Learn2Perturb: an End-to-end Feature Perturbation Learning to Improve Adversarial Robustness

by

Ahmadreza Jeddi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2020

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Content from 1 paper is used in this thesis. I was the co-author with major contributions on designing the method, implementation, and writing the paper:

A. Jeddi, M. J. Shafiee, M. Karg, C. Scharfenberger, A. Wong, "Learn2Perturb: an End-to-end Feature Perturbation Learning to Improve Adversarial Robustness". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020, pp. 1241–1250.
This paper is incorporated in Chapters 3 and 4 of this thesis.

Abstract

Deep neural networks have been achieving state-of-the-art performance across a wide variety of applications, and due to their outstanding performance, they are being deployed in safety and security critical systems. However, in recent years, deep neural networks have been shown to be very vulnerable to optimally crafted input samples called adversarial examples. Although the adversarial perturbations are imperceptible to humans, especially, in the domain of computer vision, they have been very successful in fooling strong deep models. The vulnerability of deep models to adversarial attacks limits their widespread deployment for safety-critical applications. As a result, adversarial attack and defense algorithms have drawn great attention in the literature.

Many defense algorithms have been proposed to overcome the threat of adversarial attacks, and many of these algorithms use adversarial training (adding perturbations during the training stage). Alongside other adversarial defense approaches being investigated, there has been a very recent interest in improving adversarial robustness in deep neural networks through the introduction of perturbations during the training process. However, such methods leverage fixed, pre-defined perturbations and require significant hyper-parameter tuning that makes them very difficult to leverage in a general fashion.

In this work, we introduce Learn2Perturb, an end-to-end feature perturbation learning approach for improving the adversarial robustness of deep neural networks. More specifically, we introduce novel perturbation-injection modules that are incorporated at each layer to perturb the feature space and increase uncertainty in the network. This feature perturbation is performed at both the training and the inference stages. Furthermore, inspired by the Expectation-Maximization approach, an alternating back-propagation training algorithm is introduced to train the network and noise parameters consecutively. Experimental results on CIFAR-10 and CIFAR-100 datasets show that the proposed Learn2Perturb method can result in deep neural networks which are 4-7% more robust on l_∞ FGSM and PDG adversarial attacks and significantly outperforms the state-of-the-art against l_2 C&W attack and a wide range of well-known black-box attacks

Acknowledgements

I would like to thank my supervisor, professor Alexander Wong, and my dear colleague and mentor professor Mohammad Javad Shafiee. Prof. Wong thank you for trusting in me and giving me the opportunity of doing research on the exciting topics of ML and for all the encouragement and support you have provided for me during my time in your team. Prof. Shafiee thank you for always being there to support me and discuss the new ideas; your passion for doing the research that matters is an inspiration for me, and I cannot even start to describe how much fun I have had working with you. Thank you both for teaching me how fun and exciting the research is.

I would also like to take the time to thank all of my other friends and colleagues in the Vision and Image Processing lab; it is always a joy for me to discuss various topics with you.

Many thanks to my MMath. committee members, professors Jeff Orchard and Justin Wan from the department of Computer Science at the University of Waterloo, for their time and commitment.

Most importantly, I would like to thank my beloved family specially my dear brother Kazem for all the love and support they have had for me throughout my life and studies.

Dedication

To my beloved mother

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Nomenclature

δ	Perturbation added to inputs
ϵ	Limit of perturbation added to inputs
γ	Coefficient of regularization term
$\mathcal{L}$	Cost function of neural networks
$\mathcal{N}(u, \sigma)$	Gaussian distribution
∇	Gradient of loss function (with respect to inputs or network parameters)
τ	Annealing term
θ	Noise parameters
D	A dataset
f	Function represented by a neural network
g	Regularization term
l	Distance in a norm
lr	Learning rate
P	Probabilistic formulation of neural networks

p	Norm of perturbations
Q	Noise distribution
s	Epoch to start noise injection
T	Ground truth labels
t	Training epoch
W	Network parameters
x	Input samples (images)
Y	Output of neural networks

Chapter 1

Introduction

Deep Neural Networks (DNNs) have made significant progress in various Artificial Intelligence (AI) and Machine Learning (ML) domains; they provide the state-of-the-art performances in computer vision tasks such as object recognition [1, 2], object detection [3], instance and semantic segmentation of images and videos [4, 5], pose estimation [6], video and image generative models [7]. Moreover, these powerful models are utilized for natural language processing; for text and audio recognition and translation [8, 9], sentiment analysis [10], voice and text synthesis [11, 12], and many other such tasks. ML models and specially DNN models are the building blocks for many crucial and highly security & safety sensitive systems like autonomous driving [13, 14], medical imaging [15], authentication systems [16], and malware detection tools [17]. These systems must always be prepared to defend against the adversaries who try to fool them and breach the system security towards their malicious goals.

Recently, Szegedy *et al.* [18] showed that DNN models are very vulnerable to adversarial perturbations. They generated very small perturbations that when added to input images of a strong classifier, although imperceptible to human eyes, could fool the DNN model with a high probability. These perturbed samples are called adversarial examples and Figure 1.1 illustrates an example of such perturbations. As can be seen in Figure 1.1, adversarial perturbations can be very effective even in very small magnitudes; in fact, these

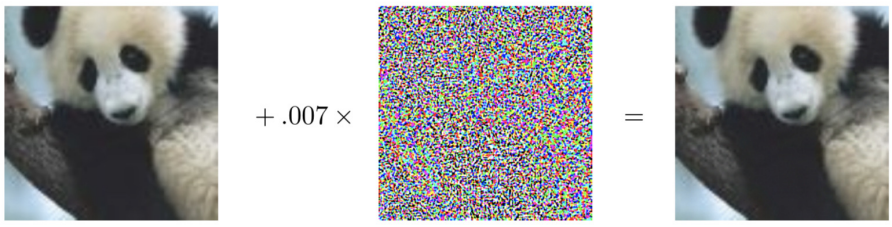

Figure 1.1: An adversarial example for an ImageNet classifier [23]. An imperceptible small perturbation correlated with the gradients of the network is added to the input, and ϵ controls the scale of this perturbation. The classifier (GoogLeNet) misclassifies the adversarial sample with a very high confidence. **Image from [23]**

.

perturbations are much more effective than random perturbations that can be caused from physical limitations or just added to the image. Since Szegedy *et al.* first fooled DNN classifiers in 2013 [18], various works have shown that it is possible to fool almost every DNN model; [19] and [20] attack object detection and semantic segmentation models, [21] generates adversarial samples for speech-to-text generation models, and [22] proposes an attack for natural language Recurrent Neural Networks (RNNs).

1.1 Problem Definition

The threat of adversarial attacks poses a great danger to DNN-based systems; especially, recent works have shown that these attacks can happen in the physical world [24], and they are transferable, which means they can attack a model even without having direct access to it [25, 26]. Achieving DNN models that are robust against adversarial perturbations has a great practical value. The goal of adversarial robustness is to provide security against attacks that may cause real-life harm.

The challenge of adversarial robustness draws great attention to understanding the phenomenon of adversarial examples [27, 23, 28]. In the last few years, many strong adversarial algorithms have been proposed in the literature [25, 29]. Each of these algorithms

craft optimized human-imperceptible perturbations to be added to the inputs of the deep models. Our goal in this work is to improve the adversarial robustness of computer vision deep models against the most powerful of these adversaries.

Numerous defense mechanisms have been proposed to boost the adversarial robustness of DNNs. Some of the more recent and promising of these techniques take advantage of network randomization, in which random perturbations are added to the inputs or within the network. While these techniques have shown promising results, they lack a systematic manner for injecting the perturbations. In our work, we propose a framework that learns how to inject noise into the system and takes great advantage of randomization towards training robust models.

This work explores the recent methods of enhancing the adversarial robustness of DNN models, and after describing the current state-of-the-art techniques, we propose and explain our Learn2Perturb [30] framework, a strong randomization-based defensive mechanism. Furthermore, we provide a deeper dive into the realm of adversarial robustness and explore the recent most effective attack and defense techniques.

1.2 Contribution

Our contributions can be folded as below:

- A highly efficient and stable end-to-end learning mechanism is introduced to learn the perturbation-injection modules to improve the model robustness against adversarial attacks. The proposed alternating back-propagation method inspired by Expectation-Maximization (EM) concept trains the network and noise parameters in a consecutive way gradually without any significant parameter-tuning effort.
- A new effective regularizer is introduced to help the network learning process which smoothly improves the noise distributions. Combining this regularizer and PGD-

adversarial training helps the proposed Learn2Perturb algorithm achieve state-of-the-art performances.

- Exhaustive experiments are conducted for various white-box and black-box adversarial attacks on CIFAR-10 and CIFAR-100 datasets, and new state-of-the-art robustness performances are reported against these adversarial algorithms.

1.3 Thesis Structure

The thesis is organized into 5 chapters. Chapter 2 provides background on deep neural networks that are used in computer vision classification tasks, as well as thorough background and discussion of the most recent works done on designing and analyzing adversarial attack and defense algorithms. In Chapter 3 we introduce our proposed Learn2Perturb framework and its components, then, we formally formulate its objective and learning process, and finally, we discuss how to setup Learn2Perturb's modules at the end of this chapter. The results of evaluating Learn2Perturb and comparing it with the state-of-the-art techniques are reported in Chapter 4; these results consist of evaluating competing defense methods against a wide range of white-box and black-box adversaries on two well-known datasets of CIFAR-10 and CIFAR-100. Lastly, the thesis is concluded in Chapter 5 and possible future directions for this work follow the conclusions.

Chapter 2

Background

In this chapter, we provide a background of the concepts related to our proposed framework and discuss the recent related works. To this end, Section 2.1 briefly describes the deep models utilized for training image classifiers and formally defines these models, and Sections 2.2 and 2.3 cover topics of adversarial attack algorithms and adversarial defense mechanisms, respectively.

2.1 Deep Learning

Deep Neural Networks (DNNs) are a class of Machine Learning (ML) tools. Like many other ML tools, DNNs are commonly trained on a (preferably fully annotated) dataset and are applied for making predictions (generalization) on unseen test data. Especially, DNN classifiers are very similar to Logistic Regression (LR) models in the sense that both types of these models use entropy-based objective functions (commonly categorical cross-entropy) and apply gradient descent for objective optimization and learning; however, the two have the major differences that DNN models are usually deeper (having more layers) than LR models (which only has 1 layer), and DNN models apply modules (such as Max-Pooling) and activation functions (such as ReLU) which makes them highly non-linear functions (LR

models are linear), thus, with having larger capacities (number of parameters and layers) and learning non-linear functions, DNNs can be much more powerful than LR models (and almost every other ML model); hence, they are a popular candidate for solving every ML problem.

2.1.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of DNN models that are frequently used for computer vision problems. Nowadays, CNNs are the *de facto* tools for vision problems. By applying the same shared convolution operations (kernels) over their gridlike inputs, they manage to drastically reduce the number of network parameters, while equipping the network with object-translation invariance at the same time. Object-translation invariance is a key feature for common vision tasks (object recognition, object detection, and segmentation), because it removes the dependence of the models on the position of objects in images and videos. CNNs have shown incredible performances in different vision tasks, such as object recognition/classification [1, 31], object detection [32, 33], object localization [34, 35], semantic and instance segmentation [36, 37, 38], and image reconstruction [39, 40]. Their success in these tasks has made them the most popular tool for critical applications like autonomous driving [41, 14, 42], medical imaging [43, 44], robotics [45, 46], and video surveillance [47, 48]. Throughout this thesis, we use the following formal definition for CNNs:

$$f(x) = l^{(n)}\left(... \, l^{(2)}\left(l^{(1)}\left(x\right)\right) ...\right) \tag{2.1}$$

where x is an input image, $l^{(i)}$ is the ith layer of the the network, for $i = 1, 2, ...k$. $f(.)$ shows the function represented by CNN.

2.1.2 CNN Architectures

Many different CNN architectures exist in the literature, and every day new task-specific CNN architectures are being proposed. Some of the most well-known and widely used

CNN architectures are AlexNet [1], LeNet [49], VGG [2], GoogLeNet [50], and ResNet [31] from simplest (oldest) network to the most complex (newest) one [51]. ResNet and its variants ResNext [52], and Wide-ResNet [53] are the most popular models in studying the adversarial robustness. Especially, more recently, techniques applying Wide-ResNet architecture have achieved tremendous success in adversarial settings [56, 54, 55].

2.2 Adversarial Attacks

Since the introduction of the adversarial examples to DNN models, academia and industry have been very interested in understanding this intriguing phenomenon; they primarily follow two goals: getting more insights into the way neural networks work and to achieve models which are robust to these perturbations so they can be deployed in safety-critical environments. Although numerous works [57, 58, 59] have tried to explain the way neural networks operate and what makes them such excellent tools, they still remain pretty much like black-box tools [60], which are fed inputs in different formats and they output customized results. Specially, these hypotheses provide fragile explanations when adversarial examples are present. Recent works of [23] and [27] propose the linearity hypothesis and the manifold assumption, respectively. These are two of the more plausible hypotheses that have been proposed to explain the phenomenon of adversarial examples. Many adversarial attack and defense algorithms have been proposed based on such hypotheses [28, 61, 62]. These two hypotheses are explained in more detail in Section 2.2.2.

While the two goals of DNN explainability and robust models have large overlaps, the bigger focus of the studies in this field has been on the robustness part. This is because the threat of adversarial attacks has slowed down the adoption of DNNs for sensitive problems such as autonomous driving, medical imaging, and fraud detection. Works such as [24] and [63] indicate that this threat is not just an abstract concept, but the adversarial attacks can take place in the physical world. Figure 2.1 shows an example of such adversarial attacks in the real world and illustrates the danger this phenomenon can pose in safety

Figure 2.1: The left image shows a traffic sign with real graffiti on it, and the right image shows an adversarial Stop sign created in a similar fashion to the real one on the left. While most humans might ignore this perturbation, the right sign can fool the Sign detection system of an autonomous vehicle with a high probability [63]. **Image from [63]**.

and security sensitive systems. Other works have shown that similar threats exist for vision models segmenting pedestrians and road users [19], as well as, speech recognition systems [64, 65].

In this section, first, we will formally define the adversarial examples in computer vision, then, we will explain two popular hypotheses on why adversarial examples exist, afterward, following [66, 51, 67] we will explain a formal systematic model for categorizing and analyzing attack and defense algorithms within an adversarial setting, named *threat* model. After these definitions, we will explore the more well-known adversarial attacks.

2.2.1 Adversarial Examples in Computer Vision

Given an image classification/recognition neural network model f, and clean input image x, assuming that f predicts the true label of x (y) correctly, an adversarial sample x' can be created by adding minimal perturbation η to x, so that $f(x') \neq y$:

$$
\begin{aligned}
\min_{x'} \|x' - x\|_p \quad &\equiv \quad \min_{\eta} \|\eta\|_p, \\
s.t. \quad & f(x) = y, \\
& f(x') = y', \\
& y \neq y', \\
& x' \in [0,1]^m, \\
& \|\eta\|_p \leq d
\end{aligned}
\tag{2.2}
$$

where $\|x' - x\|_p$ calculates the distance between x and x' (common options for p are 0, 1, 2, and ∞), and the last two parts of the equation restrict the scale of the perturbation, especially, d is usually a small positive value which forces the adversarial perturbations to be so small that they are imperceptible to human eyes.

2.2.2 Why Adversarial Examples Exist?

Adversarial examples are very surprising phenomena; the main question about them is that why can DNNs surpassing the human-level performance be so easily fooled by very small perturbations? while this question poses an open problem for which no definite answer has been proposed, numerous works have tried to provide explanations for the phenomenon of adversarial examples. Linearity of DNN models in higher dimensional space [23, 68, 26], inflexible models [69], flat decision boundaries [70, 71, 72], lack of generalization [73], and more recently, the manifold assumption [74, 27] are just some of viewpoints that have been proposed to explain the existence of adversarial samples. Each of these viewpoints has been successful in explaining some aspects and scenarios of adversarial examples, and while some have been more effective than others, none has yet been able to completely explain every aspect of these examples. However, the linearity of deep models and the manifold assumption have been more popular in the literature. Here, we will explain them in more detail.

