博士論文

Automated Detection System for Adversarial Examples
on Images with Image Transformation and Filters

画像のフィルタリングと変換を用いた
敵対例攻撃の自動検出システム

情報セキュリティ大学院大学
情報セキュリティ研究科
情報セキュリティ専攻
5674101　DANG DUY THANG

指導教員　松井俊浩

2020 年 3 月

Automated Detection System for Adversarial Examples on Images with Image Transformation and Filtering

画像のフィルタリングと変換を用いた敵対例攻撃の自動検出システム

by

Dang Duy Thang

Submitted to the Institute of Information Security
in fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

at

THE INSTITUTE OF INFORMATION SECURITY

March 2020

Author .....
Institute of Information Security
March, 2020

Certified by .....
Toshihiro Matsui
Professor
Thesis Supervisor

Accepted by .....
Atsuhiro Goto
Chairman, The Institute of Information Security

Doctoral Dissertation

Automated Detection System for Adversarial Examples on Images with Image Transformation and Filtering

画像のフィルタリングと変換を用いた敵対例攻撃の自動検出システム

by

Dang Duy Thang

Submitted to the Institute of Information Security
on March, 2020, in fulfillment of the
requirements for the degree of
DOCTOR OF PHILOSOPHY

Abstract

Deep neural networks has been applied in many task with encouraging results and reached human-level performance. These models that make predictions by learning from the training data. In this context, accuracy prediction denotes the fraction of test inputs that a model processes correctly the proportion of images that an object recognition algorithm recognizes as belonging to the correct class, and the proportion of executables that a malware detector correctly designates as benign or malicious. The estimate of a model's accuracy varies greatly with the choice of the dataset used to compute the estimate. The model's accuracy is generally evaluated on test inputs that were not used during the training process. The accuracy is usually higher if the test inputs resemble the training images more closely. For example, a image classification system trained carefully on images may obtain high accuracy when tested on other images in the same distribution. Machine learning has traditionally been developed following the assumption that the environment is benign during both training and evaluation of the model. Specifically, the inputs are usually assumed to all be drawn independently from the same probability distribution at both training and test time. This means that while test inputs are new and previously unseen during the training process, they at least have the same statistical properties as the inputs used for training. Such assumptions have been useful for designing effective machine learn-

ing algorithms but implicitly rule out the possibility that an adversary could alter the distribution at either training time or test time. However, deep neural networks have been recently found vulnerable to well-designed input samples that called adversarial examples. Such an issue causes deep neural networks to misclassify adversarial examples that are imperceptible to humans. Because of these major security issues of adversarial examples, our research is focusing on how to make the AI system more secure, able to identify and distinguish between adversarial and legitimate images. Our research mainly investigates building detection and distinguishing between adversarial examples and legitimate images in image classification tasks. In this research, we propose the-state-of-the-art detection systems for automatically detecting adversarial examples on deep neural networks by using affine transformation and image filtering. Our proposed systems can perfectly distinguish adversarial samples and benign images in an end-to-end manner without human intervention. We exploited the important role of image transformation and filters in adversarial samples and proposing state-of-the-art methods for detecting malicious samples based on our observations. We evaluated our methods on a variety of standard benchmark datasets including MNIST and ImageNet. Our methods reached out to detection rates in a range from 93.2% to 100% in many settings. Besides, we proposed a new idea to evaluate the robustness of the AI system against adversarial examples. Evaluating the robustness of AI systems is very important because of raising adversarial attacks. This idea is also discussed in this thesis.

Thesis Supervisor: Toshihiro Matsui
Title: Professor

Acknowledgments

This thesis would have been impossible without the support and mentoring of my advisor, Professor Toshihiro Matsui. Even after three years of working with him, I am constantly surprised by his amazing intelligence, infinite energy, boundless optimism, and genuine friendliness. I wish I could incorporate more of his qualities. Professor Matsui highlighted fully security on machine learning as an interesting open problem and especially adversarial examples is a subject undergoing intense study around the world at the first our meeting three years ago. I am inspired by his explanation and discussion very much. Afterward, I received that I am really interested in this research area. He also frequently pushed and corrected my English. I have also received a lot of fruitful comments and supports from my labmates Jun Fukuyama, Takumi Iwaki, Hidehito Kobayashi, Masatoshi Ogawa, Taisei Kondō and Kazuyuki Ishii in the Matsui's laboratory. I really appreciated Professor Akira Otsuka for his helpful comments in every presentation. I would not have the chance to study and settled down in Japan without enthusiasm support from Professor Atsuhiro Goto, Professor Hidehiko Tanaka, Professor Harumichi Yuasa, and Ms. Miwako Yamaguchi. I also would like to express my gratitude to my parents Dang Duy Quyet and Tran Thi Kim Thanh, my wife Nguyen Thi Quynh Trang, my son Dang Duy Nhat Minh and another my beloved family members. I can not imagine what will happen without their help and support. This work was supported by the Iwasaki Tomomi scholarship.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Background

1.1 Introduction

The rapid increase in large amounts of data processing and automation has been partly satisfied by the outstanding development of deep learning models or sometimes mentioned as deep neural networks (DNNs). Deep learning has been applied to many challenges and achieve very impressive results in big data [121], computer vision [115], natural language processing [8, 40], speech recognition [39], and playing games [103]. Because of its high performance, deep learning is also being widely applied to systems requiring high safety such as self-driving cars [46], surveillance systems based on facial recognition [41], gestures [117], and fingerprints [116]. These systems are being used very widely and continue to grow strongly in the following years. However, in recent studies, many researchers [109, 31, 10] have found the security risks associated with deep learning systems. One of the typical examples that can be considered is the safety of the self-driving car system based on automatic image processing features. What if the camera system on self-driving cars misidentified the "STOP" sign to the "SPEED UP" sign. Another example, the security system misidentifies a stranger's face as a company employee, and the fact that strangers can gain access to a successful agency through a camera system has been deceived. This is one of the security issues that are very challenging for systems based on deep learning. Typical of these risks is adversarial examples. This is a concept that is considered

to be first mentioned in the study of Szegedy et al. [109]. Adversarial examples can be interpreted as a very elaborate form of data that makes it impossible for humans to recognize the difference from the original data, but it causes deep learning models to make a completely wrong recognition. Adversarial examples not only affect the scope of image classification [31], but can also impact other fields such as natural language processing [60], speech recognition [15], malware detection [34] and object detection [112]. In the field of natural language processing, many recent studies on adversarial examples have been also proposed. Bin Liang et al [60] proposed a new attack method based on the Hot Training Phrases (HTP) to deceive a deep learning system. Ji Gao et al. [27] used two main steps for evading a target deep learning classifier. The first step is to determine the most important tokens to modify, and the second step is to change slightly the important words before fooling a target deep learning system. In the speech recognition field, Nicholas Carnili et al. [15] introduced an audio adversarial attack method that targeted to DeepSpeech [37], a state-of-the-art speech-to-text transcription system. Rohan Taori et al. [111] combined genetic algorithm and gradient method to create a black-box audio adversarial example that attacked to an automatic speech recognition (ASR) system. In Malware detection, Kathrin Grosse et al. [34] adopted the adversarial attack method based on the jacobian matrix to craft adversarial samples by using the DREBIN Android malware dataset [6] to fool a malware classifier. In image classification, adversarial examples garnered the most attention with tremendous research [3] on offensive and defensive methods.

To deal with the rising risks of adversarial examples, there are many studies have been proposed with the aim to detect or defeat adversarial examples and protect a deep learning system. Papernot et al. [87] invented a defensive method called defensive distillation to remove the effectiveness of adversarial samples on deep neural networks. Buckman et al. [12] proposed a new modification strategy to neural network architectures that make it more robustness against adversarial examples. Guneet et al. [20] introduced the stochastic activation pruning strategy for adversarial defense, this method is based on the game theory to minimize the zero-sum game between

the adversary and the model. However, there is still no perfect defensive method for adversarial examples and previous defensive methods demand many fine-tuning and intervention from a human. For solving these disadvantages of previous defensive algorithms, we proposed a state-of-the-art detection system that can automatically distinguish adversarial and benign images. Our defense strategies for defeating adversarial examples are based on image transformation and image filters that our research focus on this direction so we will review and discuss more related research works in the Chapter 3.

1.2 Background

Adversarial examples are affected not only in image processing, but also in many other research fields such as natural language processing, audio/speech processing, malware detection. But in our research direction, we only focus on image processing, especially image classification with adversarial examples. So we concentrate on image classification and it will be as default task throughout this thesis.

1.2.1 Image Classification

In this section we will introduce an Image Classification problem, which is a task of assigning an input image one label from a fixed set of categories. This is one of the core problems in Computer Vision that, despite its simplicity, has a large variety of practical applications. Moreover, as we will see later in the section, many other seemingly distinct Computer Vision tasks (such as object detection, segmentation) can be reduced to image classification. Neural networks consist of elementary computing units named neurons organized in interconnected layers. Each neuron applies an activation function to its input to produce an output. Let's start with the input of a machine learning model, in which each network layer produces an output used as input by the next layer. Networks with a single intermediate hidden layer are qualified as shallow neural networks, whereas models with multiple hidden layers are deep neural networks. Using of multiple hidden layers is interpreted as hierarchically

extracting representations from the input, eventually producing a representation relevant to solve the machine learning task and output a prediction. A neural network model F can be formalized as the composition of multidimensional and parametrized functions f_i each corresponding to a layer of the network architecture and a representation of the input, where each vector i parametrizes layer i of the network F and includes weights for the links connecting layer i to layer i_1. A set of model parameters $= i$ is learned during training. For instance, in a supervised learning manner, parameter values are learned by computing prediction errors $f(x) - y$ on a collection of known input output pairs (x, y).

1.2.2 Linear Classification

We are now going to develop a more powerful approach to image classification that we will eventually naturally extend to entire Neural Networks and Convolutional Neural Networks. The approach will have two major components: a score function that maps the raw data to class scores, and a loss function that quantifies the agreement between the predicted scores and the ground truth labels. We will then cast this as an optimization problem in which we will minimize the loss function with respect to the parameters of the score function.

Score function

The first component of this approach is to define the score function that maps the pixel values of an image to confidence scores for each class. We will develop the approach with a concrete example. Let's assume a training dataset of images $x_i \in R^D$, each associated with a label y_i. Here i=1...N and $y_i \in 1..K$. That is, we have **N** examples (each with a dimensionality **D**) and **K** distinct categories. For examples, in MNIST dataset, we have a training set of **N**=50,000 images, each with **D**=28×28×1=784 pixels, and **K**=10, since there are 10 distinct classes (digit 0, 1, ..., 9). We will now define the score function $f : R^D \mapsto R^K$ that maps the raw image pixels to class scores. Let consider the simplest possible linear classifier function, a linear mapping:

$f(x_i, W, b) = Wx_i + b$. We assume that the image x_i has all of its pixels flattened out to a single column vector of shape $[D \times 1]$. The matrix W (of size[K $\times$ D]), and the vector **b** (of size [K $\times$ 1]) are the parameters of the function. The parameters in **W** are often called the weights, and **b** is called the bias vector because it influences the output scores, but without interacting with the actual data x_i. However, you will often hear people use the terms *weights* and *papameters* interchaneably.

Loss function

In the context of an optimization algorithm, the function used to evaluate a candidate solution (i.e. a set of weights) is referred to as the objective function. We may seek to maximize or minimize the objective function, meaning that we are searching for a candidate solution that has the highest or lowest score respectively. Typically, with neural networks, we seek to minimize the error. As such, the objective function is often referred to as a cost function or a loss function and the value calculated by the loss function is referred to as simply "loss".

1.2.3 Optimization Algorithms

Optimization is the process of finding the set of parameters **W** that minimize the loss function.

Random Search

Since it is so simple to check how good a given set of parameters **W** is, the first idea that may come to mind is to simply try out many different random weights and keep track of what works best. The core idea is that finding the best set of weights **W** is a very difficult or even impossible problem (especillay once **W** contains weights for entire complex neural networks), but the problem of refining a specific set of weights **W** to be slightly better is significantly less difficult. In other words, our approach will be to start with a random **W** and then iteratively refine it, making it slightly better each time.

Gradient

We can compute the best direction along which we should change our weight vector that is mathematically guaranteed to be the direction of the steepest descend (at least in the limit as the step size goes towards zero). This direction will be related to the gradient of the loss function. In our hiking analogy, this approach roughtly corresponds to feeling the slope of the hill below our feet and stepping down the direction that feels steepest. In one-dimensional functions, the slope is the instantaneous rate of change of the function at any point you might be interested in. The gradient is a generalization of slope for functions that don't take a single number but a vector of numbers. Additionally, the gradient is just a vector of slopes (more commonly referred to as derivatives) for each dimension in the input space. The mathematical expression for the derivative of a 1-D function with respect its input is:

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f)x}{h} \tag{1.1}$$

When the function of interest take a vector of numbers instead of a single number, we call the derivatives **partial derivatives**, and the gradient is simple the vector of partial derivatives in each dimension. There are two ways to compute the gradient: A slow, approximate but easy way (numerical gradient), and a fast, exact but more error-prone way that requires calculus (analytic gradient). The numerical gradient is very simple to compute using the finite difference approximation, but the downside is that it is approximate (since we have to pick a small value of h, while the true gradient is defined as the limit as h goes to zero), and that it is very computationally expensive to compute. The second way to compute the gradient is analytically using Calculus, which allows us to derive a direct formula for the gradient (no approximations) that is also very fast to compute. However, unlike the numerical gradient it can be more error prone to implement, which is why in practice it is very common to compute the analytic gradient and compare it to the numerical gradient to check the correctness of your implementation. This is called a gradient check. Lets use the example of the SVM loss function for a single datapoint: $L_i = \sum_{j \neq y_i} [\max(0, w_j^T x_i - w_{y_i}^T x_i + \Delta)]$. We

can differentiate the function with respect to the weights. For example, taking the gradient with respect to w_{y_i} we obtain:$\bigtriangledown_{w_{y_i}} L_i = -\left(\sum_{j \neq y_i} \mathbf{1}(w_j^T x_i - w_{y_i}^T x_i + \Delta > 0)\right) x_i$, where $\mathbf{1}$ is the indicator function that is one if the condition inside is true or zero otherwise. While the expression may look scary when it is written out, when you're implementing this in code you would simply count the number of classes that did not meet the desired margin (and hence contributed to the loss function) and then the data vector x_i scaled by this number is the gradient. Notice that this is the gradient only with respect to the row of W that corresponds to the correct class. For the other rows where $j \neq y_i$ the gradient is: $\bigtriangledown_{w_{y_i}} L_i = \mathbf{1}(w_j^T x_i - w_{y_i}^T x_i + \Delta > 0) x_i$. Once you derive the expression for the gradient it is straight-forward to implement the expressions and use them to perform the gradient update.

Gradient Descent Now that we can compute the gradient of the loss function, the procedure of repeatedly evaluating the gradient and then performing a parameter update is called *Gradient Descent*. **Mini-batch gradient descent**, in large-scale applications (such as the ILSVRC challenge), the training data can have on order of millions of examples. Hence, it seems wasteful to compute the full loss function over the entire training set in order to perform only a single parameter update. A very common approach to addressing this challenge is to compute the gradient over batches of the training data. For example, in current state of the art Convolutional Neural Network, a typical batch contains 256 examples from the entire training set of 1.2 million. The extreme case of this is a setting where the mini-batch contains only a single example. This process is called **Stochastic Gradient Descent (SGD)** (or also sometimes **on-line gradient descent**). This is relatively less common to see because in practice due to vectorized code optimizations it can be computationally much more efficient to evaluate the gradient for 100 examples, than the gradient for one example 100 times. Even though SGD technically refers to using a single example at a time to evaluate the gradient, you will hear people use the term SGD even when referring to mini-batch gradient descent (i.e. mentions of MGD for **"Minibatch Gradient Descent"** , or BGD for **"Batch gradient descent"** are rare to see), where it is usually assumed that mini-batches are used. The size of the mini-batch is a

hyperparameter but it is not very common to cross-validate it. It is usually based on memory constraints (if any), or set to some value, e.g. 32, 64 or 128. We use powers of 2 in practice because many vectorized operation implementations work faster when their inputs are sized in powers of 2.

1.2.4 Convolutional Neural Networks

Architecture Overview

Convolutional Neural Networks (CNNs) take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a Convolutional Neural Network have neurons arranged in 3 dimensions: width, height, depth (note that the word depth here refers to the third dimension of an activation volume, not to the depth of a full Neural Network, which can refer to the total number of layers in a network.) Neural Networks receive an input (a single vector), and transform it through a series of hidden layers. Each hidden layer is made up of a set of neurons, where each neuron is fully connected to all neurons in the previous layer, and where neurons in a single layer function completely independently and do not share any connections. The last fully-connected layer is called the "output layer" and in classification settings it represents the class scores.

Layers in CNNs. A simple CNN is a sequence of layers, and every layer of a CNN transforms one volume of activations to another through a differentiable function. We use three main types of layers to build CNN architectures: $ConvolutionalLayer$, $PoolingLayer$, and $Fully-ConnectedLayer$ (exactly as seen in regular Neural Networks). We will stack these layers to form a full CNN architecture.

Convolutional Layer

The Convolutional Neural layer is the core building block of a Convolutional Neural Network that does most of the computational heavy lifting. Fig. 1-1 shows an architecture of Inception V3 [108], one of deep convolutional neural networks. Lets

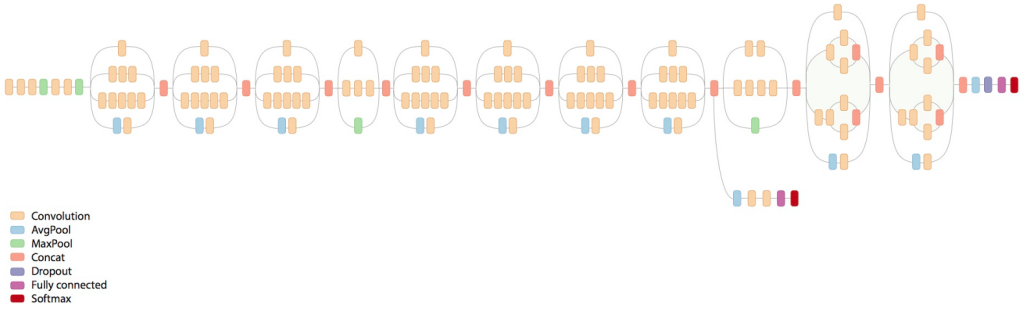


Figure 1-1: Inception V3 [108] - one of deep convolutional neural networks

first discuss what the CNN layer computes without brain/neuron analogies. The CNN layer's parameters consist of a set of learnable filters. Every filter is small space(along width and height), but extends through the full depth of the input volume. For example, a typical filter on a first layer of a CNN might have size 5×5×3 (i.e. 5 pixels width and height, and 3 because images have depth 3, the color channels). During the forward pass, we slide (more precisely, convolve) each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position. As we slide the filter over the width and height of the input volume we will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position. Intuitively, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color on the first layer, or eventually entire honeycomb or wheel-like patterns on higher layers of the network. Now, we will have an entire set of filters in each CNN layer (e.g. 12 filters), and each of them will produce a separate 2-dimensional activation map. We will stack these activation maps along the depth dimension and produce the output volume.

