Generating Adversarial Examples Traditional machine learning algorithms are known to be vulnerable to adversarial examples [13], [41], [87]. Recently, Szegedy *et al.* [196] pointed out that deep networks are also fragile to adversarial examples, and proposed a box-constrained L-BFGS method to effectively find adversarial examples. Due to the expensive computations in [196], Goodfellow *et al.* [64] then proposed the

fast gradient sign method to generate adversarial examples efficiently by performing a single gradient step. This method was further extended by Kurakin *et al.* [100] to an iterative version, and showed that the generated adversarial examples can exist in the physical world. Dong *et al.* [46] proposed a broad class of momentum-based iterative algorithms to boost the transferability of adversarial examples. The transferability can also be improved by attacking an ensemble of networks simultaneously [120], smoothing perturbation [248], exploiting intermediate feature maps [88], [91], [106], utilizing skip connections [216], penalizing interactions during the attacking process [211], and smoothing gradient [47], respectively. Recent works [8], [130], [146], [157], [158], [162], [187], [222] also suggest to leverage generative models for creating transferable adversarial examples. Besides transfer-based attacks, query-based [1], [12], [14], [25], [27], [36], [67], [68], [109], [235] attacks are also very popular in the black-box attack settings (if the feedback, *e.g.*, prediction scores, is available from the targeted model).

Our proposed input diversity is also related to EOT [5]. These two works differ in several aspects: (1) we mainly focus on the challenging black-box setting while [5] focuses on the white-box setting; (2) our work aims at alleviating overfitting in adversarial attacks, while [5] aims at making adversarial examples robust to transformations, without any discussion of overfitting; and (3) we do not apply expectation step in each attack iteration, while "expectation" is the core idea in [5].

Defending Against Adversarial Examples There are also many efforts on defending against adversarial attacks. [64], [100] proposed to inject adversarial examples into the training data to increase the network robustness. Tramèr *et al.* [200] pointed out that such adversarially trained models still remain vulnerable to adversarial examples, and proposed ensemble adversarial training, which augments training data with perturbations transferred from other models, in order to improve the network robustness further. [69], [124], [226] applied image transformations to inputs at inference time to mitigate adversarial effects. Dhillon *et al.* [43] pruned a random subset of activation

according to their magnitude to enhance network robustness. Prakash *et al.* [159] proposed a framework which combines pixel deflection with soft wavelet denoising to defend against adversarial examples. [132], [174], [186] leveraged generative models to purify adversarial images by moving them back towards the distribution of clean images. Besides directly increase model robustness against adversarial examples, many works [54], [62], [83], [93], [107], [126], [127], [134], [136], [234] suggested we can train a classifier on detecting adversarial examples.

3.3 Approach

Let X denote an image and y^{true} denote the corresponding ground-truth label, θ denote the network parameters, and $L(X, y^{\text{true}}; \theta)$ to denote the loss. To generate the adversarial example, the goal is to maximize the loss $L(X + r, y^{\text{true}}; \theta)$, under the constraint that the generated adversarial example $X^{\text{adv}} = X + r$ should look visually similar to the original image X and the corresponding predicted label $y^{\text{adv}} \neq y^{\text{true}}$. In this chapter, we use l_∞-norm to measure the perceptibility of adversarial perturbations, *i.e.*, $||r||_\infty \leq \epsilon$. The loss function is defined as

$$L(X, y^{true}; \theta) = -\mathbb{1}_{y^{true}} \cdot \log\left(\text{softmax}(l(X; \theta))\right), \tag{3.1}$$

where $\mathbb{1}_{y^{true}}$ is the one-hot encoding of the ground-truth y^{true} and $l(X; \theta)$ is the logits output. Note that all the baseline attacks have been implemented in the cleverhans library [154], which can be used directly for our experiments.

3.3.1 Family of Fast Gradient Sign Methods

In this section, we give an overview of the family of fast gradient sign methods.

Fast Gradient Sign Method (FGSM). FGSM [64] is the first member in this attack family, which finds the adversarial perturbations in the direction of the loss

gradient $\nabla_X L(X, y^{\text{true}}; \theta)$. The update equation is

$$X^{\text{adv}} = X + \epsilon \cdot \text{sign}(\nabla_X L(X, y^{\text{true}}; \theta)). \tag{3.2}$$

Iterative Fast Gradient Sign Method (I-FGSM). Kurakin *et al.* [100] extended FGSM to an iterative version, which can be written as

$$\begin{aligned} X_0^{\text{adv}} &= X \\ X_{n+1}^{\text{adv}} &= \text{Clip}_X^{\epsilon}\left\{X_n^{adv} + \alpha \cdot \text{sign}(\nabla_X L(X_n^{adv}, y^{\text{true}}; \theta))\right\}, \end{aligned} \tag{3.3}$$

where Clip_X^{ϵ} indicates the resulting image are clipped within the ϵ-ball of the original image X, n is the iteration number and α is the step size.

Momentum Iterative Fast Gradient Sign Method (MI-FGSM). MI-FGSM [46] integrates the momentum term into the attack process to stabilize update directions and escape from poor local maxima. The updating procedure is similar to I-FGSM, with the replacement of Eqn. (3.3) by:

$$\begin{aligned} g_{n+1} &= \mu \cdot g_n + \frac{\nabla_X L(X_n^{adv}, y^{\text{true}}; \theta)}{||\nabla_X L(X_n^{adv}, y^{\text{true}}; \theta)||_1} \\ X_{n+1}^{\text{adv}} &= \text{Clip}_X^{\epsilon}\left\{X_n^{adv} + \alpha \cdot \text{sign}(g_{n+1})\right\}, \end{aligned} \tag{3.4}$$

where μ is the decay factor of the momentum term and g_n is the accumulated gradient at iteration n.

3.3.2 Motivation

Let $\hat{\theta}$ denote the unknown network parameters. In general, a strong adversarial example should have high success rates on both white-box models, *i.e.*, $L(X^{adv}, y^{\text{true}}; \theta) > L(X, y^{\text{true}}; \theta)$, and black-box models, *i.e.*, $L(X^{adv}, y^{\text{true}}; \hat{\theta}) > L(X, y^{\text{true}}; \hat{\theta})$. On the one hand, the traditional single-step attacks, *e.g.*, FGSM, tend to underfit to the specific network parameters θ due to inaccurate linear appropriation of the loss $L(X, y^{\text{true}}; \theta)$, thus cannot reach high success rates on white-box models. On the other hand, the

traditional iterative attacks, *e.g.*, I-FGSM, greedily perturb the images in the direction of the sign of the loss gradient $\nabla_X L(X, y^{\text{true}}; \theta)$ at each iteration, and thus easily fall into the poor local maxima and overfit to the specific network parameters θ. These overfitted adversarial examples rarely transfer to black-box models. In order to generate adversarial examples with strong transferability, we need to find a better way to optimize the loss $L(X, y^{\text{true}}; \theta)$ to alleviate this overfitting phenomenon.

Data augmentation [72], [99], [183] is shown as an effective way to prevent networks from overfitting during the training process. Meanwhile, [69], [226] showed that adversarial examples are no longer malicious if simple image transformations are applied, which indicates these transformed adversarial images can serve as good samples for better optimization. Those facts inspire us to apply random and differentiable transformations to the inputs for improving the transferability of adversarial examples.

3.3.3 Diverse Input Patterns

DI²-FGSM. We first present the Diverse Inputs Iterative Fast Gradient Sign Method (DI²-FGSM), which applies image transformations $T(\cdot)$ to inputs with the probability p at each iteration of I-FGSM [100] to alleviate the overfitting phenomenon.

Specifically, we consider random resizing, which resizes the input images to a random size, and random padding, which pads zeros around the input images in a random manner [226], as the instantiation of the image transformations $T(\cdot)$[1]. The transformation probability p controls the trade-off between success rates on white-box models and success rates on black-box models, which can be observed from Figure 3.4. If $p = 0$, DI²-FGSM degrades to I-FGSM with poor transferability. If $p = 1$, *i.e.*, only transformed inputs are used for the attack, the generated adversarial examples tend to have much higher success rates on black-box models but lower success rates on white-box models, since the original inputs are not seen by the attackers.

[1]We also experimented with other image transformations, *e.g.*, rotation, to create diverse input patterns, and found random resizing & padding yields the most transferable adversarial examples.

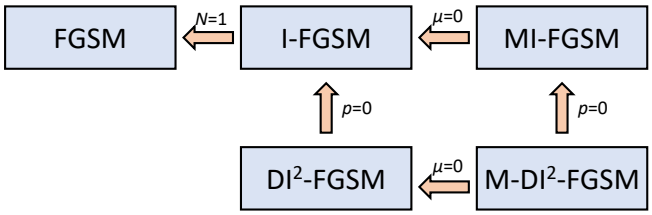


Figure 3.2. Relationships between different attacks. By setting values of the transformation probability p, the decay factor μ and the total iteration number N, we can relate these different attacks in the family of Fast Gradient Sign Methods.

In summary, the updating procedure of DI^2-FGSM is similar to I-FGSM, but with the replacement of Eqn. (3.3) by

$$X_{n+1}^{\text{adv}} = \text{Clip}_X^{\epsilon}\{X_n^{adv} + \alpha \cdot \text{sign}\left(\nabla_X L(T(X_n^{adv}; p), y^{\text{true}}; \theta)\right)\}, \tag{3.5}$$

where the stochastic transformation function $T(X_n^{adv}; p)$ is

$$T(X_n^{adv}; p) = \begin{cases} T(X_n^{adv}) & \text{with probability } p \\ X_n^{adv} & \text{with probability } 1-p \end{cases}. \tag{3.6}$$

M-DI^2-FGSM. We can combine momentum and diverse inputs together to form a much stronger attack, *i.e.*, Momentum Diverse Inputs Iterative Fast Gradient Sign Method (M-DI^2-FGSM). The overall updating procedure of M-DI^2-FGSM is similar to MI-FGSM, but with the only replacement of Eqn. (3.4) by

$$g_{n+1} = \mu \cdot g_n + \frac{\nabla_X L(T(X_n^{adv}; p), y^{\text{true}}; \theta)}{||\nabla_X L(T(X_n^{adv}; p), y^{\text{true}}; \theta)||_1}. \tag{3.7}$$

3.3.4 Relationships between Different Attacks

The attacks mentioned above all belong to the family of Fast Gradient Sign Methods, and are related via different parameter settings as shown in Figure 3.2. To summarize,

- If the transformation probability $p = 0$, M-DI^2-FGSM degrades to MI-FGSM, and DI^2-FGSM degrades to I-FGSM.
- If the decay factor $\mu = 0$, M-DI^2-FGSM degrades to DI^2-FGSM, and MI-FGSM degrades to I-FGSM.
- If the total iteration number $N = 1$, I-FGSM degrades to FGSM.

3.3.5 Attacking an Ensemble of Networks

Liu *et al.* [120] suggested that attacking an ensemble of multiple networks simultaneously can generate much stronger adversarial examples. The motivation is that if an adversarial image remains adversarial for multiple networks, then it is more likely to transfer to other networks as well. Therefore, we can use this strategy to improve the transferability even further.

We follow the ensemble strategy proposed in [46], which fuse the logit activations together to attack multiple networks simultaneously. Specifically, to attack an ensemble of K models, the logits are fused by:

$$l(X;\theta_1,...,\theta_K) = \sum_{k=1}^{K} w_k l_k(X;\theta_k) \tag{3.8}$$

where $l_k(X;\theta_k)$ is the logits output of the k-th model with the parameters θ_k, w_k is the ensemble weight with $w_k \geq 0$ and $\sum_{k=1}^{K} w_k = 1$.

3.4 Experiments

3.4.1 Experiment Setup

Dataset. It is less meaningful to attack the images that are already classified wrongly. Therefore, we randomly choose 5000 images from the ImageNet validation set that are classified correctly by all the networks which we test on, to form our test dataset. All these images are central cropped and resized to $299 \times 299 \times 3$ beforehand.

Networks. We consider four normally trained networks, *i.e.*, Inception-v3 (Inc-v3) [195], Inception-v4 (Inc-v4) [193], Resnet-v2-152 (Res-152) [72] and Inception-Resnet-v2 (IncRes-v2) [193], and three adversarially trained networks [200], *i.e.*, ens3-adv-Inception-v3 (Inc-v3$_{ens3}$), ens4-adv-Inception-v3 (Inc-v3$_{ens4}$) and ens-adv-Inception-ResNet-v2 (IncRes-v2$_{ens}$). All networks are publicly available[2,3].

[2] https://github.com/tensorflow/models/tree/master/research/slim
[3] https://github.com/tensorflow/models/tree/master/research/adv_imagenet_models

Model	Attack	Inc-v3	Inc-v4	IncRes-v2	Res-152	Inc-v3$_{ens3}$	Inc-v3$_{ens4}$	IncRes-v2$_{ens}$
Inc-v3	FGSM	64.6%	23.5%	21.7%	21.7%	8.0%	7.5%	3.6%
	I-FGSM	**99.9%**	14.8%	11.6%	8.9%	3.3%	2.9%	1.5%
	DI^2-FGSM (**Ours**)	**99.9%**	35.5%	27.8%	21.4%	5.5%	5.2%	2.8%
	MI-FGSM	**99.9%**	36.6%	34.5%	27.5%	8.9%	8.4%	4.7%
	M-DI^2-FGSM (**Ours**)	**99.9%**	**63.9%**	**59.4%**	**47.9%**	**14.3%**	**14.0%**	**7.0%**
Inc-v4	FGSM	26.4%	49.6%	19.7%	20.4%	8.4%	7.7%	4.1%
	I-FGSM	22.0%	**99.9%**	13.2%	10.9%	3.2%	3.0%	1.7%
	DI^2-FGSM (**Ours**)	43.3%	99.7%	28.9%	23.1%	5.9%	5.5%	3.2%
	MI-FGSM	51.1%	**99.9%**	39.4%	33.7%	11.2%	10.7%	5.3%
	M-DI^2-FGSM (**Ours**)	**72.4%**	99.5%	**62.2%**	**52.1%**	**17.6%**	**15.6%**	**8.8%**
IncRes-v2	FGSM	24.3%	19.3%	39.6%	19.4%	8.5%	7.3%	4.8%
	I-FGSM	22.2%	17.7%	97.9%	12.6%	4.6%	3.7%	2.5%
	DI^2-FGSM (**Ours**)	46.5%	40.5%	95.8%	28.6%	8.2%	6.6%	4.8%
	MI-FGSM	53.5%	45.9%	**98.4%**	37.8%	15.3%	13.0%	8.8%
	M-DI^2-FGSM (**Ours**)	**71.2%**	**67.4%**	96.1%	**57.4%**	**25.1%**	**20.7%**	**14.9%**
Res-152	FGSM	34.4%	28.5%	27.1%	75.2%	12.4%	11.0%	6.0%
	I-FGSM	20.8%	17.2%	14.9%	99.1%	5.4%	4.6%	2.8%
	DI^2-FGSM (**Ours**)	53.8%	49.0%	44.8%	**99.2%**	13.0%	11.1%	6.9%
	MI-FGSM	50.1%	44.1%	42.2%	99.0%	18.2%	15.2%	9.0%
	M-DI^2-FGSM (**Ours**)	**78.9%**	**76.5%**	**74.8%**	**99.2%**	**35.2%**	**29.4%**	**19.0%**

Table 3.1. The success rates on seven networks where we attack a single network. The diagonal blocks indicate white-box attacks, while the off-diagonal blocks indicate black-box attacks which are much more challenging. Experiment results show our proposed input diversity strategy substantially improves the transferability of adversarial examples.

Implementation details. For the parameters of different attackers, we follow the default settings in [100] with the step size $\alpha = 1$ and the total iteration number $N = \min(\epsilon + 4, 1.25\epsilon)$. We set the maximum perturbation of each pixel to be $\epsilon = 15$, *which is still imperceptible for human observers* [124]. For the momentum term, decay factor μ is set to be 1 as in [46]. For the stochastic transformation function $T(X; p)$, the probability p is set to be 0.5, *i.e.*, attackers put equal attentions on the original inputs and the transformed inputs. For transformation operations $T(\cdot)$, the input X is first randomly resized to a $rnd \times rnd \times 3$ image, with $rnd \in [299, 330)$, and then padded to the size $330 \times 330 \times 3$ in a random manner.

3.4.2 Attacking a Single Network

We first perform adversarial attacks on a single network. We craft adversarial examples only on normally trained networks, and test them on all seven networks. The success rates are shown in Table 3.1, where the diagonal blocks indicate white-box attacks and

Figure 3.3. Visualization of randomly selected clean images and their corresponding adversarial examples. All these adversarial examples are generated on Inception-v3 using our proposed DI^2-FGSM with the maximum perturbation of each pixel $\epsilon = 15$.

off-diagonal blocks indicate black-box attacks. We list the networks that we attack on in rows, and networks that we test on in columns.

From Table 3.1, we can observe that M-DI^2-FGSM outperforms all other baselines by a large margin on all black-box models. Meanwhile, it also maintains high success rates on all white-box models. For example, when generating adversarial examples using IncRes-v2, M-DI^2-FGSM has success rates of 67.4% on Inc-v4 (normally trained black-box model) and 25.1% on Inc-v3$_{ens3}$ (adversarially trained black-box model), while strong baselines like MI-FGSM only obtains the corresponding success rates of 45.9% and 15.3%, respectively. These results demonstrate the effectiveness of utilizing our proposed input diversity and momentum [46] simultaneously for improving the transferability of adversarial examples.

We then compare the success rates of I-FGSM and DI^2-FGSM to see the effectiveness of diverse input patterns solely. We note that DI^2-FGSM significantly improves the success rates of I-FGSM on challenging black-box models (even this model is adversarially trained), and maintains high success rates on white-box models. For example, when generating adversarial examples using Res-152, DI^2-FGSM has success rates of 99.2% on Res-152 (white-box model), 53.8% on Inc-v3 (normally trained

Model	Attack	Inc-v3	Inc-v4	IncRes-v2	Res-152	Inc-v3$_{ens3}$	Inc-v3$_{ens4}$	IncRes-v2$_{ens}$
Inc-v3	C&W	**100.0%**	5.7%	5.3%	5.1%	3.0%	2.5%	1.1%
	D-C&W (**Ours**)	**100.0%**	**16.8%**	**13.0%**	**11.2%**	**5.8%**	**3.9%**	**2.1%**
Inc-v4	C&W	15.1%	**100.0%**	9.2%	7.8%	4.4%	3.5%	1.9%
	D-C&W (**Ours**)	**29.3%**	**100.0%**	**20.1%**	**15.4%**	**7.1%**	**5.3%**	**3.1%**
IncRes-v2	C&W	15.8%	11.2%	99.9%	8.6%	6.3%	3.6%	3.4%
	D-C&W (**Ours**)	**33.9%**	**25.6%**	**100.0%**	**19.4%**	**11.2%**	**7.3%**	**4.0%**
Res-152	C&W	11.4%	6.9%	6.1%	**100.0%**	4.4%	4.1%	2.3%
	D-C&W (**Ours**)	**33.0%**	**27.7%**	**24.4%**	**100.0%**	**13.1%**	**9.3%**	**5.7%**

Table 3.2. The success rates on seven networks where we attack a single network using C&W attack. Experiment results demonstrate that the proposed input diversity strategy can enhance C&W attack for generating more transferable adversarial examples.

black-box model) and 11.1% on Inc-v3$_{ens4}$ (adversarially trained black-box model), while I-FGSM only obtains the corresponding success rates of 99.1%, 20.8% and 4.6%, respectively. Compared to FGSM, DI^2-FGSM also reaches much higher success rates on normally trained black-box models, and on-par performance on adversarially trained black-box models. Besides, we visualize 5 randomly selected pairs of such generated adversarial images and their clean counterparts in Figure 3.3. These visualizations confirm the generated adversarial perturbations are human imperceptible.

It is worth to mention that the proposed input diversity is a general way for boosting transferability of attackers. To verify this point, we incorporate C&W attack [20] with input diversity. The experiment is conducted on 1000 correctly classified images. For the parameters of C&W, the maximal iteration is 250, the learning rate is 0.01 and the confidence is 10. As Table 3.2 suggests, our method, D-C&W, obtains a significant performance improvement over C&W on black-box models.

3.4.3 Attacking an Ensemble of Networks

Though the results in Table 3.1 show that momentum and input diversity can significantly improve the transferability of adversarial examples, they are still relatively weak at attacking an adversarially trained network under the black-box setting, *e.g.*, the highest black-box success rate on IncRes-v2$_{ens}$ is only 19.0%. Therefore, we follow the strategy in [120] to attack multiple networks simultaneously in order to further

Model	Attack	-Inc-v3	-Inc-v4	-IncRes-v2	-Res-152	-Inc-v3$_{ens3}$	-Inc-v3$_{ens4}$	-IncRes-v2$_{ens}$
Ensemble	I-FGSM	96.6%	**96.9%**	98.7%	96.2%	**97.0%**	**97.3%**	94.3%
	DI^2-FGSM (**Ours**)	88.9%	89.6%	93.2%	87.7%	91.7%	91.7%	93.2%
	MI-FGSM	**96.9%**	**96.9%**	**98.8%**	**96.8%**	96.8%	97.0%	94.6%
	M-DI^2-FGSM (**Ours**)	90.1%	91.1%	94.0%	89.3%	92.8%	92.7%	**94.9%**
Hold-out	I-FGSM	43.7%	36.4%	33.3%	25.4%	12.9%	15.1%	8.8%
	DI^2-FGSM (**Ours**)	69.9%	67.9%	64.1%	51.7%	36.3%	35.0%	30.4%
	MI-FGSM	71.4%	65.9%	64.6%	55.6%	22.8%	26.1%	15.8%
	M-DI^2-FGSM (**Ours**)	**80.7%**	**80.6%**	**80.7%**	**70.9%**	**44.6%**	**44.5%**	**39.4%**

Table 3.3. The success rates of ensemble attacks. Adversarial examples are generated on an ensemble of six networks, and tested on the ensembled network (*white-box setting*) and the hold-out network (*black-box setting*). The sign "-" indicates the hold-out network. We observe that the proposed M-DI^2-FGSM significantly outperforms *all* other attacks on *all* black-box models.

improve transferability. We consider all seven networks here. Adversarial examples are generated on an ensemble of six networks, and tested on the ensembled network and the hold-out network, using I-FGSM, DI^2-FGSM, MI-FGSM and M-DI^2-FGSM, respectively. FGSM is ignored here due to its low success rates on white-box models. All ensembled models are assigned with equal weight, *i.e.*, $w_k = 1/6$.

The results are summarized in Table 3.3, where the top row shows the success rates on the ensembled network (white-box setting), and the bottom row shows the success rates on the hold-out network (black-box setting). Under the challenging black-box setting, we observe that M-DI^2-FGSM always generates adversarial examples with better transferability than other methods on all networks. For example, by keeping Inc-v3$_{ens3}$ as the hold-out model, M-DI^2-FGSM can fool Inc-v3$_{ens3}$ with an success rate of 44.6%, while I-FGSM, DI^2-FGSM and MI-FGSM only have success rates of 12.9%, 36.3% and 22.8%, respectively. Besides, compared with MI-FGSM, we observe that using diverse input patterns alone, *i.e.*, DI^2-FGSM, can reach a much higher success rate if the hold-out model is an adversarially trained network, and a comparable success rate if the hold-out model is a normally trained network.

Under the white-box setting, we observe DI^2-FGSM and M-DI^2-FGSM reach slightly lower (but still very high) success rates on ensemble models compared with I-FGSM and MI-FGSM. This is due to the fact attacking multiple networks simultaneously

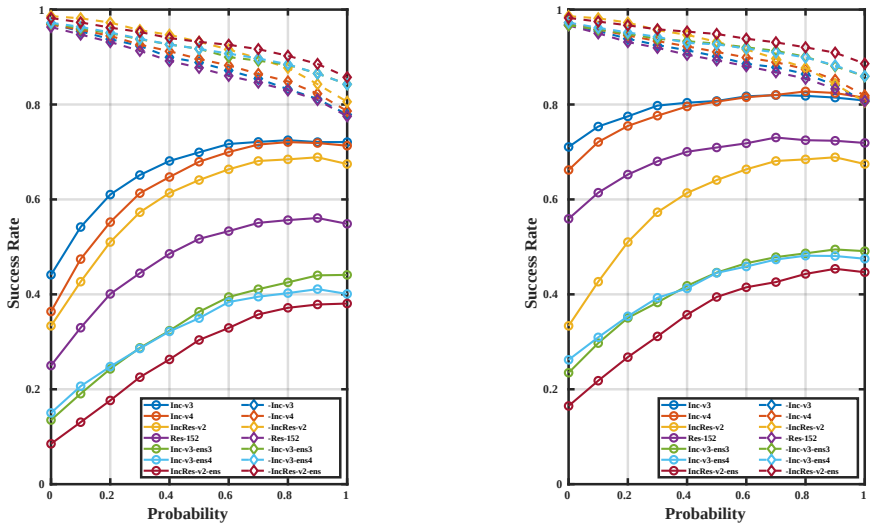


Figure 3.4. The success rates of DI^2-FGSM (a) and M-DI^2-FGSM (b) when varying the transformation probability p. "Ensemble" (white-box setting) is with dashed lines and "Hold-out" (black-box setting) is with solid lines.

is much harder than attacking a single model. However, the white-box success rates can be improved if we assign the transformation probability p with a smaller value, increase the number of total iteration N or use a smaller step size α (see Sec. 3.4.4).

3.4.4 Ablation Studies

In this section, we conduct a series of ablation experiments to study the impact of different parameters. We only consider attacking an ensemble of networks here, since it is much stronger than attacking a single network and can provide a more accurate evaluation of the network robustness. The max perturbation of each pixel ϵ is set to 15 for all experiments.

Transformation probability p. We first study the effects of the transformation probability p on the success rates under both white-box and black-box settings. We set the step size $\alpha = 1$ and the total iteration number $N = \min(\epsilon + 4, 1.25\epsilon)$. The transformation probability p varies from 0 to 1. Recall the relationships shown in Figure 3.2, M-DI^2-FGSM (or DI^2-FGSM) degrades to MI-FGSM (or I-FGSM) if $p = 0$.

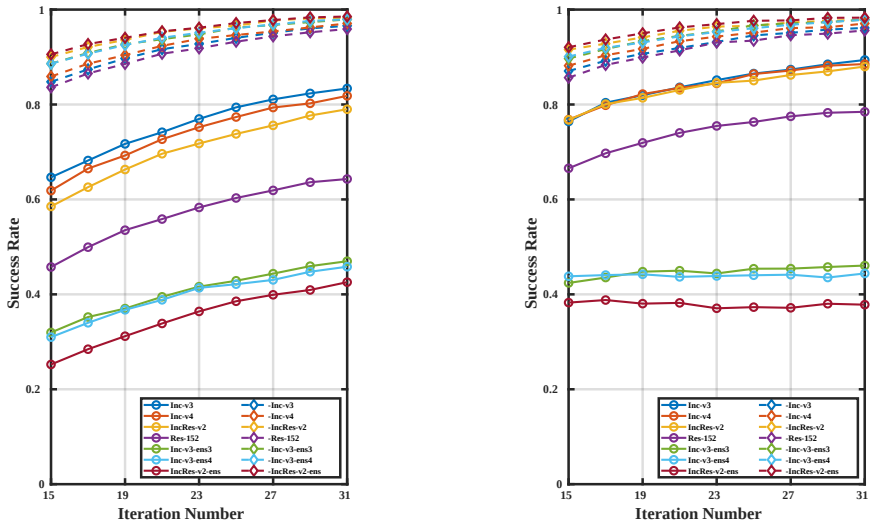


Figure 3.5. The success rates of DI^2-FGSM (a) and M-DI^2-FGSM (b) when varying the total iteration number N. "Ensemble" (white-box setting) is with dashed lines and "Hold-out" (black-box setting) is with solid lines.

We show the success rates on different networks in Figure 3.4. We observe that both DI^2-FGSM and M-DI^2-FGSM achieve a higher black-box success rates but lower white-box success rates as p increase. Moreover, for all attacks, if p is small, *i.e.*, only a small amount of transformed inputs are utilized, black-box success rates can increase significantly, while white-box success rates only drop a little. This phenomenon reveals the importance of adding transformed inputs into the attack process.