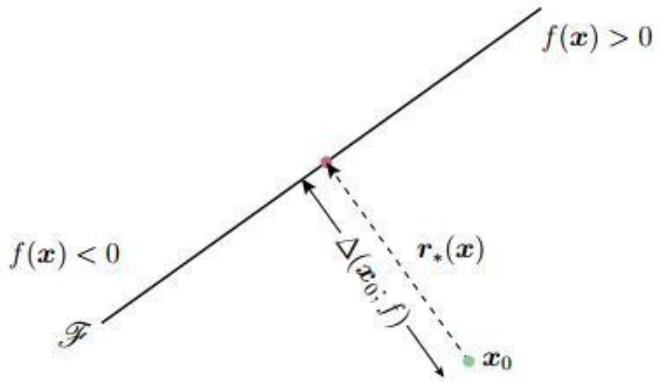

Figure 2.2: At every iteration, DeepFool finds the vector r orthogonal to the decision boundary and perturbs the image in that direction. **Image from** [76].

Linearity of Deep Models

Goodfellow *et al.* [23] suggested that DNN models are too linear in their high-dimensional latent space; and as a result of their high dimension, small changes in their input space can make significant changes in their latent space; although this is against the common belief that DNNs are highly non-linear models, the linearity hypothesis can explain FGSM (explained in Section 2.2.6) and related attacks very well [75].

DeepFool [76] is a very strong iterative adversarial attack which is designed according to the linearity hypothesis. DeepFool starts with assuming that the target clean image resides in a space where linear decision boundaries decide the label of this image; during each iteration, the adversary computes a vector that when added to the image can supposedly take the latent representation of the image beyond a decision boundary and change the classification label. These iterative perturbations are then accumulated to form the final adversarial perturbation added to the original clean image. Figure 2.2 illustrates how DeepFool operates for a binary classifier: approximating the vector orthogonal to the decision boundary and then adding perturbations until the sample crosses the decision boundary.

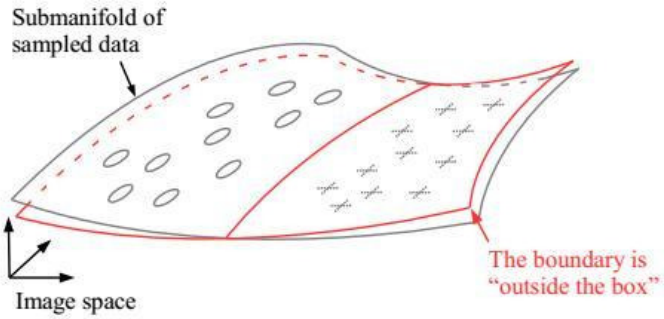

Figure 2.3: Class boundaries can extend beyond the data manifold, tilting away from it. Adversarial attacks force samples leave the data manifold and move toward crossing the class boundaries [27]. **Image from [27]**.

Manifold Assumption

In ML, the manifold hypothesis states that real-world high-dimensional data lie on a lower-dimensional latent surface called manifold. The original data form a set of connected points in their high-dimensional space which are possible to approximate using a manifold with a smaller number of dimensions [77, 78]. Instead of approximations across all of $\mathbb{R}^n$, ML algorithms assume that the distribution of the data lies on a collection of manifolds (or just one single manifold); and ML classifiers learn decision boundaries over each of these manifolds.

Numerous works have tried to explain the existence of adversarial examples by forming assumptions on the geometrical aspects of manifolds learned by DNNs, and how class boundaries and data distributions interact with each other on these latent manifolds [74, 79, 80, 81]. Tanay and Griffin [27] challenged the linearity hypothesis proposed by Goodfellow *et al.* [23], and instead proposed the *boundary tilting perspective*; this hypothesis is illustrated in Figure 2.3. They claim that class boundaries (shown in red in the Figure) lie close to the data manifolds (shown in black in the Figure); however, these boundaries extend beyond the data manifold, *tilting* away from it. While random perturbations might not be able to force a data sample to leave the manifold to make it to the other side of the manifold (hence, changing the class label), adversarial perturbations can force a sample to move in the direction of the class boundary and probably cross it.

Following [27], many other works have worked on the idea that adversarial examples leave the data manifold and are samples of different distributions than that of the clean data [82, 81, 83]. This assumption has led to the design of some adversarial detection algorithms [83, 84]; however, Carlini and Wagner [85] designed attacks that can bypass these detection systems, and hence, question the idea of adversarial examples lying off data manifold. Moreover, [74, 86] found adversarial samples that lie on the data manifold. Nonetheless, the manifold assumption has been the basis of some novel defense mechanisms [87, 88, 61].

In the recent work of [28], Stutz *et al.* showed that *regular* adversarial attacks do indeed leave the data manifold, and while they acknowledged the existence of on-manifold adversarial attacks, they showed that on-manifold adversarial attacks are essentially generalization errors (regions of data manifold that the classifier has not been successful in finding their class boundaries), and more importantly, the adversarial robustness and generalization are not necessarily at odds with each other, despite what previous works of [89] and [55] believed. Figure 2.4 illustrates the off-manifold (regular) and on-manifold (generalization errors) adversarial examples for the MNIST [90] and EMNIST [91] datasets; as seen in this Figure, regular adversarial examples tend to leave the data manifold in a direction which is near orthogonal to the manifold.

2.2.3 Threat Model

If we take a deeper look at Equation (2.2), some questions about adversarial examples might arise; questions like, what information does the adversary have about the target model? how should the adversary decide the limit of distortions (d)? can the adversary change the target model or poison the training data? should the attack be targeted (y' being a specific class) or just changing the label ($y' \neq y$) is enough? and many other such questions. Due to the existence of various possible setups for analyzing adversarial robustness, the security and the performance of systems are measured according to the capabilities and the goals of the adversary that targets the model and needs to be defended against. A typical threat

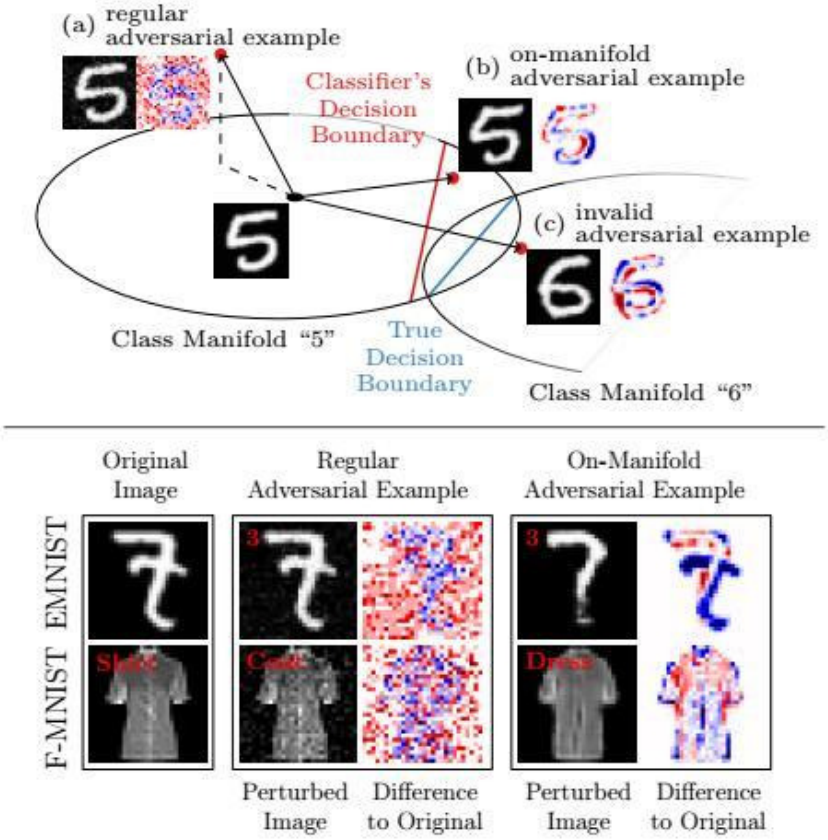

Figure 2.4: Difference between on and off manifold adversarial examples; regular off-manifold examples tend to have more noise-like effects and leave the data manifold, whereas on-manifold adversarial examples tend to change even they way human perceives the image. **Image from** [28].

model has the following components:

- **Attacker's capability:** what capabilities and information does the adversary have with regards to the target model? In this context, capabilities show at which stage the attack is performed, and information typically relates to the details of how the model is trained and what properties it has. *Poisoning* attacks are those who can inject adversarial samples during the training with the goal of affecting the model's performance on the test data. On the other hand, *evasion* attacks are the ones in which the adversary is just able to modify input samples during the inference and has no ability to change the way model works. Another factor is the amount and the level of the information that the adversary has about the model. Knowledge about the type of the model's underlying architecture, number of parameters, the value of parameters, gradients of a specific loss function with respect to the input data, and other such information. As an example, an adversary which knows every detail about the model (worst-case scenario for the defense), can be strictly more powerful in fooling networks than one which sees the model only as a black-box entity that can only be queried (giving inputs and receiving corresponding outputs).

- **Adversarial goals:** what is the goal of the adversary and what are they trying to achieve by attacking a target model? the goal can vary a lot based on the service that the target model delivers. For example, an adversary attacking an authentication system might try to extract some information about the model or its training data. Shokri *et al.* [92] proposed such a model to find the samples that are used during the model's training. Considering the capacity and the strength of DNN models, it is quite possible that these models can memorize a lot of information about their training data; hence, an adversary might try to use the model to extract the training data from a set of given samples. Another adversarial goal can be targeting the model's integrity. This is especially very common when analyzing computer vision models. Integrity refers to the different metrics that are usually used to report a model's performance; accuracy, precision, recall, F1 score, and area under the curve

are examples of such metrics. The attacks targeting the model's integrity simply try to degrade its performance, making the model misclassifying the input samples. Some attacks perform targeted perturbations; instead of simply fooling the target model, these attacks focus on changing a sample's predicted label into a specific target class. Imagine the case of autonomous driving illustrated in Figure 2.1; what would happen if the adversary could fool a street sign detection model to classify a stop sign as a speed-limit sign?

2.2.4 Black-box *vs.* White-box Attacks

Based on the task, the adversary might have some level of knowledge about the target model. Information like the model's baseline architecture, access to the model's parameters and their values, and the gradients of the loss function with respect to the input samples can give a big advantage to the adversary. The scale of the adversary's knowledge of the model is indeed one of the most important factors in designing both adversarial attacks and defenses. Adversarial attacks are divided into two categories of black-box [94, 26, 95, 93, 96, 97, 98, 99, 100, 101, 102] and white-box [18, 23, 24, 103, 29, 104, 105, 76, 25] attacks based on the level of information available to the attacker. Black-box attacks usually perform queries on the model, and they have partial information regarding the data and the structure of the targeted model [106, 93]. On the other hand, white-box attacks have a better understanding of the model that they attack; therefore, they are more powerful than black-box attacks [75, 18]. This understanding might vary between different white-box attack algorithms; nonetheless, gradients of the model's loss function with respect to the input data is the most common information utilized to modify input samples and generate adversarial examples. First-order white-box adversaries are the most common attacking algorithms which only use the first order of gradients [29, 25, 76, 18] to craft the adversarial perturbation. Both categories of black-box and white-box attacks are being extensively investigated in academia and industry. Black-box attacks provide a more practical point of view [107], while white-box attacks consider the worst-case scenario

challenging the security-sensitive systems [108].

2.2.5 The Threat Model Applied in Computer Vision

A prevalent threat model in studying the robustness of computer vision classifier models which is adopted throughout this thesis as well is:

1. **Adversarial capabilities:** The attack can only be done during the inference phase (evasion attack). This is a more realistic and practical assumption for the adversary. Therefore, the attack can only change the input data during the testing stage.

2. **Adversarial goal:** Compromising the integrity of the model is the most common goal. Especially, when attacking a classification model, most of the adversaries focus on degrading the classification accuracy. This is the adversarial goal that I will investigate throughout this work.

3. **Adversary's knowledge:** Examples and explanations of both black-box and white-box attacks are provided in this work. Moreover, adversaries of both types are used to analyze the performance of my framework, Learn2Perturb (which will be thoroughly discussed in the following sections and chapters).

2.2.6 White-box Attacks

The gradients of the loss function with respect to the input data are very common information used by adversarial attack algorithms. In this type of approach, the proposed algorithms try to maximize the loss value of the network by crafting the minimum perturbations into input data. It is worth noting that, all the white-box attacks explained here (i.e. FGSM, PGD, and C&W) are first-order adversaries, which means they only use the first-order gradients of the cost function with respect to input for adversarial sample generation.

Fast Gradient Sign Method (FGSM)

FGSM is a very simple yet very effective white-box attack [23]. For a DNN parametrized with W (i.e., where the network is encoded as $f_W(x)$) and loss function $\mathcal{L}$, for any input x, the FGSM attack computes the adversarial example x' as:

$$x' = x + \epsilon \cdot \text{sign}\Big(\nabla_x \mathcal{L}\big(f_W(x), x\big)\Big) \tag{2.3}$$

where ϵ determines the attack strength and sign$(\cdot)$ returns the sign tensor for a given tensor. Using this gradient ascent step, FGSM tries to locally maximize the loss function $\mathcal{L}$. Since it is a one-step attack it is very fast to calculate. Goodfellow *et al.* [23] proposed FGSM and they claimed that the *linearity* of DNN models is the reason FGSM can so easily fool these models. Although FGSM can be quite successful against models that are trained with no consideration of adversarial robustness, since they are single-step, compared to other attacks, FGSM attacks are easier to defend. On the other hand, the fact that they use only one iteration makes them more transferable; the transferability of the adversarial attacks is explained in Section 2.2.7. Figure 2.5 illustrates how the FGSM attack abstractly works according to the linearity hypothesis. FGSM chooses a very effective direction which can cross the decision boundaries of the classifier.

Projected Gradient Descent (PGD)

The FGSM approach is extended by PGD [25, 109] where for a number of k iterations, PGD produces $x_{t+1} = \text{bound}_{l_p}(FGSM(x_t), x_0)$, in which x_0 is the original input and $0 \leq t \leq k-1$. Using projection, the $\text{bound}_{l_p}(x', x)$ simply ensures that x' is within a specified l_p range of the original input x.

Madry *et al.* [25] illustrated that different PGD attack restarts, each with a random initialization for input within the l_∞–ball around x, find different local maxima with very similar loss values. Based on this finding, they claimed that PGD is a universal first-order adversary. Which means no other first-order adversary can outperform PGD's strength.

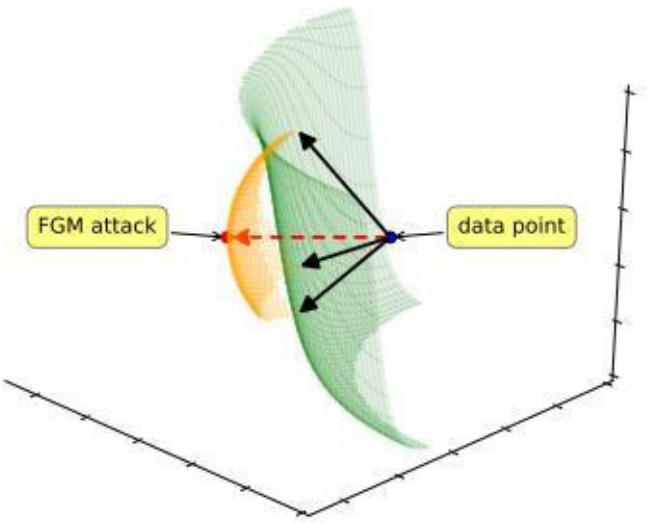

Figure 2.5: The black arrows are a set of vectors orthogonal to the decision boundaries of the classifier and are aligned with the gradient vector; these vectors form a space of potential adversarial examples [26]. **Image from [26].**

C&W Attack

Carlini and Wagner [29], proposed C&W which is another strong first-order attack algorithm that finds perturbation δ added to input x by solving the optimization problem formulated as:

$$\min_{\delta} \Big[||\delta||_p + c \cdot g(x+\delta) \Big] \quad s.t. \quad x + \delta \in [0,1]^m \tag{2.4}$$

where p shows the norm distance, and c is a constant balancing the gradient of the two terms involved in (2.4), and during the optimization is usually found by using binary search. $g(\cdot)$ encodes the objective function driving the perturbed sample to be misclassified, such that $g(x+\delta) \geq 0$ if and only if $f(x+\delta) \neq f(x)$, which means adversarial attack is successful. Carlini and Wagner [29] suggested 7 different options for $g(.)$; a very effective function of these suggestions considering $x' = x + \delta$ is

$$g(x') = max \Big[\max_{i \neq t} (Z(x')_i), -\kappa \Big] \tag{2.5}$$

where $Z(x')$ returns the softmax vector for the sample x', and t denotes the index of the true label of x. κ, the attack confidence, is a constant controlling how adversarial the

attack is; the bigger the κ's magnitude, the attacker must create stronger perturbations to fool the model. In the experiments, we will show that κ plays an important role for evaluating the robustness of defense methods that involve randomization.