Local Connectivity. When dealing with high-dimensional inputs such as images, as we saw above it is impractical to connect neurons to all neurons in the previous volume. Instead, we will connect each neuron to only a local region of the input volume. The spatial extent of this connectivity is a hyper parameter called the receptive field of the neuron (equivalently this is the filter size). The extent of the

connectivity along the depth axis is always equal to the depth of the input volume. It is important to emphasize again this asymmetry in how we treat the spatial dimensions (width and height) and the depth dimension: The connections are local in space (along width and height), but always full along the entire depth of the input volume.

Spatial arrangement. We have explained the connectivity of each neuron in the Convolutional Neural Layer to the input volume, but we haven't yet discussed how many neurons there are in the output volume or how they are arranged. Three hyperparameters control the size of the output volume: the depth, stride and zero-padding. We discuss these as bellow:

- **depth.** The depth of the output volume is a hyper parameter: it corresponds to the number of filters we would like to use, each learning to look for something different in the input. For example, if the first **Convolutional Layer** takes as input the raw image, then different neurons along the depth dimension may activate in presence of various oriented edges, or blobs of color. We will refer to a set of neurons that are all looking at the same region of the input as a **depth column**.

- **stride.** We must specify the stride with which we slide the filter. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2 (or uncommonly 3 or more, though this is rare in practice) then the filters jump 2 pixels at a time as we slide them around. This will produce smaller output volumes spatially.

- **zero-padding.** Sometimes it will be convenient to pad the input volume with zeros around the border. The size of this **zero-padding** is a hyper parameter. The nice feature of zero padding is that it will allow us to control the spatial size of the output volumes (most commonly as we will see soon we will use it to exactly preserve the spatial size of the input volume so the input and output width and height are the same).

Pooling Layer

It is common to periodically insert a Pooling layer in-between successive Convolutional Neural layers in a Convolutional Neural Network architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. The most common form is a pooling layer with filters of size 2×2 applied with a stride of 2 downsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Every MAX operation would in this case be taking a max over 4 numbers (little 2×2 region in some depth slice). The depth dimension remains unchanged. More generally, the pooling layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires two hyper parameters: their spatial extent **F** and the stride **S**
- Produces a volume of size $W_2 \times H_2 \times D_2$ where: $W_2 = (W_1 - F/)S + 1$, $H_2 = (H_1 - F/)S + 1$ and $D_2 = D_1$
- Creates zero parameters since it computes a fixed function of the input
- For Pooling layers, it is not common to pad the input using zero-padding

It is worth noting that there are only two commonly seen variations of the max pooling layer found in practice: a pooling layer with F=3, S=2 (also called overlapping pooling), and more commonly F=2, S=2. Pooling sizes with larger receptive fields are too destructive.

Normalization Layer

Many types of normalization layers have been proposed for use in CNN architectures, sometimes with the intentions of implementing inhibition schemes observed in the biological brain. However, these layers have since fallen out of favor because in practice their contribution has been shown to be minimal, if any.

Fully-connected layer

Neurons in a fully connected (FC) layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset. It is worth noting that the only difference between FC and CNN layers is that the neurons in the convolutional layer are connected only to a local region in the input, and that many of the neurons in a convolutional volume share parameters. However, the neurons in both layers still compute dot products, so their functional form is identical. Therefore, it turns out that it's possible to convert between FC and convolutional layers:

- For any convolutional layer there is an FC layer that implements the same forward function. The weight matrix would be a large matrix that is mostly zero except for at certain blocks (due to local connectivity) where the weights in many of the blocks are equal (due to parameter sharing).

- Conversely, any FC layer can be converted to a convolutional layer. For example, an FC layer with K=4,096 that is looking at some input volume of size $7\times7\times512$ can be equivalently expressed as a convolutional layer with F=7, P=0, S=1, K=4,096. In other words, we are setting the filter size to be exactly the size of the input volume, and hence the output will simply be $1 \times 1 \times 4,096$ since only a single depth column "fits" across the input volume, giving identical result as the initial FC layer.

Of these two conversions, the ability to convert an FC layer to a convolutional layer is particularly useful in practice. Consider a CNN architecture that takes a $299\times299\times3$ image, and then uses a series of convolutional layers and POOL layers to reduce the image to an activations volume of size $7 \times 7 \times 512$.

Chapter 2

Literature Review on Adversarial Examples

2.1 A taxonomy of Adversarial Attacks

In this section, we will provide a qualitative taxonomy on different terms and keywords related to adversarial attacks and categorize the threat models. Adversarial examples are considered as one variant of adversarial machine learning that has been studied for decades [19]. Adversarial machine learning covers many privacy and security problems on a machine learning system, such as biometric authentication [54], spam detection [92], and fraud detection [26]. Adversarial examples [109] is a new type of adversarial machine learning that is mainly focused on deceiving a machine learning by slightly modifying legitimate input but unrecognized by a human. A machine learning system has normally been designed with an assumption that both training and test datasets are benign during training and evaluating phases. In the training process, the data distribution is squeezed and only the weights of the model are fine-tuned by using gradient descent and backpropagation of the gradients to find the best weights for the input-output pairs (x, y). The inputs x are usually considered to be sampled independently from the unchanged distribution in the training process. These assumptions are useful to concentrate on adopting the weights of the machine learning model. The creation process of adversarial examples is performed in the

opposite way. The attackers squeeze the weights of the victim machine learning model and slightly modify the input data to cause the classifier to misclassify. The new input data is called adversarial examples. There are many ways to categorize the adversarial examples attacks and they depend on the adversary purposes or the amount of knowledge that attackers are able to learn from the victim models.

2.1.1 Targeted/Non-targeted Attacks

Firstly, the adversarial attacks can be categorized by the type of adversary goals.

- **Non-targeted Attack.** In this the case adversary's goal is to cause the classifier to predict any inccorect label. The specific incorrect label does not matter. There are several not-targeted attacks [52, 106, 81, 80] that were considered and investigated.

- **Targeted Attack.** In this case the adversary aims to change the classifier's prediction to some specific target class. This type of adversarial attack is attracted by many researchers and there are many proposed targeted attacks [109, 31, 85, 14].

2.1.2 White-box/Black-box Attacks

Second, the adversarial attacks can be categorized by the amount of knowledge the attackers have about the victim models.

- **White-box Attacks.** In the white box scenario [109, 31, 85, 14, 81, 80, 69, 14], the adversary has full knowledge of the model including model type, model architecture and values of all parameters an trainable weights.

- **Black-box Attacks.** There are two type of black-box attacks [106, 84, 64, 43, 1, 14] where attackers have limitation to access into victim models for creating adversarial examples.

  Black box with querying. In this scenario, the adversary does not know very much about the model, but can query the model, i.e. feed some inputs and

observe outputs. There are many variants of this scenario—the adversary may know the architecture but not the parameters or the adversary may not even know the architecture, the adversary may be able to observe output probabilities for each class or the adversary may only be to observe the choice of the most likely class.

Black box without querying. In the black box without querying scenario, the adversary has limited or no knowledge about the model under attack and is not allowed to query the model while constructing adversarial examples. In this case, the attacker must construct adversarial examples that fool most machine learning models.

We describe several common adversarial attack methods as below. Note that some of them can be applied to both White-box and Black-box manners.

L-BFGS. Szegedy et al. [109] used a method name L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno) to create targeted adversarial examples. This method minimize the weighted sum of perturbation size ε and loss function $L(x^*, y_{target})$ while constraining the elements of x^* to be normal pixel value.

FGSM. Goodfellow et al. [31] assumed that adversarial examples can be caused by cumulative effects of high dimensional model weights. They proposed a simple attack method but very effective, called Fast Gradient Sign Method (FGSM):

$$x^* = x + \varepsilon \cdot sign(\nabla_x L(x, y)) \tag{2.1}$$

where ε denotes the perturbation size for crafting adversarial example x^* from original input x. Given a clean image x, this method tries to create a similar image x^* in L^∞ neighborhood of x that fools the target classifier. This leads to maximize loss function $L(x, y)$ which is the cost of classifying image x as the target label y. The fast gradient sign method solves this problem by performing a one-step gradient update from x in the input space with a small size of perturbation ε. Increasing ε will lead to a higher and faster attack success rate however it may also make your adversarial sample to be more different from the original input. FGSM computes the gradients

for once, so it is much more efficient than L-BFGS. This method is very simple but it is fast and powerful for creating the adversarial examples.

PGD. Madry et al. [69] proposed Projected Gradient Descent (PGD) attack:

$$x^* = x + \delta \cdot (\bigtriangledown L\left(x, y\right)) \text{ respect to } \text{project}_{(x,\epsilon)}(x^*) \tag{2.2}$$

Where $\text{project}_{(x,\epsilon)}(x^*)$ defines a projection operator with parameter x^* on the circle area around x with radius ϵ, δ is a clip value that is searched in a box (x, ϵ). This method based on another method named Basic Iterative Method (BIM) [52] that is extended from the FGSM by applying it multiple times with a small step size and clipping values of intermediate results after each step to ensure that they are in an ϵ-neighborhood of the original input. The adversarial examples and their counterparts are defined as indistinguishable by humans. Because it is hard to model human perception, researchers use three popular distance metrics to approximate human's perception based on the L^p norm:

$$||x||_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}} \tag{2.3}$$

C&W. Carnili et al. [14] proposed a very powerful attack method by using L^0, L^2, L^∞ metrics to control adversarial perturbation. L^0 helps to calculate the number of pixels with different values at corresponding positions in two images. It indicates how many pixels between the two images are modified. L^2 is applied for measuring the Euclidean distance between two images. And L^∞ will help to measure the maximum difference for all pixels at corresponding positions in two images. So far it is unclear which one is the best distance metric for crafting adversarial examples because it depends on the proposed algorithms.

EAD:Elastic-Net Attack. EAD adversarial attack was invented by Pin-Yu Chen et al. [16] inspired from [14]. This paper uses elastic-net regularization technique [124] that is widely used in solving high-dimensional feature selection problems to invent new attack method by extending from C&W method.

Deep Fool. Moosavi-Dezfooli et al. [81] proposed an attack method to compute a minimal norm adversarial perturbation for a given image in an iterative manner. Their algorithm, i.e. DeepFool initializes with the clean image that is assumed to reside in a region confined by the decision boundaries of the classifier. This region decides the class-label of the image. At each iteration, the algorithm perturbs the image by a small vector that is computed to take the resulting image to the boundary of the polyhedron that is obtained by linearizing the boundaries of the region within which the image resides. The perturbations added to the image in each iteration are accumulated to compute the final perturbation once the perturbed image changes its label according to the original decision boundaries of the networks. The authors show that the DeepFool algorithm is able to compute perturbations that are smaller than the perturbations computed by FGSM [31] in terms of their norm while having similar fooling ratios.

One-pixel Attack. [106] proposed a new method when only one pixel in the image is changed to fool the classifier. The method successfully deceives three different network models on 70.97% of the tested images by changing just on a pixel per image. They also reported that the average confidence of the networks on the wrong labels was found to be 97.47%. For a clean image, they first created a set of 400 vectors in R^5 such that each vector contained xy-coordinates and RGB values for an arbitrary candidate pixel. They, they randomly modified the elements of the vectors to create children such that a child competes with its parent for fitness in the next iteration, while the probabilistic predicted label of the network is used as the fitness criterion. The last surviving feature is used to alter the pixel in the image.

Universal Attack. Whereas the methods like FGSM [31], ILCM [52], DeepFool [81] etc. compute perturbations to fool a network on a single image, the 'universal' adversarial perturbations computed by Moosavi-Dezfooli et al. [80] are able to fool a network on 'any' image with high probability. These image gnostic perturbations also remain quasi-imperceptible for the human vision system. The authors computed the universal perturbations by restricting their l_2 norm as well as l_∞ norm and showed that the perturbations with their norms already achieved significant fooling ratios of

around 0.8 or more for state-of-the-art image classifiers. Their iterative approach to compute a perturbation is related to the DeepFool [81] strategy of gradually pushing a data point to the decision boundary for its class. However, in that case, 'all' the training data points are sequentially pushed to the respective decision boundaries and the perturbations computed over all the images are gradually accumulated by back-projecting the accumulator to the desired l_p ball of radius ϵ every time.

Spatially Transformed Attack. Traditional adversarial attack algorithms directly modify the pixel value of an image, which changes the image's color intensity. Xiao et al. [118] proposed a new method called Spatially Transformed Attack. They perturb the image by doing slight spatial transformation: they translate, rotate and distort the local image features slightly. The perturbation is small enough to evade human inspection but it can fool the classifiers.

Jacobian-Based Saliency Map Attack. Papernot et al. [85] proposed an attack method based on Jacobian call Jacobian-Based Saliency Map Attack. That method calculates the Jacobian matrix of the score function F. It can be viewed as a greedy attack algorithm by iterative manipulating the pixel which is the most influential to the model output.

2.1.3 Poisoning Attack vs Evasion Attack

Third, the adversarial attacks can be classified by the adversary's goal. Adversaries with no access to the pre-processed data must instead poison the model's training data before its pre-processing. For instance, Perdisci et al. [82] prevented Polygraph, a worm signature generation tool, from learning meaningful signatures by inserting perturbations in worm traffic flows [88]. Polygraph combines a flow tokenizer together with a classifier that determines whether a flow should be in the signature. Polymorphic worms are crafted with noisy traffic flows such that (1) their tokenized representations will share tokens, not representative of the worm's traffic flow, and (2) they modify the classifier's threshold for using a signature to flag worms. This attack forces Polygraph to generate signatures with tokens that do not correspond to invariants of the worm's behavior. Later, Xiao et al. [119] adapted the gradient

ascent strategy introduced in feature selection algorithms like LASSO.

- **Poisoning Attacks.** The attacking algorithms [11, 48, 98] that allow an attacker to insert/modify several fake samples into the training dataset of a deep learning system. These fake samples can cause failures of the trained classifier. They can result in poor accuracy or wrong prediction [125] on some given test samples. This type of attacks frequently appears in the situation where the adversary has access to the training database. For example, web-based repositories and "honeypots" [105] often collect malware examples [35] for training, which provides an opportunity for adversaries to poison the data.

- **Evasion Attacks.** The classifiers are fixed and usually have a good performance on benign testing samples [10]. The adversaries do not have the authority to change the classifier or its parameters, but they craft some fake samples that the classifier cannot recognize. In other words, the adversaries generate some fraudulent examples to evade detection by the classifier. For example, in [24], in autonomous driving vehicles, sticking a few pieces of tapes on the stop signs can confuse the vehicles's road sign recognizer.

2.1.4 Physical Attacks

In the case of an attack in the physical world, the adversary does not have direct access to the digital representation provided to the model. Instead, the model is fed input obtained by sensors such as a camera or microphone. The adversary is able to place objects in the physical environment seen by the camera or produce sounds heard by the microphone. The exact digital representation obtained by the sensors will change based on factors like the camera angle, the distance to the microphone, ambient light or sound in the environment, etc. This means the attacker has less precise control over the input provided to the machine learning model.

Attack via Camera Phone. Kurakin et al. [52] first demonstrated that threats of adversarial attacks also exist in the physical world. To illustrate this (see Fig. 2-1), they printed adversarial images and took snapshots from a cell-phone camera. These

images were fed to the Tensor-Flow Camera Demo app that uses Google's Inception model [108] for object classification. It was shown that a large fraction of images were misclassified even when perceived through the camera. This work studies FGSM [31] and ILCM [52] attack methods for attacks in the physical world.

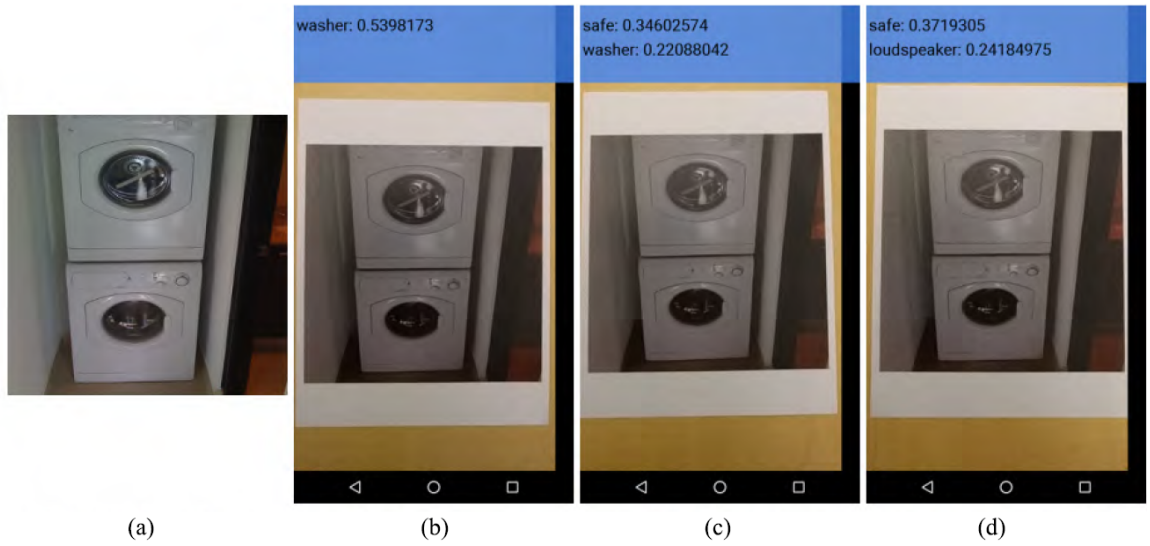


Figure 2-1: Example of the adversarial attack on mobile phone cameras: A clean image (a) was taken and used to generate different adversarial images. The images were printed and the TensorFlow Camera Demo app was used to classify them. A clean image (b) is recognized correctly as a "washer" when perceived through the camera, whereas adversarial images (c) and (d) are misclassified. The images also show network confidence in the range [0,1] for each image.

