Detecting Adversarial Examples by Measuring their Stress Response

by

Lin Sun

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved October 2019 by the
Graduate Supervisory Committee:

Rida Bazzi, Chair
Baoxin Li
Hanghang Tong

ARIZONA STATE UNIVERSITY

December 2019

ABSTRACT

Machine learning (ML) and deep neural networks (DNNs) have achieved great success in a variety of application domains, however, despite significant effort to make these networks robust, they remain vulnerable to adversarial attacks in which input that is perceptually indistinguishable from natural data can be erroneously classified with high prediction confidence. Works on defending against adversarial examples can be broadly classified as correcting or detecting, which aim, respectively at negating the effects of the attack and correctly classifying the input, or detecting and rejecting the input as adversarial. In this work, a new approach for detecting adversarial examples is proposed. The approach takes advantage of the robustness of natural images to noise. As noise is added to a natural image, the prediction probability of its true class drops, but the drop is not sudden or precipitous. The same seems to not hold for adversarial examples. In other word, the stress response profile for natural images seems different from that of adversarial examples, which could be detected by their stress response profile. An evaluation of this approach for detecting adversarial examples is performed on the MNIST, CIFAR-10 and ImageNet datasets. Experimental data shows that this approach is effective at detecting some adversarial examples on small scaled simple content images and with little sacrifice on benign accuracy.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

Deep Neural Networks (DNNs) have proved to be very effective in a variety of machine learning tasks, including security-sensitive applications like autonomous driving [1], face identification [23] and malware detection [25]. The work in this thesis is concerned with Neural Networks that are used for image classification. Such networks are typically trained on a fixed number of classes in order to classify new examples from these classes. For a given input image, the network calculates a confidence vector, with one entry per class, which aims at capturing the *probability* that the input image is in a given class. An input image is classified to be from the class for which the network has the highest confidence value. Recent studies shows that Neural Networks are susceptible to adversarial attacks [28, 8, 17, 5, 16, 3]. An attacker can take an input image which is correctly classified as being from class c_1 and slightly modify it by adding human-imperceptible perturbations so that it is classified to be from another class c_2. The research work in adversarial machine learning has witnessed a cat and mouse race between researchers developing more effective attacks and those attempting to develop defenses to thwart these attacks.

In this thesis, a new approach for defending DNNs against adversarial attacks is investigated. Before giving an outline of the approach and the results, an overview of existing attacks and some of the more effective defenses are introduced. Szegedy et al. first introduced adversarial examples against DNNs in 2013 [28]. They define an optimization problem for finding adversarial examples, and generate small perturbations by approximating a box-constrained optimization problem of maximizing loss function under small perturbation magnitudes with a box-constraind L-BFGS problem.

Later Goodfellow et al. [8] proposed a faster but effective method called Fast Gradient Sign Method for generating adversarial examples that requires less computation. It only aggregates one step gradient update along the direction of the sign of gradient at pixel level to input images. Kurakin et al. in [13] extended FGSM to targeted FGSM by not maximizing loss of original class but minimizing the loss of specified class when calculating the gradient. They call attacks that target specific classes *Targeted Attacks*. Kurakin el al. in [13] extended FGSM by running finer updates for multiple iterations and clip input during each iteration to force a magnitude constrain on perturbation. They name this finer grained attack Iterative Fast Gradient Sign Method. In order to speed up the generating process, [5] proposed Momentum FGSM attack by utilizing gradient momentum in each iterative steps. The momentum in a given iteration is derived from the momentum of the previous iteration and the gradient of the current iteration. This process greatly reduced the running time for iterative FGSM. Previous attacks treat every input feature evenly and generate perturbations without considering the significance of features to model's output. Papernot et al. proposed Jacobian-based Saliency Attack that utilize the features' different impacts in changing output and successfully fooled the network by attacking a small portion of input. Moosavi-Dezfooli et al. [17] defined a prediction polyhedron for input x to retain its original class. During each iteration, a vector that reaches the polyhedron boundary is computed and an estimate is updated. They managed to generate successful adversarial examples with very small perturbations. Leveraging their study in deep fool attack, they developed a universal attack that generates minimum attack for a small part of the dataset and keep updating the universal perturbation until most of the input images are successfully fooled. Their work was well generalized across state of art DNN architectures. Carlini and Wagner in [3] redefined objective functions for generating perturbations and proposed several attacks using multiple objective

functions. They extended their work to different norm distance measurements too.

Among defenses, Adversarial (Re)training [8] is the most extensively investigated defense against adversarial attacks. It injects perturbed data into the training data at every step of training to train a robust model. Adversarial training also increases the performance of neural networks on benign input from small datasets. Papernot et al. [22] used network distillation to extract knowledge from an original DNN and succeeded in reducing the attacks' ability to create adversarial examples to less than 5%. But this distillation defense was later bypassed by L_0 Carlini and Wagner attacks with little modifications [2]. Many researchers tried to train binary classifiers as detector to classify benign and adversarial inputs. [15] made use of the ReLu layers output as the features for detection. [9] added one novelty class in class sets so the model can detect adversarial examples by classifying the input as outliers. [6] claims that adversarial inputs do not constitute meaningful changes to the inputs thus it must push samples off from the data manifold. They use density estimation to measure how far the last hidden layer activation are from the submanifold of predicted class generated from training data. [26] trained a PixelCNN and found that the distribution of adversarial examples is different from benign data. They reject input by calculating p-value of prediction outputs. Some researchers apply regularization or smooth labels to make models less sensitive to input perturbation [10, 18]. The methods aiming to achieve output invariance to input perturbations are called gradient masking. Some researchers preprocess the input with certain transformation to remove perturbation. [10] used denoising auto-encoder as a defense. [19] applies filters in preprocessing stage. But Gu et al. in [10] mentioned that the model with image transformation can be even easier to attack if one considers the transformation layer together with the base model as one system to attack. Das et al. preprocess image with JPEG compression to defeat adversarial perturbations. [29] proposed that the feature input

spaces are unnecessarily large, and this input space provides the opportunities for adversarial examples. They proposed feature squeezing to compress feature space for defending through quantizing pixel values, perform non-differentiable local or non-local filtering. Prakash et al. [24] proposed another non-differentiable transformation by switching pixels that are less possible to capture objects of image to neighboring pixels and perform discrete wavelet transformation and filter with adaptive threshold filtering in frequency domain to recover the image classification.

Researchers have pointed out that classifiers can tolerate some natural noise [10]. Given an input image, the classifier would still be able to classify it correctly if the image if subjected to limited amount of noise. This tolerance has been exploited by some existing defenses [10, 29, 24]. What existing works did not consider is how the confidence in the original classification drops with higher levels of noise, levels at which the classification of the image cannot be expected, in general, to stay the same. By applying increasing levels of noise to an image, and measuring the degradation in the confidence of the original class (classified without noise), a vector of confidences, which we define as the *Behavior Vector*, is obtained. In this thesis, a study of the feasibility of using the behavior vector as a novel method to distinguish between benign and adversarial input is performed.

An initial part of the study consisted of studying the effects of adversarial attacks on benign images. The goal was to get an insight into adversarial perturbation that could help in devising effective patterns of noise to use in generating behavior vectors. The best scenario for a defense is when adversarial perturbations are distributed over the entire image. As the classification of benign images are mostly activated by objects in the area of interest, adversarial perturbations added pixel-wise over the image are potentially more sensitive to partial transformations. We generate adversarial examples on images from MNIST [14], CIFAT-10 [12] and ImageNet [4],

the results of the study showed that in general, adversarial perturbations tend to concentrate on salient parts of the images as well, which meant that no particular patterns in the attacks could be exploited in generating behavior vectors. Then an extensive study is done by analyzing the probability change of originally predicted class while increasing the transformation levels which is latter defined as *Stress Test*. Our study has shown that there is a behavior pattern difference between adversarial examples and benign input. Leveraging this difference in behavior vectors we trained novelty detectors using One-Class SVM and autoencoders to identify irregular inputs generated by adversarial perturbed images that deviated from normal distribution. In the end we performed a comprehensive comparison between our stress test defense and two existing defense mechanisms: Feature Squeezing [29] and Pixel Deflection [24]. Our defense is shown to have comparable results to state of the art methods on MNIST, and consistent performance for different attacking methods on CIFAR-10 and ImageNet.

The rest of this thesis is organized as follows. Chapter 2 introduces gives an overview of related works on adversarial attacks and defenses. Chapter 3 we presents analysis on perturbation distribution for different attacks. In chapter 4, describes the behavior-vector based defense and compares, on average, the behavior vectors for benign images differs from those for adversarial input for a variety of attacks and noise levels. Chapter 5 compares the behavior-vector based defense to two state-of-art defenses. Chapter 6 concludes with a discussion and directions for future.

Chapter 2

RELATED WORKS

In this chapter, we provide the background knowledge and related works about adversarial attacks and defenses. For a given classifier $f(x) : x \in \mathcal{X} \longrightarrow y \in \mathcal{Y}$, for an image x and output label y, we define adversarial perturbation as the minimum r that is sufficient to misclassify the network. There are two categories of adversarial attacks: *untargeted* and *targeted*. For untargeted attacks, the perturbation is minimized subject to $f(x+r) \neq y$. For targeted attacks, perturbation is generated subject to $f(x+r) = y^*$ where $y* \neq y$. We only consider untargeted attacks in this thesis.

2.1 Adversarial Examples

2.1.1 L-BFGS Attack

Szegedy et al. first introduced adversarial examples against deep neural networks in 2013 [28]. They define the problem as a constrained optimization problem.

$$\begin{gathered} \text{Minimize } \|r\|_2 \text{ subject to :} \\ 1. f(x+r) = l \\ 2. x + r \in [0,1]_m \end{gathered} \tag{2.1}$$

Where $\|r\|_2$ represents the L_2 distance between benign input and adversarial example, and f represents the classification function. The exact computation of this problem is hard, so they approximate the problem by solving the following equation using a line search for finding minimum $c > 0$:

$$\text{Minimize } c|r| + \text{loss}_f(x+r, l) \text{ subject to } x + r \in [0,1]_m \tag{2.2}$$

Figure 2.1: L-BFGS Adversarial Examples on AlexNet
Adopted from [28]. Left columns are original benign inputs, right columns are adversarial examples, center columns are amplified perturbations.

They applied L-BFGS attack on MNIST, QuocNet, AlexNet. For simple networks trained on MNIST, the average minimum distortion to reach 0% accuracy ranges from 0.062 to 0.14.

Aside from L-BFGS attack, they found that adversarial examples are *transferable*. A relatively large fraction of their adversarial examples can generalize across networks trained from scratch with different hyper-parameters. Even when the networks are trained from different datasets, a large fraction of adversarial examples for one model can fool other models.

2.1.2 Fast Gradient Method

Goodfellow et al. proposed one L-infinity norm attack called Fast Gradient Method to generate perturbations through one step gradient update [8]. They aggregate the sign of gradient with a fixed magnitude upon each pixel from input image through one iteration, so this FGM under L_∞ constrain is also called Fast Gradient Sign Method.

$$\eta = \epsilon sign(\nabla_x J(\theta, \mathbf{x}, y)) \tag{2.3}$$

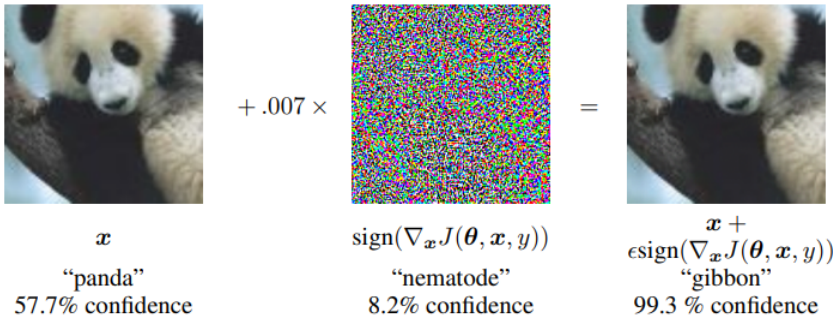

Figure 2.2: One FGSM Example Applied to GoogLeNet on ImageNet Adopted from [8].

The same approach was then extended to other norms, equation 2.4 defines the L_2 norm version of FGM.

$$\eta = \epsilon \frac{\nabla_x J(\theta, \mathbf{x}, y)}{\|\nabla_x J(\theta, \mathbf{x}, y)\|_2} \tag{2.4}$$

Where ϵ denotes the maximum perturbation on pixels, θ denotes the model parameter, $\mathbf{x}$ denotes the input image and y denotes the label of input $\mathbf{x}$. The FGM attack updates the input pixels along the direction of increasing loss between model output and input label by a fixed amount. Their simple one-step perturbation causes a shallow softmax classifier to have an error rate of 99.9% with average confidence of 79.3% using $\epsilon = 0.25$ on MNIST dataset. Their attacks induce an error rate of 87.15% and average probability of misclassified class is 96.6% with $\epsilon = 0.1$ on CIFAR-10 dataset. Values are normalized to $[0, 1]$ range.

2.1.3 Iterative Fast Gradient Method

Kurakin et al.[13] introduced a straightforward way to extend FGM by generate the attack through multiple iterations with small step size and clip pixel values of intermediate results after each iteration. This l_∞ clip ensures the attacks are in

$\epsilon - neighbourhood$ of original images.

$$X_0{}^{adv} = X,\ X_{N+1}{}^{adv} = Clip_{X,\epsilon}\{X_N{}^{adv} + \alpha \text{sign}(\nabla_X J(X_N{}^{adv}, y_{true}))\} \tag{2.5}$$

Previous works assumes adversarial examples can be directly fed into the network, but many times the image input are generated directly through devices like cameras and sensors. Surprisingly Kurakin et al. [13] discovered that even in physical work when adversarial images are printed out and recaptured by cameras, they can still fool the network with high success rate.