Lastly, in Equation (2.4), the term $x+\delta \in [0,1]^m$ ensures that the adversarial examples are valid images. The initial version of the C&W attack uses the box-constrained L-BFGS as the optimizer to solve this objective function. However, in a later version they slighly change the optimization function, so that optimizers with no box-constraint support can be used as well; δ optimization is done via a change of variable

$$\delta = \frac{1}{2}(\, tanh(\omega) + 1) - x \tag{2.6}$$

Since $-1 \leq tanh(\omega) \leq 1$, it follows that $0 \leq x + \delta \leq 1$; therefore, the crafted sample would automatically be a valid image. As a result of this change, the new C&W objective function would be

$$\min_{\omega} \left[\left|\left| \frac{1}{2}(\, tanh(\omega) + 1) - x \right|\right|_p + c \cdot g\Big(\frac{1}{2}(\, tanh(\omega) + 1)\Big) \right]. \tag{2.7}$$

Carlini and Wagner considered values of 0, 2, and ∞ for p. However, the l_2 version of this attack is a very well-formed optimization function, and C&W is most effective when $p = 2$; therefore, following literature, we will use the l_2 C&W attack as well. Finally, due to the way $g(.)$ is formulated (in Equation (2.5)), C&W can inherently be used as a very effective *targeted* attack, in which the adversarial example is classified as a specific target class; targeted attacks can strictly be more dangerous than non-targeted ones. Figure 2.6 illustrates examples of the targeted l_2 attack on the CIFAR-10 dataset [110]; as can be seen, with imperceptible perturbations, C&W is able to change the label of every sample in this Figure to any other label in the dataset.

2.2.7 Black-box Attacks

Black-box attacks can only access a model via queries; sending inputs and receiving corresponding outputs to estimate the inner working of the network. To fool a network,

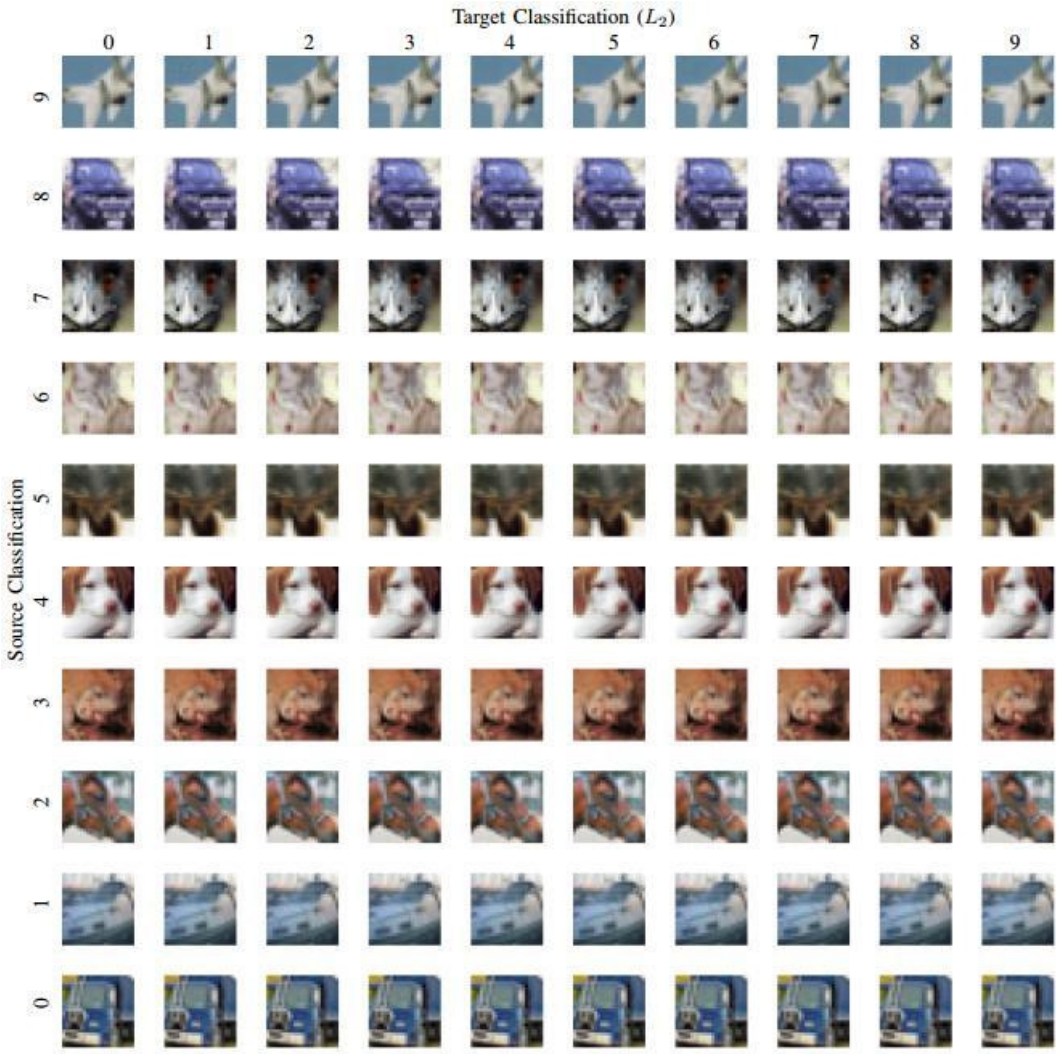

Figure 2.6: Targeted l_2 C&W attack on every possible source & target pair of the CIFAR-10 classes [29]. One image from each class of the CIFAR-10 dataset is selected; rows show the true label of the adversarial images, whereas columns show the predicted labels. Using imperceptible perturbations, C&W can change the label of each of these images into any other class. **Image from** [29].

the well-known black-box attacks either use surrogate networks [106, 93] or estimate the gradients [98, 94] via multiple queries to the targeted network.

Transferability

The phenomenon of adversarial example transferability has drawn a lot of attention in the recent years; transferability refers to the fact that in many cases adversarial examples generated for a model can fool another independently trained neural network [93, 106]. As a result of this phenomenon, neural networks can be attacked even when there is no direct access to them; this poses many practical threats to the neural networks deployed in the real world [25].

Tramer *et al.* [26] claimed that due to the high dimensionality of neural network, the subspace of their different classifiers can intersect, and the adversarial examples span a contiguous high dimension space; as a result, adversarial examples of a model can transfer to another. Papernot *et al.* [97] found that adversarial examples generated for a DNN can fool other DNNs with different baseline architectures; moreover, they can even fool other types of ML classifiers even the ones that are not differentiable, e.g. K-Nearest Neighbors (KNN) and decision trees. Furthermore, they experimentally show that transferable attacks are possible even when the two models have non-overlapping training data.

Different variations for transferable attacks are possible; in the surrogate network approach, a new network mimicking the behavior of the target model [93] is trained. Attackers perform queries on the target model and generate a synthetic dataset with the query inputs and associated outputs. Having this dataset, a surrogate network is trained. Recent works [96, 93] showed that adversarial examples fooling the surrogate model can also fool the target model with a high success rate. A simpler variant is when the surrogate model has access to the same training data as the interested network. Adversarial examples fooling the substituted network are usually transferred to (and fool) the target model as well. The latter method is the one we will follow for our evaluations when we conduct our experiments.

One-Pixel Attack

While all of the adversarial attacks have to deal with the perceptiveness problem, which means they should create adversarial perturbations that are imperceptible to human eyes, one-pixel attack [94] avoids this problem by generating adversarial examples under the extreme constraint of only modifying the value of one pixel. The one-pixel attack finds the optimized solution $e(x^*)$ in

$$\underset{e(x^*)}{\text{maximize}} \;\; f_{adv}(x + e(x)) \;\; s.t. \;\; ||e(x)||_0 \leq d \tag{2.8}$$

where in the case of one pixel attack d is 1, and what this formulation does is to maximize the loss of the network by perturbing at most d pixels. Su *et al.* applied Differential Evolution (DE) [111] to find the optimal solution. DE is a member of the Evolutionary Algorithms (EAs). Optimization function of Equation (2.8) is solved iteratively; in each iteration, a population of candidates is generated, in which each candidate contains at most d perturbed pixels and each perturbation is encoded by a tuple of 5 elements; (x, y) coordinates and the RGB values of the perturbed pixel. Creating the next generation of the pixels follows the usual EA formula

$$x_i\,(g+1) = \; x_{r_1}\,(g) + \; F\,(\,x_{r_2}\,(g) - x_{r_3}\,(g)\,) \;\; s.t. \;\; r_1 \neq r_2 \neq r_3 \tag{2.9}$$

where x_i is an element of the candidate solution in the previous population (g) of the DE algorithm, r_i is a random number, and F is the scale parameter which Sue *et al.* set to be 0.5.

The DE algorithm does not require the gradients of the network; instead, it only uses the softmax values, which makes it a *black-box* attack, and it can be effective against non-differential models (e.g. neural networks that are non-differential or other ML models like decision trees) as well. By using the DE algorithm slowly changing the value of single pixels, the one-pixel attack essentially estimates the gradient of the network with only having the black-box access; therefore, this method is a gradient estimation technique that does not need to train surrogates in order to fool the networks. Figure 2.7 shows some

examples of applying the one-pixel attack on an ImageNet [1] classifier. As can be seen, modifying the value of only one pixel on a large image (224×224 pixels) can fool the ImageNet classifier with a high confidence.

2.3 Countermeasures For Adversarial Attacks

Adversarial defense mechanisms commonly consist of two groups of algorithms: detection algorithms (reactive) [112, 113, 114, 84, 115, 116, 117], and the algorithms that train neural networks in special ways or augment them with modules during training, so that, DNNs would become robust against adversarial attacks (proactive) [118, 119, 120, 28, 121]. Although both sets of these algorithms use similar techniques and assumptions for counteracting adversarial attacks, the ones that focus on improving the robustness of a model are more popular and the majority of work in this field is done on equipping DNNs with modules and techniques that would improve their self-defense strength.

It is possible to categorize the proactive robustness algorithms into two main classes:

1. **Modifying the network's training or the training data:** methods such as adversarial training [23, 25, 122, 123, 124], data compression [125], foveation mechanism [126], and data randomization [127] that improve the robustness without changing the network itself.

2. **Network modification:** methods that modify the networks by adding more layers or augmenting the model with new modules, or just changing the regular loss function. Numerous algorithms fall in this category; deep contrastive networks [128], gradient regularization/masking, feature denoising [129], defensive distillation [130, 131, 132], logit pairing technique [133], DeepCloak [134], and network randomization [135, 136, 137] are just some of the more well-known techniques of this class of defensive techniques.

Figure 2.7: One-pixel attack on the ImageNet dataset; the perturbed pixel in each image is highlighted with a red circle, and original predictions with their confidence are in black, whereas predictions for adversarial images are in blue [94]. **Image from** [94].

In this section, we will discuss two of the most popular adversarial defense techniques that have been the building blocks of many of the recent novel defense mechanisms, namely, adversarial training and network randomization.

2.3.1 Adversarial Training

Adversarial training is considered as a very intuitive and brute-force yet very promising solution to improve the robustness of DNN models against adversarial attacks. The idea is simple; add the adversarial samples to the training data. The adversarial samples are generated by using an adversary during the training. A useful feature of adversarial training is that it is possible to combine it with other techniques; therefore, almost every powerful defensive technique incorporates its idea with adversarial training [54, 55, 138]. Adversarial training formally solves the following min-max problem:

$$\arg\min_{W} \Big[\arg\max_{||\delta||_p \leq \epsilon} \mathcal{L}\Big(f(X+\delta; W), T \Big) \Big] \tag{2.10}$$

where $(X, T) \in D$ (training data), W encodes the network parameters, and δ is perturbation with the constraint that $||\delta||_p \leq \epsilon$. The inner maximization is done by an adversary which maximizes the loss function $\mathcal{L}$ for the (X, T) pair; while the outer maximization can be the regular categorical cross-entropy loss minimization which helps the network learn from the training samples. To simply put, during the adversarial training, the adversary tries to craft the strongest perturbations while the model learns to cope with these perturbations. Adversarial training was first proposed in the seminal work of [23]; in which the authors first introduced the FGSM adversary, and then used this adversary during the training to help train robust models. However, since FGSM is not a very strong adversary, adversarial training with FGSM cannot train models with guaranteed robustness against other types of adversarial attacks. To overcome this problem, Madry *et al.* proposed PGD adversarial training [25].

Guaranteed First-order Robustness

Projected Gradient Descent (PGD) [25] is a strong adversarial attack algorithm, which makes it a good adversary to use during adversarial training. This is the idea of PGD adversarial training. Madry *et al.* [25] illustrated that adversarial training using Projected Gradient Descent (PGD) for generating on-the-fly adversarial samples during training can lead to trained models which provide robustness guarantees against all first-order adversaries. They experimentally showed that the adversarial examples in a l_∞ ball distance around the original sample with many random starts in the ball generated with PGD, all have approximately the same loss value when are fed to the network as input. Due to this fact, they provide the guarantee that as long as the attack algorithm is a first-order adversary, the local maxima of the loss value would not be significantly better than those found by PGD. Therefore, PGD is the universal first-order adversary, which means no other first-order adversary can have better performance than that of PGD, so robustness against PGD means guaranteed robustness.

2.3.2 Network Randomization

Another approach for training robust models in the recent years has been applying regularization techniques during training (and in some cases during inference as well) [139]. To do so, either new loss functions are proposed with added or embedded regularization terms (i.e., adversarial generalization) [140, 141, 142] or the network is augmented with new modules [143, 144, 145, 146] for regularization purposes making the network more robust at the end.

Network regularization with randomization approaches has recently proved to be an efficient tool for improving DNN robustness. A random noise generator as an extra module is embedded in the network architecture adding random noise to the input or the output of layers. Although the noise distribution usually follows a Gaussian distribution for its simplicity, it is possible to use different noise distributions. This noise augmentation technique

adds more uncertainty in the network and makes the adversarial attack optimization harder which improves the robustness of the model. More recently, works of [147] and [148] have theoretically proven that network randomization can result in certified robustness against all adversaries, which makes network randomization a very powerful tool for improving adversarial robustness. Parametric Noise Injection (PNI) [143], and Adversarial Bayesian Neural Network (Adv-BNN) [146] are two of the recent state-of-the-art adversarial defense mechanisms that take advantage of both network randomization and adversarial training. These two methods have similar characteristics to our proposed Learn2Perturb framework, and in our experiments, we compare the performance of Learn2Perturb to those of PNI and Adv-BNN; therefore, in the remainder of this section, we will explain these two methods.

Parametric Noise Injection

In the PNI [143] method, Gaussian noise is added to the tensors of different layers of the network. For a tensor v which can be the weights, the biases, or the layer activations of each layer the noise is injected in this way:

$$\tilde{v}_{l,i} \;=\; v_{l,i} \;+\; \alpha_l \;.\; \eta_{l,i} \;; \qquad \eta_{l,i} \approx \mathcal{N}(0, \sigma_l^2) \tag{2.11}$$

where $v_{l,i}$ shows the i-th element of the tensor v at layer l, α_l is a constant coefficient, and σ_l shows the standard deviation parameter corresponding to the tensor v. σ_l controls the amount of the injected noise and is learned during the training. $\tilde{v}_l$ is the final value of the tensor v after injecting η_l. Figure 2.8 visualizes this process in the PNI technique; a noise tensor with the same size as the clean tensor is sampled from a Gaussian distribution, and then, this noise is added to the clean tensor to create the noisy output.