Figure 3.4 also provide useful suggestions for constructing effective adversarial attacks in practice. For example, if you know the black-box model is a new network that totally different from any existing networks, you can set $p = 1$ to reach the maximum transferability. If the black-box model is a mixture of new networks and existing networks, then a moderate value of p can be selected to maximize the black-box success rates while maintaining a reasonable white-box success rate (*e.g.*, $\geq 90\%$).

Total iteration number N. We then ablate the effect of the total iteration number N on the success rates under both white-box and black-box settings. Specifically, we set the transformation probability $p = 0.5$, the step size $\alpha = 1$, and vary the total iteration number N from 15 to 31.

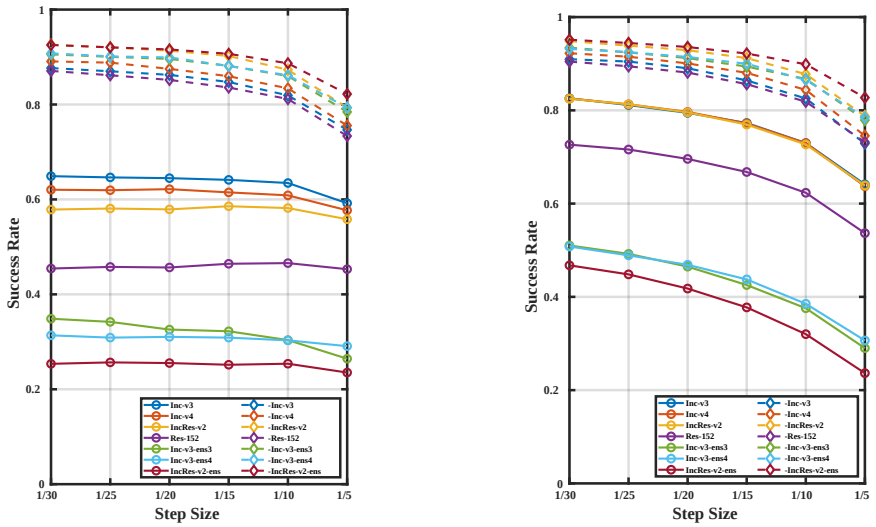


Figure 3.6. The success rates of DI^2-FGSM (a) and M-DI^2-FGSM (b) when varying the step size α. "Ensemble" (white-box setting) is with dashed lines and "Hold-out" (black-box setting) is with solid lines.

The results are showed in Figure 3.5. For DI^2-FGSM, we observe that both the black-box success rate and the white-box success rate get increased if a larger total iteration number N is performed by attackers. This phenomenon can also be observed for M-DI^2-FGSM except for the black-box success rates on adversarially trained models, *i.e.*, performing more iterations cannot boost transferability further on adversarially trained models. Moreover, we observe that the gap of success rates between M-DI^2-FGSM and DI^2-FGSM becomes smaller as N increases.

Step size α. Lastly, we study the effects of the step size α on the success rates under both white-box and black-box settings. We set the transformation probability $p = 0.5$. In order to reach the maximum perturbation ϵ even for a small step size α, we set the total iteration number be proportional to the step size, *i.e.*, $N = \epsilon/\alpha$.

From Figure 3.6, we observe that the white-box success rates of both DI^2-FGSM and M-DI^2-FGSM can be boosted if a smaller step size is provided. Under the black-box setting, the success rates of DI^2-FGSM is insensitive to the step size, while the success rates of M-DI^2-FGSM can still be improved with smaller step size.

Attack	TsAIL	iyswim	Anil Thomas	Inc-v3$_{adv}$	IncRes-v2$_{ens}$	Inc-v3	**Average**
I-FGSM	14.0%	35.6%	30.9%	**98.2%**	**96.4%**	**99.0%**	62.4%
DI^2-FGSM (**Ours**)	**22.7%**	58.4%	48.0%	91.5%	90.7%	97.3%	68.1%
MI-FGSM	14.9%	45.7%	46.6%	97.3%	95.4%	98.7%	66.4%
MI-FGSM*	13.6%	43.2%	43.9%	94.4%	93.0%	97.3%	64.2%
M-DI^2-FGSM (**Ours**)	20.0%	**69.8%**	**64.4%**	93.3%	92.4%	97.9%	**73.0%**

Table 3.4. The success rates on top defense solutions and official baselines from NIPS 2017 adversarial competition [102]. Our proposed M-DI^2-FGSM reaches an average success rate of 73.0%, outperforming the *best* attack solution in the NIPS competition by a large margin of 6.6%. * refers to the official results in the competition.

3.4.5 NIPS 2017 Adversarial Competition

In order to verify the effectiveness of our proposed attack methods in practice, we here reproduce the top defense entries and official baselines from NIPS 2017 adversarial competition [102] for testing transferability. Due to the resource limitation, we only consider the top-3 defense entries, *i.e.*, *TsAIL* [115], *iyswim* [226] and *Anil Thomas*[4], as well 3 official baselines, *i.e.*, Inc-v3$_{adv}$, IncRes-v2$_{ens}$ and Inc-v3. Specifically, we note that the No.1 solution and the No.3 solution apply significantly different image transformations (compared to random resizing & padding used in our attack method) for defending against adversarial examples. For example, the No.1 solution, *TsAIL*, applies an image denoising network for removing adversarial perturbations, and the No.3 solution, *Anil Thomas*, includes a series of image transformations, *e.g.*, JPEG compression, rotation, shifting and zooming, in the defense pipeline. The test dataset contains 5000 images which are all of the size $299 \times 299 \times 3$, and their corresponding labels are the same as the ImageNet labels.

Generating adversarial examples. When generating adversarial examples, we follow the standard pipeline in [102]: (1) split the dataset equally into 50 batches; (2) for each batch, the maximum perturbation ϵ is randomly chosen from the set $\{\frac{4}{255}, \frac{8}{255}, \frac{12}{255}, \frac{16}{255}\}$; and (3) generate adversarial examples for each batch under the corresponding ϵ constraint.

[4] https://github.com/anlthms/nips-2017/tree/master/mmd

Attacker settings. We follow [46] by attacking an ensemble of eight models, *i.e.*, Inc-v3, Inc-v4, IncRes-v2, Res-152, Inc-v3$_{ens3}$, Inc-v3$_{ens4}$, IncRes-v2$_{ens}$ and Inc-v3$_{adv}$. The ensemble weights are set as 1/7.25 equally for the first seven models and 0.25/7.25 for Inc-v3$_{adv}$. The total iteration number N is 10 and the decay factor μ is 1. Note that such configured MI-FGSM won the 1-st place in the NIPS 2017 adversarial attack competition [102]. For DI^2-FGSM and M-DI^2-FGSM, we choose $p = 0.4$ to balance the tradeoff of success rates under the white-box setting and the black-box setting.

Results. The results are summarized in Table 3.4. We also list the official competition results of MI-FGSM (*i.e.*, MI-FGSM*) as a reference to validate our implementation. The performance difference between MI-FGSM and MI-FGSM* is due to the randomness of the max perturbation magnitude introduced in the attack process. Compared with MI-FGSM, DI^2-FGSM has higher success rates on top defense solutions while slightly lower success rates on baseline models, which results in these two attacks having similar average success rates. By integrating both diverse inputs and momentum term, this enhanced attack, M-DI^2-FGSM, reports the best result, *i.e.*, achieving an average success rate of 73.0%. We note this result outperforms the the top-1 attack submission, MI-FGSM, in the NIPS competition by a large margin (*i.e.*, 73.0% *vs.* 66.4%). We believe this superior transferability can also be observed on other defense submissions which we do not evaluate on.

3.4.6 Extensions

We additionally provide a brief discussion on two extensions of this work, which further the transferability of adversarial examples.

Learning Transferable Adversarial Examples via Ghost Networks As shown in Section 3.4.3 and Section 3.4.5, the ensemble-based attacks obtain much better performance than the non-ensemble ones. Nonetheless, the ensemble-based attacks

suffer from expensive computational overhead, making it difficult to generate transferable adversarial examples efficiently, *e.g.*, we usually need to train a large amount of models from scratch for a good ensemble. To this end, we propose an alternative, Ghost Networks, which generates a vast number of *virtual* models built on a base network (a network trained from scratch), for highly efficient ensemble. Note that the word "virtual" means these networks are not stored or trained (therefore we term them as ghost networks). The ghost networks here are generated by imposing erosion on certain intermediate structures of the base network on-the-fly. Meanwhile, we propose Longitudinal Ensemble, to conduct an implicit ensemble of ghost networks during the attack to further reduce computational complexity. Consequently, adversarial examples can be easily generated without sacrificing computational efficiency. We refer interested readers to [113] for details.

Regional Homogeneity: Towards Learning Transferable Universal Adversarial Perturbations Against Defenses Also as shown in Section 3.4.5, current attacks are generally hard to transfer to adversarially trained models. But interestingly, by white-box attacking these adversarially trained models, we observe the generated perturbations exhibit the property of regional homogeneity. This phenomenon suggests constructing regionally homogeneous perturbation (RHP) could be an effective way to create strong adversarial examples. To this end, we propose an effective transforming paradigm and a customized gradient transformer module to transform existing perturbations into regionally homogeneous ones. Without explicitly forcing the perturbations to be universal, we observe that a well-trained gradient transformer module tends to output input-independent gradients (hence universal) benefiting from the under-fitting phenomenon. Extensive results demonstrate that our method substantially boosts the transferability of adversarial examples across different models (especially to strong defenses), and across different vision tasks (*i.e.*, semantic segmentation and object detection). We refer interested readers to [112] for details.

3.5 Summary

In this chapter, we propose to improve the transferability of adversarial examples with input diversity. Specifically, our method applies random transformations to the input images at each iteration in the attack process. Compared with traditional iterative attacks, the results on ImageNet show our proposed attack method gets significantly higher success rates for black-box models, and maintains similar success rates for white-box models. We improve the transferability further by integrating momentum term and attacking multiple networks simultaneously. By evaluating against the top defense submissions and official baselines from NIPS 2017 adversarial competition [102], we show that this enhanced attack reaches an average success rate of 73.0%, which outperforms the best attack solution in NIPS competition by a large margin of 6.6%. We hope our proposed attack strategy can serve as a benchmark for evaluating the robustness of networks to adversaries and the effectiveness of different defense methods in the future.

Part II

Towards Deep Networks Robust to Adversarial Attacks

Chapter 4

Feature Denoising for Improving Adversarial Robustness

Adversarial attacks to image classification systems present challenges to convolutional networks and opportunities for understanding them. This study suggests that adversarial perturbations on images lead to noise in the features constructed by these networks. Motivated by this observation, we develop new network architectures that increase adversarial robustness by performing feature denoising. Specifically, our networks contain blocks that denoise the features using non-local means or other filters; the entire networks are trained end-to-end. When combined with adversarial training, our feature denoising networks substantially improve the state-of-the-art in adversarial robustness in both white-box and black-box attack settings. On ImageNet, under 10-iteration PGD white-box attacks where prior art has 27.9% accuracy, our method achieves 55.7%; even under extreme 2000-iteration PGD white-box attacks, our method secures 42.6% accuracy. Our method was ranked first in Competition on Adversarial Attacks and Defenses (CAAD) 2018—it achieved 50.6% classification accuracy on a secret, ImageNet-like test dataset against 48 unknown attackers, surpassing the runner-up approach by ~10%. Code is available at https://github.com/facebookresearch/ImageNet-Adversarial-Training.

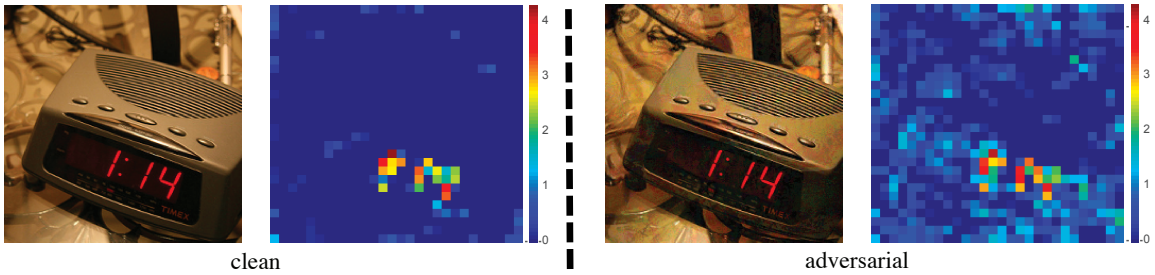


Figure 4.1. Feature map in the res_3 block of an ImageNet-trained ResNet-50 applied on a clean image (left) and on its adversarially perturbed counterpart (right). The adversarial perturbation was produced using PGD [128] with maximum perturbation $\epsilon = 16$ (out of 256). In this example, the adversarial image is incorrectly recognized as "space heater"; the true label is "digital clock".

4.1 Introduction

Adversarial attacks to image classification systems [196] add small perturbations to images that lead these systems into making incorrect predictions. While the perturbations are often imperceptible or perceived as small "noise" in the image, these attacks are highly effective against even the most successful convolutional network based systems [99], [103]. The success of adversarial attacks leads to security threats in real-world applications of convolutional networks, but equally importantly, it demonstrates that these networks perform computations that are dramatically different from those in human brains.

Figure 4.1 shows a randomly selected feature map (*e.g.*, the feature map in the res_3 block) of an ImageNet-trained ResNet-50 [72] applied on a clean image (shown in the left panel) and on its adversarially perturbed counterpart (shown in the right panel). The figure suggests that adversarial perturbations, while small in the pixel space, lead to very substantial "noise" in the feature maps of the network. Whereas the features for the clean image appear to focus primarily on semantically informative content in the image, the feature maps for the adversarial image are activated across semantically irrelevant regions as well. We provide more examples with the same "noisy" pattern in Figure 4.2.

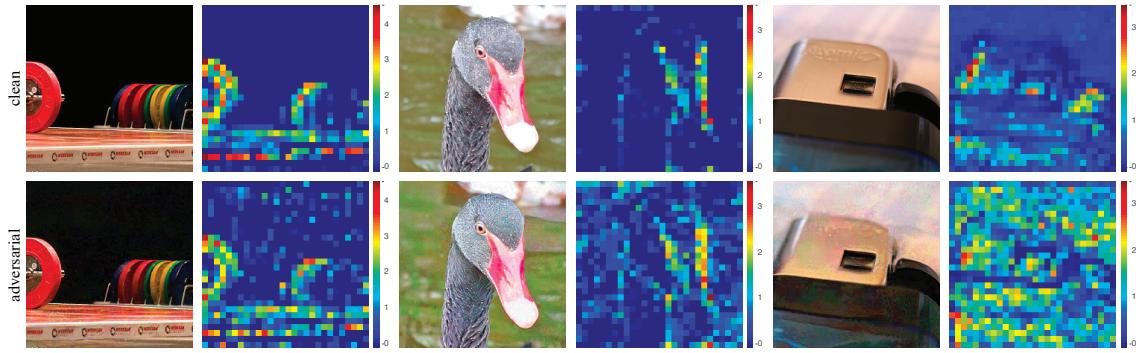


Figure 4.2. More examples similar to Figure 4.1. We show feature maps corresponding to clean images (top) and to their adversarial perturbed versions (bottom). The feature maps for each pair of examples are from the same channel of a res_3 block in the same ResNet-50 trained on clean images. The attacker has a maximum perturbation $\epsilon = 16$ in the pixel domain.

Motivated by this observation, we explore *feature denoising* approaches to improve the robustness of convolutional networks against adversarial attacks. We develop new convolutional network architectures equipped with building blocks designed to denoise feature maps. Our networks are trained end-to-end on adversarially generated samples, allowing them to learn to reduce feature-map perturbations.

Empirically, we find the best performance is achieved by networks using *non-local means* [16] for feature denoising, leading to models that are related to self-attention [204] and non-local networks [210]. Our ablation studies show that using *mean filters*, *median filters*, and *bilateral filters* [199] for feature denoising also improves adversarial robustness, suggesting that feature denoising is a good design principle.

Our models outperform the state-of-the-art in adversarial robustness against highly challenging *white-box* attacks on ImageNet [173]. Under 10-iteration PGD attacks [128], we report 55.7% classification accuracy on ImageNet, largely surpassing the prior art's 27.9% [96] with the same attack protocol. Even when faced with extremely challenging *2000-iteration* PGD attacks that have not been explored in other literature, our model achieves 42.6% accuracy. Our ablation experiments also demonstrate that feature denoising consistently improves adversarial defense results in white-box settings.

Our networks are also highly effective under the *black-box* attack setting. The network based on our method won the defense track in the recent Competition on Adversarial Attacks and Defenses (CAAD) 2018, achieving 50.6% accuracy against 48 unknown attackers, under a strict "*all-or-nothing*" criterion. This is an 10% absolute (20% relative) accuracy increase compared to the CAAD 2018 runner-up model. We also conduct ablation experiments in which we defend against the five strongest attackers from CAAD 2017 [102], demonstrating the potential of feature denoising.

4.2 Related Work

Adversarial training [64], [96], [128] defends against adversarial perturbations by training networks on adversarial images that are generated on-the-fly during training. Adversarial training constitutes the current state-of-the-art in adversarial robustness against white-box attacks; we use it to train our networks. *Adversarial logit paring* (ALP) [96] is a type of adversarial training that encourages the logit predictions of a network for a clean image and its adversarial counterpart to be similar. ALP can be interpreted as "denoising" the logit predictions for the adversarial image, using the logits for the clean image as the "noise-free" reference.

Other approaches to increase adversarial robustness include *pixel denoising*. Liao *et al*. [115] propose to use high-level features to guide the pixel denoiser; in contrast, our denoising is applied directly on features. Guo *et al*. [69] transform the images via non-differentiable image preprocessing, like image quilting [48], total variance minimization [172], and quantization. While these defenses may be effective in black-box settings, they can be circumvented in white-box settings because the attacker can approximate the gradients of their non-differentiable computations [4]. In contrast to [69], our feature denoising models are *differentiable*, but are still able to improve adversarial robustness against very strong white-box attacks.

4.3 Feature Noise

Adversarial images are created by adding perturbations to images, constraining the magnitude of perturbations to be small in terms of a certain norm (*e.g.*, L_∞ or L_2). The perturbations are assumed to be either imperceptible by humans, or perceived as noise that does not impede human recognition of the visual content. Although the perturbations are constrained to be small at the *pixel level*, no such constraints are imposed at the *feature level* in convolutional networks. Indeed, the perturbation of the features induced by an adversarial image gradually increases as the image is propagated through the network [69], [115], and non-existing activations in the feature maps are hallucinated. In other words, the transformations performed by the layers in the network exacerbate the perturbation, and the hallucinated activations can overwhelm the activations due to the true signal, which leads the network to make wrong predictions.

We qualitatively demonstrate these characteristics of adversarial images by visualizing the feature maps they give rise to. Given a clean image and its adversarially perturbed counterpart, we use the same network (here, a ResNet-50 [72]) to compute its activations in the hidden layers. Figure 4.1 and 4.2 show typical examples of the same feature map on clean and adversarial images, extracted from the middle of the network (in particular, from a res_3 block). These figures reveal that the feature maps corresponding to adversarial images have activations in regions without relevant visual content that resemble feature noise. Assuming that strong activations indicate the presence of semantic information about the image content (as often hypothesized [237]), the activations that are hallucinated by adversarial images reveal why the model predictions are altered.

In this study, we attempt to address this problem by feature denoising. In Figure 4.3, we visualize the feature maps of adversarial images, right before and right after a

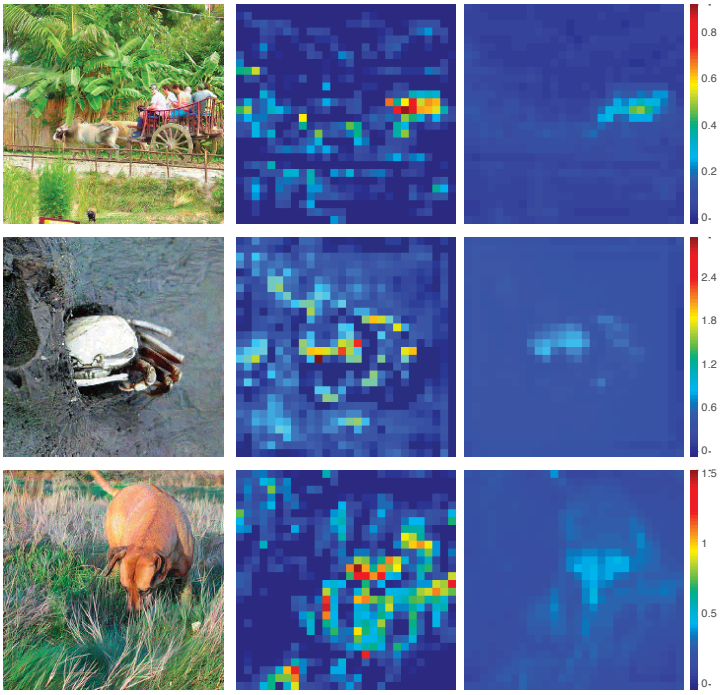


Figure 4.3. Adversarial images and their feature maps *before* (left) and *after* (right) the *denoising operation* (blue box in Figure 4.4). Here each pair of feature maps are from the same channel of a res_3 block in the same adversarially trained ResNet-50 equipped with (Gaussian) non-local means denoising blocks. The attacker has a maximum perturbation $\epsilon = 16$ for each pixel.

feature denoising operation (see the next section for details). The figure shows that feature denoising operations can successfully suppress much of the noise in the feature maps, and make the responses focus on visually meaningful content. In the next sections, we present empirical evidence showing that models that perform feature denoising operations, indeed, improve adversarial robustness.

Before we move on to describing our methods, we note although the feature noise can be easily observed *qualitatively*, it is difficult to *quantitatively* measure this noise. We found it is nontrivial to compare feature noise levels between different models, in particular, when the network architecture and/or training methods change. *E.g.*, adding a denoising block in a network, end-to-end trained, tends to change the magnitudes/distributions of all features. Nevertheless, we believe the observed *noisy* appearance of features reflects a real phenomenon associated with adversarial images.

4.4 Denoising Feature Maps

Motivated by the empirical observations above, we propose to improve adversarial robustness by adding denoising blocks at intermediate layers of convolutional networks. The denoising blocks are trained jointly with all layers of the network in an end-to-end manner using adversarial training. The end-to-end adversarial training allows the resulting networks to (partly) eliminate feature map noise that is data-dependent, *i.e.*, noise that is generated by the attacker. It also naturally handles the noise across multiple layers by considering how changes in earlier layers may impact the feature/noise distributions of later layers.

Empirically, we find that the best-performed denoising blocks are inspired by self-attention transformers [204] that are commonly used in machine translation and by non-local networks [210] that are used for video classification. In this study, we focus on the design of denoising blocks and study their denoising effects. Besides non-local means, we also experiment with simpler denoising operations such as bilateral filtering, mean filtering, and median filtering inside our convolutional networks.

4.4.1 Denoising Block

Figure 4.4 shows the generic form of our denoising block. The input to the block can be any feature layer in the convolutional neural network. The denoising block processes the input features by a *denoising operation*, such as non-local means or other variants. The denoised representation is first processed by a 1×1 convolutional layer, and then added to the block's input via a residual connection [72].[1]

The design in Figure 4.4 is inspired by self-attention [204] and non-local blocks [210]. However, only the non-local means [16] operation in the denoising block is actually doing the denoising; the 1×1 convolutions and the residual connection are

[1] In our terminology, a "denoising *operation*" refers to the computation that only performs denoising (blue box in Figure 4.4), and a "denoising *block*" refers to the entire block (all of Figure 4.4).

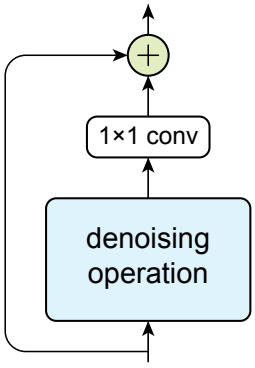


Figure 4.4. A generic denoising block. It wraps the denoising operation (*e.g.*, non-local means or median filters) with a 1×1 convolution and an identity skip connection [72].

mainly for feature combination. While various operations can suppress *noise*, they also impact *signal*. The usage of the residual connection can help the network to retain signals, and the tradeoff between removing noise and retaining signal is adjusted by the 1×1 convolution, which is learned end-to-end with the entire network. We will present ablation studies showing both the residual connection and the 1×1 convolution contribute to the effectiveness of the denoising block. The generic form of the denoising block allows us to explore various denoising operations, as introduced next.

4.4.2 Denoising Operations

We experiment with four different instantiations of the denoising operation in our denoising blocks.

Non-local means. Non-local means [16] compute a denoised feature map y of an input feature map x by taking a weighted mean of features in all spatial locations $\mathcal{L}$:

$$y_i = \frac{1}{\mathcal{C}(x)} \sum_{\forall j \in \mathcal{L}} f(x_i, x_j) \cdot x_j, \tag{4.1}$$

where $f(x_i, x_j)$ is a feature-dependent weighting function and $\mathcal{C}(x)$ is a normalization function. We note that the weighted average in Eqn. (4.1) is over x_j, rather than another embedding of x_j, unlike [204], [210]—denoising is directly on the input feature x, and the correspondence between the feature channels in y and x is kept. Following [210], we consider two forms:

- *Gaussian (softmax)* sets $f(x_i, x_j) = e^{\frac{1}{\sqrt{d}}\theta(x_i)^{\mathrm{T}}\phi(x_j)}$, where $\theta(x)$ and $\phi(x)$ are two embedded versions of x (obtained by two 1×1 convolutions), d is the number of channels, and $\mathcal{C} = \sum_{\forall j \in \mathcal{L}} f(x_i, x_j)$. By noticing that $f/\mathcal{C}$ is the *softmax* function, this version is shown in [210] to be equivalent to the softmax-based, self-attention computation of [204].

- *Dot product* sets $f(x_i, x_j) = x_i^{\mathrm{T}} x_j$ and $\mathcal{C}(x) = N$, where N is the number of pixels in x. Unlike the Gaussian non-local means, the weights of the weighted mean do *not* sum up to 1 in dot-product non-local means. However, qualitative evaluations suggest it does suppress noise in the features. Experiments also show this version improves adversarial robustness. Interestingly, we find that it is *unnecessary* to embed x in the dot-product version of non-local means for the model to work well. This is unlike the Gaussian non-local means, in which embedding is essential. The dot-product version provides a denoising operation with *no* extra parameters (blue box in Figure 4.5).

Figure 4.5, adapted from [210], shows the implementation of the denoising block based on non-local means.

Bilateral filter. It is easy to turn the non-local means in Eqn. (4.1) into a "local mean". Doing so leads to the classical *bilateral filter* [199] that is popular for edge-preserving denoising. Formally, it is defined as:

$$y_i = \frac{1}{\mathcal{C}(x)} \sum_{\forall j \in \Omega(i)} f(x_i, x_j) \cdot x_j. \tag{4.2}$$

This equation only differs from Eqn. (4.1) in the neighborhood, $\Omega(i)$, which is a local region (*e.g.*, a 3×3 patch) around pixel i. In Eqn. (4.2), we consider the Gaussian and dot product implementations of the weights as before.