2.1.4 Projected Gradient Descent

Madry et al. [16] proposed a modified iterative gradient based approach called Projected Gradient Descent. Unlike the Basic Iterative Fast Gradient Method, Projected Gradient Descent starts the perturbation by a uniformly randomly chosen data inside the L_p norm ball near benign data. After each iteration it projects the perturbation outside the norm ball to the closet data point inside the norm ball then start the next iteration of gradient update. This method iteratively applied this update k times with step-size $\geq \epsilon/k$ where ϵ refers to the L-p constrain for perturbation magnitude. Iteratively generated perturbations induce higher error rate compared to single-step perturbation, but transfer at lower rates according to [16].

2.1.5 Momentum Attack

In order to stabilize update directions and escape from poor local maxima during iterative methods, Dong et al. [5] proposed a momentum variant fast gradient sign method by using the momentum in the update function. The momentum is updated by aggregating velocity vector in the gradient direction in every iteration.

$$g_{t+1} = \mu \cdot g_t + \frac{\nabla_x J(x_t{}^*, y)}{\|\nabla_x J(x_t{}^*, y)\|_1} \tag{2.6}$$

Then the input is updated as equation 2.6

$$x_{t+1}{}^{*} = x_t{}^{*} + \alpha \cdot \text{sign}(g_{t+1}) \tag{2.7}$$

This method accelerates iterative gradient descent by accumulating a velocity vector in the gradient direction of the loss function across iterations. The "memory" of previous iterations can help the gradient update to escape from local minima or maxima thus making adversarial examples more transferable across models.

2.1.6 Jacobian-based Saliency Map Attack (JSMA)

Papernot et al. proposed an efficient saliency adversarial map called Jacobian-based Saliency Map Attack (JSMA) [21]. F denotes the N dimention output of logits layer, X denotes the M dimention input. Then the salience map is defined as equation 2.7

$$\nabla F(X) = \frac{\partial F(X)}{\partial X} = \left[\frac{\partial F_j(X)}{\partial x_i}\right]_{i \in 1..M, j \in 1..N} \tag{2.8}$$

By calculating the saliency map they found the input features that can generate most significant variation in output. A small perturbation is added on most significant features in each iteration thus they can successfully fool the network by modifying a small portion of input features. This method is slow due to the calculation of a saliency map in each iteration.

2.1.7 Deepfool Attack

Moosavi-Dezfooli et al. [17] proposed DeepFool attack by finding the closest distance from input to decision boundary. For a linear affine classifier the minimum perturbation needed for misclassification is the distance between input and decision

boundary.

$$
\begin{aligned}
r_*(x_0) := & \operatorname{argmin} \|r\|_2 \\
& \text{subject to } sign(f(x_0 + r)) \neq sign(f(x_0)) \\
= & - \frac{f(x_0)}{\|w\|_2^2}
\end{aligned}
\tag{2.9}
$$

For a multi-class linear classifier, the input should reach at lease one boundary of complement of the convex polyhedron P,

$$
P = \bigcap_{k=1}^{c} \{x : f_{\hat{k}(x_0)}(x) \geq f_k(x)\}, \tag{2.10}
$$

$\hat{k}(x_0)$ is the classification of original input. The polyhedron P is the boundary of original classification to other classes, and the minimum perturbation is the vector that projects x_0 on the faces of P.

For general non-linear classifiers, the boundary is no longer a polyhedron, but they approximate the boundary set (Figure 2.3) by a modified polyhedron $\tilde{P}_i$

$$
\tilde{P}_i = \bigcap_{k=1}^{c} \{x : f_k(x_i) - f_{\hat{k}(x_0)}(x_i) + \nabla f_k(x_i)^T x - \nabla f_{\hat{k}(x_0)}(x_i)^T x \leq 0\} \tag{2.11}
$$

At each iteration, the perturbation vector that reaches the polyhedron boundary is computed, and the current state of the input is updated. Adversarial examples generated by the DeepFool attack have less perturbation magnitudes compared to those generated by other attacks. Unlike the JSMA attack, where the number of pixels being modified is minimized, DeepFool minimize the total intensity of perturbations.

2.1.8 Carlini & Wagner Attack

Carlini and Wagner proposed a targeted attack that can defeat distillation defense [22] and most of the existing other defenses according to [3]. They defined the

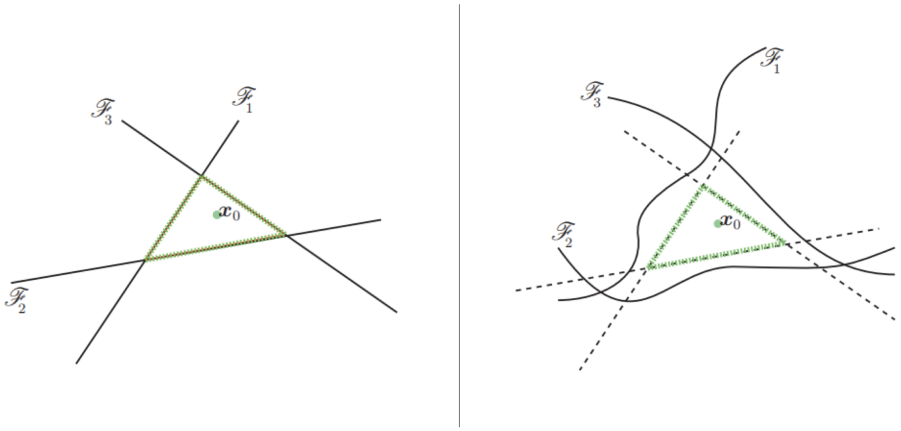

Figure 2.3: Decision Boundary for Multi-class Classifiers
Adopted from [17]. Left is the linear classifier polyhedron, right is a demonstration of general non-linear classifier decision boundary. The dotted line is the polyhedron approximation of decision boundary.

optimization problem of finding the minimum perturbation as follows

$$\begin{aligned} &\text{minimize } D(x, x+\delta) \\ &\text{such that } C(x+\delta) = t \\ &x + \delta \in [0,1]^n \end{aligned} \tag{2.12}$$

But the above formulation is difficult solve because $C(x+\delta) = t$ is highly nonlinear. So they modify the objective functions such that $C(x+\delta) = t$ if and only if $f(x+\delta) \leq 0$ and listed multiple choices for defining f. An alternative formulation for optimization problem is

$$\begin{aligned} &\text{minimize } \|\delta\|_p + c \cdot f(x+\delta) \\ &\text{such that } x + \delta \in [0,1]^n \end{aligned} \tag{2.13}$$

In terms of constant c, they use modified binary search to choose c value. During the optimization they still need to ensure the modification is in valid range where $0 \leq x_i + \delta_i \leq 1$. They introduced a new approach to solve this box-constrained problem by define perturbation δ as

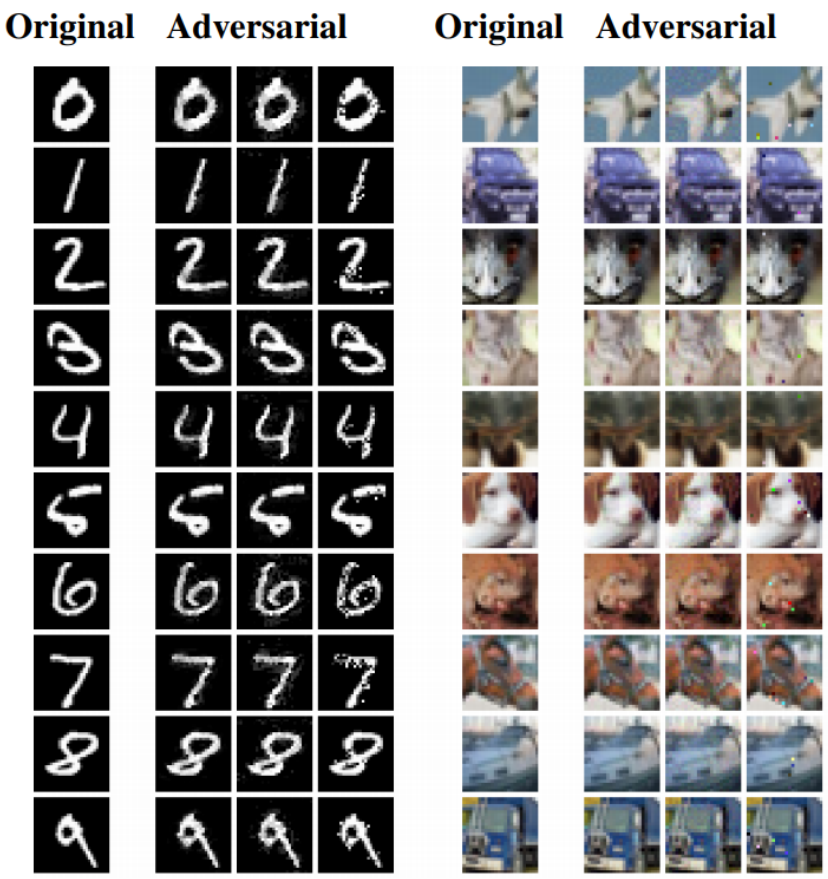

Figure 2.4: Carlini & Wagner Attack on MNIST and CIFAR-10
Adopted from [3]. For adversarial images the left columns are L_2 examples, middle columns are L_∞, right columns are L_0 examples. All images originally classified as class l and the adversarial instances are classified as class $l + 1$

$$\delta_i = \frac{1}{2}(tanh(w_i) + 1) - x_i \tag{2.14}$$

Since $-1 \leq tanh(w_i) \leq 1$, it follows that $0 \leq x_i + \delta_i \leq 1$. So the optimization solution would be automatically valid. They extend their work to different L-p distance as shown in Figure 2.4.

2.2 Defenses

2.2.1 Adversarial (Re)training

Adding adversarial examples in training is different from data augmentation where the translated images are expected to appear in the testing data set. Szegedy et al. showed that by training a mixture of benign input and adversarial input the network could be regularized somewhat. But due to the computation difficulty of L-BFGS the procedure was not demonstrated. Goodfellow et al. in [8] reformulate objective functions by including adversarial examples in the training process.

$$\widetilde{J}(\boldsymbol{\theta}, \boldsymbol{x}, y) = \alpha J(\boldsymbol{\theta}, \boldsymbol{x}, y) + (1 - \alpha) J(\boldsymbol{\theta}, x + \sigma sign(\nabla_x J(\boldsymbol{\theta}, \boldsymbol{x}, y)) \tag{2.15}$$

This approach was then applied on ImageNet in their latter work, from their results adversarial training can increase the robustness of one-step attacks but not iteratively generated attacks. Because adversarial examples generation needs extra computation, it's computational inefficient to include hard attacks in training set, and practical adversarial training on ImageNet dataset often adopts FGSM only. Madtry et al. in 2018 proposed Projected Gradient Descent which is used as the universal "first-order adversary", i.e., the strongest attack utilizing the local first order information about the network. But using PGD in adversarial training imposed a significant overhead on training time. For k-step PGD adversary, it requires additional k forward and k backward passes through the network for each batch of data. This increases the running time by a factor of $(k + 1)$.

2.2.2 Gradient Masking

Gu and Rigazio [10] first proposed preprocessing as one approach to recover adversarial examples. They first tried additive Gaussian noise at input layer and hidden

layers, then they tried gaussian blur at input layers. By applying Gaussian blur kernels of size 11 they managed to recover more then 50% of the adversarial examples with 3% sacrifice in benign accuracy. They also trained three-hidden-layer autoencoder to map adversarial input to its original input. The autoencoders are able to recover 90% of the adversarial errors, regardless of the model from which it originates. They also tried standard denoising autoencoder (DAE) without the knowledge of adversarial examples. A denoiser with gaussian $\epsilon = 0.1$ could denoise adversarial examples almost as well as an autoencoder trained on adversarial examples. But one key observation from their experiments that for any pre-processing, it is always possible to backpropagate the error signal through additional layers and generate new adversarial examples consider preprocessing layers and base model as a whole. Contractive autoencoder (CAE) is a variant of autoencoder that minimize Jacobian of hidden layers with respect to input data, so the network output can achieve "flatness" around the data points. The objective function for contractive autoencoder is shown in equation 2.15.

$$J_{CAE}(\theta) = \sum_{i=1}^{m} (L(x^{(i)}, y^{(i)}) + \lambda \| \frac{\partial h^{(i)}}{\partial x^{(i)}} \|_2) \tag{2.16}$$

where $\| \frac{\partial h^{(i)}}{\partial x^{(i)}} \|_2$ is the norm of jacobian matrix of hidden layers with respect to input. For deep contractive networks, the loss function is modified as equation 2.16.

$$J_{DCN}(\theta) = \sum_{i=1}^{m} (L(t^{(i)}, y^{(i)}) + \sum_{j=1}^{H+1} \lambda_j \| \frac{\partial {h_j}^{(i)}}{\partial {h_{j-1}}^{(i)}} \|_2) \tag{2.17}$$

They increased the minimum distortion needed of successful adversarial examples and act as practical regularizers.

2.2.3 Principle Component Analysis Detection

Hendrycks et al. reveals that adversarial images place abnormal emphasis on the lower-ranked priciple components from PCA [11]. They first center the training data

at zero, compute the covariance matrix C of the centered data and find the SVD of C which is $C = U \sum V^T$. Adversarial examples' coefficients of latter eigenvectors are consistently larger than benign input. This discover is especially remarkable for MNIST dataset where 100% of the FGSM images have coefficient's variance beyond 10 billion standard deviations from the mean coefficient variance of clean images.