Since the noise injection parameters in the PNI method are trained with the network gradients, during the training once the network starts to stabilize, gradients of the network force the noise injection parameters converge to zero. To overcome this issue, they take advantage of PGD adversarial training. In addition to helping with the convergence-to-zero problem, PGD adversarial training boosts the robustness of PNI against the adversarial

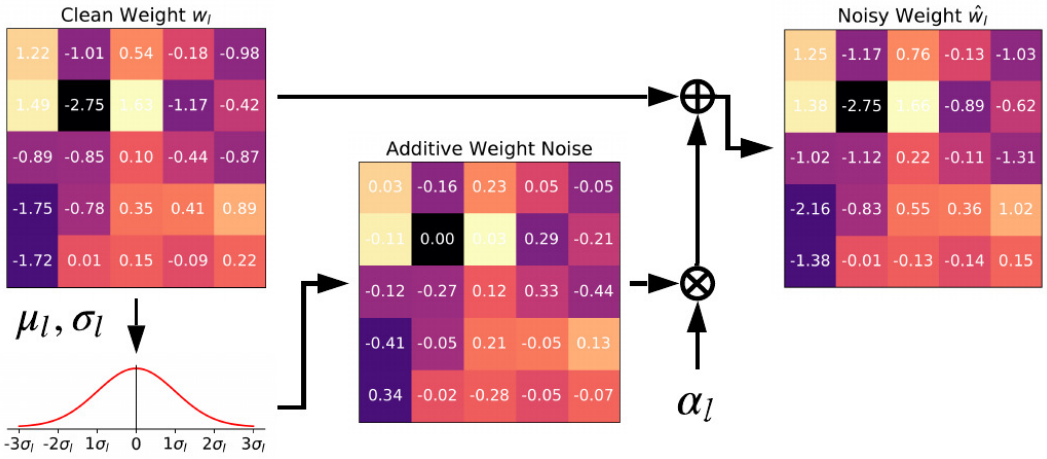

Figure 2.8: PNI injects trainable Gaussian noise to different tensors of a neural network. The noise is parameterized and its parameters can be learned during the training. The noise is injected during the inference stage as well [143]. **Image from [143]**.

algorithms. However, even with the PGD adversarial training, noise parameters eventually converge to zero.

Adv-BNN

In the Adv-BNN method [146], the ideas of Bayesian Neural Network (BNN) and PGD adversarial training are combined together. Figure 2.9 illustrates the idea of BNNs; given the random variables (x, y) from the training data, a Bayesian network tries to find the latent distribution of the network variables, w. Since finding $p\,(w \,|\, x, y) = \frac{p\,(x,y\,|\,w)\,p(w)}{p\,(x,y)}$ is intractable, they approximate it by using a parametric distribution $q_\theta\,(w)$, and they estimate θ minimizing $KL(\,q_\theta\,(w)\;||\;p\,(w\,|\,x,y)\,)$. This means that, a BNN model learns a distribution parameterized with θ for the network parameters, w. This is usually a Gaussian distribution which yields $w \sim \mathcal{N}(\mu_\theta, \sigma_\theta)$. Therefore, in order to learn the distribution of w, Adv-BNN solves the following optimization function

$$\mathcal{L}(\mu, \sigma) = -KL(\,q_{\mu,\sigma}\,(w)\;||\;p\,(w)\,) + \sum_{(x,y)\in D} \min_{||x^{adv}-x||\leq\epsilon} \mathbb{E}_{w\sim q_{\mu,\sigma}}\; log\, p(y|x^{adv}, w) \qquad (2.12)$$

where μ and σ are the parameters indicating a Gaussian distribution, and x^{adv} is the adversarial samples generated with the PGD adversary during the training.

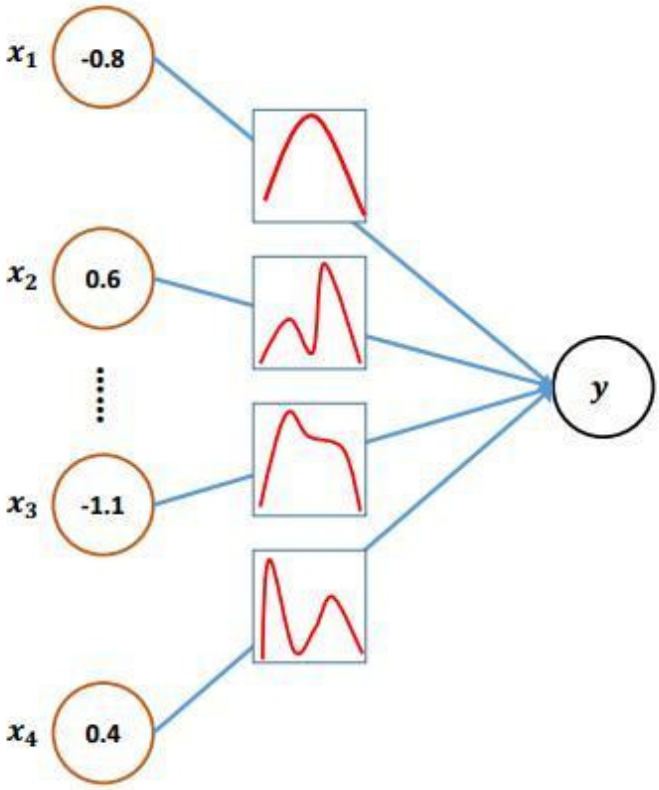

Figure 2.9: Illustration of how BNN works for a toy example neural network. Given the observable $(\mathbf{x}, \mathbf{y})$ random variables, the posterior distributions of the network weights are learned. **Image from** [146].

Adv-BNN training is a lot more computationally expensive than other adversarial robustness techniques. During the inference, for each feed-forward operation, the network needs to sample the weights of the network from their Gaussian distribution; this puts a lot of computational load on the inference step as well. Another drawback of Adv-BNN is that if the trained model has a small capacity then it will not have a good clean data performance; however, for very large models, Adv-BNN can train models that have both high clean data accuracy and a good level of robustness.

Chapter 3

Methodology

In this work, we propose a new framework called Learn2Perturb for improving the adversarial robustness of a deep neural network through end-to-end feature perturbation learning. Although it has been illustrated both theoretically and practically [136, 148] that randomization techniques can improve the robustness of deep neural networks, there is still not an effective way to select the distribution of the noise in the neural networks. In Learn2Perturb, trainable perturbation-injection modules are integrated into a deep neural network with the goal of injecting customized perturbations into the feature space at different parts of the network to increase the uncertainty of its inner workings within an optimal manner. We formulate the joint problem of learning the model parameters and the perturbation distributions of the perturbation-injection modules in an end-to-end learning framework via an alternating back-propagation approach [149]. As shown in Figure 3.1, the proposed alternating back-propagation strategy for the joint learning of the network parameters and the perturbation-injection modules is inspired from the Expectation-Maximization (EM) technique; and it comprises of two key alternating steps: i) **Perturbation-injected network training**: the network parameters are trained by gradient descent while the proposed perturbation-injection modules add layer-wise noise to the feature maps (different locations in the network). Noise injection parameters are

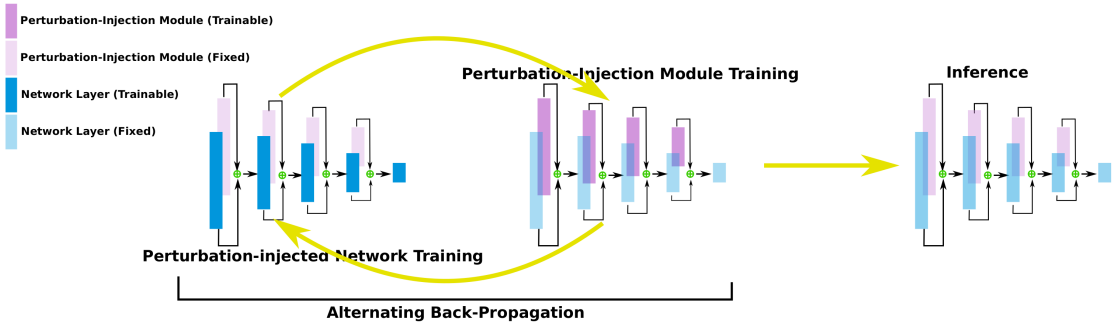

Figure 3.1: Overview of Learn2Perturb: During training, an alternating back-propagation strategy is introduced where the following two steps are performed in an alternating manner: i) the network parameters are updated in the presence of feature perturbation injection to improve adversarial robustness, and ii) the parameters of the perturbation injection modules are updated to strengthen perturbation capabilities against the improved network. The learned perturbation injection modules can be added to some or all tensors in the network to inject perturbations in feature space for two-prong adversarial robustness: i) improve robustness during training when training under perturbation injection, and ii) increase network uncertainty through interference-time perturbation injection to make it difficult to learn an adversarial attack.

fixed during this step. ii) **Perturbation-injection module training**: the parameters of the perturbation-injection modules are updated via gradient descent and based on the regularization term added to the network loss function, while network parameters are fixed.

The effect of using such a training strategy is that in step (i), the model minimizes the loss function of the classification problem when noise is being injected into multiple layers, and the model learns how to classify despite the injected perturbations. And in step (ii), the noise parameters are updated with a combination of network gradients and the regularization term applied to these parameters. The goal of this step is to let the network react to the noise injections via gradient descent and pose a bigger challenge to the network via a smooth increase of noise based on the regularizer. The trained perturbation-injection modules perturb the feature layers of the model in the inference phase as well.

3.1 Perturbation-Injection Distribution

Given the observable variables X, W as the input and the set of weights in the neural network, respectively, the goal is to model the neural network as a probabilistic model such that the output of the model, Y, is a random variable rather than a deterministic function. A probabilistic output is more robust against adversarial perturbation. As such, Y can be formulated as:

$$Y \sim P(X; W, \theta) \tag{3.1}$$

where W and θ show the set of network and noise parameters, respectively, and X is the input fed into the network. The output Y is a random variable parameterized with W and the set of independent parameters, θ.

For a given layer l of the neural network, the perturbation-injection modules can be used to achieve the following probability model for the layer's final activations:

$$P_l(X_l; W_l, \theta_l) \approx f_l(X_l, W_l) + Q(\theta_l) \tag{3.2}$$

where $f_l(X_l, W_l)$ represents the activation of layer l with weights W_l, X_l as its input, and $Q(\theta_l)$ is a random variable from the Gaussian noise distribution with θ_l as its standard deviation. While $Q(\cdot)$ can be any exponential distribution, we choose Gaussian distribution because of its simplicity and effectiveness, which can be formulated as follows:

$$Q(\theta_l) \approx \theta_l \cdot \mathcal{N}(0, 1) \tag{3.3}$$

The parameter θ_l scales the magnitude of the output from the normal distribution encoding the standard deviation of the distribution $Q(\cdot)$. Substituting the right hand-side of $Q(\cdot)$ defined in Equation (3.3) into Equation (3.2) enforces $P_l(\cdot)$ to follow a Gaussian distribution:

$$P_l(X_l; W_l, \theta_l) \approx \mathcal{N}\Big(f_l(X_l, W_l), \theta_l\Big) \tag{3.4}$$

This new probabilistic formulation of layer activations can be extended to the whole network, so instead of a deterministic output Y, network outputs $P(X; W, \theta) \approx \mathcal{N}\Big(f(X, W), \theta\Big)$, with W and θ showing the parameters of all layers.

Having this new formulation for a deep neural network, a proper training process to effectively learn both sets of these parameters is highly desired. To this end, we propose a new training mechanism to learn both network parameters and perturbation-injection modules in an alternating back-propagation approach.

3.2 Alternating Back-Propagation

The proposed neural network structure comprises of two sets of parameters, W and θ, being trained given training samples (X, T) as the input and the ground truth output to the network. However, these two sets of parameters are in conflict with each other and try to push the learning process in two opposite directions. Having the probabilistic representation $P(\cdot)$, W is mapping the input X to output T based on the mean of the distribution $P(\cdot)$, $f(X, W)$; while, the set of θ improves the generalization of the model by including perturbations into the training mechanism.

The proposed alternating back-propagation framework decouples the learning process associated with network parameters W and perturbation-injection distributions θ to effectively update both sets of parameters. To this end, the network parameters and perturbation-injection modules are updated in a consecutive manner.

The training process of the proposed Learn2Perturb is done within two main steps:

- *Perturbation-injected network training*; the parameters of the network, W, are updated via gradient descent to decrease the network loss in the presence of perturbations, caused by the currently fixed perturbation-injection distribution, $Q|_\theta$.
- *Perturbation-injection distribution training*; the parameters of the perturbation-injection distribution, θ, are updated given the set of parameters W are fixed to improve the

generalization of the network and as a result, improve its robustness against adversarial perturbation.

These two steps are performed consecutively; however, the number of iterations for each step before moving to the next step can be determined based on the application.

Algorithm 1: Alternating back-propagation of the Learn2Perturb framework

Input : Training set D $= \{(x_i, t_i), \ \ i = 1, \ldots, n\}$
Number of training epochs, I
θ_{min}, the lower bound for θ
θ^0, initial values for θ
Learning rate, lr, and constant γ

Output : Learned parameters W
Learned noise distributions $Q(\theta)$

for $t \leftarrow 1$ *to* I **do**

Perturbation-injected training: update W based on the loss function $\mathcal{L}(\cdot)$ Eq. (3.5) while θ is fixed

$W^t \leftarrow W^{t-1} - lr \cdot \nabla_W \mathcal{L}\Big(P(X; W^{t-1}, \theta^{t-1}), T\Big)$

Perturbation-injection module training:
update θ based on Eq. (3.5) while W is fixed

$\theta^t \leftarrow \theta^{t-1} - lr \cdot \nabla_\theta \mathcal{L}\Big(P(X; W^{t-1}, \theta^{t-1}), T\Big) - \gamma \cdot \nabla_\theta g(\theta^{t-1})$

Values of θ^t smaller than θ_{min} are projected to θ_{min}

end

Utilizing a generic loss function in the training of the network when the perturbation-injection modules are embedded forces the noise parameters to converge to zero and eventually removes the effect of the perturbation-injection distributions by making them very small. In other words, the neural network with generic loss tends to learn $P(\cdot)$ as a Dirac distribution where the $Q(\cdot)$ is close to zero. As such, a new regularization term is designed and added to the loss function to prevent the aforementioned problem; the new loss

function can be formulated as:

$$\arg\min_{W,\theta}\Big[\mathcal{L}\Big(P(X;W,\theta),T\Big)+\gamma\cdot g(\theta)\Big] \tag{3.5}$$

where $\mathcal{L}(\cdot)$ is the classification loss function (i.e., usually cross entropy) such that the set of parameters W need to be tuned to generate the associated output of the input X. The function $g(\theta)$ is the regularizer enforcing smooth increase in the parameters $\theta=\{\theta_{l,j}\}_{j=1:M_l}^{l=1:K}$, where $\theta_{l,j}$ shows the j^{th} noise parameter in the l^{th} layer, corresponding to an element of the output feature map. K and M_l represent the number of layers and noise parameters per layer, respectively. γ is the hyper-parameter balancing the two terms in the optimization. Independent distributions are learnt for perturbation-injection models in each layer. The regularizer function should be enforced with an annealing characteristic where the perturbation-injection distributions are gradually improved and converged thus the parameters W can be trained effectively. As such the regularization function is formulated as below:

$$g(\theta)=-\frac{\theta^{1/2}}{\tau} \tag{3.6}$$

where τ is the output of a harmonic series given the current epoch value in the training process. Using a harmonic series to determine τ, gradually decreases the effect of the regularizer function in the loss and lets the neural network converge. While the squared root of θ makes the equation easier to take the derivative, it also provides a slower rate of change for larger values of θ which helps the network to converge to a steady state smoothly. We formulate τ as below:

$$\tau(t)=\sum_{i=s}^{t}\frac{1}{i-s-1} \tag{3.7}$$

where t shows the current epoch, while s shows the first epoch number from which noise is being added to the network.

As seen in Algorithm 1, first, the perturbation-injection distributions Q and network parameters W are initialized. Then the model parameters W are updated based on the

classification loss $\mathcal{L}(\cdot)$, and this loss function is minimized in the presence of perturbation-injection modules. Then, the perturbation-injection distributions Q are updated by performing the "perturbation-injection module training" step.

One of the main advantages of this approach is that since the learning process of these two sets of parameters is decoupled, the training process can be easily performed without a significant manual hyper-parameter tweaking compared to other randomized state-of-the-art approaches. Moreover, the proposed method can help the model to converge faster as the perturbation-injection distributions are continuously improved during the training process.