Attack to Road Sign. Eykholt Kevin et al. [24] built on the attacks proposed in [14] and [64] to design robust perturbations for the physical world. They demonstrated the possibility of attacks that are robust to physical conditions, such as variation in view angles, distance, and resolution. The proposed algorithm, termed RP_2 for Robust Physical Perturbations, was used to generate adversarial examples for road sign recognition systems that achieved high fooling ratios in practical drive-by settings. Two attack classes were introduced in this work for the physical road signs, (a) poster-printing: where the attacker prints a perturbed road sign poster and places it over the real sign (see Fig. 2-2), (b) sticker perturbation: where the printing is done on a paper and the paper is stuck over the real sign. For (b) two types of perturbations were studied, (b1) subtle perturbations: that occupied the entire

sign and (b2) camouflage perturbations: that took the form of graffiti sticker on the sign. As such, all these perturbations require access to a color printer and no other special hardware. Successful generation of perturbations for both (a) and (b) such that the perturbations remained robust to natural variations in the physical world demonstrate the threat of adversarial example in the real world.

Figure 2-2: Example of road sign attack [24]: The success rate of fooling LISA-CNN classifier on all the shown images is 100%. The distance and angle to the camera are also shown. The classifier is trained using LISA dataset for road signs [78].

2.1.5 Classification-based Attacks

Adversarial attacks are also categorized by the targeted classification/recognition tasks.

Attack on Recurrent neural networks

Papernot et al. [86] successfully generated adversarial input sequences for Recurrent Neural Networks (RNNs). RNNs are deep learning models that are particularly suitable for learning mappings between sequential inputs and outputs. Papernot et al. demonstrated that the algorithms proposed to compute adversarial examples for the feed-forward neural networks can also be adapted for fooling RNNs. In particular, the authors demonstrated successful fooling of the popular Long short-term memory (LSTM) [40] RNN architecture. It is concluded that the cyclic neural network model like RNNs are also not immune to the adversarial perturbations that were originally uncovered in the context of acyclic neural networks.

Attack on Deep Reinforcement Learning

Lin et al. [63] proposed two different adversarial attacks for the agents trained by deep reinforcement learning [77]. In the first attack, called 'strategically-timed attack', the adversary minimizes the reward of the agent by attacking it at a small subset of time steps in an episode. A method is proposed to determine when an adversarial example should be crafted and applied, which enables the attack to go undetected. In the second attack, referred as 'enchanting attack', the adversary lures the agent to a designated target state by integrating a generative model and a planning algorithm. The generative model is used for predicting the future states of the agent, whereas the planning algorithm generates the actions for alluring it.

Attack on Autoencoders and generative models

Tabacof et al. [110] proposed an adversarial attack for autoencoders [89], that misleads the autoencoder to reconstruct a completely different image. Their approach attacks the internal representation of a neural network such that the representation for the adversarial image becomes similar to that of the target image. However, [110] reported that autoencoders seem to be much more robust to adversarial attacks than the typical classifier networks. Kos et al. [50] also explored methods for computing adversarial examples for deep generative models, e.g. variational autoencoder (VAE) and VAE-Generative Adversarial Networks (VAE-GANs). GANs, such as [123] are becoming exceedingly popular nowadays in Computer Vision applications due to their ability to learn data distributions and generate realistic images using those distributions. The authors introduced three different classes of attacks for VAE and VAE-GANs. Owing to the success of these attacks it is concluded that the deep generative models are also vulnerable to adversaries that can convince them to turn inputs into very different outputs. This work adds further support to the hypothesis that 'adversarial examples' are a general phenomenon for current neural network architectures'.

Attack on Semantic Segmentation and Object Detection

Semantic image segmentation and object detection are among the mainstream problems in Computer Vision. Inspired by Moosavi-Dezfooli et al. [80] and Metzen et al. [73] showed the existence of image-agnostic quasi-imperceptible perturbations that can fool a deep neural network into significantly corrupting the predicted segmentation of the images. Moreover, they also showed that it is possible to compute noise vectors that can remove a specific class from the segmented classes while keeping most of the image segmentation unchanged (e.g removing pedestrians from road scenes). Although it is argued that the 'space of the adversarial perturbations for the semantic image segmentation is presumably smaller than image classification', the perturbations have been shown to generalize well for unseen validation images with high probability. Arnab et al. [5] also evaluated FGSM [31] based adversarial attacks for semantic segmentation and noted that many observations about these attacks for classification do not directly transfer to segmentation task.

2.1.6 Miscellaneous Attacks on other Domain

The adversarial attacks discussed above are either the popular ones in the recent literature or they are representative of the research directions that are fast becoming popular.

Graph Adversarial Examples

Adversarial examples also exist in graph-structured data [125, 18]. Attackers usually slightly modify the graph structure and node features, in an effort to cause the graph neural networks [79, 47] to give the wrong prediction for node classification or graph classification tasks. These adversarial attacks, therefore, raise concerns on the security of applying graph neural networks. For example, a bank needs to build a reliable credit evaluation system where their model should not be easily attacked by malicious manipulations.

Adversarial Examples in Audio

The work in [15] studies how to attack state-of-the-art speech-to-text transcription networks, such as DeepSpeech [37]. In their settings, when given any speech waveform x, they can add an inaudible sound perturbation θ that makes the synthesized speech $x + \theta$ be recognized as any targeted desired phrase. In their attacking work, they limited the maximum decibels (dB) at any time of the added perturbation noise, so that the audio distortion is unnoticeable.

Adversarial Examples in Video

Most works concentrate on attacking static image classification models. However, success on image attacks cannot guarantee that there exist adversarial examples on videos and video classification systems. There are some works [104, 112] aim to deceive object detection systems such as YOLOv2 [90], Faster R-CNN [91].The work [112] generates adversarial patches to targets with lots of intra-class variety, namely persons. The goal is to generate a patch that is able to successfully hide a person from the state-of-the-art object detector YOLOv2 [90].

Adversarial Examples in Text

Text classification is one of the main tasks in natural language processing. In text classification, the model is devised to understand a sentence and correctly label the sentence. For example, text classification models can be applied to the IMDB dataset for characterizing user' s opinion (positive or negative) on the movies, based on the provided reviews. Recent works of adversarial attacks have demonstrated that text classifiers are easily misguided by slightly modifying the text's spelling, words or structures. There are many researches work on the adversarial example in text [76, 23, 27, 59]. The work [76] considers to add perturbation on the word embedding [74], so as to fool a LSTM [40] classifier. However, this attack only considers perturbing the word embedding, instead of the original input sentence. The work HotFlip [23] considers replacing a letter in a sentence in order to mislead a character level text classifier (each

letter is encoded to a vector). For example, changing a single letter in a sentence alters the model's prediction on its topic. The attack algorithm manages to achieve this by finding the most influential letter replacement via gradient information. These adversarial perturbations can be noticed by human readers, but they don't change the content of the text as a whole, nor do they affect human judgments. The work in [60] considers manipulating the victim sentence on word and phrase levels. They try adding, removing or modifying the words and phrases in the sentences.

Adversarial Examples on Face recognition

Face attributes are among the emerging soft biometrics for modern security systems. Although face attribute recognition can also be categorized as a classification problem, we separately review some interesting attacks in this direction because face recognition itself is treated as a mainstream problem in Computer Vision. Rozsa et al. [95, 94] explored the stability of multiple deep learning approaches using the CelebA benchmark [65] by generating adversarial examples to alter the results of facial attribute recognition. By attacking the deep network classifiers with their so-called 'Fast Flipping Attribute' technique, they found that robustness of deep neural networks against the adversarial attacks varies highly between facial attributes. It is claimed that adversarial attacks are very effective in changing the label of a target attribute to a correlated attribute. Mirjalili and Ross [75] proposed a technique that modifies a face image such that its gender (for a gender classifier) is modified, whereas its biometric utility for a face matching system remains intact. Similarly, Shen et al. [101] proposed two different techniques to generate adversarial examples for faces that can have high 'attractiveness scores' but low 'subjective scores' for the face attractiveness evaluation using deep neural network. We refer to [100] for further attacks related to the task of face recognition.

Adversarial Examples on Malware Detection

The existence of adversarial examples in safety-critical tasks, such as malware detection, should be paid much attention. The work [34] built a DNN model on the

DREBIN dataset by [51], which contains 120,000 Android application samples, where over 5,000 are malware samples. The trained model has 97% accuracy, but malware samples can evade the classifier if attackers add fake features to them. Another work [4] considers using GANs to generate adversarial malware.

Adversarial Examples on Fingerprint Recognition

Fingerprint recognition systems are also one of the most safety-critical fields where machine learning models are adopted. While there are adversarial attacks undermining the reliability of these models. For example, fingerprint spoof attacks copy an authorized person's fingerprint and replicate it on some special materials such as liquid latex or gelatin. Traditional fingerprint recognition techniques especially minutiae-based models fail to distinguish the fingerprint images generated from different materials. The works [49, 17] design a modified CNN to effectively detect the fingerprint spoof attack.

2.1.7 Summary on Adversarial Attacks

In summary, the proliferation of AI in many fields means an increase in many security risks for those AI systems themselves. Adversarial attacks have been and will continue to receive a lot of attention both in research and reality. The adversarial attacks methods are proposed in various fields such as image classification [14], object detection [80], speech recognition [15], face recognition [101], malware [4], fingerprint recognition [49, 17], and text classification [23]. It is obvious that adversarial attacks are a very challenging issue because the characteristics of it are difficult to detect by humans but it is possible to deceive the latest AI system. The adversarial attacks will certainly continue to be studied more in the future.

2.2 Defense Strategies to Adversarial Examples

2.2.1 Adversarial Training

Since the discovery of adversarial examples for the deep neural networks [109], there has been a general consensus in the related literature that the robustness of neural networks against these examples improves with adversarial training. Therefore, most of the contributions introducing new adversarial attacks, e.g. [109, 31, 81] simultaneously propose adversarial training as the first line of defense against those attacks. Although adversarial training improves the robustness of a network, it is a non-adaptive strategy that requires training to be performed using strong attacks and the architecture of the network is sufficiently expressive. Since adversarial training necessitates increased training data size, we refer to it as a "brute-force" strategy. It is also commonly observed in the literature that brute force adversarial training results in regularizing the network (e.g. see [31, 97]) to reduce overfitting, which in turn improves the robustness of the networks against the adversarial attacks. Inspired by this observation, Miyato et al. [76] proposed a "Virtual Adversarial Training" approach to smooth the output distributions of the neural networks. A related "stability training" method is also proposed by Zheng et al. [122] to improve the robustness of neural networks against small distortions to input images. It is noteworthy that whereas adversarial training is known to improve the robustness of neural networks, Moosavi-Dezfooli et al. [80] showed that effective adversarial examples can again be computed for already adversarially trained networks.

2.2.2 Gradient Masking

Ross and Doshi-Velez [93] studied input gradient regularization [21] as a method for adversarial robustness. Their method trains differentiable models (e.g. deep neural networks) while penalizing the degree of variation resulting in the output with respect to change in the input. Implying, a small adversarial perturbation becomes unlikely to change the output of the trained model drastically. It is shown that this method,

when combined with brute-force adversarial training, can result in very good robustness against attacks like FGSM [31] and JSMA [85]. However, each of these methods almost doubles the training complexity of a network, which is already prohibitive in many cases. Previously, Lyu et al. [68] also used the notion of penalizing the gradient of loss function of network models with respect to the inputs to incorporate robustness in the networks against L-BFGS [109] and FGSM [31] based attacks. Similarly, Shaham et al. [99] attempted to improve the local stability of neural networks by minimizing the loss of a model over adversarial examples at each parameter update. They minimized the loss of their model over worst-case adversarial examples instead of the original data. In related work, Nguyen and Sinha [83] introduced a masking based defense against C&W attack [14] by adding noise to the logit outputs of networks.

2.2.3 Image Compression

Dziugaite et al. [22] noted that most of the popular image classification datasets comprise JPG images. Motivated by this observation, they studied the effects of JPG compression on the perturbations computed by FGSM [31]. It is reported that JPG compression can actually reverse the drop in classification accuracy to a large extent for the FGSM perturbations. Nevertheless, it is concluded that compression alone is far from an effective defense. JPEG compression was also studied by Guo et al. [36] for mitigating the effectiveness of adversarial images. Moreover, Shin and Song [102] have demonstrated the existence of adversarial examples that can survive JPEG compression. Compression under Discrete Cosine Transform (DCT) was also found inadequate as a defense against the universal perturbations [80] in a previous work [2]. One major limitation of compression based defense is that larger compressions also result in loss of classification accuracy on clean images, whereas smaller compressions often do not adequately remove the adversarial perturbations. In another related approach, Bhagoji et al. [9] proposed to compress input data using Principal Component Analysis for adversarial robustness.

2.2.4 Image Denoising

There are some previous research works that tried to remove adversarial noises and regained the right recognition of classifiers. Liao et al. [62] proposed a method named High-level representation Guided Denoiser (HGD) as a defense for image classification systems. This paper argued that many defense models can not remove all adversarial perturbations so non-removed adversarial noises will are amplified to enormities in the top layers of the target model and this will lead to a wrong prediction. To thoroughly overcome this problem, they proposed a system in which the denoiser is trained by a high-level representation guided denoiser (HGD) loss function. However, this paper only implemented on ImageNet includes color images and not applied to a gray-scale dataset such as MNIST. This is not very important but it may not work well on gray images when the method based on the high-level representation in a very deep neural network meanwhile a simple neural network can be used and works well for MNIST. Xu et al. [120] used a strategy to reduce the degress of freedom available to an adversary by squeezing out unnecessary input features that they call “Feature Squeezing". The main idea in this paper is that they use two different denoisers to input for squeezing unnecessary features, afterward, they compare the prediction results from the target model and decide that input is adversarial or legitimate. They applied two denoising methods: (1) squeezing color bit depth and (2) spatial smoothing method. They evaluated their proposed method on variety of adversarial attacks methods, however, there is unclear how they decided variety specified thresholds on different benchmark datasets so those settings will make operators be exhausted when they have to decide the thresholds for their systems and it is obviously ineffective when this system has to cope with new and unknown dataset.

2.2.5 Image Transformation

There are several proposed defense methods based on image transformation. Lu et al. [67] proposed the method to defeat adversarial examples based on changing distance scaling. In this work, the authors evaluated the recognition capabilities of

the AI system when a distance from a road sign to a camera is 0.5 meters and 1.5 meters. This method is regarded as a scale-based method since the authors tried to test the AI system' s recognition capabilities with a range of sizes of different object sizes in images. Athalye et al. [7] proposed a new method to craft adversarial examples on a variety of synthesized images by applying distortion and affine transformation. This paper proposed the attack method, not a defense in an adversarial manner. Flowchart of this proposed method can be described as original images -> modified images (by distortion and/or affine transformation) -> create adversarial examples from modified images. Tian et al. [113] used rotation techniques to train detector neural networks for MNIST and CIFAR-10 datasets. They trained four detector networks on different training sets. This research belongs to a training classifier-based method where authors use both benign and adversarial images to train a classifier. By using this strategy, a classifier that was trained with benign and adversarial examples learned more important features and discriminated features than other networks that were trained only with benign images. After training, the author re-test detector networks by using rotated benign and adversarial images to evaluate the AI system's recognition capabilities. However, this paper only evaluated their method on small-scale datasets such as MNIST and CIFAR-10 without a large-scaled dataset like ImageNet.

2.2.6 Miscellaneous countermeasures

Liang et al. [61] treated perturbations to images as noise and used scalar quantization and spatial smoothing filter to separately detect such perturbations. In a related approach, Feinman et al. [25] proposed to detect adversarial perturbations by harnessing uncertainty estimates (of dropout neural networks) and performing density estimation in the feature space of neural networks. Eventually, separate binary classifiers are trained as adversarial example detectors using the proposed features. Gebhart and Schrater [28] viewed neural network computation as information flow in graphs and proposed a method to detect adversarial perturbations by applying persistent homology to the induced graphs.

Detector subnetwork

Metzen et al. [72] proposed to augment a targeted network with a subnetwork that is trained for a binary classification task of detecting adversarial perturbations in inputs. It is shown that appending such a network to the internal layers of a model and using adversarial training can help in detecting perturbations generated using FGSM [31], BIM [52] and DeepFool [81] methods.

Detection method based on activation function

Lu et al. [66] hypothesized that adversarial examples produce different patterns of ReLU activations in (the late stages of) networks than what is produced by clean images. Based on this hypothesis, they proposed to append a Radial Basis Function SVM classifier to the targeted models such that the SVM uses discrete codes computed by the late-stage ReLUs of the network. To detect perturbation in a test image, its code is compared against those of training samples using the SVM. Effective detection of adversarial examples generated by [31, 52, 81] is demonstrated by their framework, named SafetyNet.

Data augmentation

Grosse et al. [33] proposed to augment the potentially targeted neural network model with an additional class in which the model is trained to classify all the adversarial examples. Hosseini et al. [42] also employed a similar strategy to detect black-box attacks.

Defense against universal perturbations

Akhtar et al. [2] proposed a defense framework against the adversarial attacks generated using universal perturbations [80]. The framework appends extra 'pre-input' layers to the targeted network and trains them to rectify a perturbed image so that the classifier's prediction becomes the same as its prediction on the clean version of the same image. The pre-input layers are termed Perturbation Rectifying Network

(PRN), and they are trained without updating the parameters of the targeted network. A separate detector is trained by extracting features from the input-output differences of PRN for the training images. A test image is first passed through the PRN and then its features are used to detect perturbations. If adversarial perturbations are detected, the output of PRN is used to classify the test image.

Defense based on Generative Adversarial Networks

Lee et al. [58] used the popular framework of Generative Adversarial Networks [30] to train a network that is robust to FGSM like attacks. The authors proposed to directly train the network along with a generator network that attempts to generate perturbation for that network. During its training, the classifier keeps trying to correctly classify both the clean and perturbed images. We categorize this technique as an 'add-on' approach because the authors propose to always train any network in this fashion. In another GAN-based defense, Shen et al. [45] use the generator part of the network to rectify a perturbed image.

MagNet method

Meng and Chen [71] proposed a framework that uses one or more external detectors to classify an input image as adversarial or clean. During training, the framework aims at learning the manifold of clean images. In the testing phase, the images that are found far from the manifold are treated as adversarial and are rejected. The images that are close to the manifold (but not exactly on it) are always reformed to lie on the manifold and the classifier is fed with the reformed images. The notion of attracting nearby images to the manifold of clean images and dropping the far-off images also inspires the name of the framework, i.e. MagNet.