2.2.4 Kernel Density Estimation

Feinman et al. proposed a defense method using density estimation in subspace of deep features learned by the model and Bayesian uncertainty estimates. The intuition is that since adversarial examples are not natural input, they must push samples off the data manifold. They measure how far input x is from the manifold of deep layers for training data as equation 2.17.

$$\hat{f}(x) = \frac{1}{|X_t|} \sum_{x_i \in X_t} k(x_i, x) \tag{2.18}$$

Where $k(\cdot, \cdot)$ is the Gaussian kernel with bandwidth σ. This is for measuring the distance of input x to training submanifold of class t. While this strategy works for data far from manifolds, but may not work on data closer to manifolds. Then they added Bayesian neural network uncertainty because DNNs trained with dropout are equivalent to an approximation of the Gaussian process [7]. They sample the data T times for a neural network with dropout layer after the last pooling layer and measure the uncertainty by equation 2.18.

$$U(x^*) = \frac{1}{T} \sum_{i-1}^{T} \hat{y^*}^T \hat{y_i}^* - \left(\frac{1}{T} \sum_{i-1}^{T} \hat{y^*} \right)^T \left(\frac{1}{T} \sum_{i-1}^{T} \hat{y^*} \right) \tag{2.19}$$

They proved that the uncertainty of an adversarial sample is typically larger than benign or noisy input.

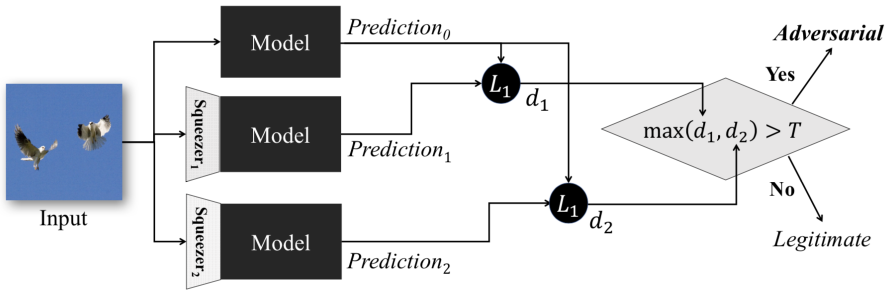

Figure 2.5: Feature Squeezing Framework for Detecting Adversarial Examples Adopted from [29]

2.2.5 Feature Squeezing

Xu et al. claims that the input feature space are often unnecessarily large, and this vast input state provides extensive opportunities for adversarial attacks. So they proposed two feature "squeezing" approaches: reduce color bit depth and spatial smoothing.

For bit depth reduction they reduce 8-bit color representation into i-th bits with integer-rounding operations. Spatial smoothing is divided into local smoothing and non-local smoothing. Local smoothing can be designed as median smoothing, mean smoothing or Gaussian smoothing. For non-local smoothing they replace the center path with similar patch found in a larger area. By comparing the squeezed classification and original classification, the detector can detect 98.2% of the MNIST adversarial examples, 84.5% of the CIFAR-10 examples and 85.9% of the ImageNet examples.

2.2.6 Distillation

Papernot el al. utilized network distillation to defense DNNs from adversarial examples [22]. The probability of the original model was used as the input to train the distilled network. The probability of the first model includes additional knowledge

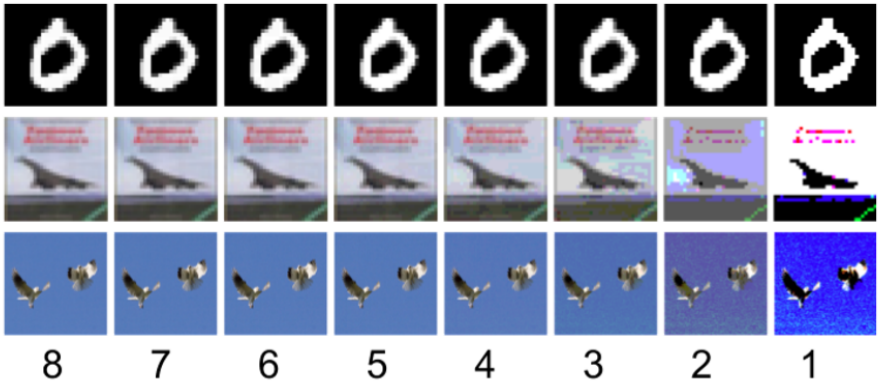

Figure 2.6: Color Bit Depth Reduction
Adopted from [29]. From leftmost to rightmost columns the color bit decrease from 8 to 1 respectively.

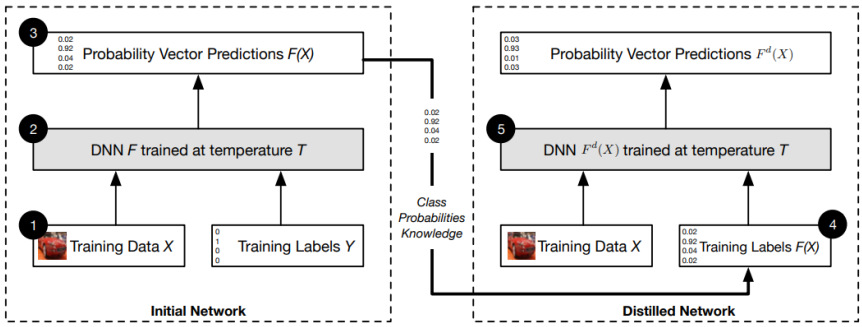

Figure 2.7: Distillation Framework
Adopted from [22]. The left network is original network the right network is the distilled network.

compared class labels, and the transformed smaller network which maintain the accuracy compared to those larger networks can be beneficial to generalization capability for input outside training data set. Softmax layer is the last layer to normalize logits into probabilities, the probability of first classifier output is presented in equation 2.19

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)} \tag{2.20}$$

T is a parameter named distillation temperature and shared across the softmax layer. All probabilities in equation 2.19 converges to $1/N$ if $T \to \infty$. The higher the temperature the more ambiguous the probability will be and lower the temperature

value the more discrete (closer to one hot vector) the probability vector will be. They proved that using higher temperature can reduce network sensitivity towards adversarial examples.

2.2.7 Pixel Deflection

We know from [10] that defense mechanisms using differentiable transformations before classification can be easily circumvented by perturbations generated through model and transformation layers together. Prakash et al. in [24] proposed Pixel Deflection by introducing a form of artificial non-differentiable noise. The noise is generated by randomly selecting pixels from input to replace with pixels within a small square neighborhood. This deflection on pixels does not alter the classification of clean images but enables the correct classification of large portion of adversarial examples.They conducted an analysis on the average localization of perturbations on images, and found that most attacks search the entire image for perturbation with less or no constrain on the number of pixels been attacked. So when they choose the pixels for deflection the probability of pixel been selected is inversely proportional to its likelihood of containing object. In attempt to soften the impact of pixel deflection, they applied discrete wavelet transform, apply soft threshold for softening and apply the inverse wavelet transform.

Chapter 3

PERTURBATION DISTRIBUTION

3.1 Experiment Setup

3.1.1 Datasets

We perform the analysis on three benchmark image datasets. MNIST is the handwritten dataset with 60000 training and 10000 testing images for handwritten digits from 0 to 9. The images are 28x28 black and white images. CIFAR-10 contains 50000 training images and 10000 testing images, all the images are 32x32 three channel colored images from 10 classes. The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. ImageNet is a large visual dataset designed for visual object recognition. More than 14 million images have been hand-annotated by the project to indicate what objects are pictured and in at least one million of the images, bounding boxes are also provided for some classes. Each year there is an ImageNet challenge with 1000 classes, in this thesis we utilize a 1000 images subset of ImageNet 2012 challenge validation dataset by utilizing the first image from each class.

3.1.2 Neural Network Model

We perform adversarial attacks on three example models, each trained on one of the datasets mentioned above. For CIFAR-10 we adopt the VGG-16 networks pre-trained on ImageNet as a base, and extract the first three blocks of weights and add dense layers on top for transfer learning. For ImageNet we utilize pre-trained model Inception V3 provided by Tensorflow. Models are trained on NVIDIA GeForce

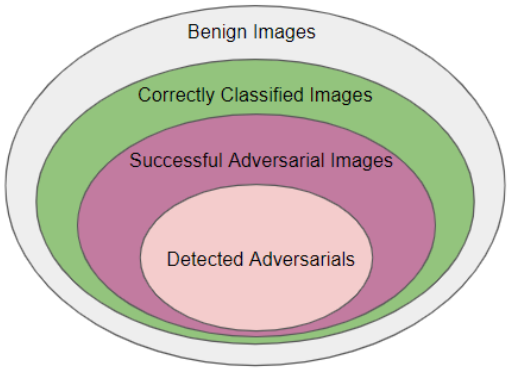

Figure 3.1: Image Relationships

Dataset	Network Architect	Top-1 Testing Acc
MNIST	2xConv2D+2xDense	99.17%
CIFAR-10	VGG-16	86.4%
ILSVRC2012	Inception V3	78.8%

Table 3.1: Target Models

GTX 1060, and tested for adversarial examples generation on AWS EC2 c5.9xlarge instance. Table 3.1 lists the Top-1 accuracy of example models.

3.1.3 Adversarial Examples Generation

Multiple L_∞ and L_2 attacks are included in our attack generation. For L_∞ attacks we include FGSM, I-FGSM, PGD, Momentum Attack and DeepFool attacks, and for L_2 attacks we include FGM, I-FGM, PGD, Momentum Attack, DeepFool attack and Carlini&Wagner attack. By the definition of adversarial attacks, misclassified images are by default successful adversarial input. This is not useful for measuring the effectiveness of adversarial attacks, so we exclude misclassified images from our attack targets. So for each dataset, a subset of 100 correctly classified images randomly selected from testing data are used as seed images for generating these attacks. Attacks are implemented using Cleverhans library [20] except DeepFool L_∞ attacks, which is

Dataset	L-p Norm	Attack	Success Rate	Confidence	L_∞	L_2
MNIST	L_∞	FGSM	91%	68.01%	0.3	-
		I-FGSM	97%	97.27%	0.18	-
		PGD	98%	96.75%	0.18	-
		Momentum	94%	90.58%	0.18	-
		DeepFool	100%	51.07%	0.16	-
	L_2	FGM	51%	78.94%	-	4
		I-FGM	79%	97.61%	-	3
		PGD	93%	99.87%	-	3
		Momentum	98%	91.54%	-	3
		DeepFool	100%	51.46%	-	1.88
		Carlini&Wagner	97%	48.98%	-	2.12
CIFAR-10	L_∞	FGSM	97%	78.50%	0.04	-
		I-FGSM	95%	92.14%	0.02	-
		PGD	94%	92.30%	0.02	-
		Momentum	90%	92.58%	0.02	-
		DeepFool	100%	54.54%	0.01	-
	L_2	FGM	96%	69.54%	-	1.5
		I-FGM	100%	95.86%	-	1
		PGD	100%	94.16%	-	1
		Momentum	92%	61.89%	-	1
		DeepFool	100%	54.06%	-	0.41
		Carlini&Wagner	100%	51.15	-	0.36
ImageNet	L_∞	FGSM	76%	48%	0.03	-
		I-FGSM	100%	98.68%	0.015	-
		PGD	100%	98.69%	0.015	-
		Momentum	100%	97.87%	0.015	-
		DeepFool	100%	36.17%	0.0014	-
	L_2	FGM	65%	51.79%	-	1.5
		I-FGM	93%	88.12%	-	1
		PGD	93%	84.91%	-	1
		Momentum	87%	82.11%	-	1
		DeepFool	100%	36.75%	-	0.36
		Carlini&Wagner	91%	57.23%	-	0.8

Table 3.2: Attacks Evaluation

generated by a slightly modified version of DeepFool algorithm from Cleverhans according to [17] description. Table 3.2 reports the results we get on eleven attacks for the three datasets.

For I-FGSM, PGD, Momentum attacks, epsilon per iteration is chosen by slightly greater than ϵ/k where k represents the maximum number of iterations allowed, which in our experiment is set to 100 for all iteratively generated attacks listed. For DeepFool and Carlini&Wagner attacks, the L_p distances are calculated after generation.

3.2 Perturbation distribution

We apply adversarial perturbations based on two norm distances: L_2 norm distance (equation 3.1) and L_∞ distance (equation 3.2), on correctly classified images (Figure 3.1). Each time when generating the perturbation, we either minimize the euclidean distance magnitude between benign input and adversarial examples, or minimize the maximum absolute difference on pixel levels between benign input and adversarial examples.

$$\|x - x^{adv}\|_2 = \sqrt{\sum_{i=1}^{n} (x_i - x_i^{adv})^2} \tag{3.1}$$

$$\|x - x^{adv}\|_\infty = \max_{1 \le i \le n} |x_i - x_i^{adv}| \tag{3.2}$$

In order to maintain the same performance between perturbation methods, we gradually increase the magnitude of perturbations and try to achieve over 90% success rate while still maintain the imperceptable nature of adversarial perturbations. We found that for MNIST dataset the L_2 perturbations generated to achieve reasonable success rates are no-longer quasi-imperceptible, so we exclude them from defense evaluations.

Figure 3.2 shows an example of applying L_∞ FGSM, I-FGSM, PGD, Momentum attack and DeepFool attack on 1000 examples with success rate near 90%. The average L_∞ distances are 0.3, 0.18, 0.18 and 0.18 respectively. As we expected, perturbations for DeepFool have better performance due to its generating mechanism while single step FGSM are no longer human-imperceptable when trying to achieve high success rate. Figure 3.3 shows the average L_1 distance between adversarial image and its benign input. Different from our expectations, the average perturbation are all concentrated in the central part of images, including gradient based adversarial

examples where perturbations are equally added to pixels based on the gradients directions. We noticed that a large portion of pixels for MNIST images have zero gradients towards output loss, and this might lead to the imbalance of perturbations added on images.