Mean filter. Perhaps the simplest form of denoising is the mean filter (average pooling with a stride of 1). Mean filters reduce noise but also smooth structures, so

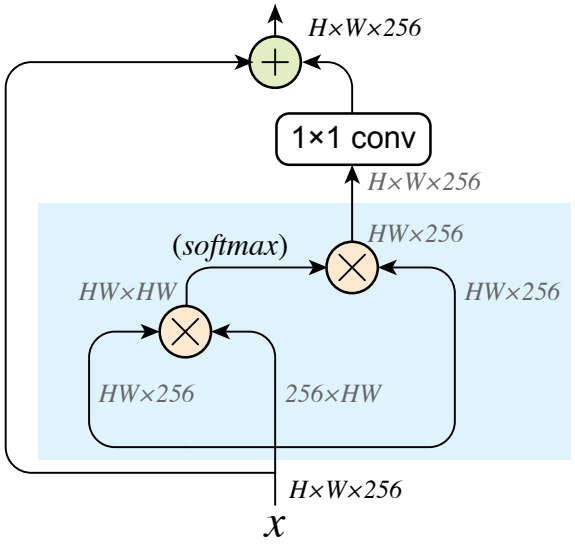


Figure 4.5. A block with *non-local means* as its denoising operation. The blue part illustrates the implementation of non-local means in Eqn. (4.1). The shapes of the feature tensors are noted, with corresponding reshaping/transposing performed: here, H and W are the height and width of the feature maps, and we use 256 channels as an example. If softmax is used, it is the Gaussian version (with appropriate 1×1 convolution embeddings used; omitted in this figure); if softmax is not used, it is the dot product version.

it is reasonable to expect them to perform worse than the above weighted means. However, somewhat surprisingly, experiments show that denoising blocks using mean filters as the denoising operation can still improve adversarial robustness.

Median filter. Lastly, we consider an interesting denoising filter that has rarely been used in deep networks: *median* filtering. The median filter is defined as:

$$y_i = \text{median}\{\forall j \in \Omega(i) : x_j\}, \tag{4.3}$$

where the median is over a local region, $\Omega(i)$, and is performed separately for each channel. Median filters are known to be good at removing salt-and-pepper noise and outliers of similar kind. Training convolutional networks that contain median filters is an open problem, but we find experimentally that using median filters as a denoising operation can also improve adversarial robustness.

In summary, our study explores a rich collection of denoising operations. Sec. 4.6 reports the results for all the denoising operations described above.

4.5 Adversarial Training

We show the effectiveness of feature denoising on top of very strong baselines. Our strong experimental results are partly driven by a successful implementation of *adversarial training* [64], [128]. In this section, we describe our implementation of adversarial training, which is used for training both baseline models and our feature denoising models.

The basic idea of adversarial training [64], [128] is to train networks on adversarially perturbed images. The adversarially perturbed images can be generated by a given white-box attacker based on the current parameters of the models. We use Projected Gradient Descent (PGD)[2] [128] as the white-box attacker for adversarial training.

PGD attacker. PGD is an iterative attacker. At each iteration, it performs a gradient descent step in the loss function w.r.t. the image pixel values, based on an adversarially selected output target. Next, it projects the resulting perturbed images into the feasible solution space—within a maximum per-pixel perturbation of ϵ of the clean image (that is, subject to an L_∞ constraint). The hyper-parameters of the PGD attacker during adversarial training are: the maximum perturbation for each pixel $\epsilon = 16$, the attack step size $\alpha = 1$, and the number of attack iterations $n = 30$. For this PGD in adversarial training, we can initialize the adversarial image by the clean image, or randomly within the allowed ϵ [128]. We randomly choose from both initializations in the PGD attacker during adversarial training: 20% of training batches use clean images to initialize PGD, and 80% use random points within the allowed ϵ.

Distributed training with adversarial images. For each mini-batch, we use PGD to generate adversarial images for that mini-batch. Then we perform a one-step SGD on these perturbed images and update the model weights. Our SGD update is based exclusively on adversarial images; the mini-batch contains no clean images.

[2]Publicly available: https://github.com/MadryLab/cifar10_challenge

Because a single SGD update is preceded by n-step PGD attack (with $n=30$) on the model, the total amount of computation in adversarial training is $\sim n\times$ bigger than standard (clean) training. To make adversarial training practical, we perform distributed training using synchronized SGD on 128 GPUs. Each mini-batch contains 32 images per GPU (*i.e.*, the total mini-batch size is $128\times 32=4096$). We follow the training recipe of [66][3] to train models with such large mini-batches. On ImageNet, our models are trained for a total of 110 epochs; we decrease the learning rate by $10\times$ at the 35-th, 70-th, and 95-th epoch. A label smoothing [195] of 0.1 is used. The total time needed for adversarial training on 128 Nvidia V100 GPUs is approximately 38 hours for the baseline ResNet-101 model, and approximately 52 hours for the baseline ResNet-152 model.

4.6 Experiments

We evaluate feature denoising on the ImageNet classification dataset [173] that has $\sim$1.28 million images in 1000 classes. Following common protocols [4], [96] for adversarial images on ImageNet, we consider *targeted* attacks when evaluating under the white-box settings, where the targeted class is selected uniformly at random; targeted attacks are also used in our adversarial training. We evaluate top-1 *classification accuracy* on the 50k ImageNet validation images that are adversarially perturbed by the attacker (regardless of its targets), also following [4], [96].

In this chapter, adversarial perturbation is considered under L_∞ norm (*i.e.*, maximum difference for each pixel), with an allowed maximum value of ϵ. The value of ϵ is relative to the pixel intensity scale of 256.

Our baselines are ResNet-101 and ResNet-152 [72]. By default, we add 4 denoising blocks to a ResNet: each is added after the last residual block of res_2, res_3, res_4, and res_5, respectively.

[3] Implemented using the publicly available Tensorpack framework [218].

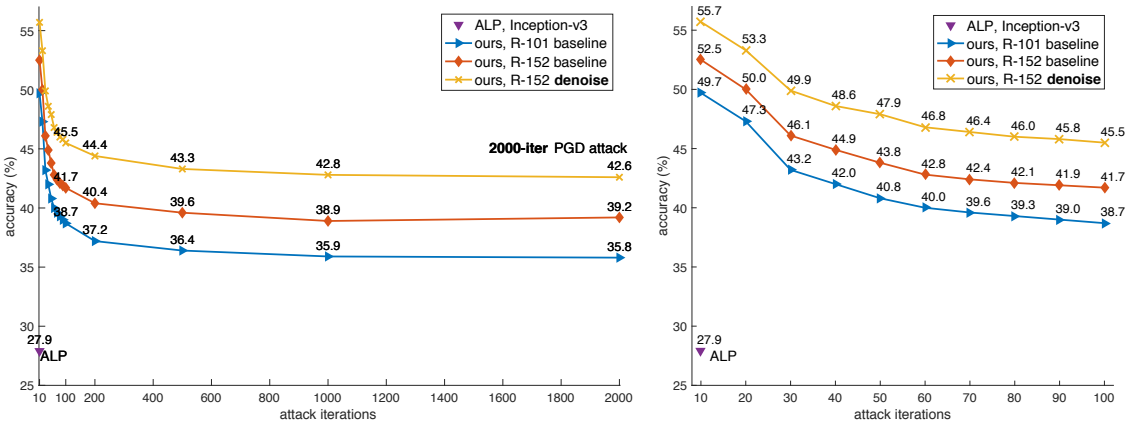


Figure 4.6. Defense against white-box attacks on ImageNet. The left plot shows results against a white-box PGD attacker with 10 to **2000** attack iterations. The right plot zooms in on the results with 10 to 100 attack iterations. The maximum perturbation is $\epsilon = 16$.

4.6.1 Against White-box Attacks

Following the evaluation protocol of ALP [96], we report defense results against PGD as the white-box attacker.[4] We evaluate with $\epsilon = 16$ (out of 255), a challenging case for defenders on ImageNet.

Following [128], the PGD white-box attacker initializes the adversarial perturbation from a random point within the allowed ϵ cube. We set its step size $\alpha = 1$, except for 10-iteration attacks where α is set to $\epsilon/10 = 1.6$. We consider a numbers of PGD attack iterations ranging from 10 to 2000.

Main results. Figure 4.6 shows the main results. We first compare with ALP [96], the previous state-of-the-art. ALP was evaluated under 10-iteration PGD attack in [96], on Inception-v3 [195]. It achieves 27.9% accuracy on ImageNet validation images (Figure 4.6, purple triangle).

ResNet-101 and ResNet-152 in Figure 4.6 are our baseline models (*without* any denoising blocks) trained using our adversarial training implementation. Even with

[4] We have also evaluated other attackers, including FGSM [64], iterative FGSM [101], and its momentum variant [46]. Similar to [96], we found that PGD is the strongest white-box attacker among them.

the lower-capacity model of R-101, our baseline is very strong—it has 49.7% accuracy under 10-iteration PGD attacks, considerably better than the ALP result. This shows that our adversarial training system is solid; we note that the comparison with ALP is on the system-level as they differ in other aspects (backbone networks, implementations, *etc.*).

"R-152, denoise" in Figure 4.6 is our model of ResNet-152 with four denoising blocks added. Here we show the best-performing version (non-local with Gaussian), which we ablate next. There is a consistent performance improvement introduced by the denoising blocks. Under the 10-iteration PGD attack, it improves the accuracy of ResNet-152 baseline by **3.2%** from 52.5% to 55.7% (Figure 4.6, right).

Our results are robust even under **2000-iteration** PGD attacks. To our knowledge, such a strong attack has *not* been previously explored on ImageNet. ALP [96] was only evaluated against 10-iteration PGD attacks (Figure 4.6), and its claimed robustness is subject to controversy [50]. Against 2000-iteration PGD attacks, our ResNet-152 baseline has 39.2% accuracy, and its denoising counterpart is **3.4%** better, achieving 42.6%. We also observe that the attacker performance diminishes with 1000~2000 attack iterations.

We note that in this *white-box* setting, the attackers can iteratively back-propagate *through* the denoising blocks and create adversarial perturbations that are tailored to the denoisers. Recent work [4] reports that *pixel* denoising methods can be circumvented by attackers in the white-box settings. By contrast, feature denoising leads to consistent improvements in white-box settings, suggesting that *feature* denoising blocks make it more difficult to fool networks.

Variants of denoising operations. Next, we evaluate variants of denoising operations in Sec. 4.4. In these ablations, we add blocks of different kinds to the baseline ResNet-152.

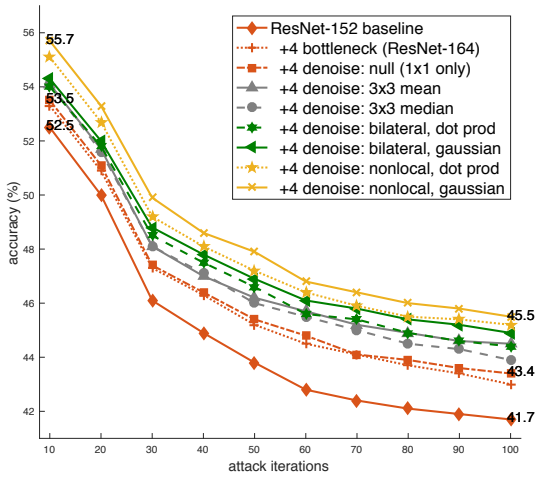


Figure 4.7. Ablation: denoising variants for defending against *white-box* attacks on ImageNet. On the ResNet-152 baseline, all other models add *4 blocks* to it. The attacker is PGD under different attack iterations, with $\epsilon = 16$. All denoising models are better than the R-152 baseline and the "null" version.

We consider the following denoising operations: 3×3 *mean filtering*, 3×3 *median filtering*, 3×3 *bilateral filtering* (Eqn. (4.2)), and *non-local filtering*. In our ablation study, we further consider a "*null*" version of the denoising block: the block in Figure 4.4 becomes trivially a residual block with a single 1×1 convolution. Further, we also compare with adding 4 standard *bottleneck* [72] blocks—essentially, ResNet-164. All models are trained by adversarial training. Figure 4.7 shows the white-box attacks results; for simplicity, we show PGD attacker with up to 100 iterations in this ablation.

All of these denoising operations have better accuracy than: (i) ResNet-152 baseline, (ii) adding 4 standard bottleneck blocks, and (iii) adding 4 "null" denoising blocks. It is worth noticing that the 1×1 null version has *the exact same number of extra parameters* as the mean filtering, median filtering, and bilateral/non-local filtering's dot product versions (which have no embedding). The null version is worse than all of them (Figure 4.7). Also, while adding standard bottleneck blocks is helpful, adding denoising blocks of *any* version is more accurate. These results suggest that the extra parameters are *not* the main reason for our accuracy improvements; feature denoising appears to be a general approach particularly useful for adversarial robustness.

attack iterations	10	100
non-local, Gaussian	55.7	45.5
removing 1×1	52.1	36.8
removing residual	NaN	NaN

Table 4.1. Ablation: denoising block design for defending against *white-box* attacks on ImageNet. Our networks have four (Gaussian) non-local means denoising blocks. We indicate the performance of models we were unable to train by "NaN".

Our best-performing model is given by the non-local (Gaussian) version, which we use by default in other parts of the chapter unless noted. Interestingly, this Gaussian version is just marginally better than the dot product version.

Design decisions of the denoising block. The denoising block in Figure 4.4 has a 1×1 layer and a residual connection. Although both components do not perform denoising, they are important for the denoising blocks to work well. Next, we ablate the behavior of the 1×1 and residual connection.

This ablation is in Table 4.1. We investigate ResNet-152 with four non-local, Gaussian denoising blocks. All models are all trained by adversarial training. When removing the 1×1 convolution in the denoising block, the accuracy drops considerably—*e.g.*, decreasing from 45.5% to 36.8% under 100-iteration PGD attacks. On the other hand, removing the residual connection makes training unstable, and its loss does not decrease in our adversarial training.

These results suggest that denoising features *in itself* is not sufficient. As suppressing noise may also remove useful signals, it appears essential to properly combine the denoised features with the input features in denoising blocks.

4.6.2 Against Black-Box Attacks

Next, we evaluate defending against *black-box* attacks. To have an unbiased yet challenging set of attackers, we study the *5 best attackers* of the NIPS 2017 CAAD competition [102], for which code is publicly available. We use the latest CAAD 2018

model	accuracy (%)
CAAD 2017 winner	0.04
CAAD 2017 winner, under 3 attackers	13.4
ours, R-152 baseline	43.1
+4 denoise: null (1×1 only)	44.1
+4 denoise: non-local, dot product	46.2
+4 denoise: non-local, Gaussian	**46.4**
+all denoise: non-local, Gaussian	**49.5**

Table 4.2. Defense against black-box attacks on ImageNet. We show top-1 classification accuracy on the ImageNet validation set. The attackers are the 5 best attackers in CAAD 2017. We adopt the CAAD 2018 "*all-or-nothing*" criterion for defenders. We note the 2017 winner has 0.04% accuracy under this strict criterion, and if we remove the 2 attackers that it is most vulnerable to, it then has 13.4% accuracy under the 3 remaining attackers.

evaluation criterion, which we call "all-or-nothing": *an image is considered correctly classified only if the model correctly classifies all adversarial versions of this image created by all attackers*. This is a challenging evaluation scenario for the defender. Following the CAAD black-box setting, the maximum perturbation for each pixel is $\epsilon = \mathbf{32}$, which also makes defense more difficult. Note that all of our models are trained with $\epsilon = 16$.

Table 4.2 shows the results of defending against black-box attacks on ImageNet validation images. To highlight the difficulty of the new "all-or-nothing" criterion, we find that the CAAD 2017 winner [115] has only 0.04% accuracy under this criterion. We find that it is mainly vulnerable to two of the five attackers[5,6]. If we remove these two attackers, [115] has 13.4% accuracy in the "all-or-nothing" setting.

With the "all-or-nothing" criterion, our ResNet-152 baseline has 43.1% accuracy against all five attackers. This number suggests that a successful implementation of adversarial training is critical for adversarial robustness.

On top of our strong ResNet-152 baseline, adding four non-local denoising blocks improves the accuracy to 46.4% (Table 4.2). Interestingly, both the Gaussian and dot

[5] https://github.com/pfnet-research/nips17-adversarial-attack
[6] https://github.com/toshi-k/kaggle-nips-2017-adversarial-attack

product versions perform similarly (46.4% *vs.* 46.2%), although the Gaussian version has more parameters due to its embedding. Furthermore, the null version has 44.1% accuracy—this is worse than the non-local, dot product version, even though they have the same number of parameters; this null version of 1×1 is 1.0% better than the ResNet-152 baseline.

We have also studied the local variants of denoising blocks, including mean, median, and bilateral filters. They have 43.6% ~ 44.4% accuracy in this black-box setting. Their results are not convincingly better than the null version's results. This suggests that non-local denoising is more important than local denoising for robustness against these black-box attackers.

Pushing the envelope. To examine the potential of our model, we add denoising blocks to *all* residual blocks (one denoising block after each residual block) in ResNet-152. We only study the non-local Gaussian version here. To make training feasible, we use the sub-sampling trick in [210]: the feature map of x_j in Eqn. (4.1) is subsampled (by a 2×2 max pooling) when performing the non-local means, noting that the feature map of x_i is still full-sized. We only use sub-sampling in this case. It achieves a number of 49.5%. This is **6.4%** better than the ResNet-152 baseline's 43.1%, under the black-box setting (Table 4.2).

CAAD 2018 challenge results. Finally, we report the results from the latest CAAD 2018 competition. The 2018 defense track adopts the "all-or-nothing" criterion mentioned above—in this case, every defense entry needs to defend against 48 *unknown* attackers submitted to the same challenge (in contrast to 5 attackers in our above black-box ablation). The test data is a secret, ImageNet-like dataset. The maximum perturbation for each pixel is $\epsilon = 32$.

Figure 4.8 shows the accuracy of the 5 best entries in the CAAD 2018 defense track. The winning entry, shown in the blue bar, was based on our method by using

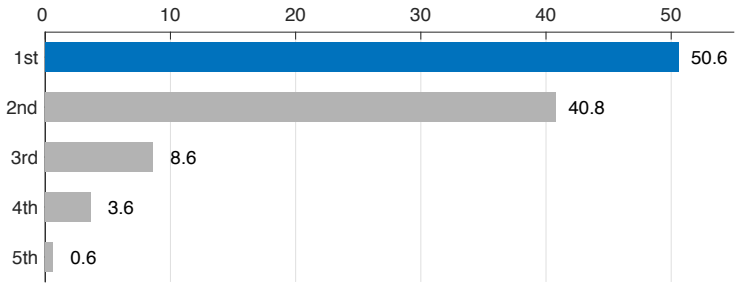


Figure 4.8. CAAD 2018 results of the adversarial defense track. The first-place entry is based on our method. We only show the 5 winning submissions here, out of more than 20 submissions.

a ResNeXt-101-32×8 backbone [232] with non-local denoising blocks added to all residual blocks. This entry only uses *single-crop, single-model* testing. It achieves **50.6%** accuracy against 48 unknown attackers. This is ~**10%** absolute (20% relative) better than the second place's 40.8% accuracy.

We also reported the white-box performance of this winning entry on ImageNet. Under 10-iteration PGD attacks and 100-iteration PGD attacks, it achieves 56.0% accuracy and 40.4% accuracy, respectively. These results are slightly worse than the robustness of ResNet-152 based models reported in Section 4.6.1. We note that this white-box robustness comparison is on the system-level as the winning entry was trained with a slightly different parameter setting.

We emphasize the CAAD 2018 defense task is very challenging because of the "all-or-nothing" criterion and many unknown (potentially new state-of-the-art) attackers. Actually, except for the two leading teams, all others have <10% accuracy and many of them have <1% accuracy. This highlights the significance of our 50.6% accuracy.

4.6.3 Denoising Blocks in Non-Adversarial Settings

Thus far we have been focusing on denoising blocks for improving adversarial defense. Because our denoising blocks are components of the convolutional networks, these networks can also be trained without adversarial training for the classification of

model	accuracy (%)
R-152 baseline	78.91
R-152 baseline, run 2	+0.05
R-152 baseline, run 3	-0.04
+4 bottleneck (R-164)	+0.13
+4 denoise: null (1×1 only)	+0.15
+4 denoise: 3×3 mean filter	+0.01
+4 denoise: 3×3 median filter	-0.12
+4 denoise: bilateral, Gaussian	+0.15
+4 denoise: non-local, Gaussian	+0.17

Table 4.3. Accuracy on clean images in the ImageNet validation set when trained on clean images. All numbers except the first row are reported as the accuracy difference comparing with the first R-152 baseline result. For R-152, we run training 3 times independently, to show the natural random variation of the same architecture. All denoising models show *no significant difference*, and are within ±0.2% of the baseline R-152's result.

"clean" images (*i.e.*, the original ImageNet dataset task). We believe studying the non-adversarial setting can help us better understand the behavior of denoising blocks.

Table 4.3 presents the clean image performance of models that were not adversarially trained. We compare the baseline R-152, adding standard bottleneck blocks, adding "null" (1×1) denoising blocks, and adding denoising blocks of various types. In the clean setting, these denoising blocks have *no obvious advantage* over the baseline R-152, adding standard bottleneck blocks, or adding "null" denoising blocks. Actually, all results are in the range of about ±0.2% of the baseline R-152's result—which have *no significant difference* if we also consider the natural variance between separate training runs of the same model (see baseline R-152 in Table 4.3).

We also find that adding non-local denoising blocks to the shallower ResNet-50 can moderately improve accuracy by 0.7% in the non-adversarial setting, but doing so on ResNet-152 has marginal gain. This, however, is not the case for adversarial images.

These results suggest that the denoising blocks could have special advantages in settings that require adversarial robustness. This observation matches our intuition that denoising blocks are designed to reduce feature noise, which only appears when classifying adversarial images.

Finally, we report our ResNet-152 baseline with *adversarial* training has 62.32% accuracy when tested on *clean* images, whereas its counterpart with "clean" training has 78.91%. For the denoising version (non-local, Gaussian), the accuracy of an adversarially trained network is 65.30% on clean images, whereas its cleanly trained counterpart has 79.08%. This tradeoff between adversarial and clean training was observed before (*e.g.*, [201]); we expect this tradeoff to be the subject of future research.

4.7 Summary

Motivated by the noisy appearance of feature maps from adversarial images, we have demonstrated the potential of feature denoising for improving the adversarial robustness of convolutional networks. Interestingly, our study suggests that there are certain *architecture designs* (*viz.*, denoising blocks) that are particularly good for adversarial robustness, even though they do not lead to accuracy improvements compared to baseline models in "clean" training and testing scenarios. When combined with adversarial training, these particular architecture designs may be more appropriate for modeling the underlying distribution of adversarial images. We hope our work will encourage researchers to start designing convolutional network architectures that have "innate" adversarial robustness.

Chapter 5

Smooth Adversarial Training

It is commonly believed that networks cannot be both accurate and robust, that gaining robustness means losing accuracy. It is also generally believed that, unless making networks larger, network architectural elements would otherwise matter little in improving adversarial robustness. Here we present evidence to challenge these common beliefs by a careful study about adversarial training. Our key observation is that the widely-used ReLU activation function significantly weakens adversarial training due to its non-smooth nature. Hence we propose *smooth adversarial training (SAT)*, in which we replace ReLU with its smooth approximations to strengthen adversarial training. The purpose of smooth activation functions in SAT is to allow it to find harder adversarial examples and compute better gradient updates during adversarial training. Compared to standard adversarial training, SAT improves adversarial robustness for "free", *i.e.*, no drop in accuracy and no increase in computational cost. For example, without introducing additional computations, SAT significantly enhances ResNet-50's robustness from 33.0% to 42.3%, while also improving accuracy by 0.9% on ImageNet. SAT also works well with larger networks: it helps EfficientNet-L1 to achieve 82.2% accuracy and 58.6% robustness on ImageNet, outperforming the previous state-of-the-art defense by 9.5% for accuracy and 11.6% for robustness. Code is available at https://github.com/cihangxie/SmoothAdversarialTraining.

5.1 Introduction

Convolutional neural networks can be easily attacked by adversarial examples, which are computed by adding small perturbations to clean inputs [196]. Many efforts have been devoted to improving network resilience against adversarial attacks [69], [119], [150], [156], [176], [226]. Among them, adversarial training [64], [100], [128], which trains networks with adversarial examples on-the-fly, stands as one of the most effective methods. Later works further improve adversarial training by feeding networks with harder adversarial examples [212], maximizing the margin of networks [44], optimizing a regularized surrogate loss [241], *etc*. While these methods achieve stronger adversarial robustness, they sacrifice accuracy on clean inputs. It is generally believed this trade-off between accuracy and robustness might be inevitable [201], unless additional computational budgets are introduced to enlarge network capacities, *e.g*., making wider or deeper networks [128], [229], adding denoising blocks [228].

Another popular direction for increasing robustness against adversarial attacks is gradient masking [4], [155], which usually introduces non-differentiable operations (e.g., discretization [17], [171]) to obfuscate gradients. With degenerated gradients, attackers cannot successfully optimize the targeted loss and fail to break such defenses. Nonetheless, gradient masking will be ineffective if its differentiable approximation is used for generating adversarial examples [4].

The bitter history of gradient masking defenses motivates us to rethink the relationship between gradient quality and adversarial robustness, especially in the context of adversarial training where gradients are applied more frequently than standard training. In addition to computing gradients to update network parameters, adversarial training also requires gradient computation for generating training samples. Guided by this principle, we identify that ReLU, a widely-used activation function in most network architectures, significantly weakens adversarial training due to its

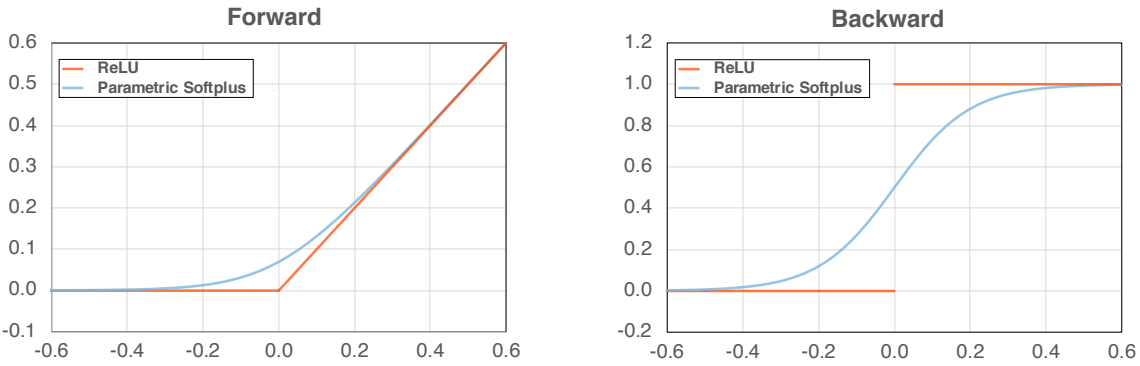


Figure 5.1. The visualization of ReLU and Parametric Softplus. *Left panel*: the forward pass for ReLU (blue curve) and Parametric Softplus (red curve). *Right panel*: the first derivatives for ReLU (blue curve) and Parametric Softplus (red curve). Different from ReLU, Parametric Softplus is smooth with continuous derivatives.

non-smooth nature, *i.e.*, ReLU's gradient gets an abrupt change when its input is zero, as illustrated in Figure 5.1.

To fix the issue induced by ReLU, in this chapter, we propose smooth adversarial training (SAT), which enforces architectural smoothness via replacing ReLU with its smooth approximations[1] for improving the gradient quality in adversarial training (Figure 5.1 shows Parametric Softplus, an example of smooth approximations for ReLU). With smooth activation functions, SAT is able to feed the networks with harder adversarial training samples and compute better gradient updates for network optimization, hence substantially strengthens adversarial training. Our experiment results show that SAT improves adversarial robustness for "free", *i.e.*, without incurring additional computations or degrading standard accuracy. For instance, by training with the economical *single-step PGD attacker* (*i.e.*, FGSM attacker with random initialization)[2] on ImageNet [173], SAT significantly improves ResNet-50's robustness by 9.3%, from 33.0% to 42.3%, while increasing the standard accuracy by 0.9% without incurring additional computational cost.

[1]More precisely, when we say a function is smooth in this chapter, we mean this function is $\mathcal{C}^1$ smooth, *i.e.*, its first derivative is continuous everywhere.

[2]In practice, we note the training time of the models with single-step PGD adversarial training is only $\sim$1.5$\times$ than the training time of their standard training counterparts.

We also explore the limits of SAT with larger networks. We obtain the best result by using EfficientNet-L1 [197], [231], which achieves 82.2% accuracy and 58.6% robustness on ImageNet, significantly outperforming the prior art [161] by 9.5% for accuracy and 11.6% for robustness.