2.2.7 Summary on Adversarial Defense

Since adversarial perturbations generated by many methods look like high-frequency noise to a human observer1 multiple authors have suggested using image preprocess-

ing and denoising as a potential defense against adversarial examples. There is a large variation in the proposed preprocessing techniques, like doing JPEG compression [22] or applying median filtering and reducing the precision of input data [120]. While such defenses may work well against certain attacks, defenses in this category have been shown to fail in the white box case, where the attacker is aware of the defense [38]. Many defenses, intentionally or unintentionally, fall into a category called "gradient masking". Most white-box attacks operate by computing gradients of the model and thus fail if it is impossible to compute useful gradients. Gradient masking consists of making the gradient useless, either by changing the model in some way that makes it non-differentiable or makes it have zero gradients in most places or make the gradients point away from the decision boundary. Essentially, gradient masking means breaking the optimizer without actually moving the class decision boundaries substantially. Because the class decision boundaries are more or less the same, defenses based on gradient masking are highly vulnerable to black-box transfer [84]. Some defense strategies (like replacing smooth sigmoid units with hard threshold units) are intentionally designed to perform gradient masking. Many defenses are based on detecting adversarial examples and refusing to classify the input if there are signs of tampering [72]. This approach works long as the attacker is unaware of the detector or the attack is not strong enough. Otherwise, the attacker can construct an attack that simultaneously fools the detector into thinking an adversarial input is a legitimate input and fools the classifier into making the wrong classification [13]. The most popular defense in current research papers is probably adversarial training [31, 109, 81]. The idea is to inject adversarial examples into the training process and train the model either on adversarial examples or on a mix of clean and adversarial examples. The approach was successfully applied to large datasets [53], and can be made more effective by using discrete vector code representations rather than real number representations of the input [12]. One key drawback of adversarial training is that it tends to overfit to the specific attack used at training time. This has been overcome, at least on small datasets, by adding noise prior to starting the optimizer for the attack [70]. Another key drawback of adversarial training is that it tends to

inadvertently learn to do gradient masking rather than to actually move the decision boundary. This can be largely overcome by training on adversarial examples drawn from an ensemble of several models [114]. A remaining key drawback of adversarial training is that it tends to overfit to specific constraint region used to generate the adversarial examples (models trained to resist adversarial examples in a max-norm ball may not resist adversarial examples based on large modifications to background pixels [29] even if the new adversarial examples do not appear particularly challenging to a human observer).

Chapter 3

Proposed Automatic Detection System for Adversarial Examples

3.1 Image Transformation for Detecting Adversarial Examples

In this section, we introduce some common methods for creating adversarial examples and defense approaches. We mainly focus on adversarial examples for image classification. We have to note that adversarial examples are effective to other tasks such as face recognition, one of image classification [32], natural language processing [27] and malware detection [35]. However those tasks are out of our research scope. In this work, we explored the robustness of deep neural networks through the very simple techniques but very effective coordinate transformation. Firstly, we craft the adversarial examples by using the PGD algorithm [69] on the original training images. Afterward, we apply our proposed method to those adversarial examples and evaluate our method through by observing the recognition performances. The results show that our method is very effective for making deep neural networks more robust against adversarial examples.

Our contributions. Our research shows the following contributions:

- We investigated and analyzed attack approaches for crafting adversarial exam-

ples. We showed attack approaches along their different strategies to provide an intuitive overview of these attack methods.

- We investigated the modern defense approaches and their variants in adversarial settings. Base on our investigation, we showed that the defense of a deep neural network is still challenging and is not yet completely satisfactory. It remains a rapidly evolving research area and that is also our motivation for this research.

- Our proposed approach is successfully applied two types of common datasets that are a small-scale dataset (MNIST) and a large-scale dataset (ImageNet). Our defense method worked well to defeat adversarial examples and in some cases it recovered the deep learning classifier's performance with high accuracy rates.

3.1.1 Crafting Adversarial Examples Phase

We consider the white-box targeted attack settings, where the attacker can fully access into the model type, model architecture, all trainable parameters and the adversary aims to change the classifier's prediction to some specific target class. The attackers use available information to identify the feature space where the model is vulnerable or try to find the victim decision boundaries. Then the victim model is exploited by altering a clean input by using adversarial example methods. To create adversarial samples that are misclassified by machine learning model, an adversary with knowledge of the model's classifier f and its trainable parameters. In this work, we use Projected Gradient Descent [70] method for crafting adversarial examples. We define classifier function $f : \mathbb{R}^n \rightarrow \Big[1...k\Big]$ that maps image pixel value vectors to a particular label. Then we assume that function f has a loss function $L : \mathbb{R}^n \times \Big[1...k\Big] \rightarrow \mathbb{R}$. For an input image $x \in \mathbb{R}^n$ and target label $y \in \Big[1...k\Big]$, our system aims to solve the following optimization problem: $\delta + L(x, y)$ subject to $x + \delta \in \Big[0, 1\Big]^n$, where δ is a perturbation noise that we add to the original image x. Whenever finding the best adversarial candidate, we have to project adversarial to constrain area (x, ϵ) by using a projection operator $project_{(x,\epsilon)}()$. We have to note that this function method would

yield the solution for $f(x)$ in the case of convex losses, however the neural networks are non-convex so we end up with an approximation in this case. In this case, we use the output of the second-to-last layer logits for calculating the gradient instead of using the output of the Softmax. So our attack method is denoted as algorithm 1. In attacking phase, we set learning rate for crafting adversarial examples is 0.01 and iterative process is 500 times. From the clean images we will create the targeted output images.

Algorithm 1: Crafting Adversarial Examples Algorithm

input : x, y_{true}, y^*, f, ϵ, α
output : x^*
parameter: learning rate = 0.01, epochs = 500

$x \leftarrow x^*$ `// initial adversarial sample`
$\delta_x \leftarrow \vec{0}$ `// initial perturbation factor`
$iter \leftarrow 1$ `// initial iteration counter`
while $||\delta_x||_\infty < \epsilon$ *and* $f(x^*) \neq y^*$ *and* $iter <= epochs$ **do**
    $x^* \leftarrow x + \delta \cdot sign(\bigtriangledown L(y^*|x^*))$
    $x^* \leftarrow \text{project}_{(x,\epsilon)}(x^*)$
    maximize $L(y^*|x^*)$ respect to $||\delta_x||_\infty$
    $\delta \leftarrow clip(x^*, x - \epsilon, x + \epsilon)$
    $iter \leftarrow iter + 1$
end
return x^*

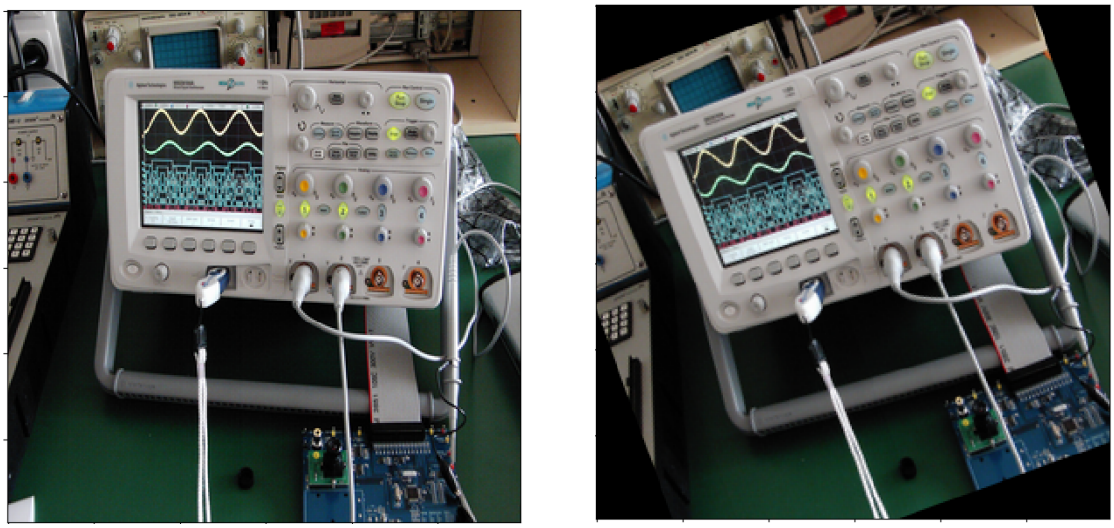
3.1.2 Coordinate Transformation

Affine transformations have been widely used in computer vision [44]. Now we define the range of defense that we want to apply into the adversarial examples for protecting a deep neural network. In this section, we describe the Coordinate Transformation manner that is one of an affine transformations, we find the parameter (i^*, j^*, α) that rotates the adversarial image with degree around the center (see Fig. 3-1) will make classifier remove the adversarial noises. Formally, the pixel at position (i, j) is rotated counterclockwise by multiplying a rotation matrix that has cos and sin

elements corresponding to the angle α:

$$\begin{bmatrix} i^* \\ j^* \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \cdot \begin{bmatrix} i \\ j \end{bmatrix} \tag{3.1}$$

So the vectors $\begin{bmatrix} i \\ j \end{bmatrix}$ and $\begin{bmatrix} i^* \\ j^* \end{bmatrix}$ have the same magnitude and they are separated by an given angle α. In our research, we set angle $\alpha \in \Big[1, 30\Big]$ because it is obviously when we increase angle degree too much the classification capabilities of machine learning system will be effected and decrease accuracy rate for original images. So the range $\Big[1, 30\Big]$ is suitable for defeating adversarial effectiveness and regain the accuracy on original images. Our experiments show that we can find the best angle that it defeats completely the adversarial noise and re-recognize the correct image.



(a) Original Oscilloscope Image

(b) Coordinate Transformation with 20 degree counterclockwise rotation

Figure 3-1: Coordinate Transformation with 20 degree counterclockwise rotation around the center

When we apply this technique on the adversarial examples that are generated, we observe that adversarial examples failed to misclassify to coordinate transformation. For more intuitive understanding, supposing we have original data (x, y) and model f. So our prediction label is y_{pred}. And the loss function $L(y, y_{pred})$ shows us how far

y_{pred} is away from y. When we create adversarial examples, the purpose is to increase the loss function $L(y, y_{pred})$ by adding a small adversarial noise to original input x. Recall the Projected Gradient Descent algorithm, equation (2.2) will be rewritten as the equation as bellow:

$$x^* = x + \delta \cdot \left(\frac{\partial L(y, y_{pred})}{\partial x} \right) \text{ respect to } \text{project}_{(x,\epsilon)}(x^*) \tag{3.2}$$

We aim to solve equation (3.2) by maximizing loss function $L(y, y_{pred})$ instead of $L(x, y)$. The logits (vector of raw prediction) is the output of the deep neural network before we feed them into the Softmax activation function for normalizing, it is described as:

$$\text{logits} = f(x) \tag{3.3}$$

$$y_{pred} = \text{softmax}(\text{logits}) \tag{3.4}$$

Softmax function takes an N-dimensional vector logits and transfers it into a vector of real number in range (0,1) which sum of them is equal to 1:

$$\text{softmax}(\text{logits}) = \frac{e^{\text{logits}}}{\sum e^{\text{logits}}} \tag{3.5}$$

And in the fundamental derivative rule, from a given $f(x) = \frac{g(x)}{h(x)}$, we have $f'(x) = \frac{g'(x)h(x) - h'(x)g(x)}{h(x)^2}$. So in equation (3.5), $g(x) = e^{\text{logits}} = e^{f(x)}$ and $h(x) = \sum e^{\text{logits}} = \sum e^{f(x)}$. In general, we need to calculate the partial derivate of softmax by pass it back to the previous layer during backpropagation and it will be defined in general form as:

$$\frac{\partial \text{softmax}(f_i(x)))}{\partial x_j} = \frac{\partial s_i}{\partial x_j} = \frac{\partial \frac{e^{x_i}}{\sum_{k=1}^{N} e^{x_k}}}{\partial x_j} \tag{3.6}$$

Where s shorts for softmax function. In this case, we have $g(x) = e^{x_i}$ and $h(x) =$

$\sum_{k=1}^{N} e^{x_k}$. It is obviously that derivative of exponential $g'(x) = e'^{x_i} = e^{x_i}$. Meanwhile, $h'(x) = e^{x_i}$ when $i = j$ and $h'(x) = 0$ when $i \neq j$.

Let consider case when $i = j$:

$$\frac{\partial \frac{e^{x_i}}{\sum_{k=1}^{N} e^{x_k}}}{\partial x_j} = \frac{e^{x_i} \sum_{k=1}^{N} e^{x_k} - e^{x_j} e^{x_i}}{\left(\sum_{k=1}^{N} e^{x_k}\right)^2} \tag{3.7}$$

$$= \frac{e^{x_i} \left(\sum_{k=1}^{N} e^{x_k} - e^{x_j}\right)}{\left(\sum_{k=1}^{N} e^{x_k}\right)^2} \tag{3.8}$$

$$= \frac{e^{x_i}}{\sum_{k=1}^{N} e^{x_k}} \frac{\sum_{k=1}^{N} e^{x_k} - e^{x_j}}{\sum_{k=1}^{N} e^{x_k}} \tag{3.9}$$

$$= s_i(1 - s_j) \tag{3.10}$$

For $i \neq j$, we have:

$$\frac{\partial \frac{e^{x_i}}{\sum_{k=1}^{N} e^{x_k}}}{\partial x_j} = \frac{0 - e^{x_j} e^{x_i}}{\left(\sum_{k=1}^{N} e^{x_k}\right)^2} \tag{3.11}$$

$$= \frac{-e^{x_j}}{\sum_{k=1}^{N} e^{x_k}} \times \frac{e^{x_i}}{\sum_{k=1}^{N} e^{x_k}} \tag{3.12}$$

$$= -s_j s_i \tag{3.13}$$

We will calculate loss between y and y_{pred} by using cross entropy function that indicates the distance between what the model believes the output distribution should be and what the original distribuation really is.

$$L(y, y_{pred}) = \text{CrossEntropy}(y, y_{pred}) \tag{3.14}$$

For two discrete probability distributions y and y_{pred}, the cross-entropy function is defined as:

$$\text{CrossEntropy}(y, y_{pred}) = -\sum_{i} y_i \log(y_{pred}) \tag{3.15}$$

By calculating partial derivative of function (3.14), we have:

$$\frac{\partial L(y, y_{pred})}{\partial x} = \frac{\partial L(y, y_{pred})}{\partial y_{pred}} \frac{\partial y_{pred}}{\partial x} \tag{3.16}$$

$$= \frac{\partial L(y, y_{pred})}{\partial f(x)} \frac{\partial f(x)}{\partial x} \tag{3.17}$$

$$= \frac{\partial(-\sum_i y_i \log(y_{pred}))}{\partial y_{pred}} \frac{\partial y_{pred}}{\partial x} \tag{3.18}$$

$$= -\sum_i y_i \frac{\partial \log(y_{pred})}{\partial y_{pred}} \frac{\partial y_{pred}}{\partial x} \tag{3.19}$$

$$= -\sum_i y_i \frac{1}{y_{pred}} \frac{\partial y_{pred}}{\partial x} \tag{3.20}$$

$$= -\sum_i y_i \frac{1}{y_{pred}} \frac{\partial \text{softmax}(f(x))}{\partial x} \tag{3.21}$$

From derivative of softmax equations (3.6), (3.7) and (3.11), we have:

$$\frac{\partial L(y, y_{pred})}{\partial x} = -y_i(1 - y_{pred(i)}) - \sum_{k \neq i} y_k \frac{1}{y_{pred(k)}}(-y_{pred(k)} . y_{pred(i)}) \tag{3.22}$$

$$= -y_i(1 - y_{pred(i)}) + \sum_{k \neq i} y_k . y_{pred(i)} \tag{3.23}$$

$$= -y_i + y_i y_{pred(i)} + \sum_{k \neq i} y_k . y_{pred(i)} \tag{3.24}$$

$$= -y_{pred(i)} \left(y_i + \sum_{k \neq i} y_k \right) - y_i \tag{3.25}$$

Because y is a one hot encoded vector for the true labels, so $\sum_k y_k = 1$ and $y_i + \sum_{k \neq i} y_k = 1$. So we have:

$$\frac{\partial L(y, y_{pred})}{\partial x} = (y_{pred} - y) \tag{3.26}$$

From equations (3.6) and (3.26), it is clear that $L(y, y_{pred})$ is influenced by product of trainable weights and activations when we calculate partial derivative of softmax by passing it back during backpropagation to find the best adversarial perturbation

δ . For examples, when we have two images with the same label, their activations in any fixed networks are similar and the weights of the network are unchanged. Consequently, $L(y, y_{pred})$ is a constant for any given image x with the same class. This means the gradient is highly correlated with true label y. Because of this property, when attacker added a adversarial noise, the classifying x^* becomes a simpler problem than the original problem of classifying x, as x^* contains extra information i.e the added noise. However, with a small change in input data by rotating adversarial images with a particular angle the system makes a different decision. And in this case, we show that deep neural network recognizes the rotated adversarial images as the true label instead of the targeted label.