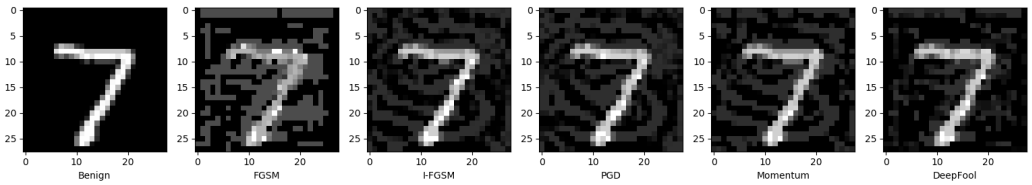

Figure 3.2: Example of L_∞ Perturbation on MNIST

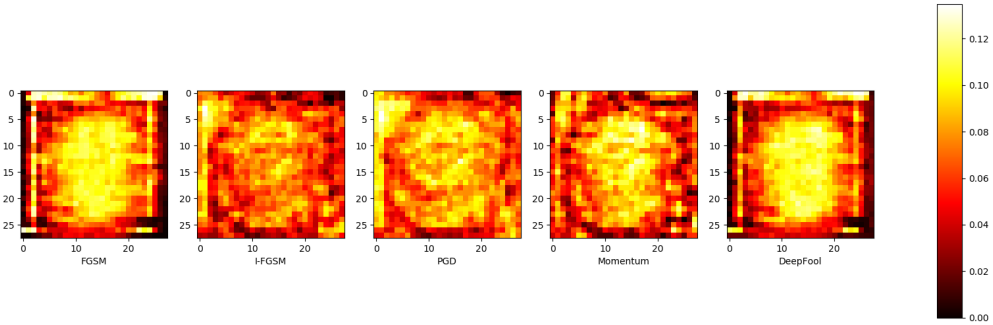

Figure 3.3: L_∞ Perturbation Distribution on MNIST

Similar to L_∞ attacks, we perform L_2 attacks FGM, I-FGM, PGD, Momentum attack, DeepFool and Carlini & Wagner L_2 attacks on MNIST dataset. Figure 3.4 shows an example of L_2 attacks when trying to reach over 90% success rate for all attacks. Figure 3.5 shows the distribution of perturbations by averaging the perturbations on each location of images. Pixel intensities are within [0, 1] range. Using L_2 attacks are minimizing the overall magnitude of image distances, so it is possible that the perturbations are obvious on some pixels while still maintain low euclidean distance. As shown from Figure 3.4 we can recognize the predicted class of this L_2 adversarial example is "9" for iterative gradient attacks. Distribution of L_2 perturbation

for MNIST are also concentrated in the central part of image.

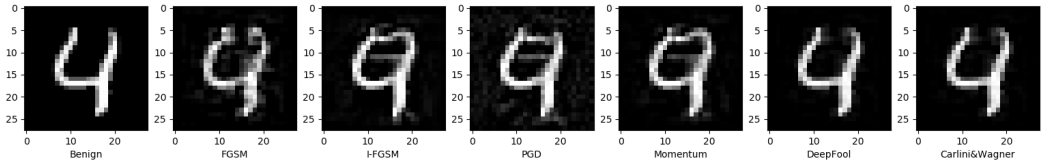

Figure 3.4: Example of L_2 Perturbation on MNIST

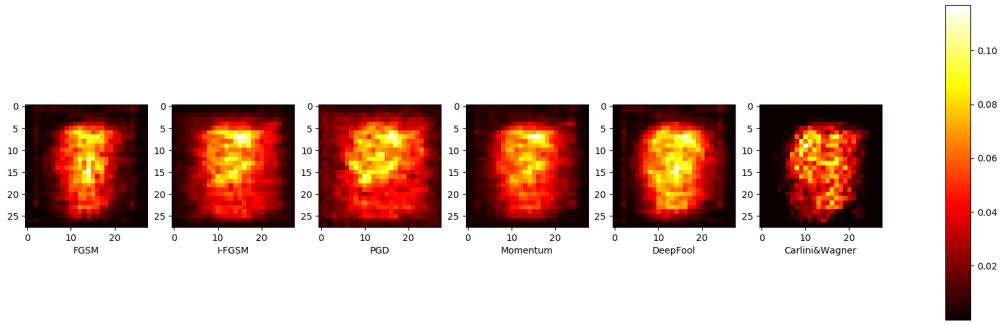

Figure 3.5: L_2 Perturbation Distribution on MNIST

We perform the same attacks on CIFAR-10 datasets. Perturbation magnitude are significantly less compared to MNIST images when achieving high success rate. The assumptions are CIFAR-10 images are 3 channel color images with slightly larger image size compare to MNIST black and white images, which provides larger space for perturbation. Figure 3.6 shows the examples of L_∞ perturbation of FGSM, I-FGSM, Momentum attack and DeepFool attack, with L_∞ norm equals to 0.04, 0.02, 0.02, 0.02 respectively. All images are originally predicted as "cat" and perturbed into "dog". Figure 3.7 shows the average perturbation distribution, where for gradient based attacks, adversarial perturbation are scattered over the entire image.

L_∞ attacks for CIFAR-10 datasets with high success rate are almost imperceptable except FGSM, where some color spots can be seen by human eyes. Figure 3.8 shows an example of L_2 attacks with just enough perturbation to achieve over 90% accuracy.

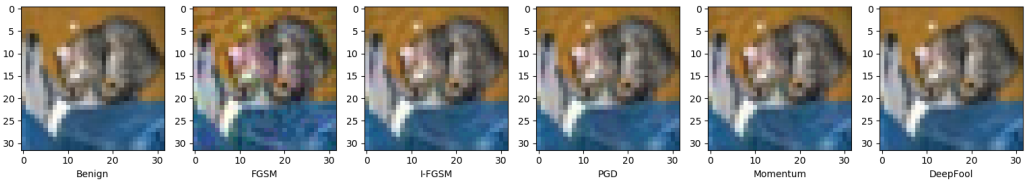

Figure 3.6: Example of L_∞ Perturbation on CIFAR-10

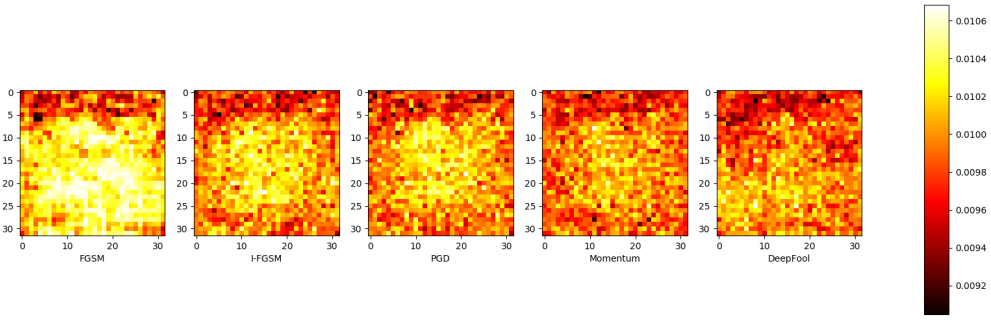

Figure 3.7: L_∞ Perturbation Distribution on CIFAR-10

Figure 3.8 is the L_2 attack example for a "ship" image, all perturbed images are recognized as "airplane". Different from L_∞ attacks, L_2 attacks are focusing on central area of images.

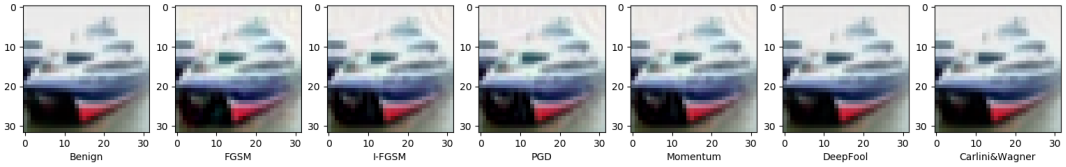

Figure 3.8: Example of L_2 Perturbation on CIFAR-10

Same experiments are performed on ImageNet subset, where adversarial examples are generated to achieve over 90% success rate while maintain small perturbation magnitudes. Figure 3.10, 3.11 shows an example of different L_∞ attacks and average perturbation magnitudes across images.

From the hot map we see for one-step gradient update L_∞ attacks or iteratively

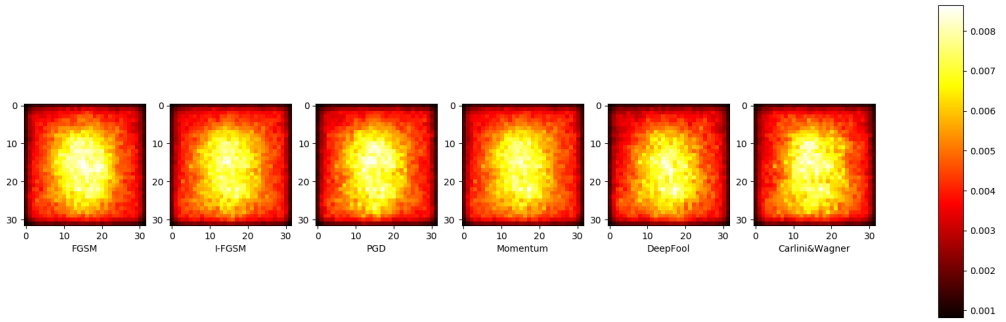

Figure 3.9: L_2 Perturbation Distribution on CIFAR-10

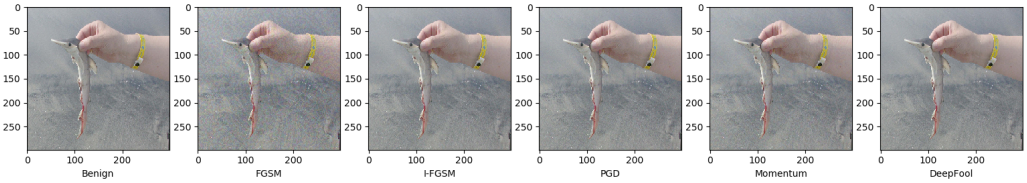

Figure 3.10: Example of L_∞ Attack On ImageNet

generated gradient L_∞ attacks on natural images from ImageNet, the perturbation tend to distribute across the entire image. For L_2 attacks where constrain in distance between adversarial and benign input is set using euclidean distance for entire image, tend to concentrate in the central part of image. Figure 3.13 displays an example of how perturbations distribute on images.

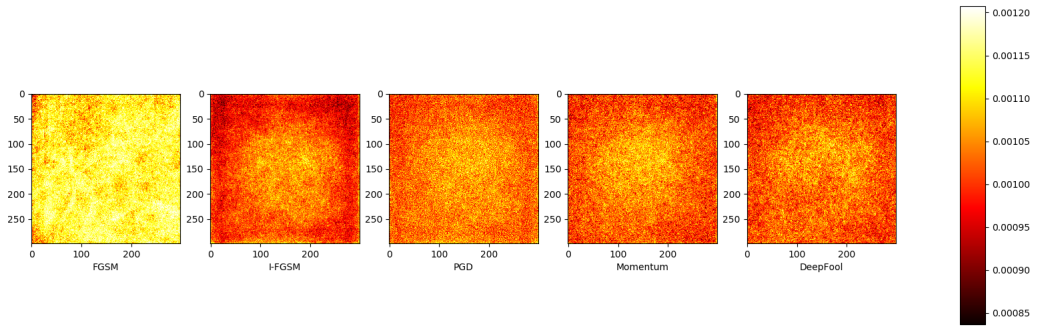

Figure 3.11: L_∞ Perturbation Distribution on ImageNet

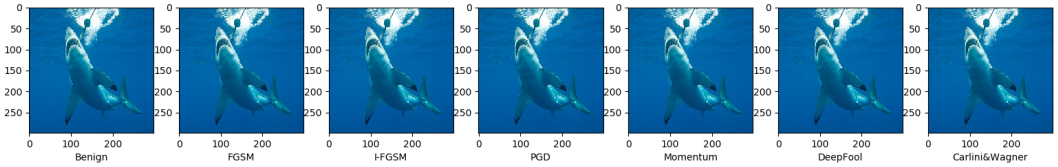

Figure 3.12: Example of L_2 Perturbation on ImageNet

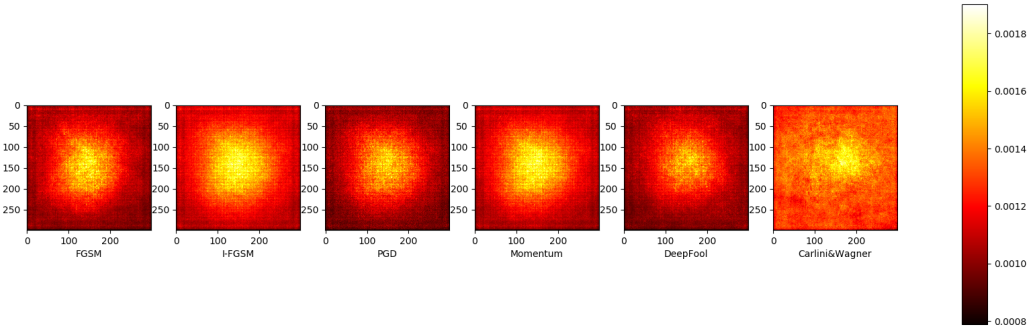

Figure 3.13: L_2 Perturbation Distribution on ImageNet

Chapter 4

STRESS TEST DEFENSE

With our assumption on difference between benign input and adversarial input towards adversarial examples, we introduce a defense mechanism to reduce vulnerabilities exposing DNNs to adversarial examples. We notice that benign images are robust to natural transformations while adversarial examples rely more on per pixel perturbation across entire image, thus adversarial examples are expected to be more sensitive to spatially partial noise applied on the image. Previous works envoling transformation as defense mainly concentrate on removing the perturbation through transformation image-wise, this leaves the potential for attackers to generate new perturbations considering stacked transformation layers and base network together. We instead are not trying to recover the adversarial input directly but add randomly distributed partial transformation on the input. By leveraging the robustness of benign input towards adversarial examples, we manage to train a detector to detect adversarial inputs using their probabilities generated for each transformation iteration.