5.2 Related Works

Adversarial training. Adversarial training improves robustness by training models on adversarial examples [64], [100], [128], [196]. Existing works suggest that, to further adversarial robustness, we need to either sacrifice accuracy on clean inputs [44], [212], [213], [241], or incur additional computational cost [128], [228], [229]. This phenomenon is referred to as *no free lunch in adversarial robustness* [145], [191], [201]. We hereby show that, with SAT, adversarial robustness can be improved for "free"—no accuracy degradation on clean images and no additional computational cost incurred.

Our work is also related to the theoretical study [184], which shows replacing ReLU with smooth alternatives can help networks get a tractable bound when certifying distributional robustness. In this chapter, we empirically corroborate the benefits of utilizing smooth activations is also observable in the practical adversarial training on the real-world dataset using large networks.

Gradient masking. Besides training models on adversarial data, other ways for improving adversarial robustness include defensive distillation [156], randomized transformations [11], [19], [43], [119], [209], [221], [226], adversarial input purification [11], [69], [115], [132], [151], [159], [174], [186], *etc.* Nonetheless, these defense methods degenerate the gradient quality, therefore induce the gradient masking issue [155], which gives a false sense of adversarial robustness [4]. In contrast to these works, we aim to improve adversarial robustness by providing networks with better gradients, but in the context of adversarial training.

5.3 ReLU Weakens Adversarial Training

We hereby perform a series of control experiments in the backward pass of gradient computations to investigate how ReLU weakens, and how its smooth approximation strengthens adversarial training.

5.3.1 Adversarial Training

Adversarial training [64], [128], [196], which trains networks with adversarial examples on-the-fly, aims to optimize the following framework:

$$\arg\min_{\theta} \mathbb{E}_{(x,y)\sim\mathbb{D}}\Big[\max_{\epsilon\in\mathbb{S}} L(\theta, x+\epsilon, y)\Big], \tag{5.1}$$

where $\mathbb{D}$ is the underlying data distribution, $L(\cdot,\cdot,\cdot)$ is the loss function, θ is the network parameter, x is a training sample with the ground-truth label y, ϵ is the added adversarial perturbation, and $\mathbb{S}$ is the allowed perturbation range. As shown in Eqn. (5.1), adversarial training consists of two computation steps: an **inner maximization step**, which computes adversarial examples, and an **outer minimization step**, which computes parameter updates.

Adversarial training setup. We choose ResNet-50 [72] as the backbone network. We apply PGD attacker [128] to generate adversarial perturbations ϵ. Specifically, we select the cheapest version of PGD, *single-step PGD* (PGD-1), to lower the training cost. Following [177], [215], we set the maximum per-pixel change $\epsilon = 4$ and the attack step size $\beta = 4$. We follow the basic ResNet training recipes on ImageNet: models are trained for a total of 100 epochs using momentum SGD optimizer, with the learning rate decreased by 10× at the 30-th, 60-th and 90-th epoch; no regularization except a weight decay of 1e-4 is applied.

When evaluating adversarial robustness, we measure the model's top-1 accuracy against the 200-step PGD attacker (PGD-200) on the ImageNet validation set, with

the maximum perturbation size $\epsilon = 4$ and the step size $\beta = 1$. We note 200 attack iteration is enough to let PGD attacker converge. Meanwhile, we report the model's top-1 accuracy on the original ImageNet validation set.

5.3.2 How Gradient Quality Affects Adversarial Training?

As shown in Figure 5.1, the widely used activation function, ReLU [71], [143], is non-smooth. ReLU's gradient takes an abrupt change, when its input is 0, therefore significantly degrades the gradient quality. We conjecture that this non-smooth nature hurts the training process, especially when we train models adversarially. This is because, compared to standard training which only computes gradients for updating network parameter θ, adversarial training requires additional computations for the inner maximization step to craft the perturbation ϵ.

To fix this problem, we first introduce a smooth approximation of ReLU, named *Parametric Softplus* [143], as follows,

$$f(\alpha, x) = \frac{1}{\alpha}\log(1 + \exp(\alpha x)), \tag{5.2}$$

where the hyperparameter α is used to control the curve shape. The derivative of this function w.r.t. the input x is:

$$\frac{d}{dx}f(\alpha, x) = \frac{1}{1 + \exp(-\alpha x)} \tag{5.3}$$

To better approximate the curve of ReLU, we empirically set $\alpha = 10$. As shown in Figure 5.1, compared to ReLU, Parametric Softplus (α=10) is smooth because it has a continuous derivative.

With Parametric Softplus, we next diagnose how gradient quality in *the inner maximization step* and *the outer minimization step* affects the accuracy and robustness of ResNet-50 in adversarial training. **To clearly benchmark the effects, we only substitute ReLU with Eqn. (5.3) in the backward pass, while leaving the forward pass unchanged**, *i.e.*, ReLU is always used for model inference.

	Improving Gradient Quality for the Adversarial Attacker	Improving Gradient Quality for the Network Parameter Updates	Accuracy (%)	Robustness (%)
ResNet-50	✗	✗	68.8	33.0
	✓	✗	68.3 (-0.5)	34.5 (+1.5)
	✗	✓	69.4 (+0.6)	35.8 (+2.8)
	✓	✓	68.9 (+0.1)	36.9 (+3.9)

Table 5.1. ReLU significantly weakens adversarial training. By improving gradient quality for either the adversarial attacker or the network optimizer, resulted models obtains better robustness than the ReLU baseline. The best robustness is achieved by adopting better gradients for both the attacker and the network optimizer.

Improving gradient quality for the adversarial attacker. We first take a look at the effects of gradient quality on computing adversarial examples (*i.e.*, *the inner maximization step*) during training. More precisely, in the inner step of adversarial training, we use ReLU in the forward pass, but Parametric Softplus in the backward pass; and in the outer step, we use ReLU in both the forward and the backward pass.

As shown in the second row of Table 5.1, when the attacker uses Parametric Softplus's gradient to generate training samples, the resulted model exhibits a performance trade-off compared to the ReLU baseline, *i.e.*, it improves adversarial robustness by 1.5% but degrades accuracy by 0.5%. We note that this performance trade-off can also observed if harder adversarial examples are applied to train networks [212], therefore motivate us to hypothesize that better gradients for the inner maximization step actually boosts the attacker's strength during training. To verify this hypothesis, we evaluate the robustness of two ResNet-50 models via PGD-1 (*vs.* PGD-200 in Table 5.1), one with standard training and one with adversarial training. Specifically, during the evaluation, the attacker uses ReLU in the forward pass, but Parametric Softplus in the backward pass. With better gradients, we note that PGD-1 attacker is strengthened and hurts models more: it can further decrease the top-1 accuracy by 4.0% (from 16.9% to 12.9%) on the model with standard training, and by 0.7% (from 48.7% to 48.0%) on the model with adversarial training (both results are not shown in Table 5.1).

Improving gradient quality for network parameter updates. We then study the role of gradient quality on updating network parameters (*i.e.*, *the outer minimization step*) during training. More precisely, in the inner step of adversarial training, we always use ReLU; but in the outer step, we use ReLU in the forward pass, and Parametric Softplus in the backward pass.

Surprisingly, this strategy improves adversarial robustness for "free". As shown in the third row of Table 5.1, without incurring additional computations, adversarial robustness is boosted by 2.8%, and meanwhile accuracy is improved by 0.6%, compared to the ReLU baseline. We note the corresponding training loss also gets lower: the cross-entropy loss on the training set is reduced from 2.71 to 2.59. These results of better robustness and accuracy, and lower training loss together suggest that, with Parametric Softplus in the backward pass of the outer minimization step, networks are able to compute better gradient updates in adversarial training. Interestingly, we also observe that better gradient updates improve the standard training, *i.e.*, with ResNet-50, training with better gradients is able to improve accuracy from 76.8% to 77.0%, and reduces the corresponding training loss from 1.22 to 1.18. These results on both adversarial training and standard training suggest that updating network parameters using better gradients could serve as a principle for improving performance in general, while keeping the inference process of the model unchanged (*i.e.*, ReLU is always used for inference).

Improving gradient quality for both the adversarial attacker and network parameter updates. Given the observation that improving ReLU's gradient for either the adversarial attacker or the network optimizer benefits robustness, we further enhance adversarial training by replacing ReLU with Parametric Softplus in all backward passes, but keeping ReLU in all forward passes.

As expected, such a trained model reports the best robustness so far, *i.e.*, as shown in the last row of Table 5.1, it substantially outperforms the ReLU baseline by 3.9%

for robustness. Interestingly, this improvement still comes for "free", *i.e.*, it reports 0.1% higher accuracy than the ReLU baseline. We conjecture this is mainly due to the positive effect on accuracy brought by computing better gradient updates (increase accuracy) slightly overriding the negative effects on accuracy brought by creating harder training samples (hurt accuracy) in this experiment.

5.3.3 Can Other Training Enhancements Remedy ReLU's Gradient Issue?

More attack iterations. It is known that increasing the number of attack iterations can create harder adversarial examples [128]. We confirm in our own experiments that by training with PGD attacker with more iterations, the resulted model exhibits a similar behavior to the case where we apply better gradients for the attacker. By increasing the attacker's cost by 2×, PGD-2 improves the ReLU baseline by 0.6% for robustness while losing 0.1% for accuracy. This result suggests we can remedy ReLU's gradient issue *in the inner step of adversarial training* if more computations are given.

Training longer. It is also known longer training lowers the training loss [81], which we explore next. Interestingly, by extending the default setup to a 2× training cost (*i.e.*, 200 epochs), though the final model indeed achieves a lower training loss (from 2.71 to 2.62), there still exhibits a trade-off between accuracy and robustness. Longer training gains 2.6% for accuracy but loses 1.8% for robustness. On the contrary, our previous experiment shows applying better gradients to optimize networks improves both robustness and accuracy. This discouraging result suggests training longer *cannot* fix the issues *in the outer step of adversarial training* caused by ReLU's poor gradient.

Conclusion. Given these results, we conclude that ReLU significantly weakens adversarial training. Moreover, it seems that the degenerated performance cannot be simply remedied even with training enhancements (*i.e.*, increasing the number of attack iterations & training longer). We identify that the key is ReLU's poor gradient—by

replacing ReLU with its smooth approximation *only in the backward pass* substantially improves robustness, even without sacrificing accuracy and incurring additional computational cost. In the next section, we show that making activation functions smooth is a good design principle for enhancing adversarial training in general.

5.4 Smooth Adversarial Training

As shown above, improving ReLU's gradient can both strengthen the attacker and provide better gradient updates. Nonetheless, this strategy may be suboptimal as there still is a discrepancy between the forward pass (which we use ReLU) and the backward pass (which we use Parametric Softplus). To fully exploit the potential of training with better gradients, we hereby propose smooth adversarial training (SAT), which enforces architectural smoothness via the exclusive usage of smooth activation functions in adversarial training. We keep all other network components the same, as most of them will not result in the issue of poor gradient.[3]

5.4.1 Adversarial Training with Smooth Activation Functions

We consider the following activation functions as the smooth approximations of ReLU in SAT (Figure 5.2 plots these functions as well as their derivatives):

- **Softplus** [143]: $\text{Softplus}(x) = \log(1 + \exp(x))$. We also consider its parametric version, *i.e.*, $\frac{1}{\alpha}\log(1 + \exp(\alpha x))$, and set $\alpha = 10$ as in Section 5.3.
- **SILU** [49], [75], [165]: $\text{SILU}(x) = x \cdot \text{sigmoid}(x)$. Compared to others, SILU has a non-monotonic "bump" when $x < 0$.
- **Gaussian Error Linear Unit** (GELU) [75]: $\text{GELU}(x) = x \cdot \Phi(x)$, where $\Phi(x)$ is the cumulative distribution function of the standard normal distribution.

[3] We ignore the gradient issue caused by max pooling, which is also non-smooth, in SAT. This is because modern architectures rarely adopt it, *e.g.* only one max pooling layer is adopted in ResNet [72], and none is adopted in EfficientNet [197].

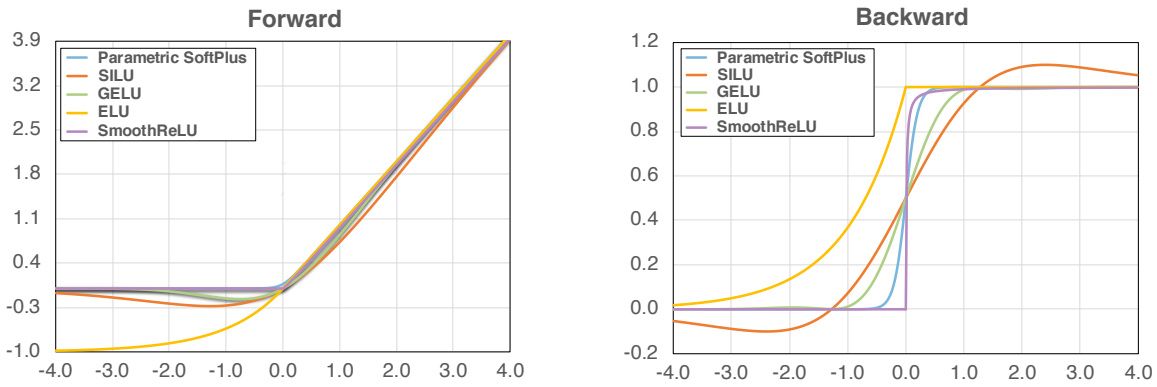


Figure 5.2. Visualizations of smooth activation functions and their derivatives.

- **Exponential Linear Unit** (ELU) [33]: $\text{ELU}(x, \alpha) = x$ if $x \geq 0$, otherwise $\text{ELU}(x, \alpha) = \alpha(\exp(x) - 1)$. We set $\alpha = 1$ as default. Note that when $\alpha \neq 1$, the gradient of ELU is not continuously differentiable anymore. We will be discussing the effects of these non-smooth variants of ELU ($\alpha \neq 1$) on adversarial training in Section 5.4.3.

Main results. We follow the settings in Section 5.3 to adversarially train ResNet-50 equipped with smooth activation functions. The results are shown in Figure 5.3. Compared to the ReLU baseline, all smooth activation functions substantially boost robustness, while keeping the standard accuracy almost the same. For example, smooth activation functions at least boost robustness by 5.7% (using Parametric Softplus, from 33% to 38.7%). We believe such improvement is generalizable to other smooth alternatives (*e.g.*, [121], [138]). Our strongest robustness is reported by SILU, which enables ResNet-50 to achieve 42.3% robustness and 69.7% standard accuracy.

Additionally, we compare to the setting in Section 5.3 where Parametric Softplus is only applied at the backward pass. Interestingly, by additionally replacing ReLU with Parametric Softplus at the forward pass, the resulted model further improves robustness by 1.8% (from 36.9% to 38.7%) while keeping the accuracy almost the same, demonstrating the importance of applying smooth activation functions in both forward and backward passes in SAT.

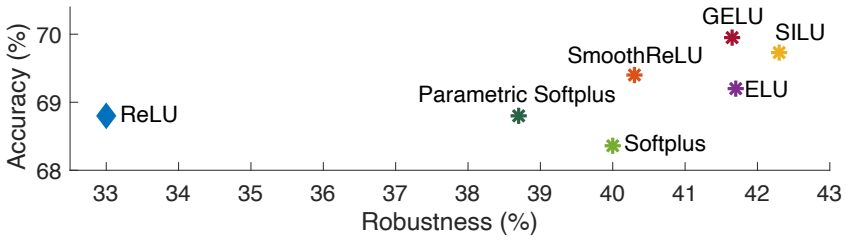


Figure 5.3. Smooth activation functions improve adversarial training. Compared to ReLU, all smooth activation functions significantly boost model robustness, while keeping the accuracy on clean images almost the same.

α	Robustness (%)	
	ELU	CELU
1	41.1	
1.2	-0.3	+0.1
1.4	-2.0	-0.3
1.6	-3.7	-0.3
1.8	-6.2	-0.2
2.0	-7.9	-0.5

Table 5.2. Robustness comparison between ELU (non-smooth when $\alpha \neq 1$) and CELU (always smooth $\forall\alpha$).

5.4.2 Ruling Out the Effect From $x < 0$

Compared to ReLU, in addition to being smooth, the functions above have non-zero responses to negative inputs ($x < 0$) which may also affect adversarial training. To rule out this factor, inspired by [178], we hereby propose SmoothReLU, which flattens the activation function by only modifying ReLU after $x \geq 0$,

$$\text{SmoothReLU}(x, \alpha) = \begin{cases} x - \frac{1}{\alpha}\log(\alpha x + 1) & \text{if } x \geq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{5.4}$$

where α is a learnable variable shared by all channels, and is constrained to be positive. We note SmoothReLU is always continuously differentiable regardless the value of α,

$$\frac{d}{dx}\text{SmoothReLU}(x, \alpha) = \begin{cases} \frac{\alpha x}{1+\alpha x} & \text{if } x \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{5.5}$$

Note SmoothReLU converges to ReLU when $\alpha \to \infty$. Additionally, in practice, the learnable parameter α needs to be initialized at a large enough value (*e.g.*, 400 in our experiments) to avoid the gradient vanishing problem at the beginning of training. We plot SmoothReLU and its first derivative in Figure 5.2.

We observe SmoothReLU substantially outperforms ReLU by 7.3% for robustness (from 33.0% to 40.3%), and by 0.6% for accuracy (from 68.8% to 69.4%), therefore clearly demonstrates the importance of a function to be smooth, and rules out the effect from having responses when $x < 0$.

5.4.3 Case Study: Stabilizing Adversarial Training with ELU using CELU

In the analysis above, we show that adversarial training can be greatly improved by replacing ReLU with its smooth approximations. To further demonstrate the generalization of SAT (beyond ReLU), we discuss another type of activation function—ELU. The first derivative of ELU is shown below:

$$\frac{d}{dx}\,\mathrm{ELU}(x,\alpha) = \begin{cases} 1 & \text{if } x \geq 0, \\ \alpha \exp(x) & \text{otherwise.} \end{cases} \tag{5.6}$$

Here we mainly discuss the scenario when ELU is non-smooth, *i.e.*, $\alpha \neq 1$. As can be seen from Eqn. (5.6), ELU's gradient is not continuously differentiable anymore, *i.e.*, $\alpha \exp(x) \neq 1$ when $x = 0$, therefore resulting in an abrupt gradient change like ReLU. Specifically, we consider the range $1.0 < \alpha \leq 2.0$, where the gradient abruption becomes more drastic with a larger value of α.

We show the adversarial training results in Table 5.2. Interestingly, we observe that the adversarial robustness is highly dependent on the value of α—the strongest robustness is achieved when the function is smooth (*i.e.*, $\alpha = 1.0$, 41.4% robustness), and all other choices of α monotonically decrease the robustness when α gradually approaches 2.0. For instance, with $\alpha = 2.0$, the robustness drops to only 33.2%, which is 7.9% lower than that of using $\alpha = 1.0$. The observed phenomenon here is consistent with our previous conclusion on ReLU—non-smooth activation functions significantly weaken adversarial training.

To stabilize the adversarial training with ELU, we apply its smooth version, CELU [9], which re-parametrize ELU to the following format:

$$\mathrm{CELU}(x,\alpha) = \begin{cases} x & \text{if } x \geq 0, \\ \alpha\left(\exp\left(\frac{x}{\alpha}\right) - 1\right) & \text{otherwise.} \end{cases} \tag{5.7}$$

The first derivatives of CELU can be written as follows:

$$\frac{d}{dx}\,\mathrm{CELU}(x,\alpha) = \begin{cases} 1 & \text{if } x \geq 0, \\ \exp\frac{x}{\alpha} & \text{otherwise.} \end{cases} \tag{5.8}$$

With this parameterization, CELU is now continuously differentiable regardless of the choice of α.

We observe that CELU greatly stabilizes adversarial training, *i.e.*, compared to $\alpha = 1.0$, the worst case in CELU is merely 0.5% lower (shown in Table 5.2). Recall that this gap for ELU is 7.9%. This case study provides another strong support on justifying the importance of performing SAT.

5.5 Exploring the Limits of Smooth Adversarial Training

Recent works [57], [229] show that, compared to standard training, adversarial training exhibits a much stronger requirement for larger networks to obtain better performance. Nonetheless, previous explorations in this direction only consider either deeper networks [229] or wider networks [128], which might be insufficient. To this end, we hereby present a systematic study on showing how network scaling up behaves in SAT. Specifically, we set SILU as the default activation function to perform SAT, as it achieves the best robustness among different candidates (as shown in Figure 5.3).

5.5.1 Scaling-up ResNet

We first perform the network scaling-up experiments with ResNet in SAT. In standard training, [197] suggest that, all three scaling-up factors, *i.e.*, *depth, width and image resolutions*, are important to further improve ResNet performance. We hereby examine the effects of these factors in SAT. We choose ResNet-50 (with the default image resolution at 224) as the baseline network.

Depth & width. Previous works already show that making networks deeper or wider can further standard adversarial training. We re-verify this conclusion in SAT. As shown in the second to fifth rows of Table 5.3, we confirm that both deeper or

	Accuracy (%)	Robustness (%)
ResNet-50	69.7	42.3
+ 2x deeper (ResNet-101)	72.9 (+3.2)	45.5 (+3.2)
+ 3x deeper (ResNet-152)	73.9 (+4.2)	46.0 (+3.7)
+ 2x wider (ResNeXt-50-32x4d)	71.2 (+1.5)	42.5 (+0.2)
+ 4x wider (ResNeXt-50-32x8d)	73.6 (+3.9)	45.1 (+2.8)
+ larger resolution 299	70.9 (+1.2)	43.8 (+1.5)
+ larger resolution 380	71.6 (+1.9)	44.1 (+1.8)
+ 3x deeper & 4x wider (ResNeXt-152-32x8d) & larger resolution 380	**78.2 (+8.5)**	**51.2 (+8.9)**

Table 5.3. Scaling-up ResNet in SAT. We observe SAT consistently helps larger networks get better performance.

wider networks consistently outperform the baseline network in SAT. For instance, by training a deeper ResNet-152, it improves ResNet-50's performance by 4.2% for accuracy and 3.7% for robustness. Similarly, by training a 4× wider ResNeXt-50-32x8d [232], it improves accuracy by 3.9% and robustness by 2.8%.

Image resolution. Though larger image resolution benefits standard training, it is generally believed that scaling up this factor will induce weaker adversarial robustness [56]. However, surprisingly, this belief is invalid when taking adversarial training into consideration. As shown in the sixth and seventh rows of Table 5.3, ResNet-50 consistently achieves better performance when training with larger image resolutions in SAT. We conjecture this improvement is possibly due to a larger image resolution (1) enables attackers to create stronger adversarial examples [56]; and (2) increases network capacity [197], therefore benefits SAT overall.

Compound scaling. So far, we have confirmed that the basic scaling of depth, width and image resolution are all important scaling-up factors in SAT. As argued in [197] for standard training, scaling up all these factors simultaneously is better than just focusing on a single dimension. To this end, we make an attempt to create a simple compound scaling for ResNet. As shown in the last row of Table 5.3, the resulted model, ResNeXt-152-32x8d with input resolution at 380, achieves a much stronger result than the ResNet-50 baseline, *i.e.*, +8.5% for accuracy and +8.9% for robustness.

Discussion on standard adversarial training. We first confirm that the basic scaling of depth, width and image resolution also matter in *standard adversarial training*, *e.g.*, by scaling up ResNet-50 (33.0% robustness), the deeper ResNet-152 achieves 39.4% robustness (+6.4%), the wider ResNeXt-50-32x8d achieves 36.7% robustness (+3.7%), and the ResNet-50 with larger image resolution at 380 achieves 36.9% robustness (+3.9%). Nonetheless, all these robustness performances are lower than the robustness achieved by the SAT's ResNet-50 (42.3%, first row of Table 5.3). In other words, scaling up networks seems less effective than replacing ReLU with smooth activation functions.

We also confirm that the compound scaling is much more effective than the basic scaling for standard adversarial training, *e.g.*, ResNeXt-152-32x8d with input resolution at 380 here reports 46.3% robustness. Although this result is better than adversarial training with the basic scaling above, it is still ~5% lower than SAT with compound scaling, *i.e.*, 46.3% v.s. 51.2%. In other words, even with larger networks, applying smooth activation functions in adversarial training is still essential for improving performance.

5.5.2 SAT with EfficientNet

The results on ResNet show that scaling up networks in SAT effectively improves performance. Nonetheless, the applied scaling policies could be suboptimal, as they are hand-designed without any optimizations. EfficientNet [197], which uses neural architecture search [250] to automatically discover the optimal factors for network scaling, provides a strong family of models for image recognition. To examine the benefits of EfficientNet, we now use it to replace ResNet in SAT. Note that all other training settings are the same as described in our ResNet experiments.

Similar to ResNet, Figure 5.4 shows stronger backbones consistently achieve better performance in SAT. For instance, by scaling the network from EfficientNet-B0 to

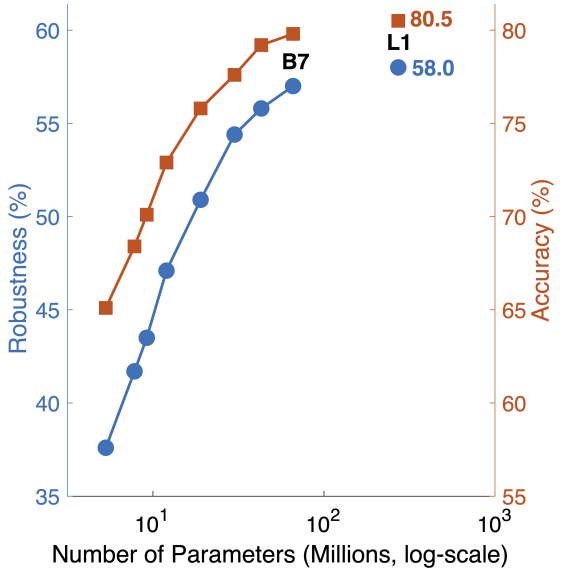


Figure 5.4. Scaling-up EfficientNet in SAT. We can observe that both robustness and accuracy get substantially improved if a larger EfficientNet is applied in SAT. Note EfficientNet-L1 is not connected to the rest of the graph because it was not part of the compound scaling suggested by [197].

EfficientNet-B7, the robustness is improved from 37.6% to 57.0%, and the accuracy is improved from 65.1% to 79.8%. Surprisingly, the improvement is still observable for larger networks: EfficientNet-L1 [231] further improves robustness by 1.0% and accuracy by 0.7% over EfficientNet-B7.

Training enhancements on EfficientNet. So far all of our experiments follow the training recipes from ResNet, which may not be optimal for EfficientNet training. To this end, we import the following settings to our experiments as in original EfficientNet training setups [197]: we change weight decay from 1e-4 to 1e-5, and add Dropout [188], stochastic depth [85] and AutoAugment [37] to regularize the training process. Besides, we train models longer (*i.e.*, 200 epochs) to better cope with these training enhancements, and adopt the early stopping strategy to prevent the catastrophic overfitting issue in robustness [215]. With these training enhancements, our EfficientNet-L1 gets further improved, *i.e.*, +1.7% for accuracy (from 80.5% to 82.2%) and +0.6% for robustness (from 58.0% to 58.6%).

	Accuracy (%)	Robustness (%)
Prior art [161]	72.7	47.0
EfficientNet+SAT	**82.2 (+9.5)**	**58.6 (+11.6)**

Table 5.4. Comparison to the previous state-of-the-art.

Comparing to the prior art [161]. Table 5.4 compares our best results with the prior art. With SAT, we are able to train a model with strong performance on both adversarial robustness and standard accuracy—our best model (EfficientNet-L1 + SAT) achieves 82.2% standard accuracy and 58.6% robustness, which largely outperforms the previous state-of-the-art method [161] by 9.5% on standard accuracy and 11.6% on adversarial robustness.