3.1.3 Geometric Translation and Combination

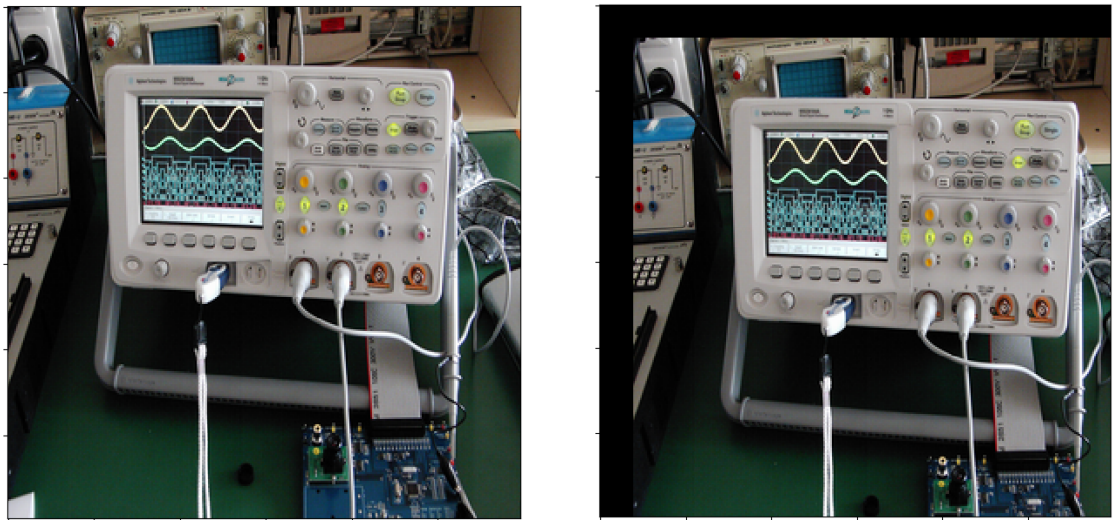
In this section, we describe about the Geometric Translation and combination method. Geometric Translation is one of method of Geometric Transformation that is interpreted as the addition of a constant vector to every point or shifting the origin of the coordinates. In our work, we applied Geometric Translation on the adversarial examples by shifting them from the top-left to bottom-right direction (see Fig. 3-2). We define the shift vector V in (v_i,v_j) direction:

$$V = \begin{bmatrix} 1 & 0 & v_i \\ 0 & 1 & v_j \end{bmatrix} \tag{3.27}$$

Before applying this transformation, we flatten adversarial example matrix to column vector, then our adversarial example x^* can be multiplied by this geometric translation matrix as:

$$V \cdot x^* = \begin{bmatrix} 1 & 0 & v_i \\ 0 & 1 & v_j \end{bmatrix} \cdot \begin{bmatrix} x_i \\ x_j \\ 1 \end{bmatrix} = \begin{bmatrix} v_i + x_i \\ v_j + x_j \end{bmatrix} = x^{**} \tag{3.28}$$

x^{**} is created by geometric translation and we feed it into the deep learning classifier. Moreover, geometric translation shifts whole image into a particular direction and this will leave blank spaces, we will fill in those blank spaces by black pixels and we call that is the black canvas. After applying the geometric translation into the adversarial examples, we use the target machine learning classifier to classify new images and observe the worst case and best case and comparing them with the classification rate on the clean inputs. Combination phase is that we will combine rotation (coordinate

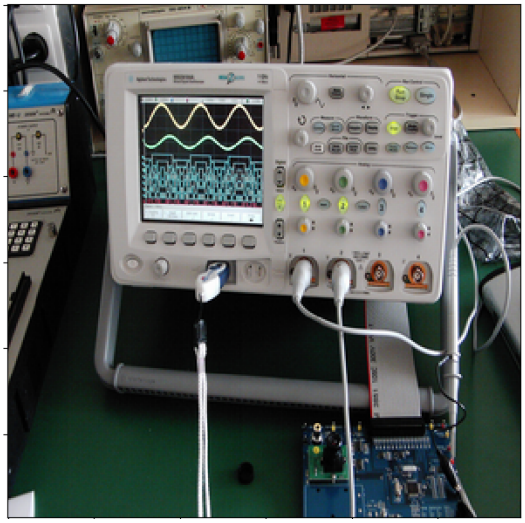


(a) Original Oscilloscope Image

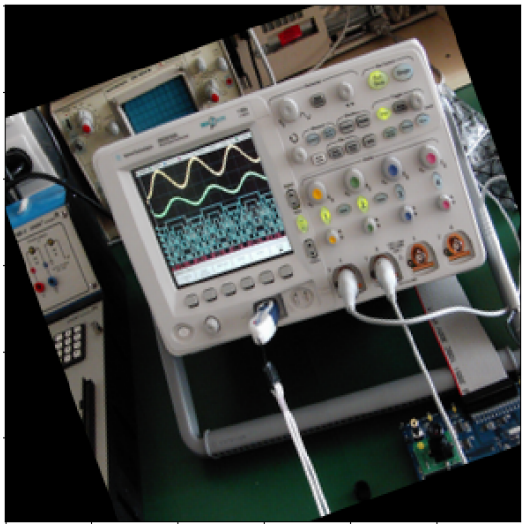
(b) Geometric Translation with 20 degree from top-left corner

Figure 3-2: Geometric Translation with 20 degree from top-lelf corner

transformation) and translation (geometric translation) into end-to-end system for defeating adversarial examples. Firstly, we applied the geometric translation to the adversarial examples with $(v_i, v_j) : \{0, 1, 2, 3\}$. By shifting the adversarial examples in very small distance, we expect to keep the key features of the images in the visible window and to avoid losing performance of our system. And by using the black canvas settings, we will ensure that there is no information of input is lost during our process. Secondly, we apply the use the coordinate transformation on the translated images and feed them into the machine learning classifier to verify our assumption. We keep the same coordinate transformation setting in this phase with $\alpha \in [0, 30]$ in the clockwise direction. The final step, we evaluate the target machine learning classifier's performance by using new combination patterns (see Fig. 3-3).



(a) Original Oscilloscope Image



(b) Combination of Coordinate Transformation and Geometric Translation with 20 degree

Figure 3-3: Combination of Coordinate Transformation and Geometric Translation.First, we shift image from top-left to bottom-right with 20 degree. Then, we rotate image with 20 degree counterclock wise from center of image

3.1.4 Proposed Defending Method

In this section, we describe about our proposed defending method for Deep Neural Networks. As we mentioned earlier, we define three main functions include TRNS, ROT and TR that stand for Geometric Translation, Coordination Transformation and Combination, respectively. We set a threshold equal to 0.9 for detecting adversarial examples. Our proposed defending method can be illustrated as Algorithm 2. Recall the loss function that we used for crafting adversarial examples in the previous section, which is used to measure the inconsistency between predicted label y^* and actual label y. It is a non-negative value, where the robustness of model increases along with the decrease of the value of loss function. And in our deep learning model, we use the Softmax classifier and cross-entropy, one of the popular loss function for evaluating softmax classifier. The Softmax classifier that has a probabilistic interpretation that we can get normalized class probabilities. Softmax function is defined as $f_i(x) = e^{x_i} / \sum_k e^{x_k}$; where f_i defines the i-th element of the vector of class function f, x_k defines the k-th element of input samples. It can be interpreted as the normalized

probability assigned to the correct label y_i given the image x_i. Exponentiating these quantities therefore gives the probabilities, and the division performs the normalization so that the probabilities sum to one. In the probabilistic interpretation, we are therefore minimizing the negative log likelihood of the correct class, which can be interpreted as performing Maximum Likelihood Estimation (MLE). A nice feature of this view is that we can now also interpret the regularization term $R(W)$ in the full loss function as coming from a Gaussian prior over the weight matrix W, where instead of MLE we are performing the Maximum a posteriori (MAP) estimation. The full details of this derivation are beyond the scope of this work so we mention these interpretations to make it more clear about Softmax classifier and the output of Softmax classifier is probabilities of all labels. In our proposed methods, we use Softmax function to compute the probabilities of primary input and modified input and compare them with threshold to make a decision which one is benign or not. In algorithm 2, we define *baseProbs* is probability of primary input x, after applying our methods, we get the modified input and measure its probability *prob*. If $|prob - baseProb| \geq 0.9$, it means probability of input dramatically drops so we can point out it is adversarial example, whereas it is benign input.

3.1.5 Experimental setup

Datasets. We consider the ImageNet dataset [96], which is a very large database designed for use in visual object recognition research and MNIST dataset [57], which is very common dataset for evaluating research works related to neural networks. The original ImageNet includes more than 14 millions images in 20,000 categories with a typical category, such as "computer mouse" or "vending machine", consisting of several hundred images. MNIST is a database of handwritten digits that is commonly used in various image processing systems. It includes 60,000 training images and 10,000 testing images that fit into a 28×28 pixel and grayscale levels.

Deep learning models. The machine learning models that we use are Google Inception V3 [108] and LeNet [55]. Google Inception V3 is a widely-used image classification model that has been shown to attain very high accuracy on the ImageNet

Algorithm 2: Proposed Defending Algorithm

input : x, y, $threshold$, ROT, $TRANS$, TR
output : 1: adversarial input, 0: clean input

$threshold \leftarrow 0.9$; $baseProb \leftarrow P(y_i|x_i) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$
if ROT **then**
    $startPoint \leftarrow 0$; $endPoint \leftarrow 30$
    for *i in range(startPoint,endPoint)* **do**
        $angle \leftarrow$ i * $\pi/180$
        $x_{ROT} \leftarrow ROT(x, angle)$
        $prob \leftarrow P(y_i|x_{ROT}) = \frac{e^{f_{y_i}}}{\sum_i e^{f_i}}$
    end
else if $TRNS$ **then**
    $minPoint \leftarrow 0$; $maxPoint \leftarrow 3$
    for *i in range(minPoint,maxPoint)* **do**
        for *j in range(minPoint,maxPoint)* **do**
            $v_i \leftarrow i$
            $v_j \leftarrow j$
            $x_{TRNS} \leftarrow TRNS(x, v_i, v_j)$
            $prob \leftarrow P(y_i|x_{TRNS}) = \frac{e^{f_{y_i}}}{\sum_i e^{f_i}}$
        end
    end
else
    $minPoint \leftarrow 0$; $maxPoint \leftarrow 3$
    for *i in range(minPoint,maxPoint)* **do**
        for *j in range(minPoint,maxPoint)* **do**
            $v_i \leftarrow i$
            $v_j \leftarrow j$
            $startPoint \leftarrow 0$; $endPoint \leftarrow 30$
            $x_{TRNS} \leftarrow TRNS(x, v_i, v_j)$
            for *q in range(startPoint,endPoint)* **do**
                $angle \leftarrow q$ * $\pi/180$
                $x_{TR} \leftarrow ROT(x_{TRNS}, angle)$
                $prob \leftarrow P(y_i|x_{TR}) = \frac{e^{f_{y_i}}}{\sum_i e^{f_i}}$
            end
        end
    end
end
if $|prob$ - $baseProb| \geq 0.9$ **then**
    return 1
else
    return 0
end

dataset. The model itself is made up of symmetric and asymmetric building blocks, including convolutional layers, average and max pooling, concats, dropouts and fully connected layers. Batchnorm is used extensively throughout the model and applied to activation inputs while loss is computed by Softmax layer. For this model, Google customized the ImageNet dataset that includes 1,331,167 images which are divided into training and evaluation sets containing 1,281,167 and 50,000 images, respectively. Its architecture that we use for crafting adversarial examples is as in table 3.1. We emphasize that we use the logits value in the last adjacency layer for compute the loss function $L(x, y)$.

Table 3.1: Google Inception Architecture [108]

type	patch size/stride or remarks	input size
conv	$3 \times 3/2$	$299 \times 299 \times 3$
conv	$3 \times 3/1$	$149 \times 149 \times 32$
conv padded	$3 \times 3/1$	$147 \times 147 \times 32$
pool	$3 \times 3/2$	$147 \times 147 \times 64$
conv	$3 \times 3/1$	$73 \times 73 \times 64$
conv	$3 \times 3/2$	$71 \times 71 \times 80$
conv	$3 \times 3/1$	$35 \times 35 \times 192$
$3 \times$ Inception	Inception filters	$35 \times 35 \times 288$
$5 \times$ Inception	Inception filters	$17 \times 17 \times 768$
$2 \times$ Inception	Inception filters	$8 \times 8 \times 1280$
pool	8×8	$8 \times 8 \times 2048$
linear	logits	$1 \times 1 \times 2048$
softmax	classifier	$1 \times 1 \times 1000$

LeNet [55] includes 7 deep layers not counting input layer that are convolutional layers and pooling operation. LeNet architecture that we use for crafting adversarial examples is as in table 3.2. This original model worked on the 32×32 input size, however we customized this model in order to accept to MNIST dataset of 28×28 input sizes.

Table 3.2: LeNet architecture [56]

type	kernel size/feature maps	input size
conv	$5 \times 5/6$	$32 \times 32 \times 1$
pool	$2 \times 2/6$	$28 \times 28 \times 6$
conv	$5 \times 5/16$	$14 \times 14 \times 6$
pool	$2 \times 2/16$	$10 \times 10 \times 16$
conv	$5 \times 5/120$	$5 \times 5 \times 16$
full connected	logits	$1 \times 1 \times 84$
softmax	classifier	$1 \times 1 \times 10$

3.1.6 Results

For evaluating our method, we consider 30 images from three categories such as Computer Mouse (C-Mouse), Computer Keyboard (Keyboard) and Vending Machine (V-Machine) in ImageNet with 10 images per each category. By considering both original images and adversarial images, the total number of samples is 60 images. Because of image copyright issues, for visualization we used three images of Keyboard, C-Mouse and V-Machine that were captured by ourself. Meanwhile, for MNIST dataset, we randomly select ten digit-0 images for crafting adversarial examples with targeted class is digit-4. The first, we use the PGD method for creating adversarial images in the white-box setting. Next, we apply the coordinate transformation, geometric translation and combination of these two transformations for those adversarial images and feed them into again the machine learning classifiers for classification. The results show that our proposed method is very effective for removing the adversarial noise and recovering the acceptable classification accuracy rate.

For attacking Google Inception model, we use 30 images from ImageNet and three our own images to craft the adversarial examples toward to ostrich label. After applying geometric translations, coordinate transformations and combination steps, we observe the effects of adversarial examples on the classifier and recognize that those adversarial perturbations are mostly removed and in some particular cases our method recover the performance of classifier with right answers. The geometric translation

settings are $v_i \in [0,3]$ and $v_j \in [0,3]$. That means we test 15 difference transformations for each image. In table 3.3 we only record the worst and the best case. In this table, the column "Primary" shows the classification's results on the original images (Clean) and adversarial images (Adv) before applying our method. The next two columns "Original class" and "Targeted class" show the results after applying our method. "Original class" column shows the worst-case and the best-case that classifier classify them as "Original" and the same meaning to "Targeted class" column. From this table, we illustrate that geometric translation mostly defeat adversarial effects when it makes confidence rate dramatically decrease. For example, in "Keyboard" case, the confidence recognition rate for "ostrich" drops from 99.9% to 5.07%.

Table 3.3: Geometric Translation results on our own images

Image	Primary		Original class		Targeted class	
	Clean	Adv	Min	Max	Min	Max
Keyboard	72.4%	99.9%	15.69%	62.01%	0.02%	5.07%
C-Mouse	86.1%	100%	3.01%	15.43%	0.1%	10.4%
V-Machine	77.9%	99.8%	3.74%	**66.71**%	0.18%	**17.2**%

The performance of the coordinate transformation is very impressive that is shown in table 3.4. We observe that deep learning classifier are correctly recognize the original class at the best-case is 92.67%, it is even better than when our method has not been applied and the maximum probability for targeted recognition is only 3.67%.

Table 3.4: Coordinate Transformation results on our own images

Image	Primary		Original class		Targeted class	
	Clean	Adv	Min	Max	Min	Max
Keyboard	72.4%	99.9%	28.27%	56.5%	0.01%	0.22%
C-Mouse	86.1%	100%	5.81%	**92.67**%	0.0%	2.16%
V-Machine	77.9%	99.8%	2.62%	55.04%	0.01%	**3.67**%

In combination setting, firstly, we perform the geometric translation on the adversarial images, after that we apply coordinate transformation in the range [0,30]. In geometric translation step, we also consider 15 cases for each image $v_i \in [0,3]$ and

$v_j \in [0, 3]$. It is obvious that it excludes $(v_i, v_j) = (0, 0)$. The result is shown in table 3.5. For combination setting, we observe that it works well on all settings and the worst record with "V-Machine" image is only 2.14%. In "Keyboard", the minimum of classification rate of targeted class all is 0.01%, and the worst case is 0.23%. Meanwhile, the best case for original label is very potential at 65.82%. For "C-Mouse", we observe that the best classification rate for original label reaches 93.86%, and classification accuracy rate for adversarial label is less than 1.83. The results confirm that the combination method gets the best effect in our experiment.

In table 3.6 summaries performance on three our own images by using TRNS, ROT and TR with Google Inception. We observe that combination method overwhelms and wins in most categories, there is only one exceptional case that TRNS gets the highest performance with clean vending machine image at 66.71%.

In table 3.7 shows the results on 60 images (includes 30 benign and 30 adversarial images) from the ImageNet dataset. We observe min, max, median and variance of probability for recognizing true label and adversarial label with original and adversarial inputs. In Geometric Transformation (TRNS), we analyze $60 \times 15 = 900$ different cases. When using TRNS, with original inputs, the median of recognition probabilities for true label that is gave by machine learning system reaches to 84.43%. When the inputs are adversarial, TRNS answered with mean of recognition probabilities for adversarial label is only 0.6% confidence. In Coordinate Transformation (ROT), we analyze $60 \times 30 = 1,800$ different cases. ROT method also performed perfectly on both original and adversarial samples. And in Combination (TR), we work with $60 \times 15 \times 30 = 27,000$ different cases. We observed that TR method outperformed almost cases to defeat adversarial noises and regain the true label. The median of recognition probabilities for adversarial label when using TR method is only 0.04% with adversarial inputs. Meanwhile, with original inputs the median of probabilities for true label is 74.65%. It is obviously that our system can completely defeat adversarial perturbation and regain true classification with acceptable rate.