4.1 Random Transformation

4.1.1 Gaussian Blur

Gaussian blur is a type of image blur filter that uses Gaussian function for calculating the transformation on pixels. 2D Gaussian blur can be represented as

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.1}$$

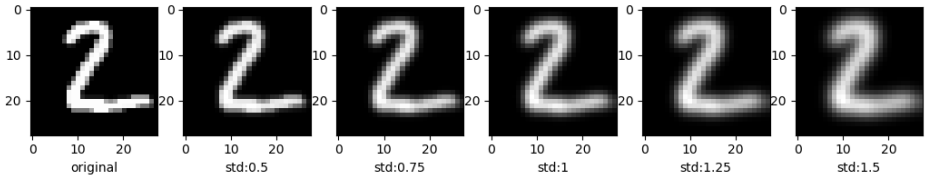

Figure 4.1: Increasing Gaussian Blur Intensity on MNIST Data

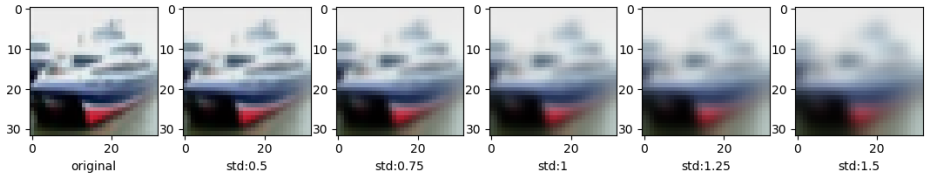

Figure 4.2: Increasing Gaussian Blur Intensity on CIFAR-10 Data

As defined in equation 4.1, increasing standard deviation for blur kernels will increase the intensity of Gaussian blur as kernel weights are more averaged. We choose kernel sizes with respect to different sigma values to include only non-zero weights in kernels. Figure 4.1 shows an example of increasing Gaussian blur sigma from 0.5 to 1.5 on MNIST. Figure 4.2 shows an example of increasing blur intensity from sigma equals to 0.5 to 1.5 on CIFAR-10.

4.1.2 Aggregated Random Transformation

Since we want to use the behavior difference of benign input and adversarial input for perturbation detection, we increase the random transformation percentage for every input through multiple iterations. *Patch* is the minimum transformation unit we use, w refers to the patch width, so the atomic transformation area on an image is $w * w$. If we use IMG_WIDTH and IMG_HEIGHT to denote the input image width and height, the total number of patches within an image is defined as

$$num\_patches = \frac{IMG\_WIDTH * IMAGE\_HEIGHT}{w * w} \tag{4.2}$$

A pattern is a shuffled array of integers from 0 to $num\_patched - 1$, with each number represents the location index of one patch on the image. A group of patches are selected in each iteration to be replaced by a fully transformed image which is generated before the process. Our goal is to transform the images patches by patches until 100% of the original image is replace by the corresponding fully transform image. Figure 4.3 represents a demonstration of applying 5% transformation using patches. We use p to represent a pattern from increasing transformation percentage from 0% to 100%, P represent the pattern set. We denote $F(x)$ to be the network output vector of original network for input image x, $F^k(x)$ refers to the probability of k^{th} class. Then we *Behavior Vector* $v_{x,p}$ of input image x as equation 4.3.

$$\begin{aligned} c &= argmax(F(x)) \\ v_(x,p) &= \begin{bmatrix} F^c({x_p}^{s*0}) \\ F^c({x_p}^{s*1}) \\ \vdots \\ F^c({x_p}^{s*n}) \end{bmatrix} \end{aligned} \tag{4.3}$$

Where c denotes the original prediction of an input image. For correctly classified benign images it represents its true label, for successful adversarial image is represents

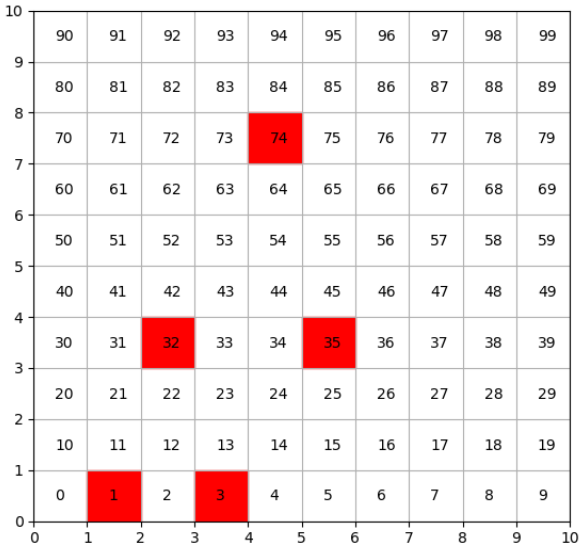

Figure 4.3: Example of Using Patches as Minimum Unit for Random Transformation

the false output from model. s denotes the transformation percentage step size, which means each time we transform s of the total image area while retain the existing transformation untouched. x_p refers to the transformed image using pattern p, ${x_p}^{s*i}$ is the image with $s * i$ percent been transformed. Figure 4.4 shows the system of generating behavior vectors and leveraging behavior vectors for anomaly detection.

For MNIST, CIFAR-10 and ImageNet subset, we use the randomly generated patterns to increase transformation percentages and record the probability of original predicted class for benign and adversarial inputs. Figure 4.5 shows examples of increasing transformation percentages on MNIST, CIFAR-10 and ImageNet.

4.2 Adversarial Detection

Once we have the partially aggregated transformed images, behavior vectors are generated for different perturbations. Utilizing the behavior vectors towards randomly added transformation we can distinguish adversarial examples from natural benign

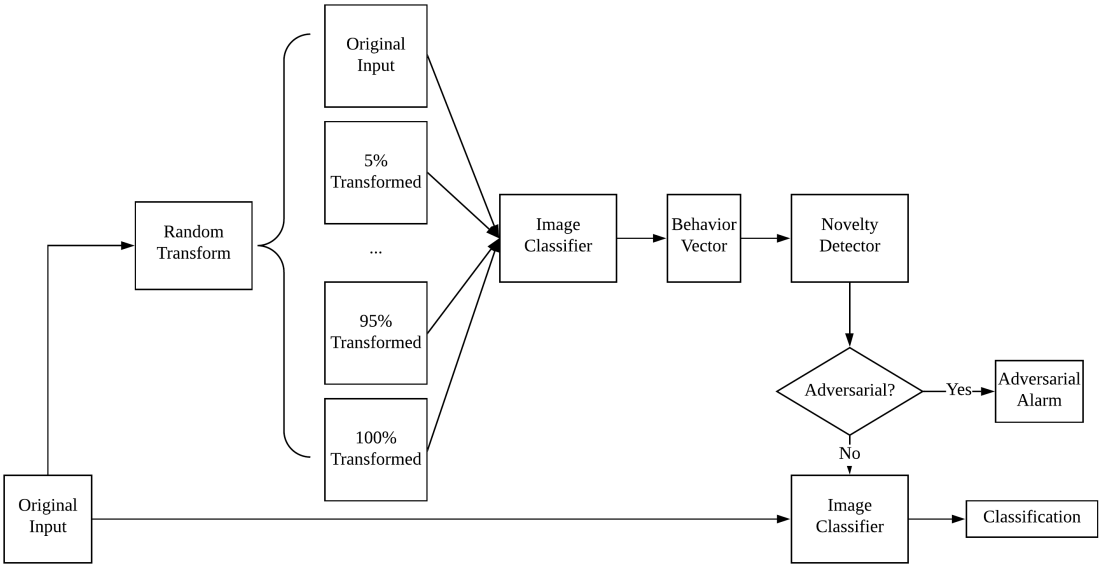

Figure 4.4: System Framework

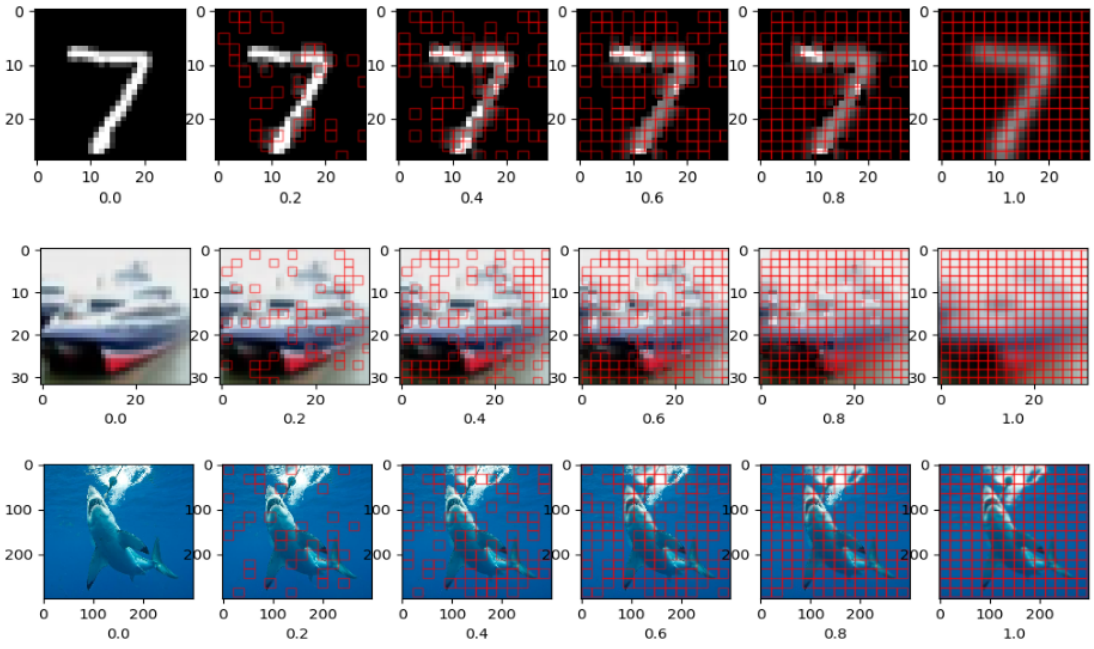

Figure 4.5: Examples of increasing transformation percentages
Area inside red grids are replaced by corresponding fully transformed images. Images are from MNIST, CIFAR-10 and ImageNet from top to bottom.

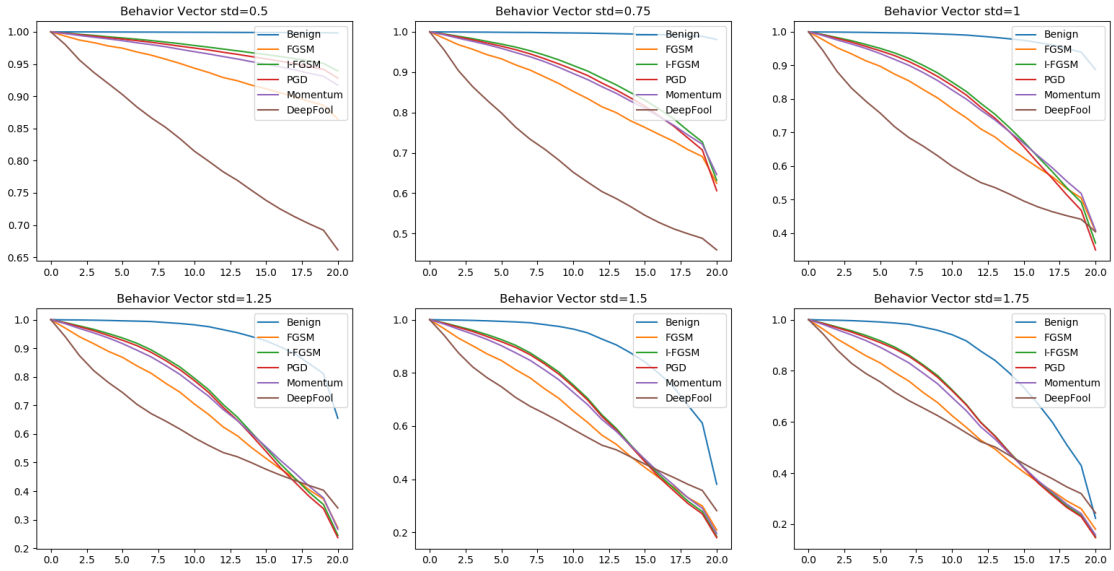

Figure 4.6: Average Probability of Original Predicted Class for L_∞ Attacks on MNIST

images.