Discussion. Finally, we emphasize a large reduction in the accuracy gap between adversarially trained models and standard trained models for large networks. For example, with the training setup above (with enhancements), EfficientNet-L1 achieves 84.1% accuracy in standard training, and this accuracy slightly decreases to 82.2% (-1.9%) in SAT. Note that this gap is substantially smaller than the gap in ResNet-50 of 7.1% (76.8% in standard training v.s. 69.7% in SAT). Moreover, it is also worth mentioning that the high accuracy of 82.2% provides strong support to [90] on arguing robust features indeed can generalize well to clean inputs.

5.6 Summary

In this chapter, we propose smooth adversarial training, which enforces architectural smoothness via replacing non-smooth activation functions with their smooth approximations in adversarial training. SAT improves adversarial robustness without sacrificing standard accuracy or incurring additional computation cost. Extensive experiments demonstrate the general effectiveness of SAT. With EfficientNet-L1, SAT reports the state-of-the-art adversarial robustness on ImageNet, which largely outperforms the prior art [161] by 9.5% for accuracy and 11.6% for robustness.

Chapter 6

Intriguing Properties of Adversarial Training at Scale

Adversarial training is one of the main defenses against adversarial attacks. In this chapter, we provide the first rigorous study on diagnosing elements of large-scale adversarial training on ImageNet, which reveals two intriguing properties.

First, we study the role of normalization. Batch Normalization (BN) is a crucial element to help deep networks achieve state-of-the-art performance on a wide range of visual benchmarks, but we show it may prevent networks from obtaining strong robustness in adversarial training. One unexpected observation is that, for models trained with BN, simply removing clean images from training data can largely boost adversarial robustness, *e.g.*, from 20.9% to 39.2% (+18.3%). We relate this phenomenon to the hypothesis that clean images and adversarial images are drawn from two different domains. This two-domain hypothesis may explain the issue of BN when training with a mixture of clean and adversarial images, as estimating normalization statistics of this mixture distribution is challenging. Guided by this two-domain hypothesis, we show disentangling the mixture distribution for normalization, *i.e.*, applying separate BNs to clean and adversarial images for statistics estimation, achieves much stronger robustness. Additionally, we find that enforcing BNs to behave consistently at training and testing can further enhance robustness.

Second, we study the role of network capacity. We find our so-called "deep" networks are still shallow for the task of adversarial learning. Unlike traditional classification tasks where accuracy is only marginally improved by adding more layers to "deep" networks (*e.g.*, ResNet-152), adversarial training exhibits a much stronger demand on deeper networks to achieve higher adversarial robustness. This robustness improvement can be observed substantially and consistently even by pushing the network capacity to an unprecedented scale, *i.e.*, ResNet-638.

6.1 Introduction

Adversarial attacks [196] can mislead neural networks to make wrong predictions by adding human imperceptible perturbations to input data. Adversarial training [64] is shown to be an effective method to defend against such attacks, which trains neural networks on adversarial images that are generated on-the-fly during training. Later works further improve robustness of adversarially trained models by imposing logits pairing [96], denoising at feature space [228], *etc*. However, these works mainly focus on justifying the effectiveness of proposed strategies and apply inconsistent pipelines for adversarial training, which leaves revealing important elements for training robust models still a missing piece in current adversarial research.

In this chapter, we provide the first rigorous diagnosis of different adversarial learning strategies, under a unified training and testing framework, on the large-scale ImageNet dataset [173]. We discover two intriguing properties of adversarial training, which are essential for training models with stronger robustness. First, though Batch Normalization (BN) [92] is known as a crucial component to help deep networks achieve state-of-the-arts performance on various visual tasks, it may become a major obstacle for securing robustness against strong attacks in the context of adversarial training. By benchmarking different strategies for boosting robustness, we observe an unexpected phenomenon—removing clean images from training data is the most effective one. We

relate this phenomenon to the conjecture that clean images and adversarial images are drawn from two different domains. This two-domain hypothesis may explain the limitation of BN when training with a mixture of clean and adversarial images, as estimating normalization statistics on this mixture distribution is challenging. We further show that adversarial training without removing clean images can also obtain strong robustness, if the mixture distribution is disentangled at BN by estimating normalization statistics separately on clean images and adversarial images, *i.e.*, one set of BN exclusively runs on adversarial images and another set of BN exclusively runs on clean images. An alternative solution to avoiding mixture distribution for normalization is to simply replace all BNs with batch-unrelated normalization layers, *e.g.*, group normalization [219], where normalization statistics are estimated on each image independently. These facts indicate that model robustness is highly related to normalization in adversarial training. Furthermore, additional performance gain is observed via enforcing consistent behavior of BN during training and testing.

Second, we find that our so-called "deep" networks are still shallow for the task of adversarial learning, and simply going deeper can effectively boost model robustness. Experiments show that directly adding more layers to "deep" networks only marginally improves accuracy for standard training, but substantial and consistent robustness improvements are observed in adversarial training, even if we push the network capacity to an unprecedented scale, *i.e.*, ResNet-638. This phenomenon suggests that larger networks are demanded for the task of adversarial learning, as the learning target, *i.e.*, adversarial images, lie in a much more complex distribution than clean images.

In summary, this chapter reveals two intriguing properties of adversarial training: (1) properly handling normalization is essential for obtaining models with strong robustness; and (2) our so-called "deep" networks are still shallow for the task of adversarial learning. We hope these findings can benefit future research on understanding adversarial training and improving adversarial robustness.

6.2 Related Work

Adversarial training. Adversarial training constitutes the current foundation of state-of-the-arts for defending against adversarial attacks. It is first developed in [64] where both clean images and adversarial images are used for training. [96] propose to improve robustness further by encouraging the logits from the pairs of clean images and adversarial counterparts to be similar. Instead of using both clean and adversarial images for training, [128] formulate adversarial training as a min-max optimization and train models exclusively on adversarial images. Subsequent works are then proposed to further improve the model robustness [21], [76], [77], [161], [202], [228], [238], [241], [243] or accelerate the adversarial training process [177], [208], [239]. However, as these works mainly focus on demonstrating the effectiveness of their own proposed mechanisms, a fair and detailed diagnosis of large-scale adversarial training strategies[1] remains as a missing piece. In this chapter, we provide the first detailed diagnosis which reveals two intriguing properties of training adversarial defenders at scale.

Normalization Layers. Normalization is an effective technique to accelerate and regularize the training of deep networks. Different methods are proposed to exploit batch-wise (*e.g.*, BN [92]), layer-wise (*e.g.*, layer normalization [6]) or channel-wise (*e.g.*, instance normalization [203] and group normalization [219]) information for estimating normalization statistics. Different from traditional visual tasks where BN usually yields much stronger performance than other normalization methods, we show that BN may become a major obstacle for achieving strong robustness in the context of adversarial training, and properly handling normalization is an essential factor to improve adversarial robustness.

[1] Though there are no prior works on diagnosing large-scale adversarial training, we notice there are several recent works on performing detailed diagnoses of adversarial training on small datasets like CIFAR-10 [65], [152], [169]. We refer interested readers to these works for details.

6.3 Adversarial Training Framework

As inconsistent adversarial training pipelines were applied in previous works [96], [228], it is hard to identify which elements are important for obtaining robust models. To this end, we provide a unified framework to train and to evaluate different models, for the sake of fair comparison.

Training Parameters. We use the publicly available adversarial training pipeline[2] to train *all* models with different strategies on ImageNet. We select ResNet-152 [72] as the baseline network, and apply projected gradient descent (PGD) [128] as the adversarial attacker to generate adversarial examples during training. The hyper-parameters of the PGD attacker are: maximum perturbation of each pixel $\epsilon = 16$, attack step size $\alpha = 1$, number of attack iterations N = 30, and the targeted class is selected uniformly at random over the 1000 ImageNet categories. We initialize the adversarial image by the clean counterpart with probability = 0.2, or randomly within the allowed ϵ cube with probability = 0.8. All models are trained for a total of 110 epochs, and we decrease the learning rate by 10× at the 35-th, 70-th, and 95-th epoch.

Evaluation. For performance evaluation, we mainly study *adversarial robustness* (rather than clean image accuracy) in this chapter. Specifically, we follow the setting in [96], [228], where the targeted PGD attacker is chosen as the white-box attacker to evaluate robustness. The targeted class is selected uniformly at random. We constrain the maximum perturbation of each pixel $\epsilon = 16$, set the attack step size $\alpha = 1$, and measure the robustness by defending against PGD attacker of 2000 attack iterations (*i.e.*, PGD-2000). As in [96], [228], we always initialize the adversarial perturbation from a random point within the allowed ϵ-cube.

In the experiments, we apply these training and evaluation settings to all models by default, unless otherwise stated.

[2] https://github.com/facebookresearch/ImageNet-Adversarial-Training

6.4 Exploring Normalization Techniques in Adversarial Training

6.4.1 On the Effects of Clean Images in Adversarial Training

We first elaborate on the effectiveness of different adversarial training strategies on model robustness. Adversarial training can be dated back to [64], where a mixture of clean images and the corresponding adversarial counterparts are used for training. We choose this strategy as our starting point, which has the following loss function:

$$\hat{J}(\theta, x, y) = \alpha J(\theta, x^{clean}, y) + (1 - \alpha) J(\theta, x^{adv}, y), \tag{6.1}$$

where $J(\cdot)$ is the loss function, θ is the network parameter, y is the ground-truth, and training pairs $\{x^{clean}, x^{adv}\}$ are comprised of clean images and their adversarial counterparts, respectively. The parameter α balances the relative importance between clean image loss and adversarial image loss. We set $\alpha = 0.5$ following [64]. With our adversarial training framework, this model can achieve 20.9% accuracy against PGD-2000 attacker. Besides this baseline, we also study the effectiveness of two recently proposed strategies [96], [128], and provide the results in the following.

Ratio of clean images. Different from the canonical form in [64], Madry *et al.* [128] apply the min-max formulation for adversarial training where no clean images are used. We note this min-max type optimization can be dated as early as [205]. We hereby investigate the relationship between model robustness and the ratio of clean images used for training. Specifically, for each training mini-batch, we keep adversarial images unchanged, but removing their clean counterparts by 20%, 40%, 60%, 80% and 100%. We report the results in Figure 6.1. Interestingly, removing a portion of clean images from training data can significantly improve model robustness, and the strongest robustness can be obtained by completely removing clean images from the training set, *i.e.*, it achieves an accuracy of 39.2% against PGD-2000 attacker, outperforming the baseline model by a large margin of 18.3%.

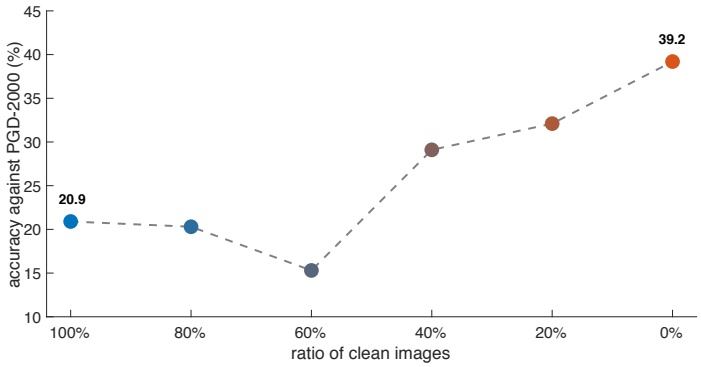


Figure 6.1. The relationship between model robustness and the portion of clean images used for training. We observe that the strongest robustness can be obtained by training completely without clean images, surpassing the baseline model by 18.3% accuracy against PGD-2000 attacker.

Adversarial logits pairing. For performance comparison, we also explore the effectiveness of an alternative training strategy, adversarial logits pairing (ALP) [96]. Compared with the canonical form in [64], ALP imposes an additional loss to encourage the logits from the pairs of clean images and adversarial counterparts to be similar. As shown in Figure 6.2, our re-implemented ALP obtains an accuracy of 23.0% against PGD-2000 attacker[3], which outperforms the baseline model by 2.1%. Compared with the strategy of removing clean images, this improvement is much smaller.

Discussion. Given the results above, we conclude that training exclusively on adversarial images is the most effective strategy for boosting model robustness. For example, by defending against PGD-2000 attacker, the baseline strategy in [64] (referred to as *100% adv + 100% clean*) obtains an accuracy of 20.9%. Adding an loss of logits pairing [96] (referred to as *100% adv + 100% clean, ALP*) slightly improves the performance by 2.1%, while completely removing clean images [128], [228] (referred to as *100% adv + 0% clean*) boosts the accuracy by 18.3%. We further plot a comprehensive evaluation curve of these three training strategies in Figure 6.2, by varying the number

[3]Surprisingly, we note our reproduced ALP result is significantly stronger than the result reported in the original ALP paper [96], as well in an independent study [50]. We identify this performance gap is mainly due to different settings of training parameter, and provide a detailed diagnosis in the supplementary material.

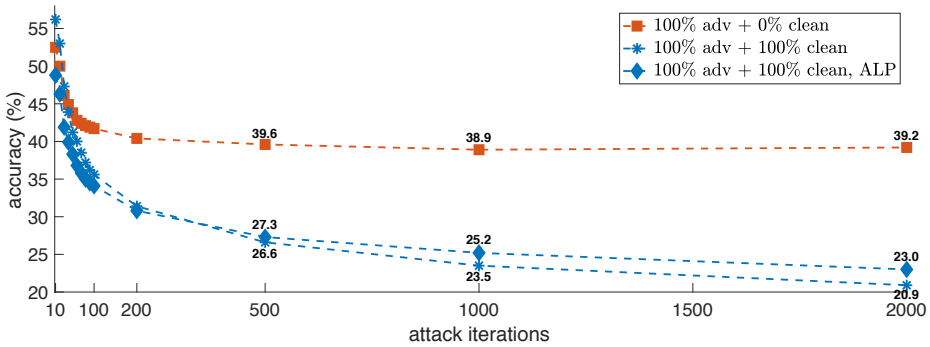


Figure 6.2. Comprehensive robust evaluation on ImageNet. For models trained with different strategies, we show their accuracy against PGD attackers with 10 to 2000 iterations. Only the curve of *100% adv + 0% clean* becomes asymptotic when evaluating against attackers with more iterations.

of PGD attack iteration from 10 to 2000. Surprisingly, only *100% adv + 0% clean* can ensure model robustness against strong attacks, *i.e.*, performance becomes asymptotic when allowing PGD attacker to perform more attack iterations. Training strategies which involve clean images for training are suspicious to result in worse robustness, if PGD attackers are allowed to perform more attack iterations. In the next section, we will study how to make these training strategies, *i.e.*, *100% adv + 100% clean* and *100% adv + 100% clean, ALP* to secure their robustness against strong attacks.

6.4.2 The Devil is in the Batch Normalization

Two-domain hypothesis. Compared to feature maps of clean images, [228] show that feature maps of their adversarial counterparts tend to be more noisy. Meanwhile, several works [54], [110], [111], [134], [149] demonstrate it is possible to build classifiers to separate adversarial images from clean images. These studies suggest that *clean images and adversarial images are drawn from two different domains*[4]. This two-domain hypothesis may provide an explanation to the unexpected observation (see Sec. 6.4.1) and we ask—why simply removing clean images from training data can largely boost adversarial robustness?

[4]Or more precisely, "natural" images collected in the datasets and the corresponding adversarial images may come from two different distributions.

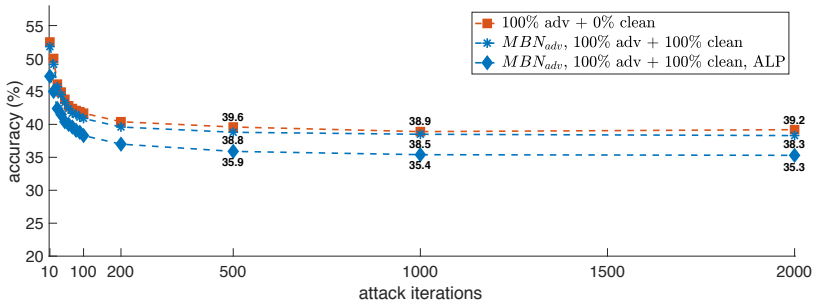


Figure 6.3. Disentangling the mixture distribution for normalization secures model robustness. Unlike the blue curves in Figure 6.2, these new curves become asymptotic when evaluating against attackers with more iterations, which indicate that the networks using MBN_{adv} can behave robustly against PGD attackers with different attack iterations, even if clean images are used for training.

As a crucial element to help deep network achieve state-of-the-art performance on various visual tasks, BN is widely adopted in many network architectures, *e.g.*, Inception [194], ResNet [72] and DenseNet [84]. The normalization statistics of BN are estimated across different images. However, exploiting batch-wise statistics is a challenging task if input images are drawn from different domains, and therefore networks fail to learn a unified representation on this mixture distribution. Given this two-domain hypothesis, when training with both clean and adversarial images, the usage of BN may fail to secure model robustness against strong adversarial attackers.

Based on the analysis above, an intuitive solution arise: *accurately estimating normalization statistics should enable models to train robustly even if clean images and adversarial images are mixed at each training mini-batch.* To this end, we explore two ways, where the mixture distribution is disentangled at normalization layers, for validating this argument: (1) maintaining separate BNs for clean/adversarial images; or (2) replacing BNs with batch-unrelated normalization layers.

Training with Mixture BN. Current networks estimate BN statistics using the mixed features from both clean and adversarial images, which leads to weak model robustness as shown in Figure 6.2. Xie *et al.* [224] show that decoupling the normalization statistics in adversarial training can effectively improve recognition models.

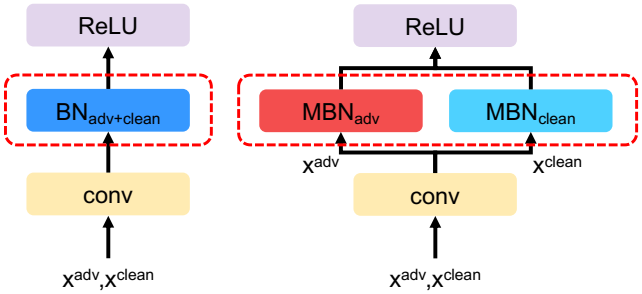


Figure 6.4. Standard BN (left) estimates normalization statistics on the mixture distribution. MBN (right) disentangles the distribution by constructing different mini-batch for clean and adversarial images to estimate normalization statistics.

Here, to study model robustness, we apply this Mixture BN (MBN) design [224] for accurate BN statistics estimation (illustrated in Figure 6.4). Specifically, in MBN, we assign one set of BN to exclusively run on adversarial images (referred to as MBN_{adv}), and another set of BN to exclusively run on clean images (referred to as $\text{MBN}_{\text{clean}}$). We do not change the structure of other layers. We verify the effectiveness of this new architecture with two (previously less robust) training strategies, *i.e.*, *100% adv + 100% clean* and *100% adv + 100% clean, ALP*.

At inference time, whether an image is adversarial or clean is unknown. We thereby measure the performance of networks by applying either MBN_{adv} or $\text{MBN}_{\text{clean}}$ separately. The results are shown in Table 6.1. We find the performance is strongly related to how BN is trained: *when using MBN_{clean}, the trained network achieves nearly the same clean image accuracy as the whole network trained exclusively on clean images; when using MBN_{adv}, the trained network achieves nearly the same adversarial robustness as the whole network trained exclusively on adversarial images.* Other factors, like whether ALP is applied for training, only cause subtle differences in performance. We further plot an extensive robustness evaluation curve of different training strategies in Figure 6.3. Unlike Figure 6.2, we observe that networks using MBN_{adv} now can secure their robustness against strong attacks, *e.g.*, the robustness is asymptotic when increasing the number of attack iteration from 500 to 2000.

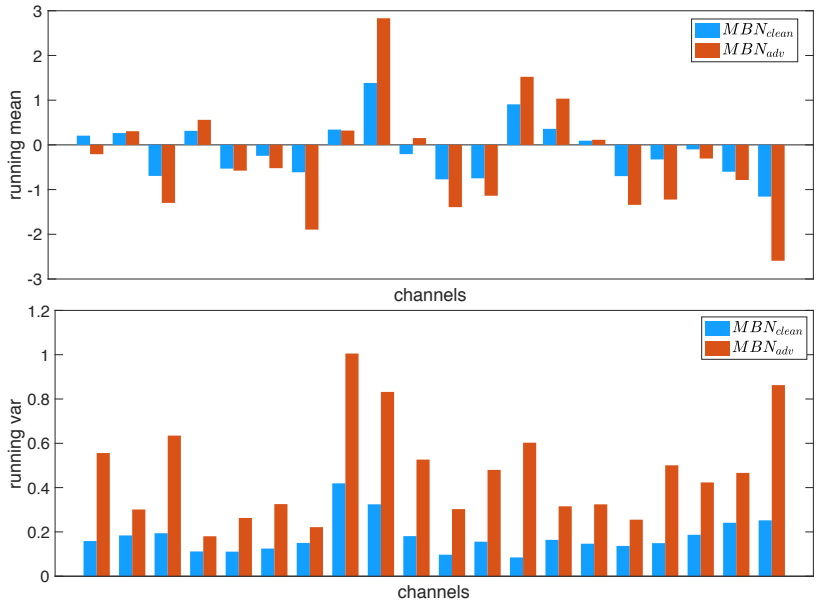


Figure 6.5. Statistics of running mean and running variance of MBN on randomly sampled 20 channels in a ResNet-152's res_3 block. This suggests that clean and adversarial images induce significantly different normalization statistics.

The results in Table 6.1 suggest that BN statistics characterize different model performance. For a better understanding, we randomly sample 20 channels in a residual block and plot the corresponding running statistics of $\text{MBN}_{\text{clean}}$ and MBN_{adv} in Figure 6.5. We observe that clean images and adversarial images induce significantly different running statistics, though these images share the same set of convolutional filters for feature extraction. This observation further supports that (1) clean images and adversarial images come from two different domains; and (2) current networks fail to learn a unified representation on these two domains. Interestingly, we also find that adversarial images lead to larger running mean and variance than clean images. This phenomenon is also consistent with the observation that adversarial images produce noisy-patterns/outliers at the feature space [228].

We note this MBN design is also used as a practical trick for training better GAN [63]. Chintala *et al.* [31] suggest to construct each mini-batch with only real or generated images when training discriminators, as generated images and real images belong to different domains at an early training stage. However, unlike our situation

training strategy	clean image accuracy (%)
0% adv + 100% clean	78.9
$\text{MBN}_{\text{clean}}$, 100% adv + 100% clean	+0.4
$\text{MBN}_{\text{clean}}$, 100% adv + 100% clean, ALP	-0.5

training strategy	PGD-2000 accuracy (%)
100% adv + 0% clean	39.2
MBN_{adv}, 100% adv + 100% clean	-0.9
MBN_{adv}, 100% adv + 100% clean, ALP	-3.9

Table 6.1. MBN statistics characterize model performance. Using $\text{MBN}_{\text{clean}}$/$\text{MBN}_{\text{adv}}$, the trained models achieve strong performance on clean/adversarial images.

where BN statistics estimated on different domains remain divergent after training, a successful training of GAN, *i.e.*, able to generate natural images with high quality, usually learns a unified set of BN statistics on real and generated images.

Training with batch-unrelated normalization layers. Instead of applying this MBN design to disentangle the mixture distribution, we can also train networks with batch-unrelated normalization layers, which avoids exploiting the batch dimension to calculate statistics, for the same purpose. We choose Group Normalization (GN) for this experiment, as GN can reach a comparable performance to BN on various vision tasks [219]. Specifically, for each image, GN divides the channels into groups and computes the normalization statistics within each group. By replacing all BNs with GNs, the mixture training strategy *100% adv + 100% clean* now can ensure robustness against strong attacks, *i.e.*, the model trained with GN achieves 39.5% accuracy against PGD-500, and increasing attack iterations to 2000 only cause a marginal performance drop by 0.5% (39.0% accuracy against PGD-2000). Exploring other batch-unrelated normalization in adversarial training remains as future works.

Exceptional cases. There are some situations where models directly trained with the original BN design can ensure their robustness against strong attacks, even if clean images are included for adversarial training. Our experiments show constraining the maximum perturbation of each pixel ϵ to be a smaller value, *e.g.*, $\epsilon = 8$, is one

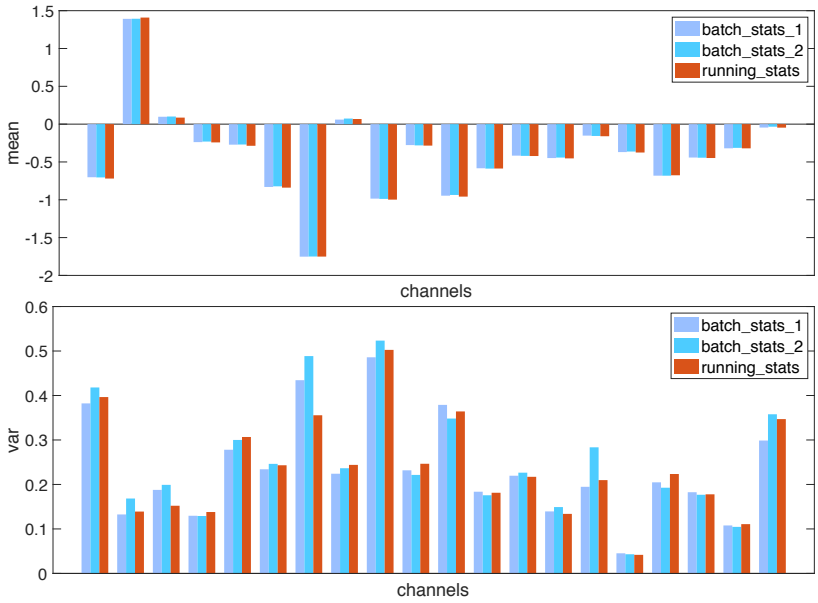


Figure 6.6. Comparison of batch statistics and running statistics of BN on randomly sampled 20 channels in a ResNet-152's res_3 block. We observe that batch mean can converge to running mean, while batch variance still differs from running variance.

of these exceptional cases. [96], [142] also show that adversarial training with clean images can secure robustness on small datasets, *i.e.*, MNIST, CIFAR-10 and Tiny ImageNet. Intuitively, generating adversarial images on these much simpler datasets or under a smaller perturbation constraint induces a smaller gap between these two domains, therefore making it easier for networks to learn a unified representation on clean and adversarial images. Nonetheless, in this chapter, we stick to the standard protocol in [96] and [228] where adversarial robustness is evaluated on ImageNet with the perturbation constraint $\epsilon = 16$.

Inconsistent behavior of BN. As the concept of "batch" is not legitimate at inference time, BN behaves differently at training and testing [92]: during training, the mean and variance are computed on each mini-batch, referred to as *batch statistics*; during testing, there is no actual normalization performed—BN uses the mean and variance pre-computed on the training set (often by running average) to normalize data, referred to as *running statistics*.

training strategy	PGD-2000 accuracy (%)
100% adv + 0% clean	39.2
100% adv + 0% clean*	+3.0
MBN_{adv}, 100% adv + 100% clean	38.3
MBN_{adv}, 100% adv + 100% clean*	+1.6
MBN_{adv}, 100% adv + 100% clean, ALP	35.3
MBN_{adv}, 100% adv + 100% clean, ALP*	+2.8

Table 6.2. Enforcing a consistent behavior of BN at the training stage and the testing stage significantly boosts adversarial robustness. * denotes that running statistics is used at the last 10 training epochs.

For traditional classification tasks, batch statistics usually converge to running statistics by the end of training, thus (practically) making the impact of this inconsistent behavior negligible. Nonetheless, this empirical assumption may not hold in the context of adversarial training. We check this statistics matching of models trained with the strategy *100% adv + 0% clean*, where the robustness against strong attacks is secured. We randomly sample 20 channels in a residual block, and plot the batch statistics computed on two randomly sampled mini-batches, together with the pre-computed running statistics. In Figure 6.6, interestingly, we observe that batch mean is almost equivalent to running mean, while batch variance does not converge to running variance yet on certain channels. Given this fact, we then study if this inconsistent behavior of BN affects model robustness in adversarial training.