3.3 Adding PGD Adversarial Training to Our Randomization Technique

We take advantage of adversarial training technique which adds on-the-fly adversarial examples into the training data, to improve the model's robustness more effectively against perturbations. As such, PGD adversarial technique is incorporated in the training to provide stronger guarantee bounds against all first-order adversaries optimizing in l_∞ space. Adding PGD adversarial training to the alternative back-propagation technique can be formulated as:

$$\arg\min_{W,\theta}\Big[\arg\max_{||\delta||_\infty\leq\epsilon}\mathcal{L}\Big(P(X+\delta;W,\theta),T\Big)\Big] \tag{3.8}$$

where W encodes the network parameters and θ shows the perturbation-injection parameters. In this formulation only adversarially generated samples are used in the training step for the outer minimization, following the original work introduced in [25].

Finally, in order to balance between the adversarial robustness and clean data accuracy [143, 55], we formulate the adversarial training as follow:

$$\arg\min_{W,\theta}\Big[\alpha\cdot\mathcal{L}\Big(P(X;W,\theta),T\Big)+\beta\cdot\arg\max_{||\delta||_\infty\leq\epsilon}\mathcal{L}\Big(P(X+\delta;W,\theta),T\Big)\Big] \tag{3.9}$$

where the first term shows the loss associated to the clean data and α is the weight for the clean data loss term, while the second shows the loss associated with the adversarially generated data with weight β. The models trained with the proposed Learn2Perturb algorithm use $\alpha = \beta = 0.5$. Equation (3.9) helps gain adversarial robustness, while maintaining a reasonably high clean data accuracy.

Equation (3.9) shows the main objective function of our Learn2Perturb framework which is optimized by using gradient descent.

3.4 Model Setup, Training, and Inference

Perturbation-injection distributions are added to the network in different locations and specifically after each convolution operation to create a new network model based on the Learn2Perturb framework. As shown in Figure 3.1, these modules generate the perturbations with the same size as the feature activation maps of that specific layer. Each perturbation-injection distribution follows independent distribution and therefore, the generated perturbation value for each feature is drawn independently.

In the training phase, the model parameters and the perturbation-injection distributions are trained in an iterative and consecutive manner and based on the proposed alternating back-propagation approach. It is worth to mention that the model parameters are trained for 20 epochs before activating the perturbation distributions to help the network parameters converge to a good initial point. After 20 epochs, the alternating back-propagation is applied to train both model parameters and perturbation-injection distributions.

The perturbation-injection distributions are applied in the inference step, as well. Adding randomization during inference will introduce a dynamic nature into the inference process and as a result, it makes it harder for the adversaries to find optimal adversarial examples to fool the network.

Chapter 4

Experimental results

In this chapter, exhaustive experimental results are conducted to evaluate the robustness and the efficiency of our proposed framework against a large set of adversarial attack algorithms; furthermore, we provide experiments comparing our framework with different state-of-the-art approaches including PGD adversarial training [25] (also denoted as Vanilla model), Parametric Noise Injection (PNI) [143], Adversarial Bayesian Neural Network (Adv-BNN) [146], Random Self-Ensemble (RSE) [145] and PixelDP (DP) [144]. Experimental results indicate that our Learn2Perturb technique outperforms other methods by a large margin.

4.1 Experimental Setup

In order to encourage the reproducible experimental results, in this section, we provide a detailed explanation of the experimental setup and environment of the reported experiments. Pytorch version 1.2 was used for developing all experiments, and our codes are available at https://github.com/Ahmadreza-Jeddi/Learn2Perturb.

4.1.1 Baseline Architecture

We use ResNet based architectures [150] as the baseline for our experiments; The classical ResNet architecture (i.e., ResNet-V1 and its variations) and the new ResNet architecture (i.e., ResNet-V2) are used for evaluation. The main difference between the two architectures is the number of stages and the number of blocks in each stage. Moreover, average pooling is utilized for down-sampling in ResNet-V1 architecture while the ResNet-V2 uses 1×1 CNN layers for this purpose. Following the observation made by Madry *et al.* [25], the capacity of networks alone can help to increase the robustness of the models against adversarial attacks. As such, we compare Learn2Perturb and competing state-of-the-art methods for various networks with different capacities.

The ResNet [150] architecture has been selected as the baseline network followed by the state-of-the-art methods and the fast convergence property of this architecture. The effect of network depth is evaluated by examining the competing methods via ResNet-V1(32), (44), (56) as well as ResNet-V1(20) where (x) shows the depth of the network. Moreover, the effect of network width is examined similar to the work done by Zagoruyko and Komodakis [53]. To increase the width of the network (i.e, experiment performed on ResNet-V1(20)), the number of input and output channels of each layer is increased by a constant multiplier, $\times 1.5$, $\times 2$, and $\times 4$ which widens the ResNet architecture. However, we do not follow the exact approach of [53] in which they applied dropout layers in the network; instead, we just increase the width of the basic convolution at each layer by increasing the number of input/output channels.

We also consider a ResNet-V2(18), which has a very large capacity compared to ResNet-V1 architecture. Not only the number of channels have increased in this architecture but also it uses 1×1 convolutions to perform the down-sampling at each residual block.

4.1.2 Datasets

For the evaluation purpose, the CIFAR-10 and CIFAR-100 datasets [110] are utilized for training and evaluating the networks. Both of these datasets contain 50,000 training data and 10,000 test data of natural color images of 32×32. While CIFAR-10 has 10 different class with 6000 images per class, CIFAR-100 has 100 classes with 600 images per class.

4.1.3 Attack Algorithms

Different white-box and black-box attacks are utilized to evaluate the proposed Learn2Perturb along with state-of-the-art methods. The competing algorithms are evaluated via white-box attacks including FGSM [18], PGD [109] and C&W attacks [29]. One-Pixel attack [94], and Transferability attack [97] are utilized as the black-box attacks to evaluate the competing methods.

4.1.4 Training and Inference Parameters

Followed by the experimental setup proposed in [143], data normalization is done via adding a non-trainable layer at the beginning of the network and the adversarial perturbations are directly added to the original input data, before normalization being applied. Both adversarial training and robustness testing setup follow the same configurations as introduced in [25] and [143]. Adversarial training with PGD and testing robustness against PGD are both done in 7 iterations with the maximum $l_\infty = 8/255$ (i.e., ϵ) and step sizes of 0.01 for each iteration. FGSM attack also uses the same $8/255$ limit for perturbation. For C&W attack, we use ADAM [151] optimizer with learning rate $5e^{-4}$. Maximum number of iterations is 1000, and for the constant c in (2.4) we choose the range $1e^{-3}$ to $1e^{10}$; furthermore to find the value of c, binary search with up to 9 steps is performed. The confidence, κ, parameter of C&W attack, which turns out to have a big effect while evaluating defense approaches involving randomization, takes values ranging from 0 to 5.

The proposed Learn2Perturb, No defense, and Vanilla methods use the same setup for gradient descent optimizer. SGD optimizer with momentum of 0.9 with Nesterov momentum and weight decay of $1e^{-4}$ is used for the training of those methods. The noise injection parameters have weight decay equal to 0. We use the batch size of 128, and 350 epochs to train the model. The initial learning rate is 0.1, then changes to 0.01 and 0.001 at epochs 150 and 250, respectively.

For the parameter γ in Equation (3.5), we choose value 10^{-4} for all of our experiments.

4.2 Analyzing Learn2Perturb

In this section, we first experimentally show that Learn2Perturb is an excellent technique for introducing randomization into neural networks, and then, we introduce a weaker variant of Learn2Perturb (Learn2Perturb-R) and evaluate both the original Learn2Perturb and this variant on the CIFAR-10 dataset. More experiments and comparisons with state-of-the-art techniques are covered in the next two sections.

4.2.1 Behaviour of Noise Distributions in Learn2Perturb *vs.* PNI

As our first experiment, we decide to compare the noise injection ability of our Learn2Perturb method to the similar work of PNI [143]. As stated in Section 2.3.2, the trained noise parameters by the PNI approach fluctuate during the training due to their objective function. The min-max optimization applied in PNI causes the training to enforce noise parameters to converge to zero as the number of training epochs increases.

This issue has been addressed in the proposed Learn2Perturb algorithm by introducing a new regularization term in the loss function. As a result, there is a trade-off between training proper perturbation-injection distribution and modeling accuracy during the training step. This trade-off would let the perturbation modules to learn properly and eventually

converge to a steady state. To this end, a harmonic series term is introduced in the proposed regularization term (Equation (3.6)) which decreases the effect of regularization as the number of training epochs increases and helps the perturbation-injection modules to converge.

Figure 4.1 shows the behaviour of noise distributions in both PNI and Learn2Perturb algorithms during the training. As seen, the proposed Learn2Perturb algorithm manages to smoothly increase the magnitude of injected noise, until finally, noise parameters converge to a steady state. However, in the case of PNI, the noise distributions are forced to zero, due to the way the loss function is formulated.

4.2.2 Reduced Learn2Perturb

We also evaluate a variation of the proposed Learn2Perturb framework (i.e. Learn2Perturb-R) where we analyze a different approach in performing the two steps of "perturbation-injected network training" and "perturbation-injection module training". In this variation, the perturbation-injection modules are only updated using the regularizer function $g(\theta)$, and network gradients are not used to update θ parameters.

To evaluate Learn2Perturb and Learn2Perturb-R, here, the two techniques are compared with Vanilla (models that trained using only PGD adversarial training), and no-defense (models trained with no knowledge of adversarial samples, the common training) models as two common reference points. The comparisons are done on three different ResNet architectures. Table 4.1 shows the effectiveness of the proposed Learn2Perturb method in improving the robustness of different network architectures. Results demonstrate that the proposed perturbation-injection modules improve the network's robustness. As seen, the proposed perturbation-injection modules can provide robust performance on both 'ResNet-V1' (with both 20 and 56 layers) and 'ResNet-V2' (18 layers) architectures against both FGSM and PGD attacks which illustrates the effectiveness of the proposed module in providing more robust networks.

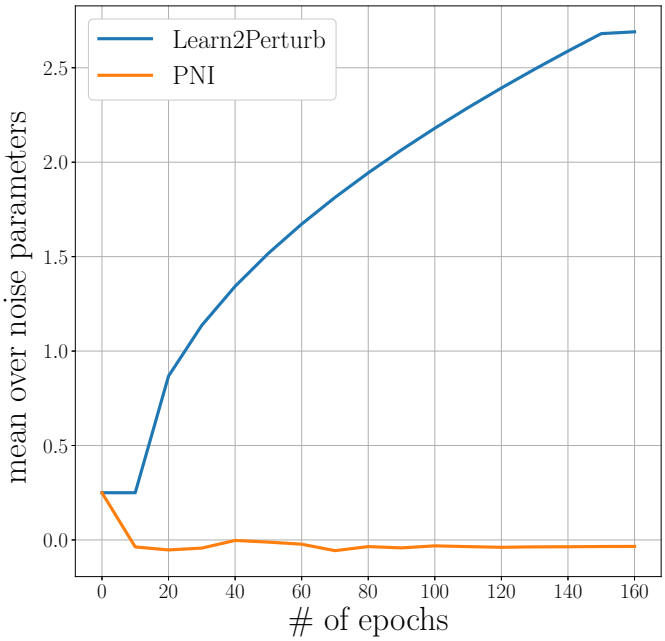

Figure 4.1: Evolution of mean over noise perturbation parameters through training epochs for ResNet-V2. As seen, while the magnitude of noise distributions are increasing in the Learn2Perturb algorithm, they converge to zero in the PNI method. The network with larger noise distributions faces larger uncertainty and learns to deal with such perturbations. Larger noise also makes it hard for an adversary to craft optimized adversarial perturbations; this is because randomization affects network gradient as well.

Moreover, as it can be seen in Table 4.1, taking advantage of both network gradient and the regularizer performs better than when we only take the regulizer into account. One reason to justify this outcome is allowing the gradient of loss function $\mathcal{L}(\cdot)$ to update perturbation-injection modules in Learn2Perturb. This would let the loss function to react to perturbations when they cannot tolerate the injected noise and updates the perturbation-injection noise modules more frequently. Nonetheless, the results in Table 4.1 show that Learn2Perturb-R still outperforms other the Vanilla method in adversarial robustness, though it provides slightly lower accuracy on clean data.

Additionally, it can be seen in Table 4.1 that clean data accuracy of a model might be at odds with its robustness. This further indicates the findings of Zhang *et al.* [55]; adversarial examples come from a different distribution than that of the clean data, therefore, there might be a trade-off between the two goals of the adversarial robustness and clean data accuracy. Recently, the works of [56] and [152] show that increasing the size of data and hence exposing network to a larger distribution during training can enhance both these goals.

4.3 CIFAR-10 Robustness Comparison

In this section, to further illustrate the effectiveness of the proposed Learn2Perturb framework, we compare the performance of Learn2Perturb with PNI [143] and Adv-BNN [146] as two state-of-the-art randomization approaches to improve on the CIFAR-10 dataset.

4.3.1 Evaluating Competing Models Against l_∞ PGD and FGSM

Table 4.2 reports comparison results against PGD and FGSM adversaries for different network architectures varying in network depth and width. We examine the effect of different network depths including ResNet-V1(20), ResNet-V1(32), ResNet-V1(44) and ResNet-V1(56) along with the effect of network width in Table 4.2 by increasing the number

Table 4.1: Evaluating the **accuracy** of the proposed Learn2Perturb and its variant (Learn2Perturb-R) and comparing them with the adversarial training algorithm (Vanilla) and a model with no defense. All the baselines are ResNets. V1 refers to ResNet-V1 and V2 refers to ResNet-V2. (x) indicates the depth of the models. C, P, and F denote Clean data accuracy, PGD accuracy, and FGSM accuracy, repectively. For the evaluations that contain randomness, we have only reported the mean of their results.

	No defense			**Vanilla[25]**			**Learn2Perturb-R**			**Learn2Perturb**		
Model	**C**	**P**	**F**	**C**	**P**	**F**	**C**	**P**	**F**	**C**	**P**	**F**
V1(20)	92.10	0.0	14.10	83.80	39.1	46.60	81.15	50.23	55.89	83.62	51.13	58.41
V1(56)	93.30	0.0	24.20	86.50	40.1	48.80	82.35	53.30	58.71	84.82	54.84	61.53
V2(18)	95.20	0.0	43.10	85.46	43.9	52.50	82.46	53.33	59.09	85.30	56.06	62.43

of filters in ResNet-V1(20) which results to ResNet-V1(20)[1.5×], ResNet-V1(20)[2×] and ResNet-V1(20)[4×]. As can be seen in this Table, while the competing methods do not provide consistent performance for different capacities of the network (increasing depth or width), the proposed framework provides consistent robustness through different network capacities. Furthermore, not only learn2Perturb significantly outperforms other methods against adversaries, but it also manages to maintain a competitive performance on clean data (data with no adversarial perturbations).

The reported results in Table 4.2 show that while PNI provides minor boosting in network accuracy on clean data, the proposed Learn2Perturb method performs with much higher accuracy when the input data is perturbed with adversarial noise. The main reason for this phenomenon is the fact that PNI reaches a very low level of noise perturbation during the training as the loss function tries to remove the effect of perturbation by making the noise parameters to zero. The results demonstrate that the proposed Learn2Perturb algorithm outperforms the PNI method by 4-7% on both FGSM and PGD adversarial attacks. The proposed method is also compared with Adv-BNN [146]. Results show that while Adv-BNN can provide robust networks in some cases compared to PNI, it is not scalable when the network width is increased and the performance of the networks drop drastically. This further indicates one of the drawbacks of Bayesian approaches; they need to be carefully designed for each network architecture, and the setup for one architecture might not work for another one; this causes major scalability problems.

We also analyze the effectiveness of the proposed method in dealing with different adversarial noise levels. To this end, the ResNet-V2(18) architecture is utilized for all competing methods. The network architectures are designed and trained via four different competing methods, and the trained networks are examined with both FGSM and PGD attacks but with a ranging variation of ϵ values.