4.2.1 Behavior Vector

As shown in figure 4.4, each image (benign or adversarial) is fed into the original neural network classifier and an original predicted class is gathered. Here we only consider correctly classified benign input and successful adversarial examples. All attacks are considered to be in *White-Box* scenario when model parameters are exposed to attackers. Each iteration partially transformed images using randomly generated patterns are fed into the classifier, and a behavior vector is collected from probability of images' original prediction. Figure 4.6, 4.7 shows the normalized average behavior vectors of MNIST benign images and adversarial images. Evaluation of adversarial images been used is listed in Table 3.2. As we expected, benign inputs have higher confidence in its original prediction as transformation area are aggregated compared to adversarial images of all generation. Comparing L_∞ attacks and L_2 attacks, when using the same Gaussian blur intensity and under same transformation percentage,

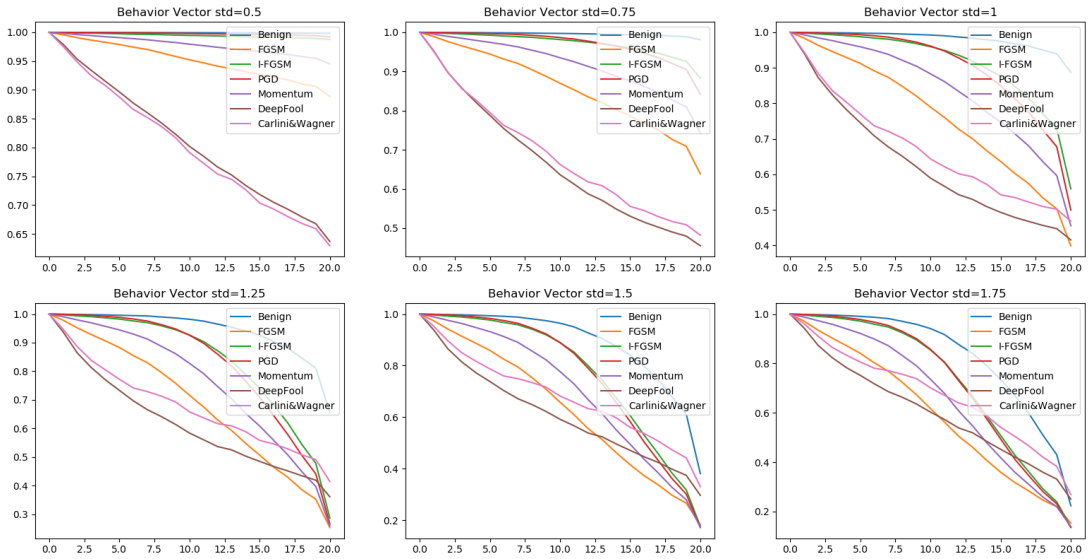

Figure 4.7: Average Probability of Original Predicted Class for L_2 Attacks on MNIST

L_2 attacks are more resistant to these natural transformation.

Different from MNIST, for CIFAR-10 we have less distinguishable difference in predicted probabilities while increasing transformation percentages. Noticed that for CIFAR-10 the averaged behavior vectors for benign inputs degrade faster while increasing transformation percentages compared to MNIST. Our assumption is MNIST are small-scaled black and white images, so the information of digits reserve after transformation.

We perform the same transformation on subset of ImageNet with 100 randomly selected images as seed images, each image is ensured to be from a unique class. All images are transformed using 20 randomly generated patterns, and each pattern includes the transformation percentages from 0% to 100%, aggregating 5% at each iteration, so the average probability are calculated using 2000 behavior vectors for benign input and 2000 behavior vectors for each adversarial generation mechanism. For adversarial examples we only consider successfully attacked input. The probability of original classification for benign input stays at relatively high value after

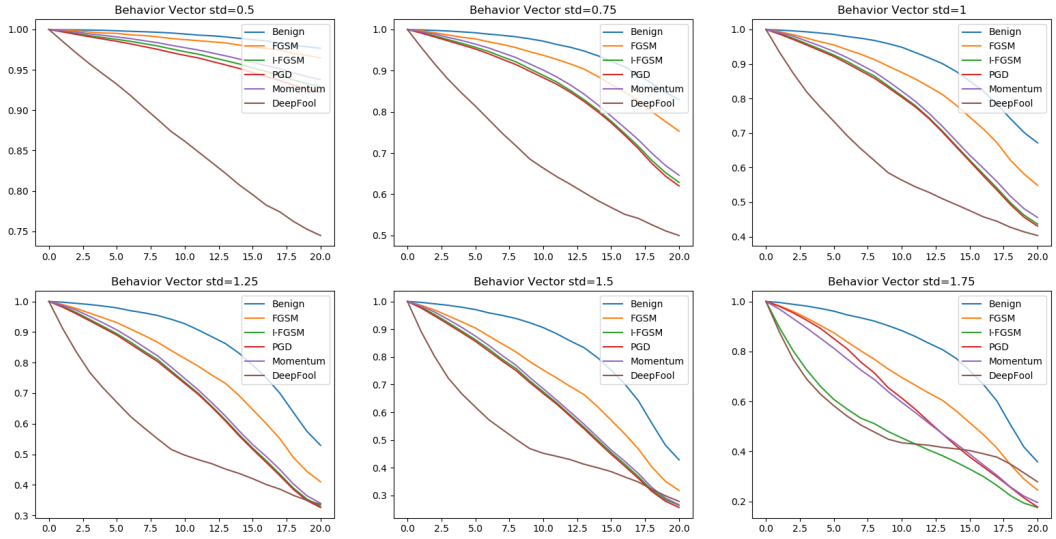

Figure 4.8: Average Probability of Original Predicted Class for L_∞ Attacks on CIFAR-10

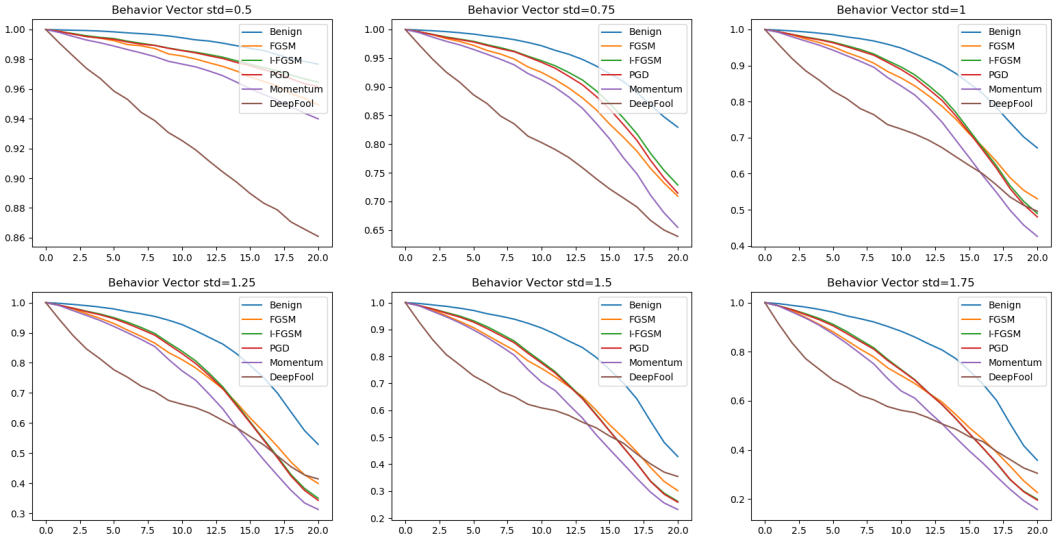

Figure 4.9: Average Probability of Original Predicted Class for L_2 Attacks on CIFAR-10

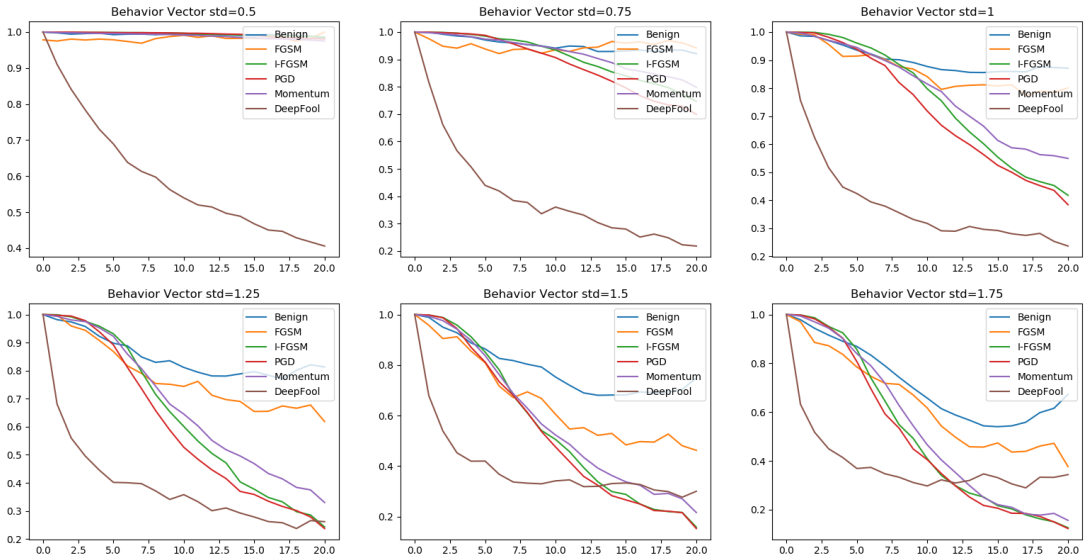

Figure 4.10: Average Probability of Original Predicted Class for L_∞ Attacks on ImageNet

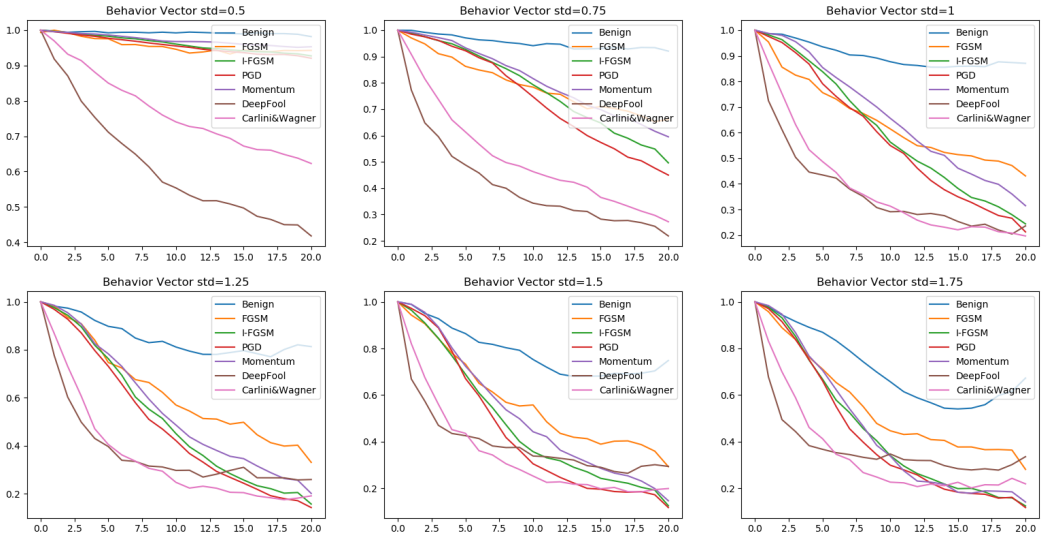

Figure 4.11: Average Probability of Original Predicted Class for L_2 Attacks on ImageNet

normalization. But we see a change in "smoothness" for averaged behavior vectors. Probability of originally classified classes are not guaranteed to drop consistently as we increase the transformation.

4.2.2 One-Class SVM Anomaly Detection

Due to the diversity of adversarial examples, it has been challenging for supervised learning algorithms to identify abnormal examples, since future anomalies may differ largely from the ones that existed. We consider the defense as an one vs all situation, where the detector should reject inputs when the behavior vectors are deviated. So in this experiment we only use benign inputs to train a detector for anomaly detection. Schölkopf et al. [27] proposed using One-Class SVM for novelty detection, it separates training data (normal data points) from the origin in feature space by a hyperplane, maximizes the distance from this hyperplane to the origin. The function returns +1 for data inside regions and output -1 for anomalies outside the regions. Equation 4.4 describes the optimization problem of One-Class SVM detector

$$\begin{aligned} &\min_{w,\xi_i,\rho} \frac{1}{2}\|w\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho \\ &s.t. \\ &(w \cdot \phi(x_i)) \geq \rho - \xi_i \text{ for all i=1,...,n} \\ &\xi_i \geq 0 \text{ for all i = 1,...,n} \end{aligned} \tag{4.4}$$

ξ is the slack variable to allow some points stay in margin. Parameter ν sets the upper bound of error rate for training and the lower bound of training examples used as support vectors. Using Lagrange and kernel function for dot product calculation, the decision function is defined in equation 4.5

	TNR							TPR
	L_∞					L_2		
Blur Sigma	FGSM	BIM	PGD	MOMENTUM	DEEPFOOL	DEEPFOOL	CW	Legitimate
std = 0.75 OCSVM	85.32%	62.95%	64.79%	63.40%	100%	100%	100%	95%
std = 1 OCSVM	79.83%	59.24%	62.21%	56.06%	100%	100%	100%	96.33%
std = 1.25 OCSVM	75.71%	45.80%	50.52%	47.02%	99.6%	99.35%	99.36%	94.5%
std=1.5 OCSVM	70.49%	33.33%	37.52%	31.86%	37.39%	98.2%	97.25%	97.07%
std = 0.5 AE	93.02%	56.29%	59.21%	63.51%	100%	100%	100%	96%
std = 0.75 AE	94.06%	71.61%	75.15%	69.84%	100%	100%	100%	95%
std = 1 AE	86.70%	70.69%	73.31%	68.61%	100%	100%	100%	94%
std = 1.25 AE	21.70%	41.07%	43.94%	26.49%	29%	33.60%	48.40%	96%
std=1.5 AE	42.52%	36.82%	41.21%	31.86%	66.75%	67.30%	73%	95%

Table 4.1: Stress Test on MNIST

$$\begin{aligned} f(x) =& sgn((w \cdot \phi(x_i)) - \rho) \\ =& sgn(\sum_{i=1}^{n} \alpha_i K(x, x_i) - \rho) \end{aligned} \tag{4.5}$$

This algorithm creates a hyperplane, normal input within hyperplane will be classified as +1 and anomaly input outside the hyperplane as -1. We evaluate the detection for One-Class SVM on MNIST using behavior vectors from correctly classified benign images and successful adversarial examples. As shown in Table 4.1, while maintaining 95% TPR (True Positive Rate) for legitimate data, the best performance we get on MNIST is when blur sigma is set to 0.75. We perform the same evaluation on CIFAR-10, results indicates that while we identify most of the DeepFool and Carlini & Wagner attacks, detection rate of gradient based attacks are low compared to MNIST results.