Intuitively, Fig A-1 shows the coordinate transformation and geometric translation on the adversarial perturbation for three our Keyboard, C-Mouse and V-Machine

Table 3.5: Combination results on our own images

Image	Translation	Original class		Targeted class	
	v_i,v_j	Min	Max	Min	Max
Keyboard	(0,1)	26.36%	58.44%	0.01%	0.19%
	(0,2)	26.76%	**65.82**%	0.01%	**0.07**%
	(0,3)	29.26%	61.13%	0.01%	0.05%
	(1,0)	27.78%	60.26%	0.01%	0.22%
	(1,1)	26.16%	57.33%	0.01%	0.16%
	(1,2)	28.92%	64.2%	0.01%	0.06%
	(1,3)	28.53%	61.69%	0.01%	0.04%
	(2,0)	23.85%	57.82%	0.01%	0.25%
	(2,1)	26.19%	62.14%	0.01%	0.18%
	(2,2)	27.3%	64.44%	0.01%	0.07%
	(2,3)	30.79%	61.67%	0.01%	0.05%
	(3,0)	28.2%	61.11%	0.01%	0.23%
	(3,1)	26.78%	58.1%	0.01%	0.16%
	(3,2)	28.29%	55.5%	0.01%	0.08%
	(3,3)	29.6%	59.83%	0.01%	0.06%
C-Mouse	(0,1)	6.41%	90.95%	0.0%	0.65%
	(0,2)	5.98%	87.82%	0.0%	0.47%
	(0,3)	6.32%	69.83%	0.0%	0.21%
	(1,0)	7.92%	88.63%	0.0%	1.83%
	(1,1)	6.55%	90.91%	0.0%	0.42%
	(1,2)	6.23%	**93.86**%	0.0%	**0.41**%
	(1,3)	4.95%	77.05%	0.0%	0.15%
	(2,0)	6.39%	86.09%	0.0%	0.7%
	(2,1)	6.95%	90.56%	0.0%	0.29%
	(2,2)	5.35%	83.72%	0.0%	0.16%
	(2,3)	4.15%	89.0%	0.0%	0.12%
	(3,0)	11.56%	82.31%	0.0%	0.36%
	(3,1)	3.43%	79.85%	0.0%	0.22%
	(3,2)	3.85%	80.86%	0.0%	0.12%
	(3,3)	3.96%	90.01%	0.0%	0.12%
V-Machine	(0,1)	2.55%	51.44%	0.01%	2.14%
	(0,2)	1.53%	48.13%	0.01%	0.08%
	(0,3)	4.53%	47.44%	0.01%	0.44%
	(1,0)	2.99%	55.62%	0.01%	1.09%
	(1,1)	1.74%	61.7%	0.01%	1.21%
	(1,2)	2.3%	62.24%	0.01%	0.78%
	(1,3)	6.48%	61.21%	0.0%	0.39%
	(2,0)	1.9%	53.28%	0.01%	1.66%
	(2,1)	1.16%	55.63%	0.01%	1.2%
	(2,2)	1.22%	54.3%	0.01%	0.63%
	(2,3)	1.42%	57.37%	0.01%	0.33%
	(3,0)	0.97%	50.23%	0.0%	0.75%
	(3,1)	0.69%	49.96%	0.01%	0.54%
	(3,2)	1.36%	48.41%	0.01%	0.25%
	(3,3)	0.84%	**63.48**%	0.01%	**0.33**%

Table 3.6: Summary performance on our own images

Image	TRNS		ROT		Combination	
	Clean	Adv	Clean	Adv	Clean	Adv
Keyboard	62.01%	5.07%	56.5%	0.22%	**65.82%**	**0.07%**
C-Mouse	15.43%	10.4%	92.67%	2.16%	**93.86%**	**0.41%**
V-Machine	**66.71%**	17.2%	55.04%	3.67%	63.48%	**0.33%**

Table 3.7: Summary performance on ImageNet dataset

Method	Cases	Input	Probability for True Label				Probability for Adv Label			
			Min	Max	Median	Var	Min	Max	Median	Var
TRNS	900	Org	7.98%	99.51%	**84.43%**	0.13%	0.0%	0.06%	**0.01%**	0.0%
		Adv	0.22%	95.69%	38.58%	2.21%	0.01%	88.15%	0.60%	3.95%
ROT	1,800	Org	1.56%	99.72%	75.71%	9.60%	0.0%	0.07%	0.01%	0.0%
		Adv	0.71%	99.69%	53.15%	3.83%	0.0%	28.2%	0.05%	0.0%
TR	27,000	Org	0.38%	99.76%	74.65%	8.76%	0.0%	0.11%	**0.01%**	0.0%
		Adv	0.15%	99.84%	**56.27%**	5.84%	0.0%	50.14%	**0.04%**	0.0%

images. In Fig A-1, the first column shows the results when we use the coordinate transformation and second one is for geometric translation. In the figures in the first column, we define that the red line illustrates the probabilities that machine learning recognizes the input as adversarial ostrich and the green line shows for the true class. We observe that with a very small coordinate transformation, the adversarial ostrich is completely defeated. In the second column of the figures, the adversarial ostrich is also defeated by geometric translation, meanwhile this method attains high probabilities of true classification.

For MNIST dataset, we randomly extract ten digit-0 from the test set for creating adversarial examples with the target of digit-4. The adversarial digits completely make the LeNe's classifier misclassify digit-0 as digit-4. With the same experimental settings as ImageNet, we also apply TRNS, ROT and combination for defeating adversarial digits and recovering the clean digits.

We iteratively use the coordinate transformation on the adversarial examples with $\alpha \in [0, 30]$ and recognize that the perturbations are defeated by this method with only small value of α.

For MNIST dataset, we also recognize that the combination method is more pow-

erful and effective for defeating adversarial perturbation than single use of TRNS or ROT. The combination results are shown in table 3.8.

Table 3.8: Summary performance on MNIST

Index	TRNS		ROT		Combination	
	Clean	Adv	Clean	Adv	Clean	Adv
4457	99.96%	0.71%	93.69%	0.33%	99.96%	**0.00**%
5584	99.13%	3.22%	96.1%	**0.01**%	99.13%	0.14%
1438	96.65%	4.01%	83.52%	0.27%	97.00%	**0.13**%
1558	89.63%	9.74%	49.57%	4.96%	89.63%	**1.74**%
7031	85.22%	1.15%	94.05%	0.12%	85.22%	**0.06**%
9962	99.36%	6.93%	97.47%	0.09%	99.36%	**0.05**%
297	98.95%	14.22%	96.38%	**0.09**%	99.56%	**0.09**%
5259	99.97%	0.23%	99.24%	0.05%	99.97%	**0.00**%
4070	99.96%	0.40%	99.09%	0.01%	99.96%	**0.00**%
4515	99.99%	0.53%	94.65%	1.04%	99.99%	**0.00**%

Fig A-2 shows the effectiveness of coordinate transformation (ROT) on adversarial examples. The sub figure besides of each picture shows the change in the probability of true label (digit 0) with green curve line and adversarial label (digit 4) with red curve line. The best case is in digit 0 with index 5584. In this best case, the probability of true label linearly increases with angle of rotation, meanwhile the probability of adversarial label decreases dramatically and reaches to 0 when angle of rotation is increased to 5 degree. The worst case is in digit 0 with index 1438. In this worst case, although the probability of adversarial label dramatically drops from nearly 100% to under 10% but it still remains and fluctuates around 10%.

3.1.7 Summary for Image Transformation for Detecting Adversarial Examples

Adversarial attack so far is a very serious problem for security and privacy on machine learning systems. This research work provides evidence that the deep neural networks can be made more robust to adversarial attacks by giving TRNS, ROT and their combination transformations to input images. As our theory and experiments, we now can develop powerful defense methods for adversarial problem. Our experiments on

ImageNet and MNIST datasets have not reached the best on all of cases. However, our results already show that our approach lead to significant increase in the robustness of deep neural network. Our best performance case (in table. 3.7) on the ImageNet images shows that it can reduce the classification rate for adversarial example from 99.9% to 0.00%. And we also believe that our findings will be further explored in our future work.

3.2 Image Filters for Detecting Adversarial Examples

In this section, we introduce new techniques to overcome adversarial examples by using Image Filters. Our proposed system can automatically detect and classify adversarial samples and legitimate samples. We assume that most of adversarial perturbations are created in high frequencies of image because adversarial examples are created only respect to loss(x^{adv},y^{adv}), so there are many properly adversarial noises exist in high frequencies. Base on this assumption, we focus on the way how to reduce high frequency adversarial noises and still keep other high frequency benign features. For proving our hypothesis right, first we put a low pass filter layer between adversarial example and target classifier. We observed that the probability of targeted class from classifier dropped significantly to near zero, meanwhile we regained the recognition results for the primary class. We will also demonstrate the correctness of these implementations into theory proof in Sec 3.2.1. Based on the previous observation, we proposed the new end-to-end system for automatically detecting adversarial by design a sieve layer between input and deep neural network to stick suspicious noises. In parallel with that process, we feed the un-sieved input to classifier and mark the highest confidence class as an anchor. We compare the probability of anchor and sieved input from classifier based on a specified fixed threshold to make a decision that one is adversarial or benign. The our key idea is depicted in Fig. 3-4.

The main contributions of this research are as follows:

- We investigated and analyzed attack approaches for crafting adversarial examples. We showed attack approaches along their different strategies to provide an intuitive overview of these attack methods.

- We investigated the modern defense approaches and their variants in adversarial settings. We assume most of adversarial perturbations are created in high frequencies. After implementing many experiments respect to theory framework, we confidently confirm our hypothesis right.

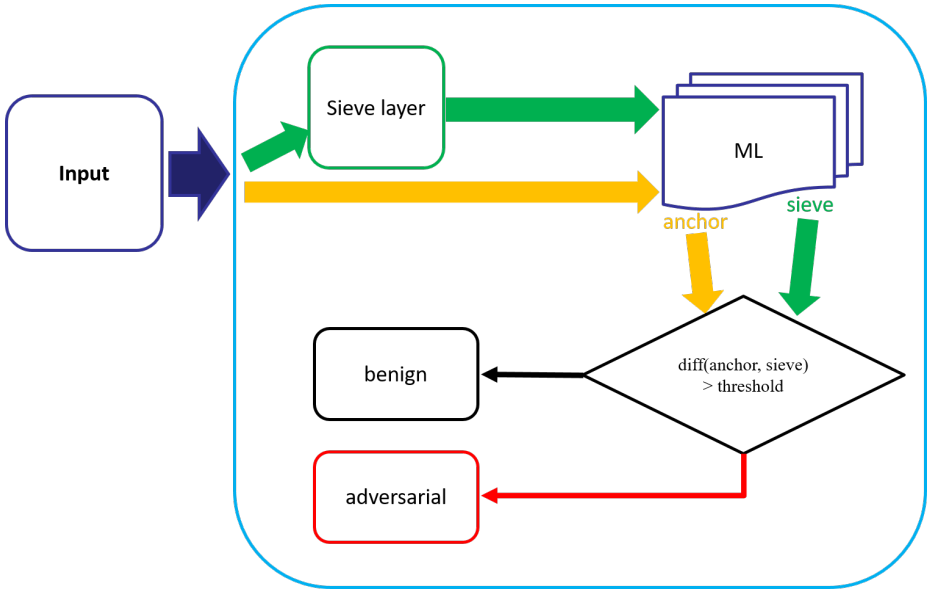


Figure 3-4: Our automated detection system for adversarial examples by using image filters

- We created new automated detection method for adversarial examples based on a thorough analysis of observations obtained from many experimentation and theoretical framework. Our approach is different from other previous research works that they usually only have experimental steps based on the original hypothesis. Our proposed approach is successfully applied two types of common datasets that are a small-scale dataset (MNIST) and a large-scale dataset (ImageNet). Our defense method worked well to classify adversarial examples and legitimate samples. Moreover, in some cases it recovered the deep learning model's classification with high accuracy rates.

3.2.1 Detection phase

We create new benchmark dataset for our detection system by combination of benign images and adversarial images that we created in attack phase. We assume that adversarial noises are high frequencies feature on images, so we will consider the way to remove them from high frequency domain on images while still keep all features in low frequency area. There are some common algorithms that are applied to reduce the noises from images for further processing such as classification. In this work, we

investigate two most known filters in image denoising field are linear and non-linear filters. Let consider an example, by constructing a new array that has the same size as the specified image. Fill each location of this new array with a weighted sum of the pixel values from the locations surrounding the corresponding location in the image, using the same set of weights each time. The result of this procedure is shift-invariant — meaning that the value of the output depends on the pattern in an image neighbourhood, rather than the position of the neighbourhood and linear meaning that the output for the sum of two images is the same as the sum of the outputs obtained for the images separately. The procedure itself is known as linear filtering. This process helps us to smooth noises in images. And one of famous linear filter is Gaussian filter that is defined as equation:

$$G_\sigma(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}} \tag{3.29}$$

Where i, j is denoted as input's coordinate signal and σ is standard deviation of the Gaussian distribution. And an alternative approach to remove noises is to think of a filter as a statistical estimator. In particular, the goal here is to estimate the actual image value at a pixel, in the presence of noisy measurements. This view leads us to a class of filters that are hard to analyze, but can be extremely useful. Smoothing an image with a symmetric Gaussian kernel replaces a pixel with some weighted average of its neighbours. If an image has been corrupted with stationary additive zero-mean Gaussian noise, then this weighted average gives a reasonable estimate of the original value of the pixel. The expected noise response is zero, and the estimate has better behaviour in terms of spatial frequency than a simple average. However, if the image noise is not stationary additive Gaussian noise, difficulties arise. In particular, consider a region of the image which has a constant dark value and there is a single bright pixel due to noise — smoothing with a Gaussian will leave a smooth, Gaussian-like, bright bump centered on this pixel. The problem here is that a weighted average can be arbitrarily badly affected by very large noise values. Thus, we can make the bright bump arbitrarily bright by making the bright pixel arbitrarily bright perhaps as

result of a transient error in reading a memory element. Estimators that do not have this most undesirable property are often known as robust estimates. The best known robust estimator involves estimating the mean of a set of values using its median. A median filter is specified by giving some form of neighbourhood shape (which can significantly affect the behaviour of the filter). This neighbourhood is passed over the image as in convolution, but instead of taking a weighted sum of elements within the neighbourhood, we take the median. If we write the neighbourhood centered at (i,j) so the filter can be described by:

$$x_{ij} = median(X_{uv} | X_{uv} \in \mathbb{N}_{ij}) \tag{3.30}$$

Where X_{uv} is denoted as the neighbourhood points of x_{ij}. By smoothing pixels in image, we can leverage adversarial noises if they are exist. In case without adversarial noises, smoothing pixels does not affect the input image quality too much so target's classifier still recognizes the right label. We name this process is sieve process (noted by green arrow in Fig. 3-4).

Our proposed detection system has two parallel processes include sieve process and anchor process. Sieve process will leverage the high frequencies in input processing while anchor process will transfer input directly to machine learning model. The probability of the highest confidence class from machine learning model will be sticked as anchor, and we use this anchor class to tracking the oscillation of the class similar to anchor class on the sieve process. If the differentiation of probability p of anchor and sieve is greater than a fixed Θ threshold, our system will confidently point out it as adversarial and vice versa. Our system is defined in algorithm 3. Where: κ is denoted for kernel size, f is machine learning function to produce probabilities of predicted class, s is sieve function. We denote the sieve function created based on Gaussian filter is Detection System based on Gaussian - DSG and another one is Detection System based on Median - DSM.

Algorithm 3: Automated Detection System of Adversarial Examples with High Frequency Sieve

input : X, Θ, s, f
output : $0, 1$
// 0: benign; 1: adversarial
parameter: $\kappa = [(3 \times 3); (5 \times 5)]$

for x *in* X **do**
  $anchor_x \leftarrow x$
  $sieve_x \leftarrow x$
  $sieve_x \leftarrow s(sieve_x, \kappa, \sigma)$
  $p(anchor_y) \leftarrow f(anchor_x)$
  $p(sieve_y) \leftarrow f(sieve_x|anchor_y)$
  if *diff*$(p(anchor_y), p(sieve_y)) > \Theta$ **then**
    return 1
  else
    return 0
  end
end

3.2.2 Datasets

In this work, we consider two common benchmark datasets for classification task include MNIST and ImageNet.

Setup for MNIST

MNIST dataset [57] includes 70,000 gray images of hand-written digits from 0 to 9. It is separated into two parts including 60,000 training images and 10,000 testing images. Each image in MNIST is 28×28 pixels with each pixel is encoded by 8-bit grayscale. We randomly extract 200 images of digit "0" from 10,000 testing images. From each of 200 images of digit "0", we create nine different adversarial images target to the rest digit (from 1 to 9). Finally we create new benchmark dataset include 2,000 images (200 benign images and 1,800 adversarial images).

Setup for ImageNet

We consider the ImageNet dataset [96] that is a very large database designed for use in visual object recognition research. The original ImageNet includes more than

14 millions images in 20,000 categories with a typical category, such as "computer mouse" or "vending machine", consisting of several hundred images. The machine learning model that we use is Google Inception V3 [108] that was trained with 1,000 common categories ImageNet. We randomly select 1,000 testing images exclude image of "ostrich". This selection of targeted class does not compromise the generality of our system. By applying FGSM and PGD method to craft adversarial images target to "ostrich", we generate new 2,000 adversarial images. We combine them together to form new benchmark repository including 3,000 images for our experiment.

3.2.3 Implementation

Adversarial examples recently attracted a lot of interest from researchers however there is still no public benchmark dataset for evaluating the robustness of defense system. So the purpose of attack phase is to create a new benchmark dataset that evaluates the detection capabilities of our detection system.

In MNIST dataset, from 200 random images of digit "0", we use FGSM method to craft adversarial images with targeted class from 1 to 9. The number of iteration (epochs) in FGSM is 1,000. Afterward, we combinate them to form a benchmark dataset for evaluating our detection system. The proposed detection system knows true label of input and we only provide input after that our system automatically processes and returns decision that the input is adversarial or benign.

In ImageNet dataset, from 360 random testing images without "ostrich", we use FGSM, PGD, CW and EAD to craft adversarial images with targeted class is "ostrich". The number of iteration (epochs) in FGSM, PGD, CW and EAD is 500 times. In Fig. 3-5, we show five samples from 390 random picked images that we use for crafting adversarial examples. In the first row, there are original images (or we usually call as benign images) includes dish, hammerhead, mosque, oscilloscope and parachute. The probability row shows the highest probability of class in class name in brackets. We finally create 1,800 images including 360 original images and 1,440 adversarial images.

In sieve process, we set kernel sizes for Gaussian and Median filter are (3×3) and

(5×5). We observe the change of probability of anchor before and after applying filters to make a decision base on a given threshold.