Figure 4.2 demonstrates the robustness of four competing methods in dealing with the FGSM adversarial attack. As seen, while increasing ϵ decreases the robustness of all trained networks, the network designed and trained by the proposed Learn2Perturb

Table 4.2: The effect of network capacity on the **accuracy** of the proposed method and other state-of-the-art algorithms. The proposed Learn2Perturb is compared with Parametric Noise Injection (PNI) method [143] and Adv-BNN [146]. Results shows the effectiveness of the proposed Learn2Perturb algorithm in training robust neural network models. To have a fair comparison, we evaluated methods on different network sizes and capacities. Result are reported by standard deviation because of the randomness involved in these methods. All the baselines are ResNets. V1 refers to ResNet-V1 and V2 refers to ResNet-V2. (x) indicates the depth of the models. C, P, and F denote Clean data accuracy, PGD accuracy, and FGSM accuracy, repectively.

	PNI [143]			**Adv-BNN [146]**			**Learn2Perturb**		
Model	**C**	**P**	**F**	**C**	**P**	**F**	**C**	**P**	**F**
V1(20)	84.90±0.1	45.90±0.1	54.50±0.4	65.76±5.92	44.95±1.21	51.58±1.49	83.62±0.02	**51.13±0.08**	58.41±0.07
V1(32)	85.90±0.1	43.50±0.3	51.50±0.1	62.95±5.63	50.42±0.06	50.29±2.70	84.19±0.06	**52.62±0.06**	59.94±0.11
V1(44)	84.70±0.2	48.50±0.2	55.80±0.1	76.87±0.24	51.33±0.03	58.55±0.49	85.61±0.01	**53.24±0.03**	61.32±0.13
V1(56)	86.80±0.2	46.30±0.3	53.90±0.1	77.20±0.02	51.16±0.13	57.88±0.02	84.82±0.04	**54.75±0.07**	61.53±0.04
V1(20)[1.5×]	86.00±0.1	46.70±0.2	54.50±0.2	65.58±0.42	28.07±1.11	36.11±1.29	85.40±0.08	**53.32±0.02**	61.10±0.06
V1(20)[2×]	86.20±0.1	46.10±0.2	54.60±0.2	79.03±0.04	53.46±0.06	58.30±0.14	85.89±0.10	**54.29±0.02**	61.61±0.05
V1(20)[4×]	87.70±0.1	49.10±0.3	57.00±0.2	82.31±0.03	52.61±0.12	59.01±0.04	86.09±0.05	**55.75±0.07**	61.32±0.02
V2(18)	87.21±0.0	49.42±0.01	58.06±0.02	82.15±0.06	53.62±0.06	60.04±0.01	85.30±0.09	**56.06±0.08**	62.43±0.06

Table 4.3: Comparison results of the **accuracy** of the proposed Learn2Perturb and state-of-the-art methods in providing a robust network model. Some of the numbers are extracted from [143]. The reported results are either based on the maximum accuracy achieved in the literature or our own results if we achieved higher level of accuracy.

Defense Method	Model	Clean	PGD
RSE [145]	ResNext	87.5	40
DP [144]	28-10 wide ResNet	87	25
Adv-BNN [146]	ResNet-V1(56)	77.20	54.62±0.06
PGD adv. training [25]	ResNet-V1(20) [4×]	87	46.1±0.1
PNI [143]	ResNet-V1(20) [4×]	**87.7±0.1**	49.1±0.3
Learn2Perturb	ResNet-V2(18)	85.3±0.1	**56.3±0.1**

approach outperforms other methods through all variations of adversarial noise values (ϵ's).

To confirm the results shown in Figure 4.2, a similar experiment is conducted to examine the robustness of the trained networks against PGD attack. Figure 4.3 shows this version. While the PGD attack is more powerful in fooling the networks, results show that the network designed and trained by the proposed Learn2Perturb framework still outperforms other state-of-the-art approaches.

In order to compare Learn2Perturb with a wider range of defensive techniques, Table 4.3 shows the best reported results of the recent state-of-the-art approaches. The proposed Learn2Perturb method outperforms other state-the-art methods and provides a more robust network with better performance when dealing with PGD attack.

4.3.2 Evaluation of Competing Models Against L_2 C&W Attack

It has been shown that there is no guarantee that methods robust against l_∞ attacks would provide the same level of robustness against l_2 attacks [136]. Araujo *et al.* [136] experimentally illustrated that randomization during the training of a model trained with

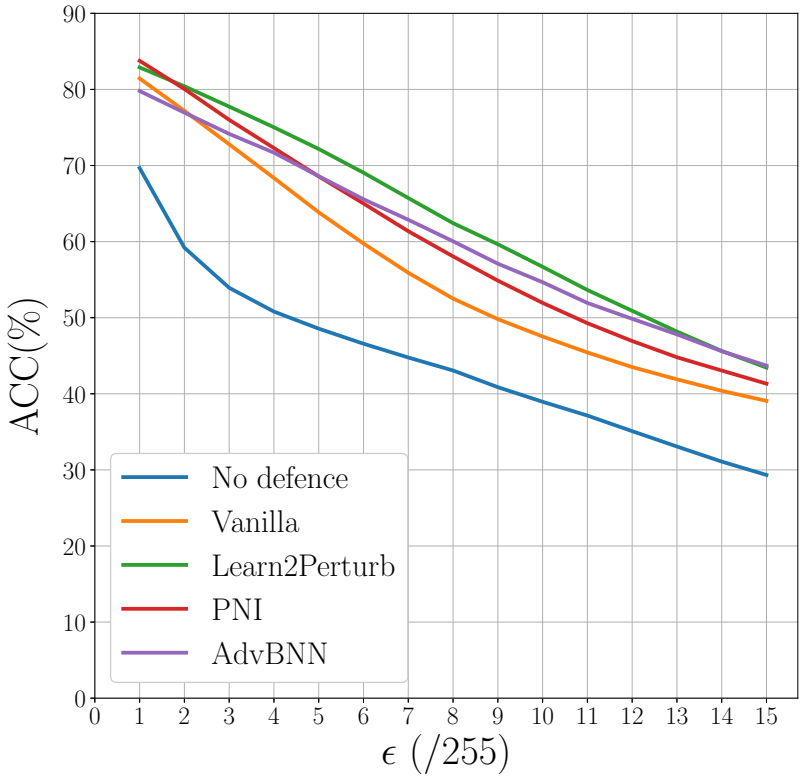

Figure 4.2: Analyzing the effectiveness of the proposed method compared to state-of-the-art algorithms on different ϵ values for FGSM attack for the task of image classification on CIFAR-10 dataset.

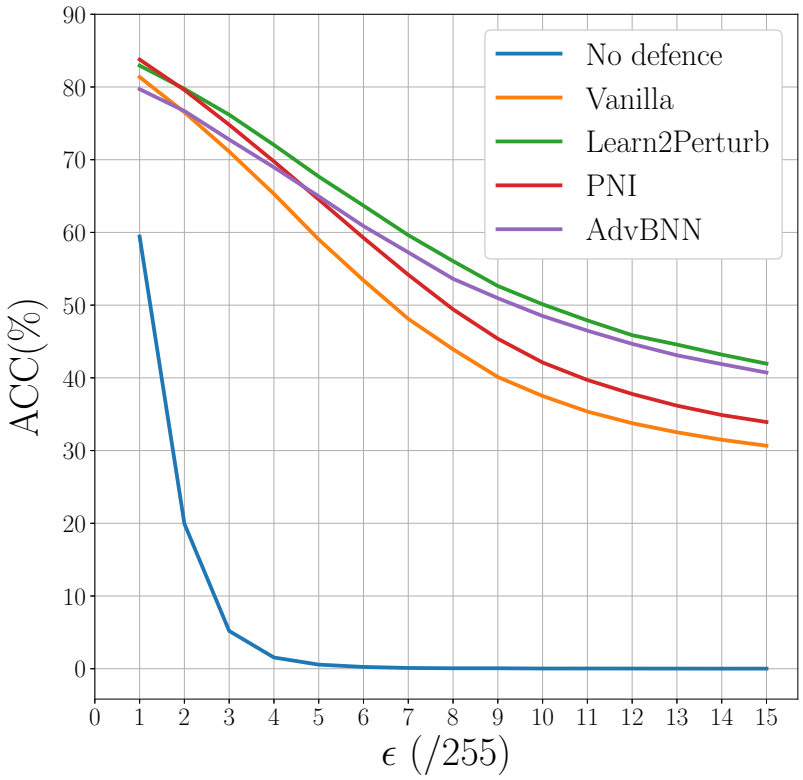

Figure 4.3: Evaluating the robustness of the proposed Learn2Perturb compared with other state-of-the-art methods through different ϵ based on PGD attack.

Table 4.4: Comparison results of the **accuracy** of the proposed Learn2Perturb and competing methods on C&W [29] attack.

Confidence	**No defense**	**Vanilla**	**PNI**	**Adv-BNN**	**Learn2Perturb**
$\kappa = 0.0$	0.0	0.0	66.9	78.9	**83.6**
$\kappa = 0.1$	0.0	0.0	66.1	78.1	**84.0**
$\kappa = 1.0$	0.0	0.0	34.0	65.1	**76.4**
$\kappa = 2.0$	0.0	0.0	16.0	49.1	**66.5**
$\kappa = 5.0$	0.0	0.0	0.8	16.0	**34.8**

an l_∞ adversary (e.g. PGD) can improve the robustness against l_2 attacks as well. In this work, we further validate this finding. In order to provide more powerful l_2 attacks challenging the effect of randomization, we apply C&W attacks with different confidence values, κ. The parameter κ enforces the $f(\cdot)$ in (2.4) to be $\leq -\kappa$ rather than simply ≤ 0. As seen in Table 4.4, for bigger values of κ the success rate of the C&W attack increases; nonetheless, our proposed method outperforms the other competing methods with a big margin for all values of κ.

4.3.3 Expectation Over Transformation (EOT)

Athalye *et. al* [153] showed that many of the defense algorithms that take advantage of injecting randomization to network's interior layers or applying random transformations on the input before feeding it to the network achieve robustness through false stochastic gradients. They further stated that these methods obfuscate the gradients that attackers utilize to perform iterative attacking optimizations. As such, they proposed the EOT attack (originally introduced in [154]) to evaluate these types of defense mechanisms. They showed that the false gradients cannot protect the network when the attack uses the gradients which are the expectation over a series of transformations.

Since our Learn2Perturb algorithm and other competing methods involve randomization, the tested algorithms in this study are evaluated via the EOT attack method as well.

To do so, followed by [148], at every iteration of PGD attack, the gradient is achieved as the expectation calculated from a Monte Carlo method with 80 simulations of different transformations. Results show that the network trained via PNI can provide 48.65% robustness compared to Adv-BNN which provides 51.19% robustness for the CIFAR-10 dataset against this attack. The experimental result illustrates that the proposed Learn2Perturb approach can produce a model that achieves 53.34% robustness and outperforms the other two state-of-the-art algorithms.

It is worth mentioning that the experimental results showed that neither the proposed Learn2Perturb method nor the other competing approaches studied in this work suffer from obfuscated gradients. Furthermore, the proposed Learn2Perturb method successfully passes the five metrics introduced in [153], and thus further illustrates that Learn2Perturb is not subjected to obfuscated gradients.

4.4 CIFAR-10 Robustness Against Black-Box Attacks

In this section, the robustness of the proposed method and the competing algorithms against black-box attacks are evaluated. Two different attacks including few-pixel attack [94] and transferability attack [97] are used to evaluate the competing methods.

4.4.1 Few-Pixel Attack

For the few pixel attack, we use a population size of 400 and maximum iteration steps of 75 for the differential evolution algorithm. The attack strength is controlled by the number of pixels that are allowed to be modified. In this comparison, we consider the {1,2,3}-pixel attacks.

Table 4.5 shows the comparison results of the competing methods against few-pixel attack. Two different network architectures (ResNet-V1(20) and ResNet-V2(18)) are used

Table 4.5: Few-pixel attack; the **accuracy** of the competing methods are evaluated via few-pixel [94] attack base on two network architectures of ResNet-V1(20) and ResNet-V2(18). {1,2,3} pixels are changed in the test samples to perturbed the images.

Network	Attack Strength	No defense	Vanilla	PNI	Adv-BNN	Learn2Perturb
ResNet-V1(20)	**1-pixel**	21.45	65.20	67.40	58.40	**70.15**
	2-pixel	2.55	48.35	61.75	56.20	**63.90**
	3-pixel	1.10	36.40	58.10	55.70	**61.85**
ResNet-V2(18)	**1-pixel**	23.44	56.10	50.90	**68.60**	64.45
	2-pixel	3.20	33.20	39.00	**64.55**	60.05
	3-pixel	0.95	23.95	35.40	**59.70**	53.90

to evaluate the competing algorithms. As seen, the proposed Learn2Perturb method outperforms other state-of-the-art methods when the baseline network architecture is ResNet-V1(20). However, Adv-BNN provides better performance when the baseline network architectures are ResNet-V2(18), while the proposed Learn2Perturb algorithm provides comparable performance for this baseline.

4.4.2 Transferability Attack

As the most practical attack and threat to DNN models, transferability attack refers to cases where adversarial samples of a model can fool another DNN or general ML model as well. Here, we evaluate the robustness of the competing models against the transferable adversarial samples. Table 4.6 demonstrates the results for this comparison. Results again show that the proposed Learn2Perturb method provides robust predictions against this attack as well; while our results are not significantly better than the competing models, Table 4.6 indicates that samples transfered from other methods have a smaller success rate in fooling Learn2Perturb than that of the PGD attack.

Table 4.6: Comparing the robustness of models against transferability attack; the competing methods are attacked within the context of Transferability where the perturbed images utilized to evaluate the robustness of the model are generated by another method. The 'Source Model' is the model which the perturbed samples are generated from to attack each competing algorithm.

		Vanilla		**PNI**		**Adv-BNN**		**Learn2Perturb**	
Network	**Source Model**	**FGSM**	**PGD**	**FGSM**	**PGD**	**FGSM**	**PGD**	**FGSM**	**PGD**
Resnet20 - V1	**Vanilla**	–	–	60.32±0.05	58.27±0.01	49.22±0.90	48.63±3.10	58.86±0.03	56.75 ± 0.01
	PNI	66.31±0.02	63.04±0.01	–	–	51.12±1.22	49.59±0.83	63.26±0.10	59.31 ± 0.06
	Adv-BNN	74.38±0.16	72.02±0.02	73.05±0.12	70.26±0.05	–	–	72.48±0.05	69.25 ± 0.06
	Learn2Perturb	70.66±0.01	67.32±0.01	68.46±0.03	64.77±0.01	54.16±2.36	52.23±1.52	–	–
Resnet18 - V2	**Vanilla**	–	–	69.52±0.02	68.01±0.02	67.20±0.04	65.88±0.03	67.32±0.04	65.58 ± 0.04
	PNI	69.56±0.01	67.09±0.02	–	–	67.63±0.03	64.87±0.04	67.63±0.05	64.61 ± 0.01
	Adv-BNN	73.66±0.01	71.23±0.01	74.02±0.03	71.51±0.02	–	–	71.67±0.09	68.75 ± 0.04
	Learn2Perturb	73.49±0.01	70.44±0.00	73.89±0.04	70.61±0.01	70.22±0.04	67.33±0.04	–	–

4.5 CIFAR-100 Robustness Comparison

A detailed analysis of the experimental results for the CIFAR-100 dataset is provided as follows. The CIFAR-100 dataset [110] is very similar to the CIFAR-10 dataset, however, the image samples are categorized into 100 class labels. All the models involving PGD adversarial training are trained with $\epsilon = \frac{8}{255}$ during training. Figures 4.4 and 4.5 demonstrate the performance comparison of the proposed Learn2Perturb with other state-of-the-art methods on CIFAR-100 dataset based on FGSM and PGD attacks.

As seen, the proposed Learn2Perturb method outperforms other competing algorithms for epsilons up to $\frac{8}{255}$; however for bigger *epsilon* values it provides comparable performance with Adv-BNN, which has the best result. It is worth mentioning that for the models that have smaller capacity (in terms of the number of network parameters), Adv-BNN lacks the ability to converge to a stable well-performing model; moreover, its training is highly computationally expensive, which makes it a less practical technique.