For ImageNet data we achieve lower than 50% TNR (True Negative Rate) for most of the attacks. Our assumption is that as the variety of image content increases, the pattern in behavior vectors tends to change according to image content. When increasing the coverage of blur patches, the prediction confidence of original predicted

	TNR											TPR
	L_∞					L_2						
Blur Sigma	FGSM	BIM	PGD	MOMENTUM	DEEPFOOL	FGSM	BIM	PGD	MOMENTUM	DEEPFOOL	CW	Legitimate
std=1.25 OCSVM	34.28%	23.80%	22.42%	22.63%	76.6%	38.88%	35.32%	32.42%	21.9%	76.9%	74.65%	96%
std=1.5 OCSVM	38.35%	27.03%	26.14%	26.57%	79.9%	45.05%	35.25%	34.36%	23.87%	81.2%	78.5%	95%
std=1.5 AE	12.68%	22.71%	23.21%	22.31%	28.45%	19.12%	13.27%	14.22%	12.20%	29.00%	32%	96%
std=1.75 AE	35.97%	30.72%	33.42%	31.42%	83.65%	42.96%	11.50%	13.57%	11.77%	85.35%	83.80%	88.67%

Table 4.2: Stress Test on CIFAR-10

	TNR											TPR
	L_∞					L_2						
Blur Sigma	FGSM	BIM	PGD	MOMENTUM	DEEPFOOL	FGSM	BIM	PGD	MOMENTUM	DEEPFOOL	CW	Legitimate
std=1 OCSVM	19.40%	61%	57%	63%	46%	24.61%	45.26%	49.46%	33%	45%	40.21%	96%
std=1.25 OCSVM	19.40%	55%	48%	45%	32%	10.77%	33.68%	32.61%	20.69%	33%	28.26%	93%
std=1.5 OCSVM	17.91%	56%	58%	57%	32%	75%	35.06%	31.17%	21.50%	32%	28.26%	93%
std=1.75 OCSVM	5.97%	54%	49%	46%	20%	12.31%	22.07%	29.35%	15.27%	20%	8.69%	93%

Table 4.3: Stress Test on ImageNet

class are not monotonically decreasing. This raise the difficulty of learning an anomaly detector.

4.2.3 Autoencoder Anomaly Detection

An autoencoder is a type of neural network used to learn efficient data codings in a unsupervised manner. Autoencoders consist of two parts: an encode stage that maps the input to a bottleneck internal coded representation, and an decode stage to maps the compressed data to a reconstruction of the input. Autoencoders are often applied to data denoising, dimentionality reduction. We utilized a simple autoencoder to map the behavior vectors from benign images and setting proper threshold for identifying adversarial examples to maintain over 95% TPR. We use square root of mean squared error between input vector and reconstructed vector as the metric to evaluate whether the output is a good representation of input.

Figure 4.12 shows how autoencoder works for identifying abnormal behavior vectors. The autoencoder is trained on benign input only and classify input as irregular when reconstruction error is under threshold. Figure 4.13 is the ROC curve for 3-layer

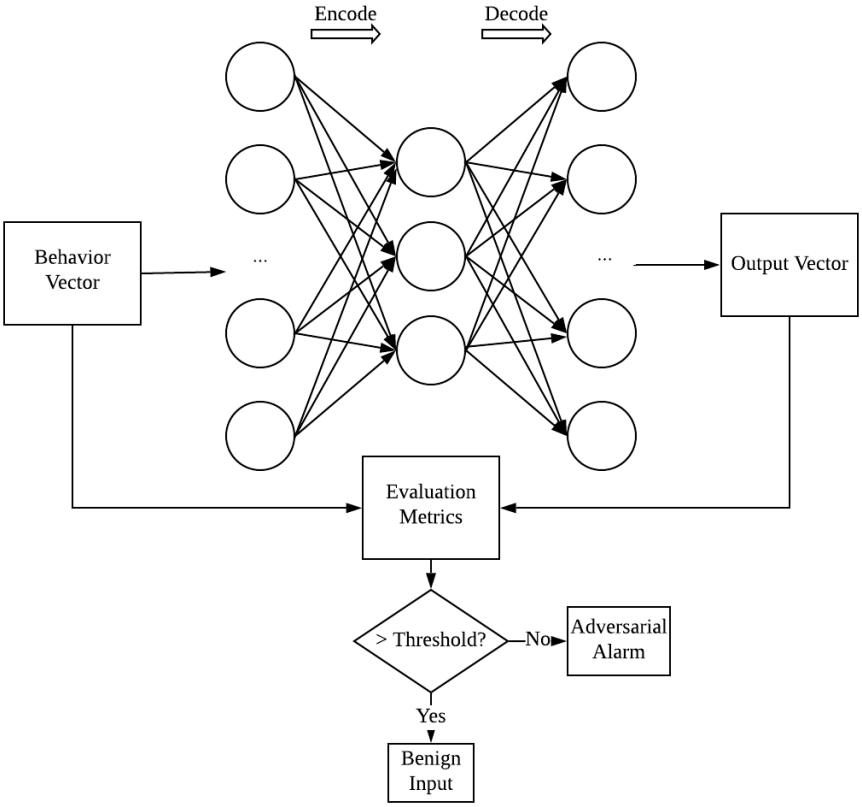

Figure 4.12: Autoencoder for Anomaly Detection

autoencoder trained on benign input from MNIST. Looking deep into the training data, Gaussian blur tend to have consistent impact on image classification when images are from small search spaces like MNIST. The same does not hold for complex input like ImageNet. When the variance of individual data from averaged behavior vectors is small, it is easy for autoencoders to learn the compress and reconstruction pattern.

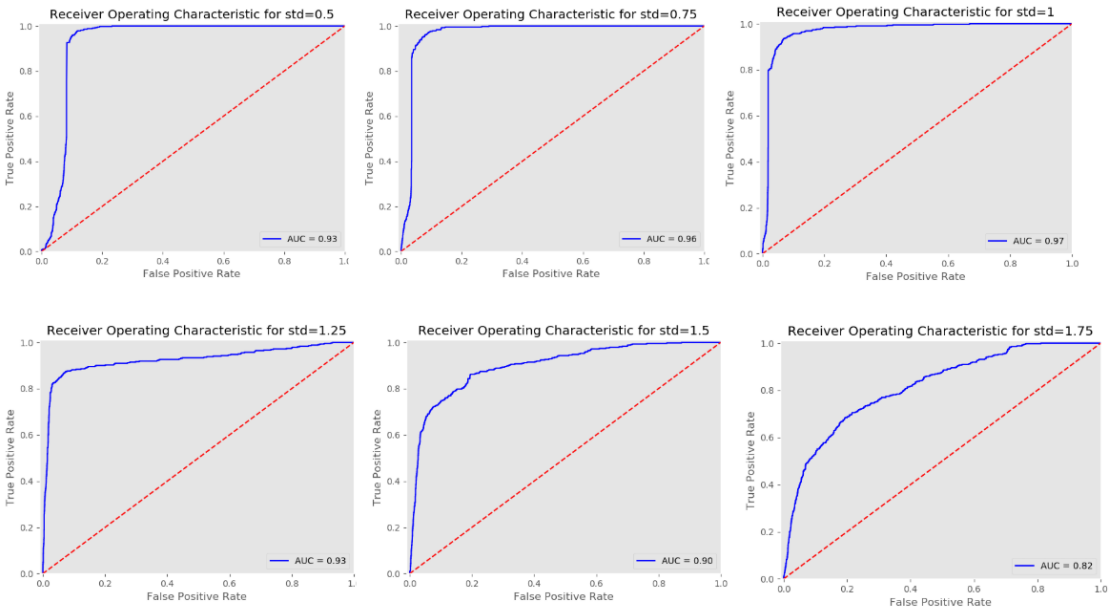

Figure 4.13: ROC Curve for 3-layer Autoencoder Anomaly Detection on MNIST

Chapter 5

COMPARISON OF RESULTS

Due to the fact that many attacks and defenses primarily work on datasets with smaller size images, such as MNIST and CIFAR-10, only few of them report results on large-scaled images like ImageNet. Researchers in the field report their results based on different L_p norms, different attack mechanisms, different constraints on perturbation magnitude and different models been attacked and defended. In an attempt to perform a fair comparison, we reproduced two defense methods [29, 24] that recover the classification of adversarial examples. We evaluate the defenses using the same models and attacks we used in the experiments on MNIST, CIFAR-10 and ImageNet images. Because our defense is a *detection* defense, we modified their defenses to output detection results instead of recovering the images. For Feature Squeezing we output anomaly alarms when L_1 distance of predictions from squeezed and non-squeezed input exceeds threshold value learned. For Pixel Deflection we output adversarial alarms when base model's classification differs from that of pixel deflected images. We define the proportion of successfully perturbed images that detected by defense approaches the *Detection Rate*.

Table 5.1 compares our results with feature squeezing defense on MNIST. For feature squeezing we utilize color-bit reduction, median smoothing and non-local mean filtering and the best joint defense mentioned in the original paper as our squeezing methods. Detail of this defense is introduced in Section 2.2.5. We set the threshold of L_1 distance before and after squeezing to ensure less than 5% error rate on benign input. Notice that feature squeezing works extremely well on small-scaled black and white images due to the simplicity of searching spaces. Perturbations added to

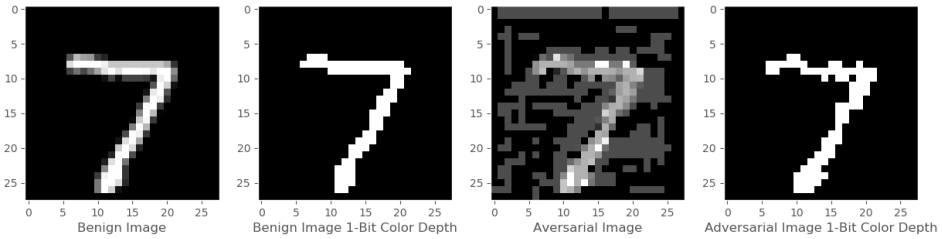

Figure 5.1: Example of 1-bit Color Depth Reduction on MNIST

each pixel will either suppressed to 0 or increased to 255 by 1-bit color reduction. Figure 5.1 shows an example of using 1-bit color depth reduction on MNIST. For the stress test defense, the best results achieved is by using an autoencoder for detecting adversarial examples under transformation $\sigma = 0.75$, which best distinguishes the anomaly behavior vectors from benign behavior vectors. For the MNIST dataset, the best results are achieved by Joint defense using 1-bit color reduction and local median smoothing from feature squeezing, it achieves near 100%detection rate on different attacks, while our stress test defense achieves similar defense rate on L_2 DeepFool and Carlini Wagner attacks and L_∞ attack.

We performed the same defenses on CIFAR-10 dataset. For feature squeezing we applied color bit reduction, local median filtering and non-local filtering. The best joint defense from feature squeezing is using 5-bit color reduction joint with 2x2 local median filtering, and 13-3-2 non-local filtering. Here 2x2 for local filtering refers to using 2x2 sliding window for filtering, and 13-3-2 refers to non-local filtering using 13x13 searching window, with a 3x3 patch size and 2 filtering strength. We include pixel deflection on CIFAR-10 dataset and it achieves lower detection rate compared to feature squeezing. Stress Defense tends to have consistent results on different attacks since it rejects behavior vectors as long as it differs from the normal distribution, independently from the perturbation generation mechanisms. Joint defense from

	L-infinity					L-2		
Defense Parameters	FGSM	BIM	PGD	MOMENTUM	DEEPFOOL	DEEPFOOL	CW	Legitimate
Feature Squeezing								
1-bit color depth	**96.70%**	97.93%	96.52%	97.94%	**100%**	**100%**	**100%**	96%
2-bit color depth	93.41%	28.50%	43.33%	56.70%	**100%**	**100%**	**100%**	95%
2x2 median filter	86.81%	63.44%	61.25%	64.89%	**100%**	**100%**	**100%**	96%
3x3 median filter	93.40%	40.86%	41.40%	48.93%	98%	**100%**	98.93%	96%
Best Joint (1-bit + 2x2 median)	**96.70%**	**100%**	**100%**	**100%**	100%	**100%**	100%	94%
Stress Test								
std = 0.75 OCSVM	85.32%	62.95%	64.79%	63.40%	**100%**	**100%**	**100%**	95%
std = 1 OCSVM	79.83%	59.24%	62.21%	56.06%	**100%**	**100%**	**100%**	96.33%
std = 1.25 OCSVM	75.71%	45.80%	50.52%	47.02%	99.6%	99.35%	99.36%	94.5%
std=1.5 OCSVM	70.49%	33.33%	37.52%	31.86%	37.39%	98.2%	97.25%	97.07%
std = 0.5 AE	93.02%	56.29%	59.21%	63.51%	**100%**	**100%**	**100%**	96%
std = 0.75 AE	94.06%	71.61%	75.15%	69.84%	**100%**	**100%**	**100%**	95%
std = 1 AE	86.70%	70.69%	73.31%	68.61%	**100%**	**100%**	**100%**	94%
std = 1.25 AE	21.70%	41.07%	43.94%	26.49%	29%	33.60%	48.40%	96%
std=1.5 AE	42.52%	36.82%	41.21%	31.86%	66.75%	67.30%	73%	95%

Table 5.1: Comparison Between Different Defense Methods on MNIST

feature squeezing has the best results overall compared with the other two defenses.

For natural images from ImageNet, color bit reduction have inconsistent results over different attacks when used independently. It achieves high detection rates on some attacks and have near zero detection rates on other attacks. The assumptions are quantizing color intensity values by certain threshold can't detect adversarial input under diverse perturbation magnitudes. Pixel deflection achieves highest detection rate on one step FGSM attack, and some iteratively generated attacks. Similar to CIFAR-10 results, Stress Test defense has consistent results over different attacks. On ImageNet dataset, the best joint defense from feature squeezing has the highest detection rates.