6.4.3 Revisiting Statistics Estimation of BN

A heuristic approach. Instead of developing a new training strategy to make batch statistics converge to running statistics by the end of training, we explore a more heuristic solution: applying pre-computed running statistics for model training during the last 10 epochs. We report the performance comparison in Table 6.2. By enabling BNs to behave consistently at training and testing, this approach can further boost the model robustness by 3.0% with the training strategy *100% adv + 0% clean*. We also successfully validate the generality of this approach on other two robust training

strategies. For example, it can improve the model robustness under the training strategies *MBN$_{adv}$, 100% adv + 100% clean* and *MBN$_{adv}$, 100% adv + 100% clean, ALP* by 1.6% and 2.8%, respectively. These results suggest that model robustness can be benefited from a consistent behavior of BN at training and testing. Moreover, this approach comes for "free"—no additional training cost is incurred.

6.4.4 Beyond Adversarial Robustness

On the importance of training convolutional filters adversarially. In Section 6.4.2, we study the model performance when the mixture distribution is disentangled for normalization—by applying either MBN$_{\text{clean}}$ or MBN$_{\text{adv}}$, the trained models achieve strong performance on either clean images or adversarial images. This result suggests that clean and adversarial images share the same convolutional filters to effectively extract features. We hereby further explore whether the filters learned exclusively on adversarial images can extract features effectively on clean images, and vice versa. We first take a model trained with the strategy *100% adv + 0% clean*, and then finetune BNs using only clean images for a few epochs. Interestingly, we find the accuracy on clean images can be significantly boosted from 62.3% to 73%, which is only 5.9% worse than the standard training setting, *i.e.*, 78.9%.

These result indicates that convolutional filters learned exclusively on adversarial images can also be effectively applied to clean images. However, we find the opposite direction does not work—convolutional filters learned on clean images cannot extract features robustly on adversarial images (*e.g.*, 0% accuracy against PGD-2000 after finetuning BNs with adversarial images). This phenomenon indicates the importance of training convolutional filters adversarially, as such learned filters can also extract features from clean images effectively. The findings here also are related to the discussion of robust/non-robustness features in [90]. Readers with interests are recommended to refer to this concurrent work for more details.

Limitation of adversarial training. We note that our adversarially trained models exhibit a performance trade-off between clean accuracy and robustness—the training strategies that achieve strong model robustness usually result in relatively low accuracy on clean images. For example, *100% adv + 0% clean*, MBN_{adv}*, 100% adv + 100% clean* and MBN_{adv}*, 100% adv + 100% clean, ALP* only report 62.3%, 64.4% and 65.9% of clean image accuracy. By replacing BNs with GNs, *100% adv + 100% clean* achieves much better clean image accuracy, *i.e.*, 67.5%, as well maintaining strong robustness. We note that this trade-off is also observed in the prior work [201]. Though this performance trade-off may be inevitable for a given network, the recent work [7] show it is possible to make adversarially trained models to exhibit a better trade-off between clean accuracy and robustness. Future attentions are deserved on exploring this direction further.

6.5 Going Deeper in Adversarial Training

As discussed in Section 6.4.2, current deep networks are not capable of learning a unified representation on clean and adversarial images. It may suggest that the "deep" network we used, *i.e.*, ResNet-152, still underfits the complex distribution of adversarial images, which motivates us to apply larger networks for adversarial training. We simply instantiate the concept of larger networks by going deeper, *i.e.*, adding more residual blocks. For traditional image classification tasks, the benefits brought by adding more layers to "deep" networks is diminishing, *e.g.*, the blue curve in Figure 6.7 shows that the improvement of clean image accuracy becomes saturated once the network depth goes beyond ResNet-200.

For a better illustration, we train deeper models exclusively on adversarial images and observe a possible underfitting phenomenon as shown in Figure 6.7. In particular, we apply the heuristic training strategy in Section 6.4.3 to mitigate the possible (negative) effects brought by BN. We observe that adversarial learning task exhibits

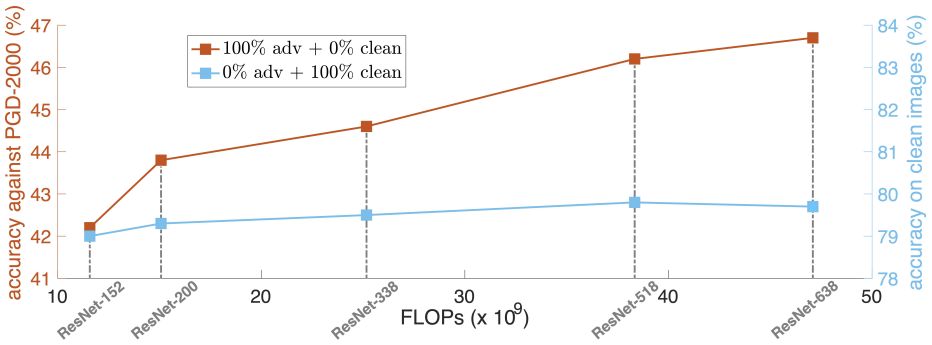


Figure 6.7. Compared to traditional image classification tasks, adversarial training exhibits a stronger demand on deeper networks. The performance gain of traditional image classification becomes marginal after ResNet-200 while the adversarial robustness keeps increasing even for the very deep ResNet-638.

a strong "thirst" on deeper networks to obtain stronger robustness. For example, increasing depth from ResNet-152 to ResNet-338 significantly improves the model robustness by 2.4%, while the corresponding improvement in the "clean" training setting (referred to as *0% adv + 100% clean*) is only 0.5%. Surprisingly, this observation still holds even by pushing the network capacity to an unprecedented scale, *i.e.*, the very deep ResNet with 638 layers (ResNet-638).

Given the empirical results above, we can conclude that our so-called "deep" networks (*e.g.*, ResNet-152) are still shallow for the task of adversarial learning, and larger networks should be used for fitting this complex distribution. Besides our findings on network depth, [128] show increase network width also substantially improve network robustness. These empirical observations also corroborate with the recent theoretical studies [57], [145] which argues that robust adversarial learning needs much more complex classifiers.

Besides adversarial robustness, we observe that these adversarially trained models also exhibit a consistent performance gain on clean image accuracy when increasing the network depth. It is worth to mention that, our deepest network, ResNet-638, achieves an accuracy of 68.7% on clean images, substantially outperforming the relatively shallow network ResNet-152 by 6.1%.

6.6 Summary

In this chapter, we reveal two intriguing properties of adversarial training at scale: (1) conducting normalization in the right manner is essential for training robust models on large-scale datasets like ImageNet; and (2) our so-called "deep" networks are still shallow for the task of adversarial learning. Our discoveries may also be inherently related to our two-domain hypothesis—clean images and adversarial images are drawn from different distributions. We hope these findings can facilitate fellow researchers for better understanding of adversarial training as well as further improvement of adversarial robustness.

Part III

Robust Representation Learning Improves Generalization

Chapter 7

Adversarial Examples Improve Image Recognition

Adversarial examples are commonly viewed as a threat to ConvNets. Here we present an opposite perspective: adversarial examples can be used to **improve image recognition models** if harnessed in the right manner. We propose AdvProp, an enhanced adversarial training scheme which treats adversarial examples as additional examples, to prevent overfitting. Key to our method is the usage of a separate auxiliary batch norm for adversarial examples, as they have different underlying distributions to normal examples.

We show that AdvProp improves a wide range of models on various image recognition tasks and performs better when the models are bigger. For instance, by applying AdvProp to the latest EfficientNet-B7 [197] on ImageNet, we achieve significant improvements on ImageNet (+0.7%), ImageNet-C (+6.5%), ImageNet-A (+7.0%) and Stylized-ImageNet (+4.8%). With an enhanced EfficientNet-B8, our method achieves the state-of-the-art **85.5%** ImageNet top-1 accuracy **without** extra data. This result even surpasses the best model in [129] which is trained with 3.5B Instagram images (~3000× more than ImageNet) and ~9.4× more parameters. Models are available at https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet.

7.1 Introduction

Adversarial examples, which are crafted by adding imperceptible perturbations to images, can lead deep networks to make wrong predictions. The existence of adversarial examples not only reveals the limited generalization ability of deep networks, but also poses security threats on the real-world deployment of these models. Since the first discovery of the vulnerability of deep networks to adversarial attacks [196], many efforts [28], [64], [95], [96], [100], [128], [153], [190], [200], [228], [241] have been made to improve network robustness.

In this chapter, rather than focusing on defending against adversarial examples, we shift our attention to leveraging adversarial examples to improve accuracy. Previous works show that training with adversarial examples can enhance model generalization but are restricted to certain situations—the improvement is only observed either on small datasets (*e.g.*, MNIST) in the fully-supervised setting [64], [108], or on larger datasets but in the semi-supervised setting [139], [160]. Meanwhile, recent works [96], [100], [228] also suggest that training with adversarial examples on large datasets, *e.g.*, ImageNet [173], with supervised learning results in performance degradation on clean images. To summarize, it remains an open question of how adversarial examples can be used effectively to help vision models.

We observe all previous methods jointly train over clean images and adversarial examples without distinction even though they should be drawn from different underlying distributions. We hypothesize this distribution mismatch between clean examples and adversarial examples is a key factor that causes the performance degradation in previous works [96], [100], [228].

Here we propose AdvProp, short for Adversarial Propagation, a new training scheme that bridges the distribution mismatch with a simple yet highly effective two-batchnorm approach. Specifically, we propose to use two batch norm statistics, one for clean images

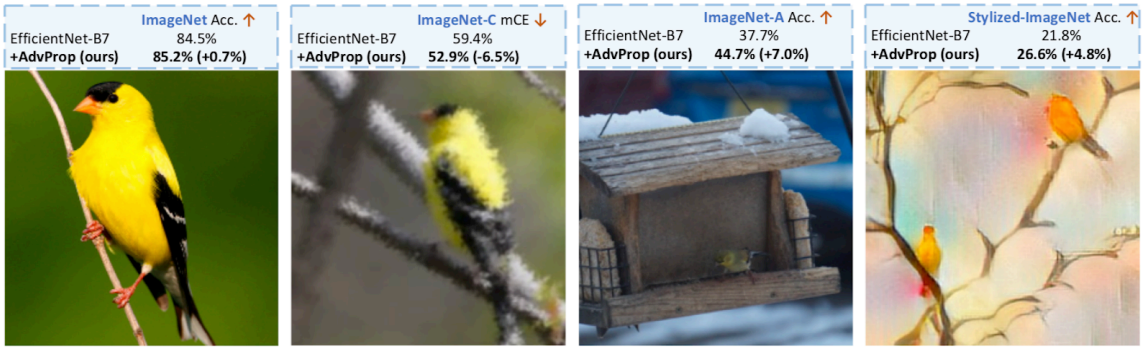


Figure 7.1. AdvProp improves image recognition. By training models on ImageNet, AdvProp helps EfficientNet-B7 [197] to achieve 85.2% accuracy on ImageNet [173], 52.9% mCE (mean corruption error, lower is better) on ImageNet-C [74], 44.7% accuracy on ImageNet-A [78] and 26.6% accuracy on Stylized-ImageNet [59], beating its vanilla counterpart by 0.7%, 6.5%, 7.0% and 4.8%, respectively. Theses sample images are randomly selected from the category "goldfinch".

and one auxiliary for adversarial examples. The two batchnorms properly disentangle the two distributions at normalization layers for accurate statistics estimation. We show this distribution disentangling is crucial, enabling us to successfully improve, rather than degrade, model performance with adversarial examples.

To our best knowledge, our work is the first to show adversarial examples can improve model performance in the fully-supervised setting on the large-scale ImageNet dataset. For example, an EfficientNet-B7 [197] trained with AdvProp achieves 85.2% top-1 accuracy, beating its vanilla counterpart by 0.8%. The improvement by AdvProp is more notable when testing models on distorted images. As shown in Figure 7.1, AdvProp helps EfficientNet-B7 to gain an absolute improvement of 9.0%, 7.0% and 5.0% on ImageNet-C [74], ImageNet-A [78] and Stylized-ImageNet [59], respectively.

As AdvProp effectively prevents overfitting and performs better with larger networks, we develop a larger network, EfficientNet-B8, by following similar compound scaling rules in [197]. With AdvProp, EfficientNet-B8 achieves the state-of-the-art **85.5%** top-1 accuracy on ImageNet **without** extra data. This result even surpasses the best model in [129], which is pretrained on 3.5B extra Instagram images (~3000× more than ImageNet) and requires ~9.4× more parameters than our EfficientNet-B8.

7.2 Related Work

Adversarial Training. Adversarial training, which trains networks with adversarial examples, constitutes the current foundation of state-of-the-arts for defending against adversarial attacks [64], [100], [128], [228]. Although adversarial training significantly improves model robustness, how to improve clean image accuracy with adversarial training is still under-explored. VAT [139] and deep co-training [160] attempt to utilize adversarial examples in semi-supervised settings, but they require enormous extra unlabeled images. Under supervised learning settings, adversarial training is typically considered hurting accuracy on clean images [164], *e.g.*, ~10% drop on CIFAR-10 [128] and ~15% drop on ImageNet [228]. Tsipras *et al.* [201] argue that the performance tradeoff between adversarial robustness and standard accuracy is provably inevitable, and attribute this phenomenon as a consequence of robust classifiers learning fundamentally different feature representations than standard classifiers. Other works try to explain this tradeoff phenomenon from the perspective of the increased sample complexity of adversary [137], [145], [189], the limited amount of training data [21], [144], [175], [202], [238], or network overparameterization [163].

This chapter focuses on standard supervised learning without extra data. Although using similar adversarial training techniques, we stand on an opposite perspective to previous works—we aim to use adversarial examples to improve clean image accuracy.

Benefits of Learning Adversarial Features. Many works corroborate that training with adversarial examples brings additional features to deep networks. For example, compared with clean images, adversarial examples make network representations align better with salient data characteristics and human perception [201]. Moreover, such trained models are much more robust to high frequency noise [236]. Zhang *et al.* [243] further suggest these adversarially learned feature representations are less sensitive to texture distortions and focus more on shape information.

Our AdvProp can be characterized as a training paradigm which fully exploits the complementarity between clean images and their corresponding adversarial examples. The results further suggest that adversarial features are indeed beneficial for recognition models, which agree with the conclusions drawn from these aforementioned studies.

Data augmentation. Data augmentation, which applies a set of label-preserving transformations to images, serves as an important and effective role to prevent networks from overfitting [72], [99], [183]. Besides traditional methods like flipping and cropping, many advanced augmentations are proposed recently, *e.g.*, applying masking out [42] or adding Gaussian noise [123] to regions in images, or mixing up pairs of images and their labels in a convex manner [242]. Recent works also demonstrate it is possible to learn augmentation policies automatically for achieving better performance on image classification [37], [38], [104], [116], [244] and object detection [38], [249].

Our work can be regarded as one type of data augmentation: creating additional training samples by injecting noise. However, nearly all previous attempts, by augmenting either with random noise (*e.g.*, Table 5 in [100] shows the result of training with random normal perturbations) or adversarial noise [96], [100], [200], fail to improve accuracy on clean images.

7.3 A Preliminary Way to Boost Performance

Madry *et al.* [128] formulate adversarial training as a min-max game and train models exclusively on adversarial examples to effectively boost model robustness. However, such trained models usually cannot generalize well to clean images as shown in [128], [228]. We validate this result by training a medium-scale model (EfficientNet-B3) and a large-scale model (EfficientNet-B7) on ImageNet using PGD attacker[1] [128]—both adversarially trained models obtain much lower accuracy on clean images compared

[1]For PGD attacker, we set the maximum perturbation per pixel ϵ=4, the step size α=1 and the number of attack iteration $n = 5$.

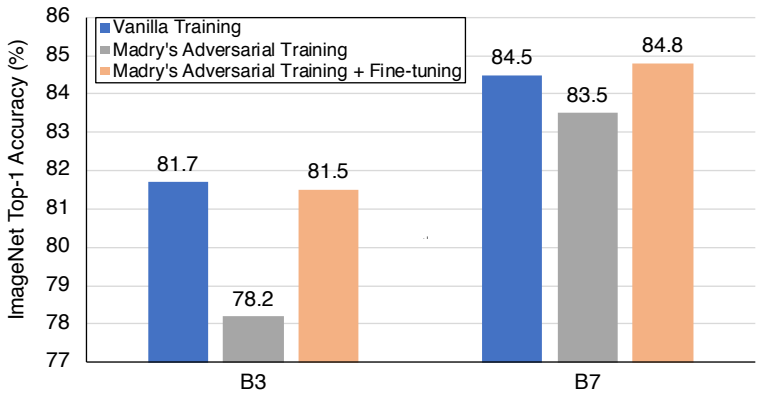


Figure 7.2. Two take-home messages from the experiments on ImageNet: (1) training exclusively on adversarial examples results in performance degradation; and (2) simply training with adversarial examples and clean images in turn can improve network performance on clean images. Fine-tuning details: we train networks with adversarial examples in the first 175 epochs, and then fine-tune with clean images in the rest epochs.

to their vanilla counterparts. For instance, such adversarially trained EfficientNet-B3 only obtains an accuracy of 78.2% on the clean images, whereas vanilla trained EfficientNet-B3 achieves 81.7% (see Figure 7.2).

We hypothesize such performance degradation is mainly caused by *distribution mismatch*—adversarial examples and clean images are drawn from two different domains therefore training exclusively on one domain cannot well transfer to the other. If this distribution mismatch can be properly bridged, then performance degradation on clean images should be mitigated even if adversarial examples are used for training. To validate our hypothesis, we hereby examine a simple strategy—pre-train networks with adversarial examples first, and then fine-tune with clean images.

The results are summarized in Figure 7.2. As expected, this simple fine-tuning strategy (marked in light orange) always yields much higher accuracy than Madry's adversarial training baseline (marked in grey), *e.g.*, it increases accuracy by 3.3% for EfficientNet-B3. Interestingly, while compared to the standard vanilla training setting where only clean images are used (marked in blue), this fine-tuning strategy sometimes even help networks to achieve superior performance, *e.g.*, it increases EfficientNet-B7 accuracy by 0.3%, achieving 84.8% top-1 accuracy on ImageNet.

The observation above delivers a promising signal—adversarial examples can be beneficial for model performance if harnessed properly. Nonetheless, we note that this approach fails to improve performance in general, *e.g.*, though such trained EfficientNet-B3 significantly outperforms the Madry's adversarial training baseline, it is still slightly below (-0.2%) the vanilla training setting. Therefore, a natural question arises: is it possible to distill valuable features from adversarial examples in a more effective manner and boost model performance further generally?

7.4 Approach

The results in Sec. 7.3 suggest that properly integrating information from both adversarial examples and clean images even in a simple manner improves model performance. However, such fine-tuning strategy may partially override features learned from adversarial examples, leading to a sub-optimal solution. To address this issue, we propose a more elegant approach, named AdvProp, to jointly learn from clean images and adversarial examples. Our method handles the issue of distribution mismatch via explicitly decoupling batch statistics on normalization layers, and thus enabling a better absorption from both adversarial and clean features. In this section, we first revisit the adversarial training regime in Sec. 7.4.1, and then introduce how to enable disentangled learning for a mixture of distributions via auxiliary BNs in Sec. 7.4.2. Finally, we summarize the training and testing pipeline in Sec. 7.4.3.

7.4.1 Adversarial Training

We first recall the vanilla training setting, and the objective function is

$$\arg\min_{\theta} \mathbb{E}_{(x,y)\sim\mathbb{D}}\Big[L(\theta, x, y)\Big], \tag{7.1}$$

where $\mathbb{D}$ is the underlying data distribution, $L(\cdot,\cdot,\cdot)$ is the loss function, θ is the network parameter, and x is training sample with ground-truth label y.

Consider Madry's adversarial training framework [128], instead of training with original samples, it trains networks with maliciously perturbed samples,

$$\arg\min_{\theta} \mathbb{E}_{(x,y)\sim\mathbb{D}}\Big[\max_{\epsilon\in\mathbb{S}} L(\theta, x+\epsilon, y)\Big], \tag{7.2}$$

where ϵ is a adversarial perturbation, $\mathbb{S}$ is the allowed perturbation range. Though such trained models have several nice properties as described in [201], [236], [243], they cannot generalize well to clean images [128], [228].

Unlike Madry's adversarial training, our main goal is to improve network performance on clean images by leveraging the regularization power of adversarial examples. Therefore we treat adversarial images as additional training samples and train networks with a mixture of adversarial examples and clean images, as suggested in [64], [100],

$$\arg\min_{\theta} \Big[\mathbb{E}_{(x,y)\sim\mathbb{D}}\Big(L(\theta, x, y) + \max_{\epsilon\in\mathbb{S}} L(\theta, x+\epsilon, y)\Big)\Big]. \tag{7.3}$$

Ideally, such trained models should enjoy the benefits from both adversarial and clean domains. However, as observed in former studies [64], [100], directly optimizing Eqn. (7.3) generally yields lower performance than the vanilla training setting on clean images. We hypothesize that the distribution mismatch between adversarial examples and clean images prevents networks from accurately and effectively distilling valuable features from both domains. Next, we will introduce how to properly disentangle different distributions via our auxiliary batch norm design.

7.4.2 Disentangled Learning via An Auxiliary BN

Batch normalization (BN) [92] serves as an essential component for many state-of-the-art deep networks [72], [84], [195]. Specifically, BN normalizes input features by the mean and variance computed *within each mini-batch*. One intrinsic assumption of utilizing BN is that the input features should come from a single or similar distributions. This normalization behavior could be problematic if the mini-batch contains data from different distributions, therefore resulting in inaccurate statistics estimation.

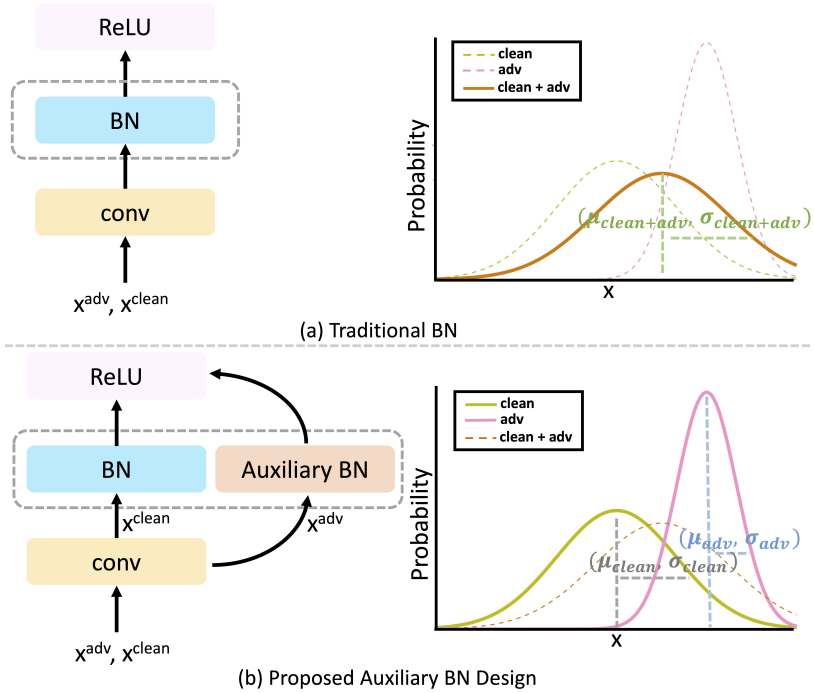


Figure 7.3. Comparison between (a) traditional BN usage and (b) the utilization of auxiliary BN. The left and right panels illustrate the information flow in the corresponding network architectures and the estimated normalization statistics when facing a mixture of adversarial and clean images, respectively.

We argue that adversarial examples and clean images have different underlying distributions, and the adversarial training framework in Eqn. (7.3) essentially involves a two-component mixture distribution. To disentangle this mixture distribution into two simpler ones respectively for the clean and adversarial images, we hereby propose an auxiliary BN to guarantee its normalization statistics are exclusively preformed on the adversarial examples. Specifically, as illustrated in Figure 7.3(b), our proposed auxiliary BN helps to disentangle the mixed distributions by keeping separate BNs to features that belong to different domains. Otherwise, as illustrated in Figure 7.3(a), simply maintaining one set of BN statistics results in incorrect statistics estimation, which could possibly lead to performance degradation.

Note that we can generalize this concept to multiple auxiliary BNs, where the number of auxiliary BNs is determined by the number of training sample sources. For example, if training data contains clean images, distorted images and adversarial

Algorithm 2: Pseudo code of AdvProp

Data: A set of clean images with labels;
Result: Network parameter θ;
for *each training step* **do**
  Sample a clean image mini-batch x^c with label y;
  Generate the corresponding adversarial mini-batch x^a using the auxiliary BNs;
  Compute loss $L^c(\theta, x^c, y)$ on clean mini-batch x^c using the main BNs;
  Compute loss $L^a(\theta, x^a, y)$ on adversarial mini-batch x^a using the auxiliary BNs;
  Minimize the total loss w.r.t. network parameter $\arg\min_\theta L^a(\theta, x^a, y) + L^c(\theta, x^c, y)$.
end
return θ

images, then two auxiliary BNs should be maintained. Ablation studies in Sec. 7.5.4 demonstrates that such fine-grained disentangled learning with multiple BNs can improve performance further. A more general usage of multiple BNs will be further explored in future works, *e.g.*, [80], [94], [114], [133], [181], [206].

7.4.3 AdvProp

We formally propose AdvProp in Algorithm 2 to accurately acquire clean and adversarial features during training. For each clean mini-batch, we first attack the network using the auxiliary BNs to generate its adversarial counterpart; next we feed the clean mini-batch and the adversarial mini-batch to the same network but applied with different BNs for loss calculation, *i.e.*, use the main BNs for the clean mini-batch and use the auxiliary BNs for the adversarial mini-batch; finally we minimize the total loss w.r.t. the network parameter for gradient updates. In other words, except BNs, all other layers (*e.g.*, convolutional layers) are jointly optimized for both adversarial examples and clean images.

Note the introduction of auxiliary BN in AdvProp only increases a negligible amount of extra parameters for network training, *e.g.*, 0.5% more parameters than the baseline on EfficientNet-B7. *At test time, these extra auxiliary BNs are all dropped, and we only use the main BNs for inference.*

Experiments show that such disentangled learning framework enables networks to get much stronger performance than the adversarial training baseline [64], [100]. Besides, compared to the fine-tuning strategy in Sec. 7.3, AdvProp also demonstrates superior performance as it enables networks to jointly learn useful feature from adversarial examples and clean examples at the same time.

7.5 Experiments

7.5.1 Experiments Setup

Architectures. We choose EfficientNets [197] at different computation regimes as our default architectures, ranging from the light-weight EfficientNet-B0 to the large EfficientNet-B7. Compared to other networks, EfficientNet achieves much better accuracy and efficiency. We follow the settings in [197] to train these networks: RMSProp optimizer with decay 0.9 and momentum 0.9; batch norm momentum 0.99; weight decay 1e-5; initial learning rate 0.256 that decays by 0.97 every 2.4 epochs; a fixed AutoAugment policy [37] is applied to augment training images.

Adversarial Attackers. We train networks with a mixture of adversarial examples and clean images as in Eqn. (7.3). We choose Projected Gradient Descent (PGD) [128] under L_∞ norm as the default attacker for generating adversarial examples on-the-fly. We try PGD attackers with different perturbation size ϵ, ranging from 1 to 4. We set the number iteration for the attackers $n{=}\epsilon{+}1$, except for the case $\epsilon{=}1$ where n is set to 1. The attack step size is fixed to $\alpha{=}1$.

Datasets. We use the standard ImageNet dataset [173] to train all models. In addition to reporting performance on the original ImageNet validation set, we go beyond by testing the models on the following test sets:

- **ImageNet-C** [74]. The ImageNet-C dataset is designed for measuring the network robustness to common image corruptions. Specifically, it consists of 15 diverse

corruption types, and each type of corruption has five levels of severity, resulting in 75 distinct corruptions.