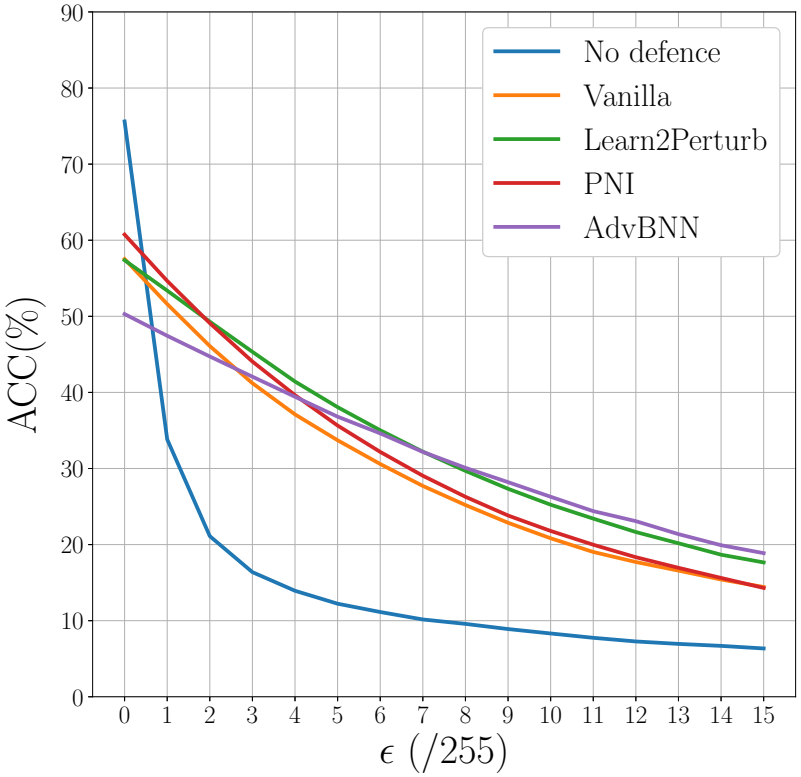

Figure 4.4: FGSM attack on CIFAR-100 with different epsilons for the l_∞ ball on ResNet-V2(18).

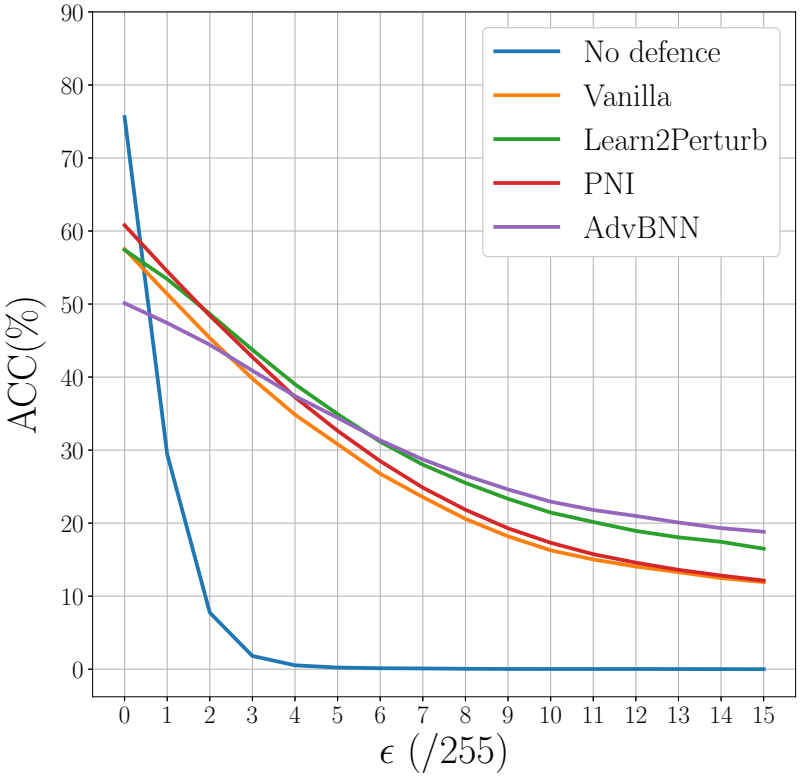

Figure 4.5: PGD attack on CIFAR-100 with different epsilons for the l_∞ ball on ResNet-V2(18).

Chapter 5

Conclusion

In this thesis, we explored the concept of adversarial examples and the effects that these examples can have on real-life systems. We provided explanations of some of the more plausible hypotheses on why adversarial examples exist. We also reviewed the common, strongest white-box and black-box adversarial attack algorithms that are used to evaluate and compare our proposed framework with the competing state-of-the-art methods. Finally, after describing some of the adversarial defense mechanisms and their strengths and weaknesses, we introduced our Learn2Perturb framework, an end-to-end feature perturbation learning approach for improving adversarial robustness of deep neural networks. Experimental results provided in Chapter 4 showed that Learn2Perturb can significantly boost the adversarial robustness of the DNN models.

5.1 Thesis Contribution Insights

We introduced a new algorithm for improving the adversarial robustness of computer vision models. Parameterized perturbation injection modules were introduced to increase uncertainty during both training and inference to make it harder to craft successful adversarial examples. Moreover, a novel alternating back-propagation approach was introduced

to learn both network parameters and perturbation-injection module parameters in an alternating fashion. To overcome the problem of noise parameters converging to zero a new regularization term was added to the objective function that the network tries to minimize in order to learn a robust generalized model (refer to Equation (3.5)).

The goal of this regularization term is to first stop noise distribution from converging to zero and then, smoothly increase the magnitude of the injected noise so that more uncertainty is introduced during both training and inference. Figure 4.1 compares our noise injection technique with the PNI [143] method; as can be seen in this Figure, we are able to inject better customized noise to the network.

Experimental results on both different black-box and white-box attacks demonstrated the efficacy of the proposed Learn2Perturb algorithm, which outperformed the state-of-the-art methods in improving robustness against different adversarial attacks. On the CIFAR-10 dataset for different network capacities, we improve the l_∞ FGSM and PGD robustness by $4-7\%$ while maintaining high clean data accuracy. For the l_2 C&W attack, we constantly outperform competing methods by a large margin for different values of κ (C&W attack confidence). This further indicates the finding of Araujo *et al.* [136] that using randomization during adversarial training with an l_∞ adversary can improve the robustness against l_2 attacks. We also show that our model is very robust against black-box few-pixel and transferable attacks. Similarly, on the CIFAR-100 dataset, we outperform the state-of-the-art robustness against the FGSM and PGD adversaries.

5.2 Future Work

Although our Learn2Perturb framework succeeds in learning models that have either better or comparable adversarial robustness compared to state-of-the-art approaches, it has some limitations as well. The major limitation of Learn2Perturb is that even though it has a comparable clean-data accuracy (performance on data with no adversarial perturbations) to other methods, the reduction of this accuracy due to the adversarial training during

the learning process might deprive real-world applications of the complete benefit of the neural networks. In future work, we will focus on addressing this problem and designing adversarially robust models with no loss in clean data generalization.

To address the abovementioned generalization problem and train more robust models, future work involves combining the Learn2Perturb framework with some of the recent ideas that have been proposed in the field of adversarial robustness. Especially, two lines of work have shown great success in this field; first are the ones that work on the sample complexity of adversarial examples believing that adversarial robustness and better generalization require more training data than simple standard training. The second line of work focuses more on designing better adversarial training and regularization techniques. These two lines of work and how we believe Learn2Perturb can take advantage of them are explained in the remainder of this section.

5.2.1 More Training Data Can Improve Robustness

In their recent work, Schmidt *et al.* [152] experimentally showed that the PGD adversarial training on CIFAR-10 dataset tends to highly overfit on the training data; based on this observation they claimed that adversarial robustness requires a larger sample complexity, and then, they theoretically proved that adding more training data can help improve the adversarial robustness. Following Schmidt *et al.* [152], the very recent works of [56] and [155] showed that even using *unlabeled* data can increase the adversarial robustness and the generalization of the classification models.

A direction for us is to study a defense mechanism that has larger training data and takes advantage of randomization. Carmon *et al.* [56] augmented the CIFAR-10 dataset with 500K unlabeled data which are achieved by applying semi-supervised techniques on the 80 Million Tiny Images dataset [156]. Moreover, they showed that adding these unlabeled data during the training of CIFAR-10 classifiers can improve the robustness of these models by almost 10%. Recently, Yalniz *et al.* [157] augmented the ImageNet dataset by using semi-supervised techniques on the YFCC-100M dataset [158]. Since the augmented

unlabeled data might not have the same quality as the original training data, the main focus of the training is still on the original dataset; however, the augmented dataset is utilized as well though less frequent than the original one, so that the model is exposed to a larger input distribution.

Augmenting the training data by applying semi-supervised techniques on unlabeled data may not always be a viable option, since in some cases the overall amount of the data (labeled or unlabeled) is small. Moreover, augmenting the training data means that the training process would be more computationally expensive. Nonetheless, this technique is simple to implement, and it has been shown to be very effective for training robust models. We believe that increasing the size of the actual training data together with randomization can help train models that are more robust to out-of-distribution samples. Therefore, designing better semi-supervised models for extracting relevant unlabeled data and more efficient sampling techniques over these unlabeled data would help our Learn2Perturb framework.

5.2.2 Advanced Adversarial Training Techniques

Although the PGD adversarial training [25] is a very effective method which yields guaranteed first-order robustness, simply generating adversarial examples for every training sample, independent from others, cannot achieve high levels of certified robustness. Therefore, some recent works have tried to improve PGD adversarial training. Instead of maximizing the network loss with respect to the input sample (as it is done in PGD adversarial training), for each clean sample, Zhang *et al.* [55] generated a duplicate and iteratively maximized the l_2 distance between the $softmax$ vectors of the two samples. In a more recent work, instead of independently generating adversarial examples for each image, Zhang *et al.* [54] generated adversarial samples for groups of images, and they achieve a very high performance on CIFAR-10 (73% adversarial robustness).

Works like [55] and [54] indicate that the relationships among different training samples or even among different classes can have a big impact on the robustness of the trained

model. We have conducted a set of experiments that further validate these findings; furthermore, we notice that the classification of some of the classes in a dataset might be harder than other classes; for example, in the CIFAR-10 dataset the two classes of *cat* and *dog* are often misclassified as one another. This is even more intensified when the model is under adversarial attack. Since the simple adversarial training generates adversarial samples with no consideration of these relations, it seems that this method is not sufficient to find robust decision boundaries between those kind of classes.

Designing new better adversarial training methods like the ones of [54] and [55] combined with randomization techniques can help find more robust decision boundaries, which would promote both the generalization and the adversarial robustness of neural networks. Since it is easy to combine randomization techniques with other methods, we believe that we can take advantage of the recent adversarial training techniques and extend the Learn2Perturb idea to further improve the adversarial robustness.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[2] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[3] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[4] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[5] Liang-Chieh Chen et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848.

[6] Alexander Toshev and Christian Szegedy. "Deeppose: Human pose estimation via deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1653–1660.

[7] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[8] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

[9] Tom Ko et al. "Audio augmentation for speech recognition". In: *Sixteenth Annual Conference of the International Speech Communication Association*. 2015.

[10] Rushlene Kaur Bakshi et al. "Opinion mining and sentiment analysis". In: *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE. 2016, pp. 452–455.

[11] Aaron van den Oord et al. "Wavenet: A generative model for raw audio". In: *arXiv preprint arXiv:1609.03499* (2016).

[12] Scott Reed et al. "Generative adversarial text to image synthesis". In: *arXiv preprint arXiv:1605.05396* (2016).

[13] Ahmad EL Sallab et al. "Deep reinforcement learning framework for autonomous driving". In: *Electronic Imaging* 2017.19 (2017), pp. 70–76.

[14] Chenyi Chen et al. "Deepdriving: Learning affordance for direct perception in autonomous driving". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2722–2730.

[15] June-Goo Lee et al. "Deep learning in medical imaging: general overview". In: *Korean journal of radiology* 18.4 (2017), pp. 570–584.

[16] Andrew Boles and Paul Rad. "Voice biometrics: Deep learning-based voiceprint authentication system". In: *2017 12th System of Systems Engineering Conference (SoSE)*. IEEE. 2017, pp. 1–6.

[17] Zhenlong Yuan et al. "Droid-sec: deep learning in android malware detection". In: *Proceedings of the 2014 ACM conference on SIGCOMM*. 2014, pp. 371–372.

[18] Christian Szegedy et al. "Intriguing properties of neural networks". In: *arXiv preprint arXiv:1312.6199* (2013).

[19] Cihang Xie et al. “Adversarial examples for semantic segmentation and object detection”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 1369–1378.

[20] Dawn Song et al. “Physical adversarial examples for object detectors”. In: *12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18)*. 2018.

[21] Nicholas Carlini and David Wagner. “Audio adversarial examples: Targeted attacks on speech-to-text”. In: *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE. 2018, pp. 1–7.

[22] Moustafa Alzantot et al. “Generating natural language adversarial examples”. In: *arXiv preprint arXiv:1804.07998* (2018).

[23] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and harnessing adversarial examples”. In: *arXiv preprint arXiv:1412.6572* (2014).

[24] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. “Adversarial examples in the physical world”. In: *arXiv preprint arXiv:1607.02533* (2016).

[25] Aleksander Madry et al. “Towards deep learning models resistant to adversarial attacks”. In: *arXiv preprint arXiv:1706.06083* (2017).

[26] Florian Tramèr et al. “The space of transferable adversarial examples”. In: *arXiv preprint arXiv:1704.03453* (2017).

[27] Thomas Tanay and Lewis Griffin. “A boundary tilting persepective on the phenomenon of adversarial examples”. In: *arXiv preprint arXiv:1608.07690* (2016).

[28] David Stutz, Matthias Hein, and Bernt Schiele. “Disentangling adversarial robustness and generalization”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6976–6987.

[29] Nicholas Carlini and David Wagner. “Towards evaluating the robustness of neural networks”. In: *2017 IEEE symposium on security and privacy (sp)*. IEEE. 2017, pp. 39–57.

[30] Ahmadreza Jeddi et al. “Learn2Perturb: an End-to-end Feature Perturbation Learning to Improve Adversarial Robustness”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 1241–1250.

[31] Kaiming He et al. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[32] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[33] Joseph Redmon and Ali Farhadi. “YOLO9000: better, faster, stronger”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.

[34] Yang Long et al. “Accurate object localization in remote sensing images based on convolutional neural networks”. In: *IEEE Transactions on Geoscience and Remote Sensing* 55.5 (2017), pp. 2486–2498.

[35] Spyros Gidaris and Nikos Komodakis. “Object detection via a multi-region and semantic segmentation-aware cnn model”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1134–1142.

[36] Liang-Chieh Chen et al. “Rethinking atrous convolution for semantic image segmentation”. In: *arXiv preprint arXiv:1706.05587* (2017).

[37] Kaiming He et al. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[38] Bernardino Romera-Paredes and Philip Hilaire Sean Torr. “Recurrent instance segmentation”. In: *European conference on computer vision*. Springer. 2016, pp. 312–329.

[39] Bo Zhu et al. “Image reconstruction by domain-transform manifold learning”. In: *Nature* 555.7697 (2018), pp. 487–492.

[40] Jo Schlemper et al. “A deep cascade of convolutional neural networks for dynamic MR image reconstruction”. In: *IEEE transactions on Medical Imaging* 37.2 (2017), pp. 491–503.

[41] Marvin Teichmann et al. “Multinet: Real-time joint semantic reasoning for autonomous driving”. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2018, pp. 1013–1020.

[42] Moritz Menze and Andreas Geiger. “Object scene flow for autonomous vehicles”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3061–3070.

[43] Bradley J Erickson et al. “Machine learning for medical imaging”. In: *Radiographics* 37.2 (2017), pp. 505–515.

[44] Baris Kayalibay, Grady Jensen, and Patrick van der Smagt. “CNN-based segmentation of medical imaging data”. In: *arXiv preprint arXiv:1701.03056* (2017).

[45] Dmitry Kalashnikov et al. “Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation”. In: *arXiv preprint arXiv:1806.10293* (2018).

[46] Fangyi Zhang et al. “Towards vision-based deep reinforcement learning for robotic motion control”. In: *arXiv preprint arXiv:1511.03791* (2015).

[47] Khan Muhammad et al. “Efficient deep CNN-based fire detection and localization in video surveillance applications”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 49.7 (2018), pp. 1419–1434.