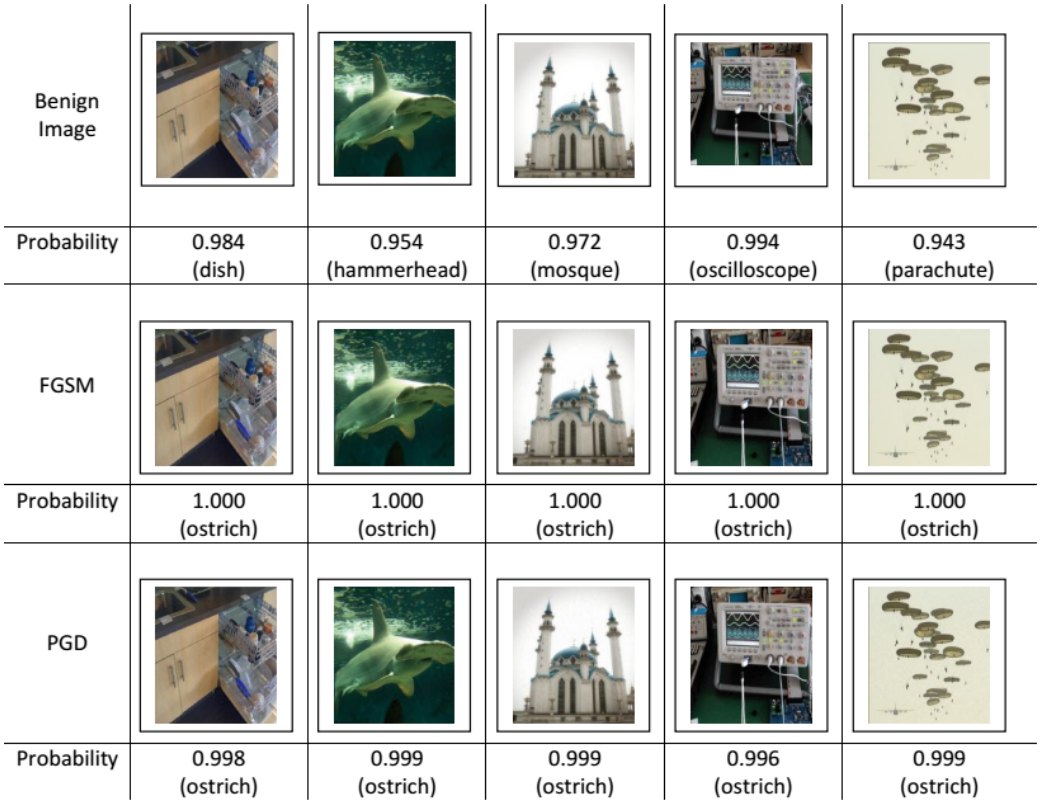


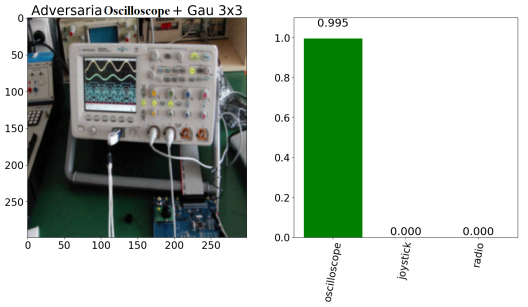
Figure 3-5: Attack Phase Samples.

3.2.4 Results

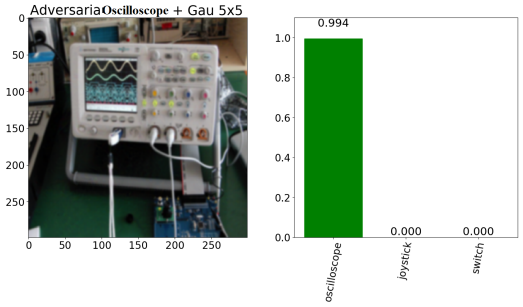
We compare our result with Xu et al. [120]. Our system is more convenient that Xu's system in either detection accuracy and easy-to-setup, our system consistently uses a fixed threshold while Xu's system has to adapt a variety of threshold values. We also report our system's performance by using F1-scores metric. Our detection system based on Gaussian named as DSG and another one is Detection System based on Median named as DSM.

We observe and analyze a typical case with image of oscilloscope. From benign oscilloscope image with probability is around 99.7%, we created two adversarial ostrich by using FGSM and PGD method (Fig. 3-5). Afterward, we used DSG and

DSM function for sieving adversarial ostrich noises and regain oscilloscope features. Fig. 3-6 has expressed probabilities change dramatically on targeted ostrich and legitimate oscilloscope when processed by DSG function. This observation confirms our assumption that adversarial noises are high frequencies and by adapting low pass filter in our model, we can proposed a powerful detection model for adversarial examples.



(a) Observation on probabilities of Oscilloscope label



(b) Observation on probabilities of Oscilloscope label

Figure 3-6: Adversarial ostrich image (true class: Oscilloscope) suffers to our sieve process

Adversarial Detection results on MNIST dataset are summated in Table 3.9. In Table 3.9, "-" means we do not have information from the other research work. Although the number of images in comparison is the same, the way we create test set is more challenge than Xu et al. [120]. Xu kept a balanced dataset by creating 1,000 legitimate images and 1,000 adversarial examples while we created 1,800 adversarial images from 200 legitimate inputs so our model had to cope with imbalanced [107] dataset in our experiment. Our detection rates are very competitive and performed better than Xu's work. Besides that, our automatic detection system used a fixed threshold for all settings while Xu's work used a variety of threshold respect to settings. In the implementation results on MNIST dataset, the DSG method achieves the best performance on detection rate at 99.9% while Xu's the best result is only at 98.2%.

In ImageNet dataset, we observed our detection rates reached the highest score for comparing to Xu's work. In Table 3.10 also shows that the number of files we used in this implementation is larger than Xu et al. and we still keep our benchmark dataset is imbalance for evaluating our proposed model. Our detection rate achieved around

Table 3.9: Detection Rate on MNIST dataset

	Our Method		Xu et al. [120]		
	DSG	DSM	Bit-Depth	Smoothing	Best-Joint
No. Files	2,000	2,000	2,000	2,000	2,000
Threshold	0.1	0.1	0.0005	0.0029	0.0029
True Positive	1799	1796	-	-	-
True Negative	198	195	-	-	-
False Positive	2	5	-	-	-
False Negative	1	4	-	-	-
Accuracy	0.999	0.996	-	-	-
Precision	0.999	0.997	-	-	-
Recall	**0.999**	0.998	0.903	0.868	0.982
F1 score	0.999	0.998	-	-	-

Table 3.10: Detection Rate on ImageNet dataset

	Our Method		Xu et al. [120]		
	DSG	DSM	Bit-Depth	Smoothing	Best-Joint
No. Files	1,800	1,800	1,800	1,800	1,800
Threshold	0.9	0.9	1.4417	1.1472	1.2128
True Positive	1,380	1,433	-	-	-
True Negative	356	350	-	-	-
False Positive	4	10	-	-	-
False Negative	60	7	-	-	-
Accuracy	0.964	0.991	-	-	-
Precision	0.997	0.993	-	-	-
Recall	0.958	**0.995**	0.751	0.816	0.859
F1 score	0.997	0.994	-	-	-

95.8% with DSG and 99.5% with DSM. DSM worked perfectly to detect all adversarial examples while Xu's the best performance is only 85.9% detection accuracy rate. We use a fix threshold at 0.9 in both DSM and DSG methods. We recognize that the DSM works better than DSG to detect adversarial image but worse to recognize the original images. From these observation, we find out that the DSM is very strong method to remove the features on the high-frequency and in some case it also remove the importance features that makes false positive is higher than DSG method.

3.2.5 Summary for Denoising Detection System

In this section, we investigated the high frequencies in the adversarial examples. We assume that adversarial noises are high frequencies feature on images, so we will consider the way to remove them from high frequency domain on images while still keep all features in low frequency area. There are some common algorithms that are applied to reduce the noises from images for further processing such as classification. In this work, we investigate two most known filters in image denoising field are linear (Gaussian filter) and non-linear filter (Median filter). Based on assumption and theory framework, we demonstrated the effectiveness of low pass filter in remove high frequency adversarial noises. From this observation we proposed a automated detection system for adversarial examples.

On MNIST dataset, we use FGSM to create adversarial examples and our detection system achieved detection accuracy rate up to 99.9% with DSG method and 99.8% with DSM method.

On ImageNet dataset, we use FGSM, PGD, CW and EAD methods to create adversarial examples. Our detection system also reached to 99.5% detection accuracy rate with DSM method and 95.8% with DSG method.

From our experimental results, we recognized that Median filter is better than Gaussian filter for defeating adversarial noises although Median filter will take more time than Gaussian in computation. In evaluation of our system, we setup the new benchmark datasets are more challenge than [120, 62] when they used images from training set for evaluating rather than testing set, while we use testing images for implementation. We also challenged our model by keeping imbalanced datasets and our detection system still reached state-of-the-art performance. Another important contribution in this work is that we not only defeated adversarial noises, but also our systems regained the legitimate class from effectiveness of adversarial examples.

3.3 Geometric Transformation and Denoiser for identifying Adversarial Examples

In this section, we describe the state-of-the-art automatic detection system that detects and distinguishes between adversarial and legitimate samples from unknown inputs. We assume that most of the adversarial noises are crafted with only respect to deceive a classifier and imperceptible to perceive by a human. By adopting slightly the pixel values of input image toward to new targeted label, adversarial attack methods are inattentive to spatial constraints from the original image, and it leads to drop adversarial noises on the high-frequency domain. Moreover, most of the adversarial attack methods based on gradient descent to create malicious patterns and optimize loss function between legitimate input with a targeted label, it tries to find the best candidate with the targeted classifier. This leads to a variant of overfitting when fine-tuning the important features of the original image. Besides, the image classification systems based on the deep learning are very sensitive to transformation so we deeply investigate how to use transformation to break the effectiveness of adversarial examples on the classification systems.

To prove our assumption, we first set up a transform layer and a denoiser layer between the adversarial image and a classifier. We observe the probabilities of outputs and analyze the changing of probabilities of the original label and adversarial label before and after using layers. From those observations, we propose a state-of-the-art automatic adversarial identification system with three parallel layers between input images and a classifier. Three parallel layers include a forward layer, transform layer and denoiser layer. The forward layer's role is forwarding the unknown input to the classifier. The transform layer and denoiser layer perform on the unknown input to deform and remove the adversarial features in the high-frequency band. The observed outputs from the classifier are the labels with the highest probabilities to form into a vector. We set up a decision system to evaluate the vector to decide which unknown input is adversarial or benign. Our proposed model is described in Fig. 3-7

Algorithm 4: Automatic Identification System with Image Transformation Layers

input : x, f, L, GR, GT, $GT\&R$

output : 1: adversarial input, 0: clean input

$l_{marked} \leftarrow f(x)$ `// generating a marked label`
if GR **then**
    for i *in range(*5,20*)* **do**
        $angle \leftarrow$ i * $\pi/180$
        $x_{GRi} \leftarrow GR(x, angle)$; $l_i \leftarrow f(x_{GRi})$; $L \leftarrow l_i$
    end
else if GT **then**
    for i *in range(*5,20*)* **do**
        for j *in range(*5,20*)* **do**
            $v_i \leftarrow i$; $v_j \leftarrow j$; $x_{GTi} \leftarrow GT(x, v_i, v_j)$
            $l_i \leftarrow f(x_{GTi})$; $L \leftarrow l_i$
        end
    end
else
    for i *in range(*5,20*)* **do**
        for j *in range(*5,20*)* **do**
            $v_i \leftarrow i$; $v_j \leftarrow j$
            $x_{GTij} \leftarrow GT(x, v_i, v_j)$
            for q *in range(*5,20*)* **do**
                $angle \leftarrow q$ * $\pi/180$
                $x_{GRq} \leftarrow GR(x_{GTij}, angle)$
                $l_i \leftarrow f(x_{GRq})$; $L \leftarrow l_i$
            end
        end
    end
end

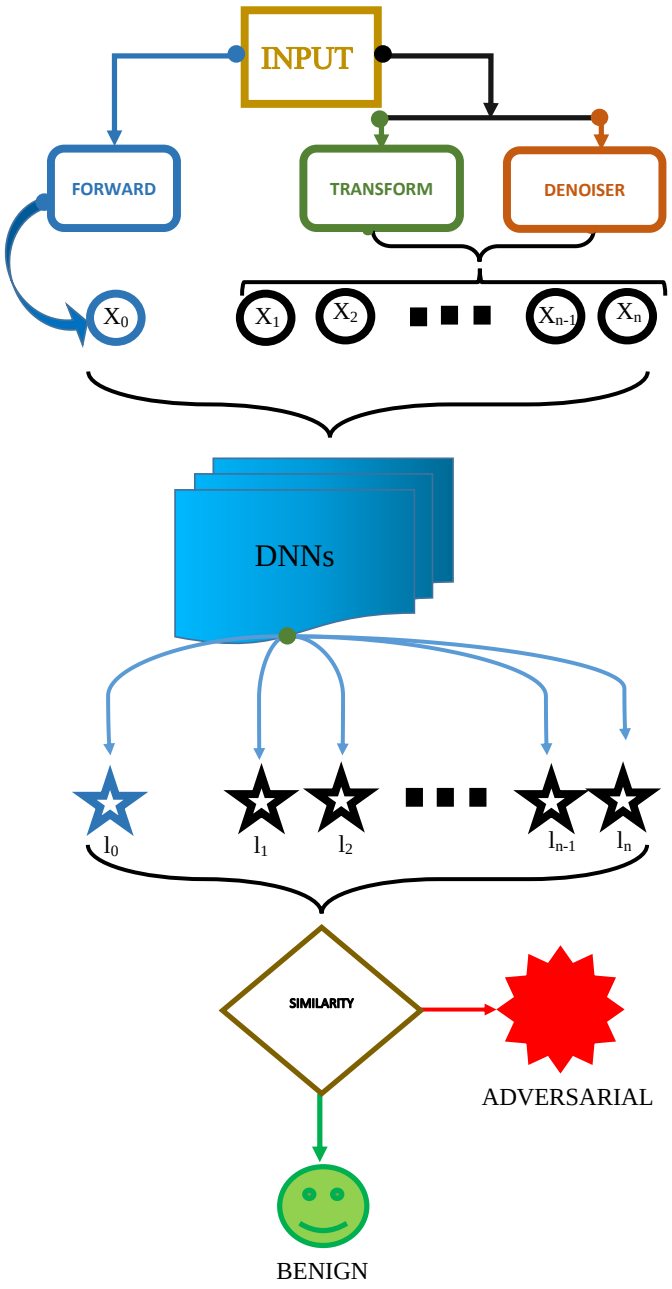


Figure 3-7: Adversarial Examples Identification System

In this model, we set up three layers simultaneously between the input and the classifier. The forward layer is passed directly into the classifier. Transform layer

using image transformation algorithm (geometric rotation, geometric translation, and combination). The Denoiser layer uses algorithms to eliminate noise in the high-frequency domain. Afterward, the data has been processed through these three layers, from each input data we have a set of data $X = \{X_0, X_1, ..., X_{n-1}, X_n\}$. These data will be the input of the classifier. The outputs of the classifier are labels with the highest probability $l = \{l_0, l_1, ..., l_{n-1}, l_n\}$. Based on the set of labels l, we use a similarity function to compare the similarities of the elements in the set l to decide whether the input data is adversarial or benign. Our system performs to analyze adversarial data automatically and there is no need to set a threshold value to make a final decision. The filter layer places the high-frequency components in the input processing while the forward layer transfers the input directly to the target classifier. The highest-confidence class from the classifier is assigned as the marked label. The filter process then tracks the labels similar to the marked label. If the marked label is equal to all filtered labels, our system confidently determines the input as benign, but in the case, where are more than two different marked labels that are different from the marked label, the detection system points out it as an adversarial example. Our system proceeds by algorithm 3, where x defines the input image, L is filtered labels, l_{marked} is marked label, κ denotes the kernel sizes, f is a machine learning function that computes the predicted label with the highest probability, and s is the filter function. The filter function based on the Gaussian filter is called the label-based identification model based on Gaussian (LIMG); the other filter function is our identification model based on Median (LIMM). We set the kernel size in range [3×3, 5×5].In the transform layer (algorithm 4), we define three main functions include GR, GT, and GT&R that stand for geometric rotation, geometric translation and a combination of them, respectively. For GR method, we chose the rotation angle in range [5,20]. In GT method, we select the translation in range[5,20]. For GT&R method, the first we apply the GT, after that we use GR on the image input. The composite model that combines transform and denoiser layers we call CMTD.

3.3.1 Evaluation Metrics

We use the some commonn metrics for evaluating our implementation as below.

- **True Positive (TP)** is an outcome where the model correctly predicts the *positive* class.
- **True Negative (TN)** is an outcome where the model correctly predicts the *negative* class.
- **False Positive (FP)** is an outcome where the model incorrectly predicts the *positive* class.
- **False Negative (FN)** is an outcome where the model incorrectly predicts the *negative* class.
- **Precision** or positive predictive value is the proportion of positive and negative results and equal to TP/(TP+FP).
- **Recall** or sensitive or true positive rate measures the proportion of actual positives that are correctly identified and it is defined as TP/(TP+FN).
- **Accuracy** is the closeness of the measurements to a specific value, and it is defined as (TP+TN)/(TP+TN+FP+FN).
- **F1 score** is a measure of a test's accuracy, and it equals to 2*(Precision * Recall)/(Precision + Recall).

3.3.2 Implementation and Settings

In this section, we describe the setup and settings of the proposed automatic identification models. In general, we use two benchmark data sets: small scale MNIST and large scale ImageNet to build synthesized data sets. In this study, we use the state-of-the-art deep neural network is Google Inception V3 [108] as a victim classifier to create adversarial images as well as evaluating the identification results because the

problem we focus on is the white-box manner. In the creation of adversarial examples, we use four attack methods: FGSM, PDG, CW_L_2, and EAD. In the transform layer, we use three methods of GT, GR, and GTR. In the denoiser layer, we use two methods of LIMG, LIMM. The proposed composite model uses the transform and denoiser layers. The test was performed on Intel (R) Core (TM) i9-9900X 20 CPUs 3.50 GHz, 64 GB Memory, NVIDIA GeForce RTX 2080Ti 11GB.

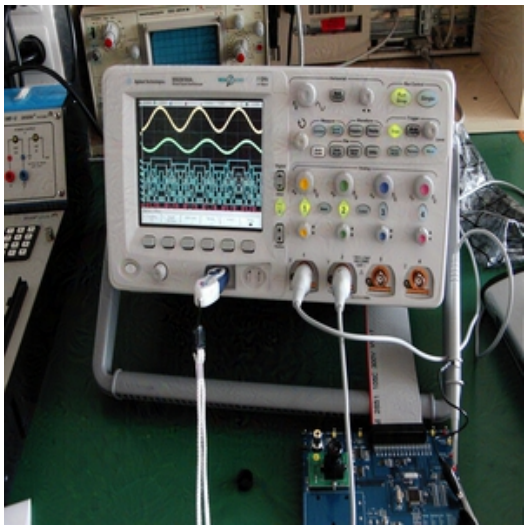
Setup of MNIST

The MNIST dataset [57] includes 70,000 gray images of hand-written digits ranging from 0 to 9. It is separated into 60,000 training images and 10,000 testing images. A single MNIST image is composed of 28×28 pixels, and is encoded by an 8-bit grayscale. We randomly extract 200 images of the digit "0" from the 10,000 testing images. From each of these 200 images, we create nine adversarial images targeting the remaining digits (1-9). Finally, we create a new synthesized benchmark dataset of 2,000 images (200 benign images and 1,800 adversarial images).