From this comparison, we can see that our stress test using behavior vectors generated from spatial random transformation is suitable to identify adversarial inputs on small scaled black and white images. On MNIST our stress test defense is comparable

	L-infinity					L-2						
Defense Parameters	FGSM	I-FGSM	PGD	MOMENTUM	DEEPFOOL	FGM	I-FGM	PGD	MOMENTUM	DEEPFOOL	CW	Legitimate
Feature Squeezing												
4-bit color depth	23.71%	22.34%	25.53%	17.77%	82%	19.15%	10%	11%	10.10%	61%	75%	94%
5-bit color depth	7.22%	12.77%	11.70%	4%	55%	13.83%	7%	8%	4%	52%	56.94%	92%
2x2 median filter	58.76%	63.83%	64.89%	63.33%	81%	64.89%	60%	65%	60.60%	79%	77%	95%
3x3 median filter	65.98%	67.02%	69.14%	65.55%	89%	64.89%	68%	72%	64.64%	**88%**	**88.89%**	93%
11X3X2 non-local mean	28.86%	38.30%	37.23%	34.44%	**97%**	24.47%	13%	17%	12.12%	79%	79.16%	88%
13X3X2 non-local mean	28.86%	38.30%	38.30%	34.44%	**97%**	24.47%	14%	18%	13.13%	79%	79.16%	87%
Best Joint (5-bit, 2x2 median, 13-3-2 non-local)	**81.44%**	**94.68%**	**93.61%**	**92.22%**	86%	**84.04%**	**99%**	**100%**	**95.96%**	85%	76.38%	95%
Pixel Deflection												
PD+wavelet	11.34%	20.21%	19.15%	14.44%	85%	15.96%	6%	9%	12.12%	**85%**	80%	97%
Stress Test												
std=1.25 OCSVM	34.28%	23.80%	22.42%	22.63%	76.6%	38.88%	35.32%	32.42%	21.9%	76.9%	74.65%	96%
std=1.5 OCSVM	38.35%	27.03%	26.14%	26.57%	79.9%	45.05%	35.25%	34.36%	23.87%	81.2%	78.5%	95%
std=1.5 AE	12.68%	22.71%	23.21%	22.31%	28.45%	19.12%	13.27%	14.22%	12.20%	29.00%	32%	96%
std=1.75 AE	35.97%	30.72%	33.42%	31.42%	83.65%	42.96%	11.50%	13.57%	11.77%	85.35%	83.80%	88.67%

Table 5.2: Comparison Between Different Defense Methods on CIFAR-10

	L-infinity					L-2						
Defense Parameters	FGSM	I-FGSM	PGD	MOMENTUM	DEEPFOOL	FGM	I-FGM	PGD	MOMENTUM	DEEPFOOL	CW	Legitimate
Feature Squeezing												
2-bit color depth	4.47%	67%	**68%**	65%	3%	4.61%	66%	63.44%	57.47%	2%	12%	94%
5-bit color depth	10.44%	2%	1.00%	2%	94%	35.38%	26.88%	31.18%	24.13%	95%	91.30%	95%
2x2 median filter	4%	47%	53%	33.00%	3%	7.69%	54.84%	60.21%	41.37%	2%	22.82%	96%
3x3 median filter	3%	46%	53%	33%	3%	2%	64.51%	69.89%	55.17%	4%	22.83%	95%
11X3X2 non-local mean	10.93%	6.06%	5%	5.00%	92%	38%	25.33%	26.88%	31.5	84%	68%	95%
13X3X2 non-local mean	13.28%	6.06%	5%	5.00%	92%	45.90%	26.67%	27.96%	32.88%	83%	71%	95%
Best Joint (5-Bit, 2x2 Median, 11-3-2 Non-local)	15.42%	47%	53%	**73%**	**96%**	56.80%	**72.42%**	**74.30%**	**75.45%**	**97%**	**97%**	96%
Pixel Deflection												
PD+DWT	**38.28%**	39.39%	39.24%	31.00%	91%	**65.57%**	65%	64%	54.79%	91%	88%	92%
Stress Test												
std=1 OCSVM	19.40%	61%	57%	63%	46%	24.61%	45.26%	49.46%	33%	45%	40.21%	96%
std=1.25 OCSVM	19.40%	55%	48%	45%	32%	10.77%	33.68%	32.61%	20.69%	33%	28.26%	93%
std=1.5 OCSVM	17.91%	56%	58%	57%	32%	75%	35.06%	31.17%	21.50%	32%	28.26%	93%
std=1.75 OCSVM	5.97%	54%	49%	46%	20%	12.31%	22.07%	29.35%	15.27%	20%	8.69%	93%
Best Joint std=1.25, std=1.5	16.41%	**69%**	**68%**	64%	36%	16.92%	44.08%	43.01%	33.33%	36%	27.17%	93%

Table 5.3: Comparison Between Different Defense Methods on ImageNet

to state-of-art defenses. While on three channel colored image dataset CIFAR-10 our defense is not competitive compared to other defense approaches. The assumptions are the differences between adversarial behavior vectors and benign behavior vectors are minor (Figure 4.8, Figure 4.9), this increases the difficulty for anomaly detection. On the ImageNet dataset the behavior vectors are not necessarily monotonically decreasing while increasing transformation percentages (Figure 4.10, Figure 4.11), this reduces the utility of using behavior vectors to detect adversarial examples.

Chapter 6

CONCLUSIONS AND FUTURE WORKS

In this thesis we analyze the adversarial perturbations distribution on L_2 and L_∞ attacks. We propose a defense against adversarial examples by leveraging the robustness of benign input to spatial transformations. From our study we found the aggregated transformation can achieve consistent impact on benign inputs, by learning this behavior of benign inputs during natural transformation, our defense mechanism achieves similar results compared to state-of-art defenses on small scaled simple content images like MNIST. Using behavior vectors as a defense can detect anomaly adversarial inputs on complex colored images in some circumstances, but the detection rate is not competitive compared to state-of-art defenses.

Due to the significant computation required for generating adversarial examples, calculating behavior vectors and reproduce defenses, our experiments were limited to small subsets of database using seed images (100 randomly selected seed images for each attack). One can argue that by carefully changing the parameters of generating adversarial examples and exploring a larger dataset, it is possible to generate an adversarial example that follows the same pattern as averaged benign vectors when raising transformation intensity. We haven't validated the performance of our defense on L_0 attacks when the portion of pixels being attacked are limited to smaller ranges. Because our defense is dependent on random partial transformation, L_0 attacks which only perturb small portion of the images without considering perturbation magnitudes can potentially increase the variety of adversarial behavior vectors, and lower the detection rates of our defense. In terms of attack scenarios, our vanilla white-box attacks haven't considered potential attacks when attackers are also aware of the

defenses and trying to bypass the defenses by using an approximation of partial blur, but such an attack seems hard to formulate. It is possible that an attacker can bypass the stress test defense by using a differentiable approximation of partial blur and modification of loss functions to include the anomaly detector model's loss in the adversarial generation. In the future, a more comprehensive evaluation would need to consider such attacks.

REFERENCES

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[2] N. Carlini and D. Wagner. Defensive distillation is not robust to adversarial examples. *arXiv preprint arXiv:1607.04311*, 2016.

[3] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[5] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

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

[7] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[8] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[9] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

[10] S. Gu and L. Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

[11] D. Hendrycks and K. Gimpel. Early methods for detecting adversarial images. *arXiv preprint arXiv:1608.00530*, 2016.

[12] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-10 (canadian institute for advanced research).

[13] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial machine learning at scale. 2016.

[14] Y. LeCun, C. Cortes, and C. J. Burges. The mnist database of handwritten digits, 1998. *URL http://yann. lecun. com/exdb/mnist*, 10:34, 1998.

[15] J. Lu, T. Issaranon, and D. Forsyth. Safetynet: Detecting and rejecting adversarial examples robustly. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 446–454, 2017.

[16] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[17] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[18] A. Nayebi and S. Ganguli. Biologically inspired protection of deep networks from adversarial attacks. *arXiv preprint arXiv:1703.09202*, 2017.

[19] M. Osadchy, J. Hernandez-Castro, S. Gibson, O. Dunkelman, and D. Pérez-Cabo. No bot expects the deepcaptcha! introducing immutable adversarial examples, with applications to captcha generation. *IEEE Transactions on Information Forensics and Security*, 12(11):2640–2653, 2017.

[20] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, A. Matyasko, V. Behzadan, K. Hambardzumyan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber, and R. Long. Technical report on the cleverhans v2.1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2018.

[21] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[22] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016.

[23] O. M. Parkhi, A. Vedaldi, A. Zisserman, et al. Deep face recognition. In *bmvc*, volume 1, page 6, 2015.

[24] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. Storer. Deflecting adversarial attacks with pixel deflection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8571–8580, 2018.

[25] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi. Microsoft malware classification challenge. *arXiv preprint arXiv:1802.10135*, 2018.

[26] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[27] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pages 582–588, 2000.

[28] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[29] W. Xu, D. Evans, and Y. Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

APPENDIX A

RAW DATA

L-p norm	Attack	L-infinity	Success Rate	Prediction Confidence
L_infinity	FGSM	0.1	12%	82.43%
		0.2	51%	77.16%
		0.3	91%	68.01%
		0.4	100%	70.74%
		0.5	99%	79.43%
	I-FGSM	0.1	31%	87.35%
		0.12	52%	89.47%
		0.14	69%	93.91%
		0.16	89%	94.08%
		0.18	97%	97.27%
	PGD	0.1	31%	88.24%
		0.12	48%	91.51%
		0.14	71%	92.77%
		0.16	89%	93.72%
		0.18	98%	96.75%
	Momentum	0.1	27%	88.10%
		0.12	46%	85.37%
		0.14	64%	89.87%
		0.16	79%	91.54%
		0.18	94%	90.58%
	DeepFool	0.16	100%	51.07%

Table A.1: Evaluation of L-infinity Attacks on MNIST

L-p norm	Attack	L-2	Success Rate	Prediction Confidence
L_2	FGM	1	7%	78.2%
		2	32%	82.96%
		3	47%	78.61%
		4	51%	78.94%
		5	52%	79.18%
	I-FGM	0.5	4%	59.88%
		1	19%	90.68%
		2	73%	96.52%
		3	79%	97.61%
		4	79%	98.02%
	PGD	0.5	2%	88.24%
		1	18%	91.51%
		2	77%	96.98%
		3	93%	99.87%
		4	98%	99.95%
	Momentum	0.5	2%	88.10%
		1	46%	85.37%
		2	76%	89.87%
		3	98%	91.54%
		4	99%	90.58%
	DeepFool	1.88	100%	51.46%
	C&W	2.12	97%	48.98%

Table A.2: Evaluation of L-2 Attacks on MNIST

L-p norm	Attack	L-infinity	Success Rate	Prediction Confidence
L_infinity	FGM	0.01	47%	71.03%
		0.02	77%	76.03%
		0.03	89%	79.69%
		0.04	97%	78.50%
		0.05	100%	78.12%
	I-FGM	0.005	30%	61.58%
		0.01	59%	80.72%
		0.02	95%	92.14%
		0.03	100%	96.10%
		0.04	100%	96.92%
	PGD	0.005	30%	61.72%
		0.01	59%	80.70%
		0.02	94%	92.30%
		0.03	100%	95.98%
		0.04	100%	95.36%
	Momentum	0.005	29%	62.27%
		0.01	58%	80.64%
		0.02	90%	92.58%
		0.03	100%	93.39%
		0.04	100%	95.11%
	DeepFool	0.01	100%	54.54%

Table A.3: Evaluation of L-infinity Attacks on CIFAR-10

L-p norm	Attack	L-2	Success Rate	Prediction Confidence
L_2	FGM	0.5	75%	63.29%
		1	92%	69.49%
		1.5	96%	69.54%
		2	96%	69.61%
	I-FGM	0.5	78%	85.59%
		1	100%	95.86%
		1.5	100%	98.29%
	PGD	0.5	78%	84.98%
		1	100%	94.16%
		1.5	100%	97.29%
	Momentum	0.5	90%	71.35%
		1	92%	61.89%
		1.5	92%	52.55%
	DeepFool	0.41	100%	54.06%
	C&W	0.36	100%	51.15%

Table A.4: Evaluation of L-2 Attacks on CIFAR-10

L-p norm	Attack	L-infinity	Success Rate	Prediction Confidence
L_infinity	FGM	0.03	67%	47.54%
		0.04	66%	46.00%
		0.05	64%	44.70%
	I-FGM	0.005	99%	95.40%
		0.01	100%	97.61%
		0.015	100%	98.68%
	PGD	0.005	99%	95.73%
		0.01	100%	97.62%
		0.015	100%	98.69%
	Momentum	0.005	100%	94.11%
		0.01	100%	96.45%
		0.015	100%	97.87%
	DeepFool	0.0014	100%	36.17%

Table A.5: Evaluation of L-infinity Attacks on ImageNet

L-p norm	Attack	L-2	Success Rate	Prediction Confidence
L_2	FGM	0.5	50%	49.82%
		1	61%	50.89%
		1.5	65%	51.79%
	I-FGM	0.5	77%	94.29%
		1	93%	88.12%
		1.5	95%	87.31%
	PGD	0.5	77%	92.12%
		1	93%	84.91%
		1.5	92%	88.63%
	Momentum	0.5	72%	88.92%
		1	87%	82.11%
		1.5	93%	79.18%
	DeepFool	0.36	100%	36.75%
	C&W	0.8	91%	57.23%

Table A.6: Evaluation of L-2 Attacks on ImageNet