[48] Chenqiang Gao et al. “People counting based on head detection combining Adaboost and CNN in crowded surveillance environment”. In: *Neurocomputing* 208 (2016), pp. 108–116.

[49] Quoc V Le. “Building high-level features using large scale unsupervised learning”. In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 8595–8598.

[50] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[51] Xiaoyong Yuan et al. “Adversarial examples: Attacks and defenses for deep learning”. In: *IEEE transactions on neural networks and learning systems* 30.9 (2019), pp. 2805–2824.

[52] Saining Xie et al. “Aggregated residual transformations for deep neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1492–1500.

[53] Sergey Zagoruyko and Nikos Komodakis. “Wide residual networks”. In: *arXiv preprint arXiv:1605.07146* (2016).

[54] Haichao Zhang and Jianyu Wang. “Defense against adversarial attacks using feature scattering-based adversarial training”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 1831–1841.

[55] Hongyang Zhang et al. “Theoretically principled trade-off between robustness and accuracy”. In: *arXiv preprint arXiv:1901.08573* (2019).

[56] Yair Carmon et al. “Unlabeled data improves adversarial robustness”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 11192–11203.

[57] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “” Why should I trust you?” Explaining the predictions of any classifier”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.

[58] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Deep inside convolutional networks: Visualising image classification models and saliency maps”. In: *arXiv preprint arXiv:1312.6034* (2013).

[59] Seunghoon Hong et al. “Online tracking by learning discriminative saliency map with convolutional neural network”. In: *International conference on machine learning*. 2015, pp. 597–606.

[60] Walt Woods, Jack Chen, and Christof Teuscher. "Adversarial explanations for understanding image classification decisions and improved neural network robustness". In: *Nature Machine Intelligence* 1.11 (2019), pp. 508–516.

[61] Lukas Schott et al. "Towards the first adversarially robust neural network model on MNIST". In: *arXiv preprint arXiv:1805.09190* (2018).

[62] Ajil Jalal et al. "The robust manifold defense: Adversarial training using generative models". In: *arXiv preprint arXiv:1712.09196* (2017).

[63] Kevin Eykholt et al. "Robust physical-world attacks on deep learning visual classification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1625–1634.

[64] Nicholas Carlini et al. "Hidden voice commands". In: *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 2016, pp. 513–530.

[65] Guoming Zhang et al. "Dolphinattack: Inaudible voice commands". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 103–117.

[66] Nicolas Papernot et al. "Towards the science of security and privacy in machine learning". In: *arXiv preprint arXiv:1611.03814* (2016).

[67] Battista Biggio et al. "Evasion attacks against machine learning at test time". In: *Joint European conference on machine learning and knowledge discovery in databases*. Springer. 2013, pp. 387–402.

[68] Pedro Tabacof and Eduardo Valle. "Exploring the space of adversarial images". In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2016, pp. 426–433.

[69] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. "Analysis of classifiers' robustness to adversarial perturbations". In: *Machine Learning* 107.3 (2018), pp. 481–508.

[70] Seyed-Mohsen Moosavi-Dezfooli et al. "Analysis of universal adversarial perturbations". In: *arXiv preprint arXiv:1705.09554* (2017).

[71] Seyed-Mohsen Moosavi-Dezfooli et al. "Robustness of classifiers to universal perturbations: A geometric perspective". In: *International Conference on Learning Representations*. 2018.

[72] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. "Robustness of classifiers: from adversarial to random noise". In: *Advances in Neural Information Processing Systems*. 2016, pp. 1632–1640.

[73] Andras Rozsa, Manuel Günther, and Terrance E Boult. "Are accuracy and robustness correlated". In: *2016 15th IEEE international conference on machine learning and applications (ICMLA)*. IEEE. 2016, pp. 227–232.

[74] Justin Gilmer et al. "Adversarial spheres". In: *arXiv preprint arXiv:1801.02774* (2018).

[75] Naveed Akhtar and Ajmal Mian. "Threat of adversarial attacks on deep learning in computer vision: A survey". In: *IEEE Access* 6 (2018), pp. 14410–14430.

[76] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. "Deepfool: a simple and accurate method to fool deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2574–2582.

[77] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[78] Alexandru Constantin Serban, Erik Poll, and Joost Visser. "Adversarial examples-a complete characterisation of the phenomenon". In: *arXiv preprint arXiv:1810.01185* (2018).

[79] Zhengli Zhao, Dheeru Dua, and Sameer Singh. "Generating natural adversarial examples". In: *arXiv preprint arXiv:1710.11342* (2017).

[80] Tom B Brown et al. "Unrestricted adversarial examples". In: *arXiv preprint arXiv: 1809.08352* (2018).

[81] Yang Song et al. "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples". In: *arXiv preprint arXiv:1710.10766* (2017).

[82] Hyeungill Lee, Sungyeob Han, and Jungwoo Lee. "Generative adversarial trainer: Defense to adversarial perturbations with gan". In: *arXiv preprint arXiv:1705.03387* (2017).

[83] Partha Ghosh, Arpan Losalka, and Michael J Black. "Resisting adversarial attacks using gaussian mixture variational autoencoders". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 541–548.

[84] Dongyu Meng and Hao Chen. "Magnet: a two-pronged defense against adversarial examples". In: *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 2017, pp. 135–147.

[85] Nicholas Carlini and David Wagner. "Adversarial examples are not easily detected: Bypassing ten detection methods". In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 2017, pp. 3–14.

[86] Can Kanbak, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. "Geometric robustness of deep networks: analysis and improvement". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4441–4449.

[87] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. "Prior convictions: Black-box adversarial attacks with bandits and priors". In: *arXiv preprint arXiv:1807.07978* (2018).

[88] P Samangouei, M Kabkab, and R Defense-GAN Chellappa. "Protecting classifiers against adversarial attacks using generative models. arXiv 2018". In: *arXiv preprint arXiv:1805.06605* (2018).

[89] Dimitris Tsipras et al. "Robustness may be at odds with accuracy". In: *arXiv preprint arXiv:1805.12152* (2018).

[90] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[91] Gregory Cohen et al. “EMNIST: Extending MNIST to handwritten letters”. In: *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2017, pp. 2921–2926.

[92] Reza Shokri et al. “Membership inference attacks against machine learning models”. In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2017, pp. 3–18.

[93] Nicolas Papernot et al. “Practical black-box attacks against machine learning”. In: *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. ACM. 2017, pp. 506–519.

[94] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. “One pixel attack for fooling deep neural networks”. In: *IEEE Transactions on Evolutionary Computation* 23.5 (2019), pp. 828–841.

[95] Yinpeng Dong et al. “Discovering adversarial examples with momentum”. In: *CoRR abs/1710.06081* (2017).

[96] Yanpei Liu et al. “Delving into transferable adversarial examples and black-box attacks”. In: *arXiv preprint arXiv:1611.02770* (2016).

[97] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. “Transferability in machine learning: from phenomena to black-box attacks using adversarial samples”. In: *arXiv preprint arXiv:1605.07277* (2016).

[98] Pin-Yu Chen et al. “Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models”. In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 2017, pp. 15–26.

[99] Weiwei Hu and Ying Tan. “Generating adversarial malware examples for black-box attacks based on gan”. In: *arXiv preprint arXiv:1702.05983* (2017).

[100] Moustafa Alzantot et al. “Genattack: Practical black-box attacks with gradient-free optimization”. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. 2019, pp. 1111–1119.

[101] Arjun Nitin Bhagoji et al. “Practical black-box attacks on deep neural networks using efficient query mechanisms”. In: *European Conference on Computer Vision*. Springer. 2018, pp. 158–174.

[102] Arjun Nitin Bhagoji et al. “Exploring the space of black-box attacks on deep neural networks”. In: *arXiv preprint arXiv:1712.09491* (2017).

[103] Seyed-Mohsen Moosavi-Dezfooli et al. “Universal adversarial perturbations”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1765–1773.

[104] Shumeet Baluja and Ian Fischer. “Adversarial transformation networks: Learning to generate adversarial examples”. In: *arXiv preprint arXiv:1703.09387* (2017).

[105] Nicolas Papernot et al. “The limitations of deep learning in adversarial settings”. In: *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE. 2016, pp. 372–387.

[106] Andrew Ilyas et al. “Black-box adversarial attacks with limited queries and information”. In: *arXiv preprint arXiv:1804.08598* (2018).

[107] Pu Zhao et al. “On the design of black-box adversarial examples by leveraging gradient-free optimization and operator splitting method”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 121–130.

[108] Javid Ebrahimi et al. “Hotflip: White-box adversarial examples for text classification”. In: *arXiv preprint arXiv:1712.06751* (2017).

[109] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. “Adversarial machine learning at scale”. In: *arXiv preprint arXiv:1611.01236* (2016).

[110] Alex Krizhevsky and Geoffrey Hinton. “Learning multiple layers of features from tiny images”. In: (2009).

[111] Swagatam Das and Ponnuthurai Nagaratnam Suganthan. “Differential evolution: A survey of the state-of-the-art”. In: *IEEE transactions on evolutionary computation* 15.1 (2010), pp. 4–31.

[112] Jiajun Lu, Theerasit Issaranon, and David Forsyth. “Safetynet: Detecting and rejecting adversarial examples robustly”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 446–454.

[113] Jan Hendrik Metzen et al. “On detecting adversarial perturbations”. In: *arXiv preprint arXiv:1702.04267* (2017).

[114] Weilin Xu, David Evans, and Yanjun Qi. “Feature squeezing: Detecting adversarial examples in deep neural networks”. In: *arXiv preprint arXiv:1704.01155* (2017).

[115] Weilin Xu, David Evans, and Yanjun Qi. “Feature squeezing mitigates and detects carlini/wagner adversarial examples”. In: *arXiv preprint arXiv:1705.10686* (2017).

[116] Lewis Smith and Yarin Gal. “Understanding measures of uncertainty for adversarial example detection”. In: *arXiv preprint arXiv:1803.08533* (2018).

[117] Bin Liang et al. “Detecting adversarial image examples in deep neural networks with adaptive noise reduction”. In: *IEEE Transactions on Dependable and Secure Computing* (2018).

[118] Tianyu Pang et al. “Improving adversarial robustness via promoting ensemble diversity”. In: *arXiv preprint arXiv:1901.08846* (2019).

[119] Chongli Qin et al. “Adversarial robustness through local linearization”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 13847–13856.

[120] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. “Fundamental limits on adversarial robustness”. In: *Proc. ICML, Workshop on Deep Learning*. 2015.

[121] Nicholas Carlini et al. “On evaluating adversarial robustness”. In: *arXiv preprint arXiv:1902.06705* (2019).

[122] Aman Sinha, Hongseok Namkoong, and John Duchi. "Certifying some distributional robustness with principled adversarial training". In: *arXiv preprint arXiv:1710.10571* (2017).

[123] Florian Tramèr and Dan Boneh. "Adversarial training and robustness for multiple perturbations". In: *Advances in Neural Information Processing Systems*. 2019, pp. 5866–5876.

[124] Florian Tramèr et al. "Ensemble adversarial training: Attacks and defenses". In: *arXiv preprint arXiv:1705.07204* (2017).

[125] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. "A study of the effect of jpg compression on adversarial images". In: *arXiv preprint arXiv:1608.00853* (2016).

[126] Yan Luo et al. "Foveation-based mechanisms alleviate adversarial examples". In: *arXiv preprint arXiv:1511.06292* (2015).

[127] Qinglong Wang et al. "Learning adversary-resistant deep neural networks". In: *arXiv preprint arXiv:1612.01401* (2016).

[128] Shixiang Gu and Luca Rigazio. "Towards deep neural network architectures robust to adversarial examples". In: *arXiv preprint arXiv:1412.5068* (2014).

[129] Cihang Xie et al. "Feature denoising for improving adversarial robustness". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 501–509.

[130] Nicolas Papernot et al. "Distillation as a defense to adversarial perturbations against deep neural networks". In: *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2016, pp. 582–597.

[131] Nicolas Papernot and Patrick McDaniel. "Extending defensive distillation". In: *arXiv preprint arXiv:1705.05264* (2017).

[132] Nicolas Papernot and Patrick McDaniel. "On the effectiveness of defensive distillation". In: *arXiv preprint arXiv:1607.05113* (2016).

[133] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. "Adversarial logit pairing". In: *arXiv preprint arXiv:1803.06373* (2018).

[134] Ji Gao et al. "Deepcloak: Masking deep neural network models for robustness against adversarial samples". In: *arXiv preprint arXiv:1702.06763* (2017).

[135] Cihang Xie et al. "Mitigating adversarial effects through randomization". In: *arXiv preprint arXiv:1711.01991* (2017).

[136] Alexandre Araujo et al. "Robust neural networks using randomized adversarial training". In: *arXiv preprint arXiv:1903.10219* (2019).

[137] Yuchen Zhang and Percy Liang. "Defending against whitebox adversarial attacks via randomized discretization". In: *arXiv preprint arXiv:1903.10586* (2019).

[138] Ali Shafahi et al. "Adversarial training for free!" In: *Advances in Neural Information Processing Systems*. 2019, pp. 3358–3369.

[139] Alberto Bietti et al. "A kernel perspective for regularizing deep neural networks". In: *International Conference on Machine Learning*. 2019, pp. 664–674.

[140] Gamaleldin Elsayed et al. "Large margin deep networks for classification". In: *Advances in neural information processing systems*. 2018, pp. 842–852.

[141] Matthias Hein and Maksym Andriushchenko. "Formal guarantees on the robustness of a classifier against adversarial manipulation". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2266–2276.

[142] Shizhao Sun et al. "On the depth of deep neural networks: A theoretical view". In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[143] Zhezhi He, Adnan Siraj Rakin, and Deliang Fan. "Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 588–597.

[144] Mathias Lecuyer et al. "Certified robustness to adversarial examples with differential privacy". In: *arXiv preprint arXiv:1802.03471* (2018).

[145] Xuanqing Liu et al. “Towards robust neural networks via random self-ensemble”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 369–385.

[146] Xuanqing Liu et al. “Adv-bnn: Improved adversarial defense through robust bayesian neural network”. In: *arXiv preprint arXiv:1810.01279* (2018).

[147] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. “Certified adversarial robustness via randomized smoothing”. In: *arXiv preprint arXiv:1902.02918* (2019).

[148] Rafael Pinot et al. “Theoretical evidence for adversarial robustness through randomization”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 11838–11848.

[149] Tian Han et al. “Alternating back-propagation for generator network”. In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[150] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[151] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[152] Ludwig Schmidt et al. “Adversarially robust generalization requires more data”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 5014–5026.

[153] Anish Athalye, Nicholas Carlini, and David Wagner. “Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples”. In: *arXiv preprint arXiv:1802.00420* (2018).

[154] Anish Athalye et al. “Synthesizing robust adversarial examples”. In: *arXiv preprint arXiv:1707.07397* (2017).

[155] Jonathan Uesato et al. “Are labels required for improving adversarial robustness?” In: *arXiv preprint arXiv:1905.13725* (2019).

[156] Antonio Torralba, Rob Fergus, and William T Freeman. "80 million tiny images: A large data set for nonparametric object and scene recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 30.11 (2008), pp. 1958–1970.

[157] I Zeki Yalniz et al. "Billion-scale semi-supervised learning for image classification". In: *arXiv preprint arXiv:1905.00546* (2019).

[158] Bart Thomee et al. "YFCC100M: The new data in multimedia research". In: *Communications of the ACM* 59.2 (2016), pp. 64–73.

[159] Daniel Lowd and Christopher Meek. "Adversarial learning". In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 2005, pp. 641–647.

[160] Paolo Russu et al. "Secure kernel machines against evasion attacks". In: *Proceedings of the 2016 ACM workshop on artificial intelligence and security*. 2016, pp. 59–69.

[161] Fei Zhang et al. "Adversarial feature selection against evasion attacks". In: *IEEE transactions on cybernetics* 46.3 (2015), pp. 766–777.

[162] Olga Russakovsky et al. "Imagenet large scale visual recognition challenge". In: *International journal of computer vision* 115.3 (2015), pp. 211–252.

[163] Bai Li et al. "Certified adversarial robustness with additive noise". In: *Advances in Neural Information Processing Systems*. 2019, pp. 9464–9474.

[164] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. "Certified defenses against adversarial examples". In: *arXiv preprint arXiv:1801.09344* (2018).