- **ImageNet-A** [78]. The ImageNet-A dataset adversarially collects 7,500 natural, unmodified but "hard" real-world images. These images are drawn from some challenging scenarios (*e.g.*, occlusion and fog scene) which are difficult for recognition.
- **Stylized-ImageNet** [59]. The Stylized-ImageNet dataset is created by removing local texture cues while retaining global shape information on natural images via AdaIN style transfer [89]. As suggested in [59], networks are required to learn more shape-based representations to improve accuracy on Stylized-ImageNet.

Compared to ImageNet, images from ImageNet-C, ImageNet-A and Stylized-ImageNet are much more challenging, even for human observers.

7.5.2 ImageNet Results and Beyond

ImageNet Results. Figure 7.4 shows the results on the ImageNet validation set. We compare our method with the vanilla training setting. The family of EfficientNets provides a strong baseline, *e.g.*, EfficientNet-B7's 84.5% top-1 accuracy is the prior art on ImageNet [197].

As different networks favor different attacker strengths when trained with AdvProp (which we ablate next), we first report the best result in Figure 7.4. Our proposed AdvProp substantially outperforms the vanilla training baseline on all networks. This performance improvement is proportional to the network capacity and larger networks tend to perform better if they are trained with AdvProp. For example, the performance gain is *at most* 0.4% for networks smaller than EfficientNet-B4, but is *at least* 0.6% for networks larger than EfficientNet-B4.

Compared to the prior art, *i.e.*, 84.5% top-1 accuracy, an EfficientNet-B6 trained with AdvProp (with ~2× less FLOPs than EfficientNet-B7) already surpasses it by

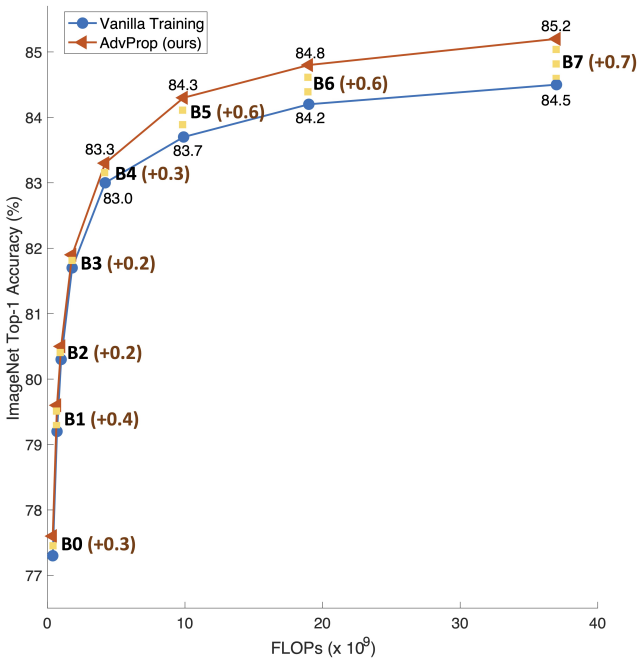


Figure 7.4. AdvProp boosts model performance over the vanilla training baseline on ImageNet. This improvement becomes more significant if trained with larger networks. Our strongest result is reported by the EfficientNet-B7 trained with AdvProp, *i.e.*, 85.2% top-1 accuracy on ImageNet.

0.3%. Our strongest result is obtained by the EfficientNet-B7 trained with AdvProp which achieves 85.2% top-1 accuracy on ImageNet, beating the prior art by 0.7%.

Generalization on Distorted ImageNet Datasets. Next, we evaluate models on distorted ImageNet datasets, which are much more difficult than the original ImageNet dataset. For instance, though ResNet-50 demonstrates reasonable performance on ImageNet (76.7% accuracy), it only achieves 74.8% mCE (mean corruption error, lower is better) on ImageNet-C, 3.1% top-1 accuracy on ImageNet-A and 8.0% top-1 accuracy on Stylized-ImageNet.

The results are summarized in Table 7.1. Again, our proposed AdvProp consistently outperforms the vanilla training baseline for all models on all distorted datasets. The improvement here is much more significant than that on the original ImageNet. For example, AdvProp improves EfficientNet-B3 by 0.2% on ImageNet, and substantially boosts the performance by 5.1% on ImageNet-C and 3.6% on Stylized-ImageNet.

Model	ImageNet-C* [74]	ImageNet-A [78]	Stylized-ImageNet* [59]
	mCE ↓	Top-1 Acc. ↑	Top-1 Acc. ↑
ResNet-50	74.8	3.1	8.0
EfficientNet-B0	70.7	6.7	13.1
+ AdvProp (**ours**)	**66.2 (-4.5)**	**7.1 (+0.4)**	**14.6 (+1.5)**
EfficientNet-B1	65.1	9.0	15.0
+ AdvProp (**ours**)	**60.2 (-4.9)**	**10.1 (+1.1)**	**16.7 (+1.7)**
EfficientNet-B2	64.1	10.8	16.8
+ AdvProp (**ours**)	**61.4 (-2.7)**	**11.8 (+1.0)**	**17.8 (+1.0)**
EfficientNet-B3	62.9	17.9	17.8
+ AdvProp (**ours**)	**57.8 (-5.1)**	**18.0 (+0.1)**	**21.4 (+3.6)**
EfficientNet-B4	60.7	26.4	20.2
+ AdvProp (**ours**)	**58.6 (-2.1)**	**27.9 (+1.5)**	**22.5 (+1.7)**
EfficientNet-B5	62.3	29.4	20.8
+ AdvProp (**ours**)	**56.2 (-6.1)**	**34.4 (+5.0)**	**24.4 (+3.6)**
EfficientNet-B6	60.6	34.5	20.9
+ AdvProp (**ours**)	**53.6 (-7.0)**	**40.6 (+6.1)**	**25.9 (+4.0)**
EfficientNet-B7	59.4	37.7	21.8
+ AdvProp (**ours**)	**52.9 (-6.5)**	**44.7 (+7.0)**	**26.6 (+4.8)**

Table 7.1. AdvProp significantly boost models' generalization ability on ImageNet-C, ImageNet-A and Stylized-ImageNet. The highest result on each dataset is 52.9%, 44.7% and 26.6% respectively, all achieved by the EfficientNet-B7 trained with AdvProp. *For ImageNet-C and Stylized-ImageNet, we follow the previous setup [59], [74] to *always fix the testing image size at the scale of 224×224×3* for a fair comparison.

The EfficientNet-B7 trained with AdvProp reports the strongest results on these datasets, *i.e.*, 52.9% mCE on ImageNet-C, 44.7% top-1 accuracy on ImageNet-A, and 26.6% top-1 accuracy on Stylized-ImageNet. These are the best results so far *if models are not allowed to train with corresponding distortions [59] or extra data [129], [231].*

To summarize, the results suggest that AdvProp significantly boosts the generalization ability by allowing models to learn much richer internal representations than the vanilla training. The richer representations not only provide models with global shape information for better classifying Stylized-ImageNet dataset, but also increase model robustness against common image corruptions.

Ablation on Adversarial Attacker Strength. We now ablate the effects of attacker strength used in AdvProp on network performance. Specifically, the attacker strength here is determined by perturbation size ϵ, where larger perturbation size indicates stronger attacker. We try with different ϵ ranging from 1 to 4, and report the corresponding accuracy on the ImageNet validation set in Table 7.2.

	B0	B1	B2	B3	B4	B5	B6	B7
PGD5 (ϵ=4)	77.1	79.2	80.3	81.8	**83.3**	**84.3**	**84.8**	**85.2**
PGD4 (ϵ=3)	77.3	79.4	80.4	**81.9**	**83.3**	**84.3**	84.7	85.1
PGD3 (ϵ=2)	77.4	79.4	80.4	**81.9**	83.1	**84.3**	84.7	85.0
PGD1 (ϵ=1)	**77.6**	**79.6**	**80.5**	81.8	83.1	**84.3**	84.6	85.0

Table 7.2. ImageNet performance of models trained with AdvProp at different attack strength. In general, smaller networks favor weaker attackers, while larger networks favor stronger attackers.

With AdvProp, we observe smaller networks favor weaker attackers. For example, the light-weight EfficientNet-B0 achieves the best performance by using 1-step PGD attacker with perturbation size 1 (PGD1 (ϵ=1)), significantly outperforms the counterpart which trained with 5-step PGD attacker with perturbation size 4 (PGD5 (ϵ=4)), *i.e.*, 77.6% *vs.* 77.1%. This phenomenon is possibly due to that small networks are limited by their capacity to effectively distill information from strong adversarial examples, even the mixture distributions are well disentangled via auxiliary BNs.

Meanwhile, networks with enough capacity tend to favor stronger attackers. By increasing attacker strength from PGD1 (ϵ=1) to PGD5 (ϵ=4), AdvProp boosts EfficientNet-B7's accuracy by 0.2%. This observation motivate our later ablation on keeping increasing attackers strength to fully exploit the potential of large networks.

7.5.3 Comparisons to Adversarial Training

As shown in Figure 7.4 and Table 7.1, AdvProp improves models for better recognition than the vanilla training baseline. These results contradict previous conclusions [96], [100], [200] that the performance degradation is always observed if adversarial examples are used for training. We hereby provide a set of ablations for explaining this inconsistency. We choose the PGD5 (ϵ=4) as the default attacker to generate adversarial examples during training.

Comparison Results. We compare AdvProp to traditional adversarial training [64], and report evaluation results on ImageNet validation set in Figure 7.5. Compared to

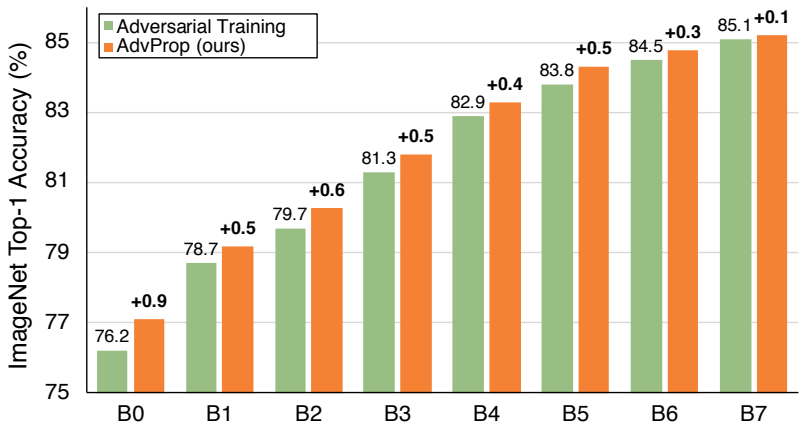


Figure 7.5. AdvProp substantially outperforms adversarial training [64] on ImageNet, especially for small models.

the traditional adversarial training, our method consistently achieves better accuracy on all models. This result suggests that carefully handling BN statistics estimation is important for training better models with adversarial examples.

From Figure 7.5, we note that the biggest improvement is obtained when using EfficientNet-B0, *i.e.*, our proposed AdvProp beats the traditional adversarial training by 0.9%. While by using larger models, this improvement becomes smaller—it stays at ~0.5% until scaling to EfficientNet-B5, and then drops to 0.3% for EfficientNet-B6 and 0.1% for EfficientNet-B7, respectively.

Quantifying Domain Differences. One possible hypothesis for the observation above is that more powerful networks have stronger ability to learn a unified internal representations on the mixed distributions, therefore mitigate the issue of distribution mismatch at normalization layers even without the help of auxiliary BNs. To support this hypothesis, we take models trained with AdvProp, and compare the performance difference between the settings that use either the main BNs or the auxiliary BNs. As such resulted networks share all other layers except BNs, the corresponding performance gap empirically captures the degree of distribution mismatch between adversarial examples and clean images. We use ImageNet validation set for evaluation, and summarize the results in Table 7.3.

	B0	B1	B2	B3	B4	B5	B6	B7
BN	77.1	79.2	80.3	81.8	83.3	84.3	84.8	85.2
Auxiliary BN	73.7	75.9	77.0	78.6	80.5	82.1	82.7	83.3
$\triangle$	+3.4	+3.3	+3.3	+3.2	+2.8	+2.2	+2.1	+1.9

Table 7.3. Performance comparison between settings that use either the main BNs and auxiliary BNs on ImageNet. This performance difference captures the degree of distribution mismatch between adversarial examples and clean images.

By training with larger networks, we observe this performance difference gets smaller. Such gap for EfficientNet-B0 is 3.4%, but then is reduced to 1.9% for EfficientNet-B7. It suggests that the internal representations of adversarial examples and clean images learned on large networks are much more similar than that learned on small networks. Therefore, with a strong enough network, it is possible to accurately and effectively learn a mixture of distributions even without a careful handling at normalization layers.

Why AdvProp? For small networks, our comparison shows that AdvProp substantially outperforms the adversarial training baseline. We attribute this performance improvement mainly to the successful disentangled learning via auxiliary BNs.

For larger networks, though the improvement is relatively small on ImageNet, AdvProp consistently outperforms the adversarial training baseline by a large margin on distorted ImageNet datasets. As shown in Table 7.4, AdvProp improves EfficientNet-B7 by 3.1% on ImageNet-C, 4.3% on ImageNet-A and 1.5% on Stylized-ImageNet over the adversarial training baseline.

Moreover, *AdvProp enables large networks to perform better if trained with stronger attackers*. For example, by slightly increasing attacker strength from PGD5 (ϵ=4) to PGD7 (ϵ=6), AdvProp further helps EfficientNet-B7 to achieve **85.3%** top-1 accuracy on ImageNet. Conversely, applying such attacker to traditional adversarial training decreases EfficientNet-B7's accuracy to 85.0%, possibly due to a more severe distribution mismatch between adversarial examples and clean images.

Model	ImageNet-C [74]	ImageNet-A [78]	Stylized-ImageNet [59]
	mCE ↓	Top-1 Acc. ↑	Top-1 Acc. ↑
B6 + Adv. Training	55.8	37.0	24.7
B6 + AdvProp (**ours**)	**53.6**	**40.6**	**25.9**
B7 + Adv. Training	56.0	40.4	25.1
B7 + AdvProp (**ours**)	**52.9**	**44.7**	**26.6**

Table 7.4. AdvProp demonstrates much stronger generalization ability on distorted ImageNet datasets (*e.g.*, ImageNet-C) than the adversarial training baseline for large models (*e.g.*, EfficientNet-B6 and EfficientNet-B7).

In summary, our proposed AdvProp enables networks to enjoy the benefits of adversarial examples even with limited capacity. For networks with large enough capacity, compared to adversarial training, AdvProp still demonstrates much stronger generalization ability and better at exploiting model capacity for improving models' recognition performance further.

Missing Pieces in Traditional Adversarial Training. In our reproduced adversarial training, we note that it is already better than the vanilla training setting on large EfficientNet models. For example, our adversarially trained EfficientNet-B7 achieves 85.1% top-1 accuracy on ImageNet, which beats the vanilla training baseline by 0.6%. However, interestingly, previous works [96], [100] show adversarial training *always* degrades model performance.

Compared to [96], [100], we make two changes in our re-implementation: (1) using stronger networks; and (2) training with weaker attackers. For examples, previous works use networks like Inception [194] or ResNet [72] for training, and set the perturbation size ϵ=16; while we use much stronger EfficientNet for training, and limit the perturbation size to a much smaller value ϵ=4. Intuitively, weaker attackers push the distribution of adversarial examples less away from the distribution of clean images, and larger networks are better at bridging domain differences. Both factors mitigate the issue of distribution mismatch, thus making networks much easier to learn valuable feature from both domains.

	B0	B1	B2	B3	B4	B5	B6	B7
AdvProp	77.6	79.6	80.5	81.9	83.3	84.3	**84.8**	**85.2**
Fine-Grained AdvProp	**77.9**	**79.8**	**80.7**	**82.0**	**83.5**	**84.4**	**84.8**	**85.2**

Table 7.5. Fine-grained AdvProp substantially boosts model accuracy on ImageNet, especially for small models. We perform fine-grained disentangled learning by keeping an additional auxiliary BN for AutoAugment images.

7.5.4 Ablations

Fine-grained Disentangled Learning via Multiple Auxiliary BNs. Following [197], our networks are trained with AutoAugment [37] by default, which include operations like rotation and shearing. We hypothesize these operations (slightly) shift the original data distribution and propose to add an extra auxiliary BN to disentangle these augmented data further for fine-grained learning. In total, we keep one main BN for clean images *without* AutoAugment, and two auxiliary BNs for clean images *with* AutoAugment and adversarial examples, respectively.

We try PGD attackers with perturbation size ranging from 1 to 4, and report the best result on ImageNet in Table 7.5. Compared to the default version, this fine grained AdvProp improves performance further. It helps EfficientNet-B0 to achieve **77.9%** accuracy with just 5.3M parameters, which is the state-of-the-art performance for mobile networks. As a comparison, MobileNetv3 has 5.4M parameters with 75.2% accuracy [82]. These results encourage the future investigation on more fine-grained disentangled learning in general, not just for adversarial training.

Comparison to AutoAugment. Training with adversarial examples is a form of data augmentation. We choose the standard Inception-style pre-processing [194] as baseline, and compare the benefits of additionally applying AutoAugment or AdvProp. We train networks with PGD5 (ϵ=4) and evaluate performance on ImageNet.

Results are summarized in Table 7.6. For small models, AutoAugment is slightly better than AdvProp although we argue this gap can be bridged by adjusting the

	B0	B1	B2	B3	B4	B5	B6	B7
Inception Pre-process [194]	76.8	78.8	79.8	81.0	82.6	83.2	83.7	84.0
+ AutoAugment [37]	+0.5	+0.4	+0.5	+0.7	+0.4	+0.5	+0.5	+0.5
+ AdvProp (**ours**)	+0.3	+0.3	+0.2	+0.4	+0.3	+0.8	+0.9	+0.9
+ **Both** (**ours**)	+0.3	+0.4	+0.5	+0.8	+0.7	+1.1	+1.1	+1.2

Table 7.6. Both AutoAugment and AdvProp improves model performance over the Inception-style pre-processing baseline on ImageNet. Large Models generally perform better with AdvProp than AutoAugment. Training with a combination of both is better than using AdvProp alone on all networks.

attacker strength. For large models, AdvProp significantly outperforms AutoAugment. Note that training with both strategies is better than using AdvProp alone.

Attackers Other Than PGD. We hereby study the effects of applying different attackers in AdvProp on model performance. Specifically, we try two different modifications on PGD: (1) we no longer limit the perturbation size to be within the ϵ-ball, and name this attacker to Gradient Descent (GD) as it removes the projection step in PGD; or (2) we skip the random noise initialization step in PGD, turn it to I-FGSM [100]. Other attack hyper-parameters are unchanged: the maximum perturbation size ϵ=4 (if applicable), number of attack iteration n=5 and attack step size α=1.0.

For simplicity, we only experiment with EfficientNet-B3, EfficientNet-B5 and EfficientNet-B7, and report the ImageNet performance in Table 7.7. We observe that all attackers substantially improve model performance over the vanilla training baseline. This result suggests that our AdvProp is not designed for a specific attacker (*e.g.*, PGD), but a general mechanism for improving image recognition models with different adversarial attacker.

	B3	B5	B7
Vanilla Training	81.7	83.7	84.5
PGD [128]	81.8	84.3	85.2
I-FGSM [100]	81.9	84.3	85.2
GD	81.7	84.3	85.3

Table 7.7. ImageNet performance when trained with different attackers. With AdvProp, all attackers successfully improve model performance over the vanilla training baseline.

	ResNet-50	ResNet-101	ResNet-152	ResNet-200
Vanilla Training	76.7	78.3	79.0	79.3
Adversarial Training	-3.2	-1.8	-2.0	-1.4
AdvProp (**ours**)	**+0.4**	**+0.6**	**+0.8**	**+0.8**

Table 7.8. Performance comparison among vanilla training, adversarial training and AdvProp on ImageNet. AdvProp reports the best result on all ResNet models.

ResNet Results. We additionally experiment with ResNet. We compare AdvProp against two baselines: vanilla training and adversarial training. We apply PGD5 (ϵ=4) to generate adversarial examples, and follow the settings in [72] to train all networks.

We report model performance on ImageNet in Table 7.8. Compared to vanilla training, adversarial training always degrades model performance while AdvProp consistently leads to better accuracy on all ResNet models. Take ResNet-152 for example, adversarial training *decreases* the baseline performance by 2.0%, but our AdvProp further *boosts* the baseline performance by 0.8%.

In Sec. 7.5.3, we show that adversarial training can improve performance if large EfficientNets are used for training. However, this phenomenon is not observed on ResNet, *e.g.*, adversarial training still leads to inferior accuracy even trained with the large ResNet-200. It may suggest that architecture design also plays an important role when training with adversarial example, and we leave it as a future work.

Pushing The Envelope with a Larger Model. Previous results suggest AdvProp performs better with larger networks. To push the envelope, we train a larger network, EfficientNet-B8, by scaling up EfficientNet-B7 further according to the compound scaling rule in [197].

Our AdvProp improves the accuracy of EfficientNet-B8 from 84.8% to 85.5%, achieving a new state-of-the-art accuracy on ImageNet without using extra data. This result even surpasses the best model reported in [129], which is pretrained on 3.5B extra Instagram images (~3000× more than ImageNet) and requires ~9.4× more parameters (829M vs. 88M) than our EfficientNet-B8.

7.6 Summary

Previous works commonly view adversarial examples as a threat to deep networks, and suggest training with adversarial examples lead to accuracy drop on clean images. Here we offer a different perspective: to use adversarial examples for improving accuracy of deep networks. As adversarial examples have different underlying distributions to normal examples, we propose to use an auxiliary batch norm for disentangled learning by processing adversarial examples and clean images separately at normalization layers. Our method, AdvProp, significantly improves accuracy of all networks in our experiments. Our best model reports the state-of-the-art 85.5% top-1 accuracy on ImageNet without any extra data.

Chapter 8

Discussion and Conclusion

Deep network are powerful for visual recognition. But meanwhile, many works suggest current deep networks cannot generalize as well as the human perception, *e.g.*, they are sensitive to small input changes, viewpoint variation or occlusion. In this thesis, we focus on adversarial examples, and explore how they can fool deep networks and how deep networks can be robust against them. We additionally demonstrate that adversarial examples can in turn efficiently regularize the training of deep networks for boosting the generalization ability. Nonetheless, it remains an open challenge for deep networks to be as robust/generalizable as the human perception. Towards the goal of building human-level computer vision systems, we plan to explore the following directions in the future.

Towards trustworthy recognition models. The vulnerability of deep networks to adversarial examples raises profound security concerns. Principally, we can build a robust model from the following two aspects:

- *Robust learning algorithm.* Currently, adversarial training stands as the one of the most effective ways for defending against adversarial attacks. But it suffers from several shortcomings, *e.g.*, adversarial training is computationally expensive, and such trained models exhibit a performance tradeoff between clean image accuracy and adversarial robustness. It is possible to improve this

learning paradigm by designing loss objective with adaptive regularization terms, reducing the cost of generating adversarial examples, *etc*. Besides, as suggested in [229], we can explore the possible connections between domain adaption and adversarial robustness, to build an unified learning framework for gaining adversarial robustness.

- *Robust architectures design*. We also seek to build novel neural architectures that have "innate" adversarial robustness. Following [225], [231], we can explore other architectural elements to smooth feature representations for improving model robustness. Meanwhile, as demonstrated in [22], [70], it is promising to apply neural architecture search to discover the design principle of robust neural architecture topologies. Besides, we will also try to theoretically understand how and why robust neural architectures can gain adversarial robustness for "free", which plausibly break the no free lunch theorem in adversarial robustness [201].

Besides focusing on the general methodologies for increasing model robustness, we will also attend to a specific application setting—defending against physical adversarial examples. Physical adversarial examples bring safety and reliability concerns to the deep networks based cyber-physical systems, *e.g.*, autonomous vehicles. These concerns are further aggravated by the fact that current approaches [3], [30], [105], [131], [217], [220] are still limited at handling physical adversarial examples. By noticing physical adversarial attack can be alternatively viewed as a combination of the perturbation attack and the occlusion attack, we can explore the idea of equipping models with both robust feature representations (for adversarial noise) as well as the ability to represent objects via local visual cues in a flexible and adaptive manner (for occlusion). This architecture design would be easily scaled/extended to handle other real-world challenging scenarios (*e.g.*, viewpoint variation), as richer representations (*e.g.*, 2D or 3D geometry) can be encoded as visual cues to obtain more advanced models.

Exploring the benefits of learning with "hard" data. Previous works generally treat adversarial examples as a threat to deep networks. Nonetheless, as demonstrated in [224], adversarial examples can also be beneficial, *i.e.*, they significantly boosts the generalization of deep networks. Another showcase is provided in [114], where cue conflict images [59] are utilized to help deep networks learn better shape-texture representations. Both works suggest that it is promising to explore whether learning with "hard" data, which traditionally is viewed as the blind spot of deep networks, can serve as a general principle for benefiting the representation learning of deep networks. On the one hand, learning with "hard" data usually helps deep networks learn valuable features that cannot be obtained from traditional data augmentation methods, *e.g.*, adversarial examples can help models learn robust features in a much more effective and efficient way than other data augmentation methods. On the other hand, augmenting training set with "hard" data sometimes is a more general way to perform data augmentation. For example, flipping operation is only applicable to image-based data, while adversarial examples almost exist for every tasks (including vision, NLP, reinforcement learning) and can be used to augment data universally.

Another direction worth exploring is that we can use "hard" data to help the model interpretability. Prior studies [90], [201] show that adversarial examples can help deep network learn more human alignable feature representations. This conclusion may be extendable to the setting of learning with other "hard" data.

Summary Studying robustness is an important research topic for the computer vision community, not only for the purpose of building a trustworthy recognition system, but also for deeper understandings of our "black-box" models. This thesis has summarized a few interesting aspects and promising directions to go, which we hope will shed lights on future research.

References

[1] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: A query-efficient black-box adversarial attack via random search," in *ECCV*, 2020.

[2] A. Arnab, O. Miksik, and P. H. Torr, "On the robustness of semantic segmentation models to adversarial attacks," in *CVPR*, 2018.

[3] M. Arvinte, A. Tewfik, and S. Vishwanath, "Detecting patch adversarial attacks with image residuals," *arXiv preprint arXiv:2002.12504*, 2020.

[4] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *ICML*, 2018.

[5] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *ICML*, 2018.

[6] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv:1607.06450*, 2016.

[7] Y. Balaji, T. Goldstein, and J. Hoffman, "Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets," *arXiv preprint arXiv:1910.08051*, 2019.

[8] S. Baluja and I. Fischer, "Learning to attack: Adversarial transformation networks," in *AAAI*, 2018.

[9] J. T. Barron, "Continuously differentiable exponential linear units," *arXiv preprint arXiv:1704.07483*, 2017.

[10] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, 2008.

[11] A. N. Bhagoji, D. Cullina, C. Sitawarin, and P. Mittal, "Enhancing robustness of machine learning systems via data transformations," in *CISS*, 2018.

[12] A. N. Bhagoji, W. He, B. Li, and D. Song, "Practical black-box attacks on deep neural networks using efficient query mechanisms," in *ECCV*, 2018.

[13] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *ECML-PKDD*, 2013.

[14] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," in *ICLR*, 2018.

[15] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," in *ICLR*, 2018.

[16] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *CVPR*, 2005.

[17] J. Buckman, A. Roy, C. Raffel, and I. Goodfellow, "Thermometer encoding: One hot way to resist adversarial examples," in *ICLR*, 2018.

[18] J. Canny, "A computational approach to edge detection," *TPAMI*, 1986.

[19] X. Cao and N. Z. Gong, "Mitigating evasion attacks to deep neural networks via region-based classification," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, 2017.

[20] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *SP*, 2017.

[21] Y. Carmon, A. Raghunathan, L. Schmidt, J. C. Duchi, and P. S. Liang, "Unlabeled data improves adversarial robustness," in *NeurIPS*, 2019.

[22] H. Chen, B. Zhang, S. Xue, X. Gong, H. Liu, R. Ji, and D. Doermann, "Anti-bandit neural architecture search for model defense," *arXiv preprint arXiv:2008.00698*, 2020.

[23] H. Chen, H. Zhang, P.-Y. Chen, J. Yi, and C.-J. Hsieh, "Attacking visual language grounding with adversarial examples: A case study on neural image captioning," in *ACL*, 2018.

[24] 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," *TPAMI*, 2017.

[25] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *ACM Workshop on Artificial Intelligence and Security*, 2017.

[26] S.-T. Chen, C. Cornelius, J. Martin, and D. H. P. Chau, "Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector," in *ECML-PKDD*, 2018.

[27] M. Cheng, T. Le, P.-Y. Chen, H. Zhang, J. Yi, and C.-J. Hsieh, "Query-efficient hard-label black-box attack: An optimization-based approach," in *ICLR*, 2018.

[28] M. Cheng, Q. Lei, P.-Y. Chen, I. Dhillon, and C.-J. Hsieh, "Cat: Customized adversarial training for improved robustness," *arXiv preprint arXiv:2002.06789*, 2020.

[29] M. Cheng, J. Yi, P.-Y. Chen, H. Zhang, and C.-J. Hsieh, "Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples," in *AAAI*, 2020.

[30] P.-Y. Chiang*, R. Ni*, A. Abdelkader, C. Zhu, C. Studor, and T. Goldstein, "Certified defenses for adversarial patches," in *ICLR*, 2020.

[31] S. Chintala, E. Denton, M. Arjovsky, and M. Mathieu, *How to train a gan? tips and tricks to make gans work*, https://github.com/soumith/ganhacks, 2016.

[32] M. Cisse, Y. Adi, N. Neverova, and J. Keshet, "Houdini: Fooling deep structured visual and speech recognition models with adversarial examples," in *NIPS*, 2017.

[33] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," in *ICLR*, 2016.

[34] 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*, 2016.

[35] C. Cosgrove and A. Yuille, "Adversarial examples for edge detection: They exist, and they transfer," in *WACV*, 2020.

[36] F. Croce, M. Andriushchenko, N. D. Singh, N. Flammarion, and M. Hein, "Sparse-rs: A versatile framework for query-efficient sparse black-box adversarial attacks," *arXiv preprint arXiv:2006.12834*, 2020.

[37] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," in *CVPR*, 2019.

[38] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical data augmentation with no separate search," in *CVPR*, 2020.

[39] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *NIPS*, 2016.

[40] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.

[41] N. Dalvi, P. Domingos, S. Sanghai, D. Verma, *et al.*, "Adversarial classification," in *SIGKDD*, 2004.

[42] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[43] G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, and A. Anandkumar, "Stochastic activation pruning for robust adversarial defense," in *ICLR*, 2018.

[44] G. W. Ding, Y. Sharma, K. Y. C. Lui, and R. Huang, "Max-margin adversarial (mma) training: Direct input space margin maximization through adversarial training," in *ICLR*, 2020.

[45] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *ICML*, 2014.

[46] Y. Dong, F. Liao, T. Pang, H. Su, X. Hu, J. Li, and J. Zhu, "Boosting adversarial attacks with momentum," in *CVPR*, 2018.

[47] Y. Dong, T. Pang, H. Su, and J. Zhu, "Evading defenses to transferable adversarial examples by translation-invariant attacks," in *CVPR*, 2019.

[48] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *SIGGRAPH*, 2001.

[49] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Networks*, 2018.

[50] L. Engstrom, A. Ilyas, and A. Athalye, "Evaluating and understanding the robustness of adversarial logit pairing," in *NeurIPS Workshop on SECML*, 2018.

[51] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *IJCV*, 2010.

[52] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramer, A. Prakash, T. Kohno, and D. Song, "Physical adversarial examples for object detectors," in *USENIX WOOT*, 2018.

[53] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *CVPR*, 2018.

[54] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *arXiv preprint arXiv:1703.00410*, 2017.

[55] V. Fischer, M. C. Kumar, J. H. Metzen, and T. Brox, "Adversarial examples for semantic image segmentation," in *ICLR Workshop*, 2017.

[56] A. Galloway, A. Golubeva, T. Tanay, M. Moussa, and G. W. Taylor, "Batch normalization is a cause of adversarial vulnerability," *arXiv preprint arXiv:1905.02161*, 2019.

[57] R. Gao, T. Cai, H. Li, L. Wang, C.-J. Hsieh, and J. D. Lee, "Convergence of adversarial training in overparametrized networks," in *NeurIPS*, 2019.

[58] R. Geirhos, J. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, "Shortcut learning in deep neural networks," *arXiv preprint arXiv:2004.07780*, 2020.

[59] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness," in *ICLR*, 2018.

[60] R. Girshick, "Fast R-CNN," in *ICCV*, 2015.

[61] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.

[62] Z. Gong, W. Wang, and W.-S. Ku, "Adversarial and clean data are not twins," *arXiv preprint arXiv:1704.04960*, 2017.

[63] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014.

[64] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *ICLR*, 2015.

[65] S. Gowal, C. Qin, J. Uesato, T. Mann, and P. Kohli, "Uncovering the limits of adversarial training against norm-bounded adversarial examples," *arXiv preprint arXiv:2010.03593*, 2020.

[66] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch SGD: Training ImageNet in 1 hour," *arXiv:1706.02677*, 2017.

[67] C. Guo, J. S. Frank, and K. Q. Weinberger, "Low frequency adversarial perturbation," in *UAI*, 2018.

[68] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger, "Simple black-box adversarial attacks," in *ICML*, 2019.

[69] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," in *ICLR*, 2018.

[70] M. Guo, Y. Yang, R. Xu, Z. Liu, and D. Lin, "When nas meets robustness: In search of robust architectures against adversarial attacks," in *CVPR*, 2020.

[71] R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature*, 2000.

[72] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[73] J. Hendrik Metzen, M. Chaithanya Kumar, T. Brox, and V. Fischer, "Universal adversarial perturbations against semantic image segmentation," in *ICCV*, 2017.

[74] D. Hendrycks and T. G. Dietterich, "Benchmarking neural network robustness to common corruptions and surface variations," *ICLR*, 2019.

[75] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[76] D. Hendrycks, K. Lee, and M. Mazeika, "Using pre-training can improve model robustness and uncertainty," *ICML*, 2019.

[77] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, "Using self-supervised learning can improve model robustness and uncertainty," in *Advances in Neural Information Processing Systems*, 2019.

[78] D. Hendrycks, K. Zhao, S. Basart, J. Steinhardt, and D. Song, "Natural adversarial examples," *arXiv preprint arXiv:1907.07174*, 2019.

[79] C.-H. Ho, B. Leung, E. Sandstrom, Y. Chang, and N. Vasconcelos, "Catastrophic child's play: Easy to perform, hard to defend adversarial attacks," in *CVPR*, 2019.

[80] C.-H. Ho and N. Vasconcelos, "Contrastive learning with adversarial examples," in *NeurIPS*, 2020.

[81] E. Hoffer, I. Hubara, and D. Soudry, "Train longer, generalize better: Closing the generalization gap in large batch training of neural networks," in *NIPS*, 2017.

[82] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, *et al.*, "Searching for mobilenetv3," in *ICCV*, 2019.

[83] B. Huang, Y. Wang, and W. Wang, "Model-agnostic adversarial detection by random perturbations," in *IJCAI*, 2019.

[84] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, 2017.

[85] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *ECCV*, 2016.

[86] L. Huang, C. Gao, Y. Zhou, C. Xie, A. Yuille, C. Zou, and N. Liu, "Universal physical camouflage attacks on object detectors," in *CVPR*, 2020.

[87] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar, "Adversarial machine learning," in *ACM Workshop on Security and Artificial Intelligence*, 2011.

[88] Q. Huang, I. Katsman, H. He, Z. Gu, S. Belongie, and S.-N. Lim, "Enhancing adversarial example transferability with an intermediate level attack," in *ICCV*, 2019.

[89] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *ICCV*, 2017.

[90] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," in *NeurIPS*, 2019.

[91] N. Inkawhich, K. J. Liang, B. Wang, M. Inkawhich, L. Carin, and Y. Chen, "Perturbing across the feature hierarchy to improve standard and strict blackbox attack transferability," *arXiv preprint arXiv:2004.14861*, 2020.

[92] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.

[93] M. Javaheripi, M. Samragh, B. D. Rouhani, T. Javidi, and F. Koushanfar, "Curtail: Characterizing and thwarting adversarial deep learning," *TDSC*, 2020.

[94] Z. Jiang, T. Chen, T. Chen, and Z. Wang, "Robust pre-training by adversarial contrastive learning," in *NeurIPS*, 2020.

[95] C. Jin and M. Rinard, "Manifold regularization for adversarial robustness," *arXiv preprint arXiv:2003.04286*, 2020.

[96] H. Kannan, A. Kurakin, and I. Goodfellow, "Adversarial logit pairing," *arXiv preprint arXiv:1803.06373*, 2018.

[97] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *IJCV*, 1988.

[98] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby, "Large scale learning of general visual representations for transfer," *arXiv preprint arXiv:1912.11370*, 2019.

[99] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[100] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *ICLR*, 2017.

[101] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *ICLR Workshop*, 2017.

[102] A. Kurakin, I. Goodfellow, S. Bengio, Y. Dong, F. Liao, M. Liang, T. Pang, J. Zhu, X. Hu, C. Xie, *et al.*, "Adversarial attacks and defences competition," in *The NIPS'17 Competition: Building Intelligent Systems*, Springer, 2018.

[103] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, 1989.

[104] J. Lemley, S. Bazrafkan, and P. Corcoran, "Smart augmentation learning an optimal data augmentation strategy," *IEEE Access*, 2017.

[105] A. Levine and S. Feizi, "(de) randomized smoothing for certifiable defense against patch attacks," *arXiv preprint arXiv:2002.10733*, 2020.

[106] Q. Li, Y. Guo, and H. Chen, "Yet another intermediate-level attack," in *ECCV*, 2020.

[107] X. Li and F. Li, "Adversarial examples detection in deep networks with convolutional filter statistics," in *ICCV*, 2017.

[108] Y. Li, E. X. Fang, H. Xu, and T. Zhao, "Inductive bias of gradient descent based adversarial training on separable data," in *ICLR*, 2020.

[109] Y. Li, L. Li, L. Wang, T. Zhang, and B. Gong, "Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks," in *ICML*, 2019.

[110] Y. Li, N. Wang, J. Liu, and X. Hou, "Demystifying neural style transfer," *arXiv preprint arXiv:1701.01036*, 2017.

[111] Y. Li, L. Xie, Y. Zhang, R. Zhang, Y. Wang, and Q. Tian, "Defending adversarial attacks by correcting logits," *arXiv preprint arXiv:1906.10973*, 2019.

[112] Y. Li, S. Bai, C. Xie, Z. Liao, X. Shen, and A. Yuille, "Regional homogeneity: Towards learning transferable universal adversarial perturbations against defenses," in *ECCV*, 2020.

[113] Y. Li, S. Bai, Y. Zhou, C. Xie, Z. Zhang, and A. Yuille, "Learning transferable adversarial examples via ghost networks," in *AAAI*, 2020.

[114] Y. Li, Q. Yu, M. Tan, J. Mei, P. Tang, W. Shen, A. Yuille, and C. Xie, "Shape-texture debiased neural network training," *arXiv*, 2020.

[115] F. Liao, M. Liang, Y. Dong, and T. Pang, "Defense against adversarial attacks using high-level representation guided denoiser," in *CVPR*, 2018.

[116] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim, "Fast autoaugment," in *NeurIPS*, 2019.

[117] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *CVPR*, 2017.

[118] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *ECCV*, 2018.

[119] X. Liu, M. Cheng, H. Zhang, and C.-J. Hsieh, "Towards robust neural networks via random self-ensemble," in *ECCV*, 2018.

[120] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *ICLR*, 2017.

[121] V. S. Lokhande, S. Tasneeyapant, A. Venkatesh, S. N. Ravi, and V. Singh, "Generating accurate pseudo-labels in semi-supervised learning and avoiding overconfident predictions via hermite polynomial activations," in *CVPR*, 2020.

[122] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

[123] R. G. Lopes, D. Yin, B. Poole, J. Gilmer, and E. D. Cubuk, "Improving robustness without sacrificing accuracy with patch gaussian augmentation," *arXiv preprint arXiv:1906.02611*, 2019.

[124] Y. Lou, X. Boix, G. Roig, T. Poggio, and Q. Zhao, "Foveation-based mechanisms alleviate adversarial examples," Center for Brains, Minds and Machines (CBMM), arXiv, Tech. Rep., 2016.

[125] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, 2004.

[126] J. Lu, T. Issaranon, and D. Forsyth, "Safetynet: Detecting and rejecting adversarial examples robustly," in *ICCV*, 2017.

[127] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. Wijewickrema, G. Schoenebeck, D. Song, M. E. Houle, and J. Bailey, "Characterizing adversarial subspaces using local intrinsic dimensionality," in *ICLR*, 2018.

[128] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *ICLR*, 2018.

[129] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten, "Exploring the limits of weakly supervised pretraining," in *ECCV*, 2018.

[130] X. Mao, Y. Chen, Y. Li, Y. He, and H. Xue, "Gap++: Learning to generate target-conditioned adversarial examples," *arXiv preprint arXiv:2006.05097*, 2020.

[131] M. McCoyd, W. Park, S. Chen, N. Shah, R. Roggenkemper, M. Hwang, J. X. Liu, and D. Wagner, "Minority reports defense: Defending against adversarial patches," *arXiv preprint arXiv:2004.13799*, 2020.

[132] D. Meng and H. Chen, "Magnet: A two-pronged defense against adversarial examples," in *CCS*, 2017.

[133] A. Merchant, B. Zoph, and E. D. Cubuk, "Does data augmentation benefit from split batchnorms," *arXiv preprint arXiv:2010.07810*, 2020.

[134] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *ICLR*, 2018.

[135] C. Michaelis, B. Mitzkus, R. Geirhos, E. Rusak, O. Bringmann, A. S. Ecker, M. Bethge, and W. Brendel, "Benchmarking robustness in object detection: Autonomous driving when winter is coming," *arXiv preprint arXiv:1907.07484*, 2019.

[136] D. Miller, Y. Wang, and G. Kesidis, "When not to classify: Anomaly detection of attacks (ada) on dnn classifiers at test time," *Neural computation*, 2019.

[137] Y. Min, L. Chen, and A. Karbasi, "The curious case of adversarially robust models: More data can help, double descend, or hurt generalization," *arXiv preprint arXiv:2002.11080*, 2020.

[138] D. Misra, "Mish: A self regularized non-monotonic neural activation function," *arXiv preprint arXiv:1908.08681*, 2019.

[139] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: A regularization method for supervised and semi-supervised learning," *TPAMI*, 2018.

[140] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *CVPR*, 2017.

[141] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *CVPR*, 2016.

[142] M. Mosbach, M. Andriushchenko, T. Trost, M. Hein, and D. Klakow, "Logit pairing methods can fool gradient-based attacks," *arXiv preprint arXiv:1810.12042*, 2018.

[143] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.

[144] A. Najafi, S.-i. Maeda, M. Koyama, and T. Miyato, "Robustness to adversarial perturbations in learning from incomplete data," in *NeurIPS*, 2019.

[145] P. Nakkiran, "Adversarial robustness may be at odds with simplicity," *arXiv preprint arXiv:1901.00532*, 2019.

[146] M. M. Naseer, S. H. Khan, M. H. Khan, F. S. Khan, and F. Porikli, "Cross-domain transferability of adversarial perturbations," in *NeurIPS*, 2019.

[147] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *CVPR*, 2015.

[148] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, 1979.

[149] T. Pang, C. Du, Y. Dong, and J. Zhu, "Towards robust detection of adversarial examples," in *NeurIPS*, 2018.

[150] T. Pang, K. Xu, C. Du, N. Chen, and J. Zhu, "Improving adversarial robustness via promoting ensemble diversity," in *ICML*, 2019.

[151] T. Pang, K. Xu, and J. Zhu, "Mixup inference: Better exploiting mixup to defend adversarial attacks," in *ICLR*, 2020.

[152] T. Pang, X. Yang, Y. Dong, H. Su, and J. Zhu, "Bag of tricks for adversarial training," *arXiv preprint arXiv:2010.00467*, 2020.

[153] T. Pang, X. Yang, Y. Dong, K. Xu, H. Su, and J. Zhu, "Boosting adversarial training with hypersphere embedding," in *NeurIPS*, 2020.

[154] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, *et al.*, "Technical report on the cleverhans v2. 1.0 adversarial examples library," *arXiv preprint arXiv:1610.00768*, 2016.

[155] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *ASIACCS*, 2017.

[156] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *SP*, 2016.

[157] O. Poursaeed, T. Jiang, H. Yang, S. Belongie, and S.-N. Lim, "Fine-grained synthesis of unrestricted adversarial examples," *arXiv preprint arXiv:1911.09058*, 2019.

[158] O. Poursaeed, I. Katsman, B. Gao, and S. Belongie, "Generative adversarial perturbations," in *CVPR*, 2017.

[159] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. Storer, "Deflecting adversarial attacks with pixel deflection," in *CVPR*, 2018.

[160] S. Qiao, W. Shen, Z. Zhang, B. Wang, and A. Yuille, "Deep co-training for semi-supervised image recognition," in *ECCV*, 2018.

[161] 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 *NeurIPS*, 2019.

[162] H. Qiu, C. Xiao, L. Yang, X. Yan, H. Lee, and B. Li, "Semanticadv: Generating adversarial examples via attribute-conditional image editing," in *ECCV*, 2020.

[163] A. Raghunathan, S. M. Xie, F. Yang, J. Duchi, and P. Liang, "Understanding and mitigating the tradeoff between robustness and accuracy," in *ICML*, 2020.

[164] A. Raghunathan, S. M. Xie, F. Yang, J. C. Duchi, and P. Liang, "Adversarial training can hurt generalization," *arXiv preprint arXiv:1906.06032*, 2019.

[165] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," in *ICLR Workshop*, 2018.

[166] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: An astounding baseline for recogniton," in *CVPR Workshop on DeepVision*, 2014.

[167] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *AAAI*, 2019.

[168] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *TPAMI*, 2017.

[169] L. Rice, E. Wong, and J. Z. Kolter, "Overfitting in adversarially robust deep learning," *arXiv preprint arXiv:2002.11569*, 2020.

[170] A. Rosenfeld, R. Zemel, and J. K. Tsotsos, "The elephant in the room," *arXiv preprint arXiv:1808.03305*, 2018.

[171] A. Rozsa and T. E. Boult, "Improved adversarial robustness by reducing open space risk via tent activations," *arXiv preprint arXiv:1908.02435*, 2019.

[172] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: nonlinear phenomena*, 1992.

[173] 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," *IJCV*, 2015.

[174] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-GAN: Protecting classifiers against adversarial attacks using generative models," in *ICLR*, 2018.

[175] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Madry, "Adversarially robust generalization requires more data," in *NeurIPS*, 2018.

[176] L. Schott, J. Rauber, M. Bethge, and W. Brendel, "Towards the first adversarially robust neural network model on mnist," in *ICLR*, 2019.

[177] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" In *NeurIPS*, 2019.

[178] G. Shamir, D. Lin, and L. Coviello, "Smooth activations and reproducibility in deep networks," *arXiv preprint arXiv:2010.09931*, 2020.

[179] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang, "Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection," in *CVPR*, 2015.

[180] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *CVPR*, 2016.

[181] M. Shu, Z. Wu, M. Goldblum, and T. Goldstein, "Preparing for the worst: Making networks less brittle with adversarial batch normalization," *arXiv preprint arXiv:2009.08965*, 2020.

[182] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *ICCV*, 2015.

[183] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[184] A. Sinha, H. Namkoong, and J. Duchi, "Certifying some distributional robustness with principled adversarial training," in *ICLR*, 2018.

[185] C. Sitawarin, A. N. Bhagoji, A. Mosenia, M. Chiang, and P. Mittal, "Darts: Deceiving autonomous cars with toxic signs," *arXiv preprint arXiv:1802.06430*, 2018.

[186] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," in *ICLR*, 2018.

[187] Y. Song, R. Shu, N. Kushman, and S. Ermon, "Constructing unrestricted adversarial examples with generative models," in *NeurIPS*, 2018.

[188] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *JMLR*, 2014.

[189] D. Stutz, M. Hein, and B. Schiele, "Disentangling adversarial robustness and generalization," in *CVPR*, 2019.

[190] D. Stutz, M. Hein, and B. Schiele, "Confidence-calibrated adversarial training: Generalizing to unseen attacks," in *ICML*, 2020.

[191] D. Su, H. Zhang, H. Chen, J. Yi, P.-Y. Chen, and Y. Gao, "Is robustness the cost of accuracy?–a comprehensive study on the robustness of 18 deep image classification models," in *ECCV*, 2018.

[192] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *ICCV*, 2017.

[193] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *ICLR Workshop*, 2016.

[194] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

[195] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *CVPR*, 2016.

[196] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *ICLR*, 2014.

[197] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML*, 2019.

[198] S. Thys, W. Van Ranst, and T. Goedemé, "Fooling automated surveillance cameras: Adversarial patches to attack person detection," in *CVPR Workshop on CV-COPS*, 2019.

[199] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *ICCV*, 1998.

[200] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *ICLR*, 2018.

[201] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," in *ICLR*, 2019.

[202] J. Uesato, J.-B. Alayrac, P.-S. Huang, A. Fawzi, R. Stanforth, and P. Kohli, "Are labels required for improving adversarial robustness?" In *NeurIPS*, 2019.

[203] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv:1607.08022*, 2016.

[204] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.

[205] A. Wald, "Statistical decision functions which minimize the maximum risk," *Annals of Mathematics*, 1945.

[206] H. Wang, T. Chen, S. Gui, T.-K. Hu, J. Liu, and Z. Wang, "Once-for-all adversarial training: In-situ tradeoff between robustness and accuracy for free," in *NeurIPS*, 2020.

[207] J. Wang, C. Xie, Z. Zhang, J. Zhu, L. Xie, and A. Yuille, "Detecting semantic parts on partially occluded objects," in *BMVC*, 2017.

[208] J. Wang and H. Zhang, "Bilateral adversarial training: Towards fast training of more robust models against adversarial attacks," in *ICCV*, 2019.

[209] S. Wang, X. Wang, P. Zhao, W. Wen, D. Kaeli, P. Chin, and X. Lin, "Defensive dropout for hardening deep neural networks under adversarial attacks," in *ICCAD*, 2018.

[210] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *CVPR*, 2018.

[211] X. Wang, J. Ren, S. Lin, X. Zhu, Y. Wang, and Q. Zhang, "A unified approach to interpreting and boosting adversarial transferability," *arXiv preprint arXiv:2010.04055*, 2020.

[212] Y. Wang, X. Ma, J. Bailey, J. Yi, B. Zhou, and Q. Gu, "On the convergence and robustness of adversarial training," in *ICML*, 2019.

[213] Y. Wang, D. Zou, J. Yi, J. Bailey, X. Ma, and Q. Gu, "Improving adversarial robustness requires revisiting misclassified examples," in *ICLR*, 2020.

[214] X. Wei, J. Zhu, S. Yuan, and H. Su, "Sparse adversarial perturbations for videos," in *AAAI*, 2019.

[215] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," in *ICLR*, 2020.

[216] D. Wu, Y. Wang, S.-T. Xia, J. Bailey, and X. Ma, "Skip connections matter: On the transferability of adversarial examples generated with resnets," in *ICLR*, 2019.

[217] T. Wu, L. Tong, and Y. Vorobeychik, "Defending against physically realizable attacks on image classification," in *ICLR*, 2020.

[218] Y. Wu *et al.*, *Tensorpack*, https://github.com/tensorpack/, 2016.

[219] Y. Wu and K. He, "Group normalization," in *ECCV*, 2018.

[220] C. Xiang, A. N. Bhagoji, V. Sehwag, and P. Mittal, "Patchguard: Provable defense against adversarial patches using masks on small receptive fields," *arXiv preprint arXiv:2005.10884*, 2020.

[221] C. Xiao and C. Zheng, "One man's trash is another man's treasure: Resisting adversarial examples by adversarial examples," in *CVPR*, 2020.

[222] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," in *IJCAI*, 2018.

[223] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song, "Spatially transformed adversarial examples," in *ICLR*, 2018.

[224] C. Xie, M. Tan, B. Gong, J. Wang, A. Yuille, and Q. V. Le, "Adversarial examples improve image recognition," in *CVPR*, 2020.

[225] C. Xie, M. Tan, B. Gong, A. Yuille, and Q. V. Le, "Smooth adversarial training," *arXiv preprint arXiv:2006.14536*, 2020.

[226] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, "Mitigating adversarial effects through randomization," in *ICLR*, 2018.

[227] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, "Adversarial Examples for Semantic Segmentation and Object Detection," in *ICCV*, 2017.

[228] C. Xie, Y. Wu, L. van der Maaten, A. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in *CVPR*, 2019.

[229] C. Xie and A. Yuille, "Intriguing properties of adversarial training at scale," in *ICLR*, 2020.

[230] C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, and A. L. Yuille, "Improving transferability of adversarial examples with input diversity," in *CVPR*, 2019.

[231] Q. Xie, E. Hovy, M.-T. Luong, and Q. Le, "Self-training with noisy student improves imagenet classification," in *CVPR*, 2020.

[232] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *CVPR*, 2017.

[233] S. Xie and Z. Tu, "Holistically-nested edge detection," in *ICCV*, 2015.

[234] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *NDSS*, 2018.

[235] C. Yang, A. Kortylewski, C. Xie, Y. Cao, and A. Yuille, "Patchattack: A black-box texture-based attack with reinforcement learning," in *ECCV*, 2020.

[236] D. Yin, R. G. Lopes, J. Shlens, E. D. Cubuk, and J. Gilmer, "A fourier perspective on model robustness in computer vision," in *NeurIPS*, 2019.

[237] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional neural networks," in *ECCV*, 2014.

[238] R. Zhai, T. Cai, D. He, C. Dan, K. He, J. Hopcroft, and L. Wang, "Adversarially robust generalization just requires more unlabeled data," *arXiv preprint arXiv:1906.00555*, 2019.

[239] D. Zhang, T. Zhang, Y. Lu, Z. Zhu, and B. Dong, "You only propagate once: Accelerating adversarial training via maximal principle," 2019.

[240] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, Z. Zhang, H. Lin, Y. Sun, T. He, J. Mueller, R. Manmatha, M. Li, and A. Smola, "Resnest: Split-attention networks," *arXiv preprint arXiv:2004.08955*, 2020.

[241] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *ICML*, 2019.

[242] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," in *ICLR*, 2018.

[243] T. Zhang and Z. Zhu, "Interpreting adversarially trained convolutional neural networks," in *ICML*, 2019.

[244] X. Zhang, Q. Wang, J. Zhang, and Z. Zhong, "Adversarial autoaugment," in *ICLR*, 2020.

[245] Y. Zhang, H. Foroosh, P. David, and B. Gong, "Camou: Learning physical vehicle camouflages to adversarially attack detectors in the wild," in *ICLR*, 2018.

[246] Z. Zhang, S. Qiao, C. Xie, W. Shen, B. Wang, and A. L. Yuille, "Single-shot object detection with enriched semantics," in *CVPR*, 2018.

[247] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *ICCV*, 2015.

[248] W. Zhou, X. Hou, Y. Chen, M. Tang, X. Huang, X. Gan, and Y. Yang, "Transferable adversarial perturbations," in *ECCV*, 2018.

[249] B. Zoph, E. D. Cubuk, G. Ghiasi, T.-Y. Lin, J. Shlens, and Q. V. Le, "Learning data augmentation strategies for object detection," *arXiv preprint arXiv:1906.11172*, 2019.

[250] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *ICLR*, 2017.

[251] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *CVPR*, 2018.

Vita

Cihang Xie is completing his Ph.D. degree of Computer Science at the Johns Hopkins University, under the supervision of Bloomberg Distinguished Professor Alan L. Yuille. Cihang received his M.S. degree from UCLA in 2015, and B.S. degree from Huazhong University of Science and Technology in 2014. Cihang's research interests lie in the fields of computer vision, deep learning and machine learning. His research goal is on building human-level computer vision systems, particularly in securing model performance under the worst-case scenario and endowing models with interpretability.