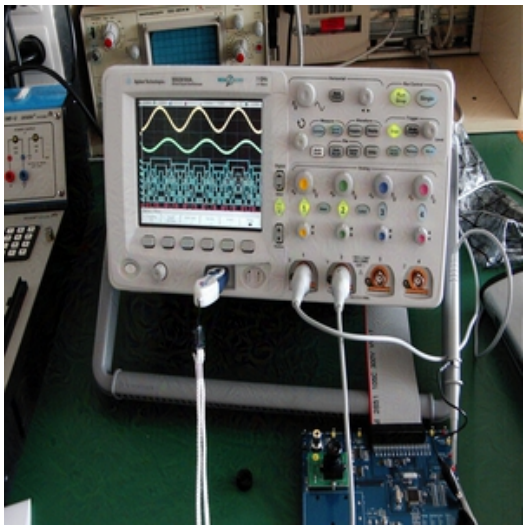
Setup of ImageNet

We consider the ImageNet dataset [96] that is a very large database created for using in many tasks such as image classification, and object detection. The original ImageNet includes more than 14 million images in 20,000 categories with a typical category, such as "oscilloscope" or "ostrich", consisting of several hundred images. The machine learning model that we use is Google Inception V3 [108] that was trained with 1,000 common categories ImageNet. We randomly select 360 testing images. By applying FGSM, PGD, CW_L_2 and EAD methods to craft adversarial images target to randomly chosen labels, we generate new 1,440 adversarial images. We combine them together to form a new benchmark repository including 1,800 images for our experiment. Although adversarial examples have recently attracted much interest from researchers, a public benchmark dataset for evaluating the performance and quality of defense systems remains lacking. In the attack phase of our system, we thus created a new benchmark dataset for evaluating the detection capabilities.

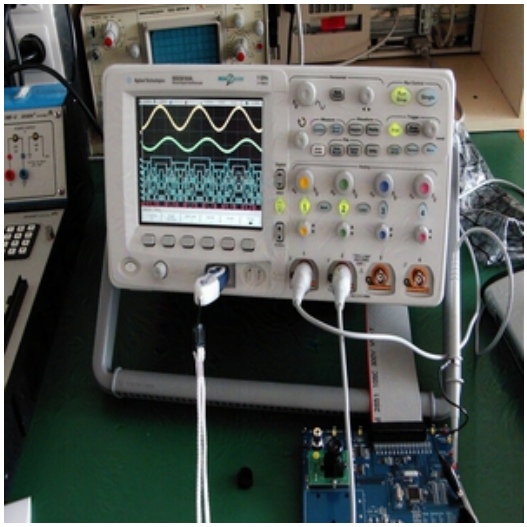
3.3.3 Results



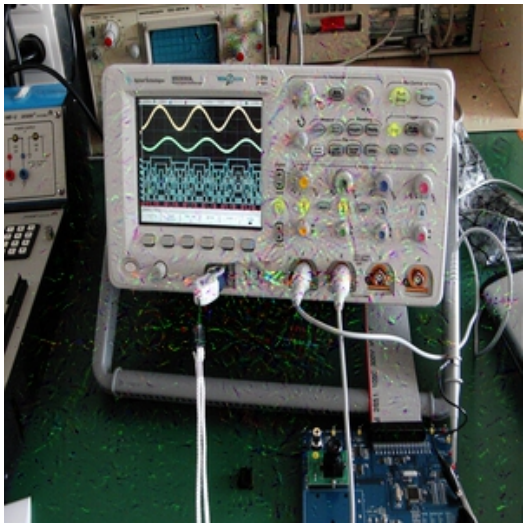
(a) FGSM Adversarial Image, 99.99% as ostrich label



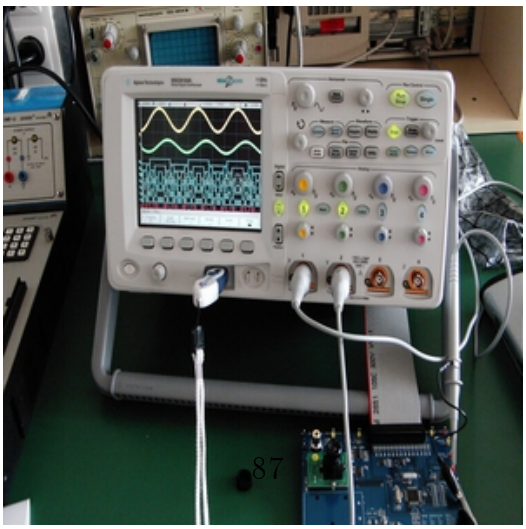
(b) PGD Adversarial Image, 99.90% as ostrich label



(c) CW_L_2 Adversarial Image, 99.85% as ostrich label

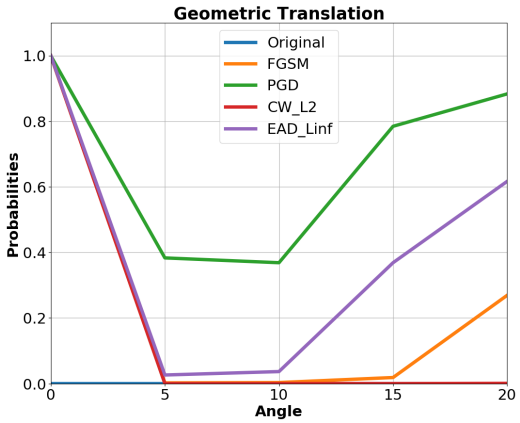


(d) EAD Adversarial Image, 99.89% as ostrich label

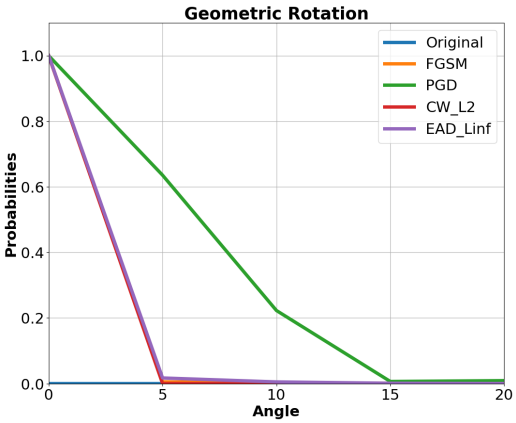


(e) Original Image, 99.35% as the ground

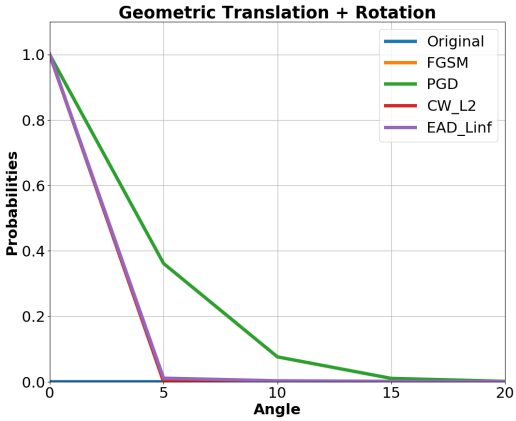
We compare our results with Xu et al. [120]. Our system is more convenient than Xu's system in either detection accuracy and easy-to-setup, our system does not use a threshold for making the decision while Xu's system has to carefully choose a set of threshold values. Our system only setup the paramters for the transform layer and denoiser layer such as kernel sizes, rotation angles, and translation angles. All of these parameter's values are selected based on our observation in the implementation results. We also report our system's performance by using the F1-scores metric. We observe and analyze a typical case with an image from the ImageNet dataset. In Figure 3-8, from a random image in the ImageNet dataset, the classifier system produces the highest probability of 99.7% of the *oscilloscope* label. We use four different methods FGSM, PGD, CW_L_2 and EAD to create adversarial images with targeted labels as a *ostrich* label by randomly selected from 1,000 labels in the ImageNet dataset.



(a) Observation on probability of targeted label when using GT method



(b) Observation on probability of targeted label when using GR method

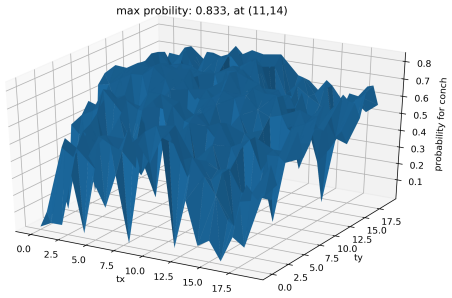


(c) Observation on probability of targeted label when using GT&R method

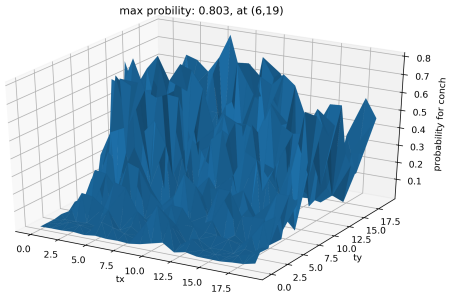
Figure 3-9: Our proposed Model 1 defeats the adversarial perturbation's affect

Figure 3-9 shows the changing of probability classification for the targeted label before and after using the transform layer. We observe that the probabilities of the targeted labels suddenly drop after using the transform layer. In figure 3-9.a, when the angle value is zero (meaning GT is not used), the probability of the targeted label is close to 1.0 for adversarial images, but for blue lines (the observation on the benign image), the probability is 0.0. However, when the angle value increases to 5 degrees, we can see that the probability of targeted labels on adversarial images decreases to approximately 0.0 (exclude PGD adversarial image in green line). And this is maintained when increasing the value of the angle up to 10 degrees for adversarial examples created using the FGSM, PGD and CW_L2 methods. The probabilities of

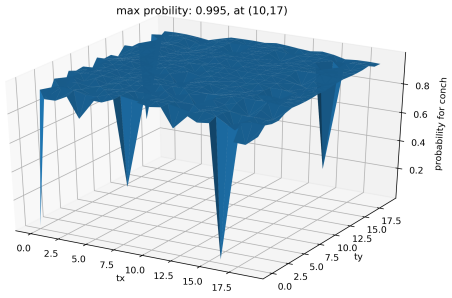
the targeted label tend to increase in the range [10,20] when using the GT method in this case. In figure 3-9.b, the GR method works more effectively than the GT method when the angle value increases from 5 to 20 degrees, the probability of targeted label decreases and remains at 0.0 for all adversarial images created by four attack methods. In figure 3-9.c, the GT&R method also works well when the effect of the adversarial perturbation on the classifier system has been eliminated. This is a very important observation that we decide on the range of angles of transformation in our identification system in range $[1, 5]$ in the percentage of an input image.



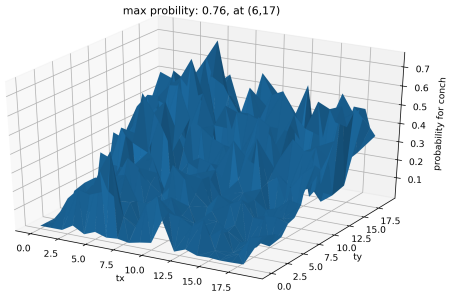
(a) Input: FGSM Adversarial Image



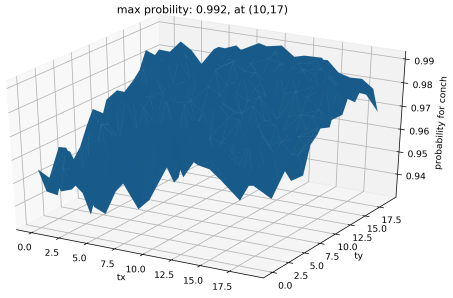
(b) Input: PGD Adversarial Image



(c) Input: CW_L2 Adversarial Image



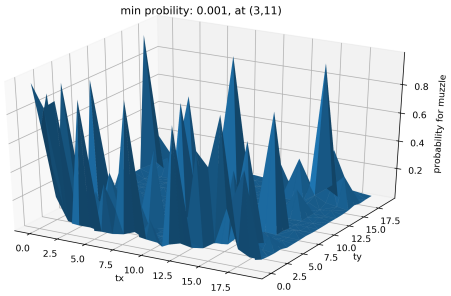
(d) Input: EAD Adversarial Image



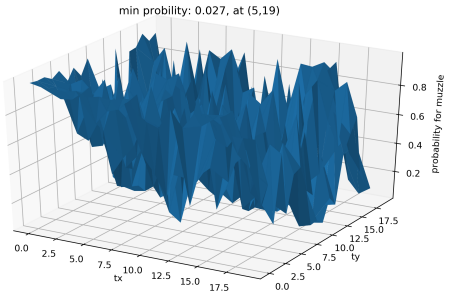
(e) Input: Original Image

Figure 3-10: Observation on the probabilities of the ground truth label with GT&R method

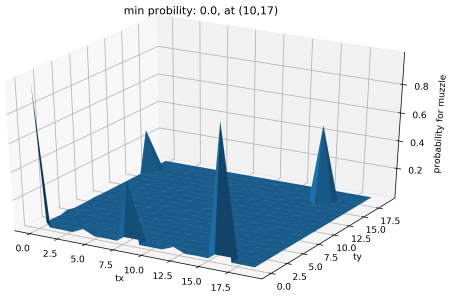
Figure 3-10 shows the observations on the probabilities of the ground truth labels when we use the GT&R method. We can observe that in some cases (such as Figure 3-10.c and Figure 3-10.e), the probabilities of the ground truth label are even better than without GT&R.



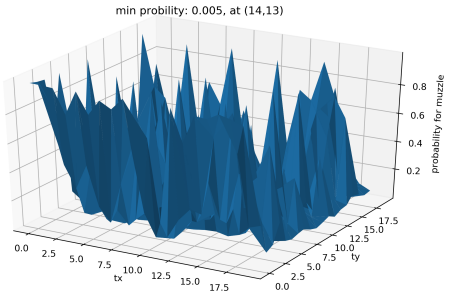
(a) Input: FGSM Adversarial Image



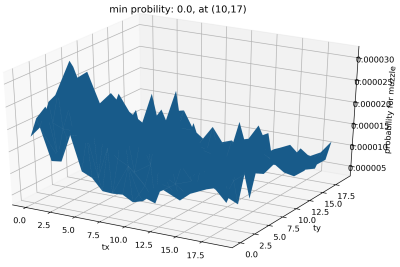
(b) Input: PGD Adversarial Image



(c) Input: CW_L2 Adversarial Image



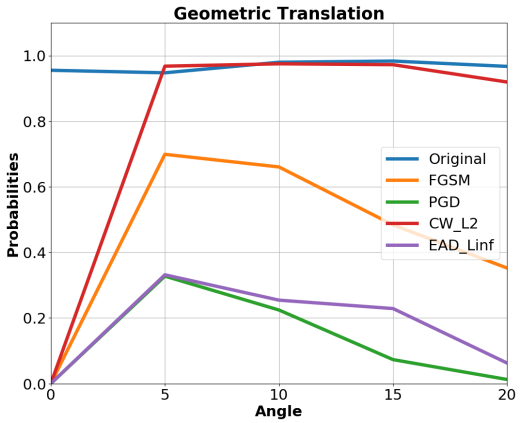
(d) Input: EAD Adversarial Image



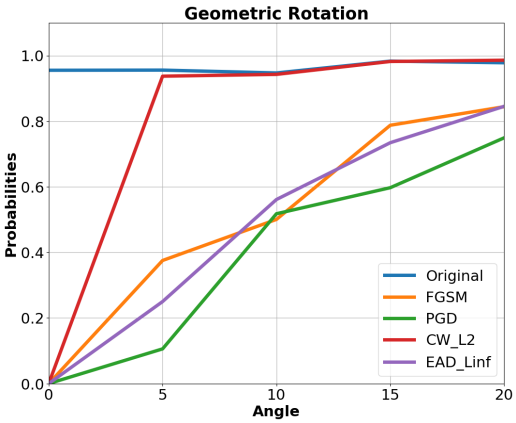
(e) Input: Original Image

Figure 3-11: Observation on the probabilities of the targeted label with GT&R method

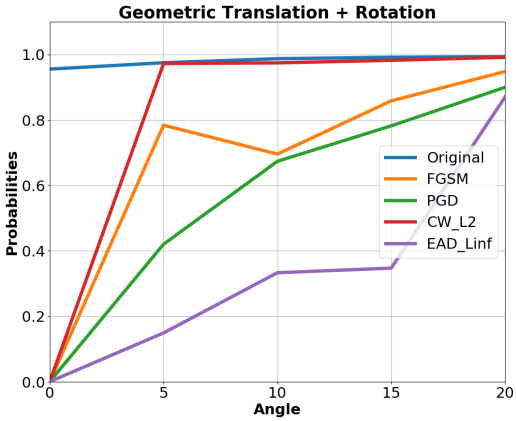
Figure 3-11 shows observations of the probabilities of a targeted label change when the GT&R method is applied. It is obvious to see that in many cases the GT&R method suppresses the probability of targeted labels to around 0.0.



(a) Observation on probability of ground truth label when using GT method



(b) Observation on probability of ground truth label when using GR method

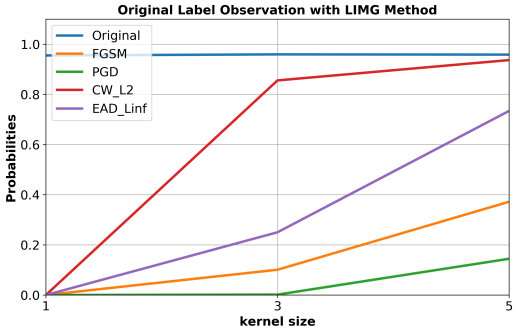


(c) Observation on probability of ground truth label when using GT&R method

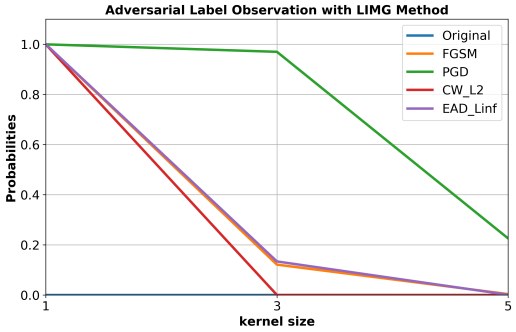
Figure 3-12: Our proposed Model 1 regain the true recognition

Figure 3-12 shows the probability classification results for the ground truth label are stable when input is the original image after and before using the transform layer. On the other hand, probability classification for the ground truth label increased dramatically when inputs are adversarial images after using transform. Figure 3-12.a shows the probability of ground truth label when the input images are adversarial. These probabilities are recovered when the angle value increases to 5 degrees. But when the angle value increases more, the probabilities tend to go down, excluding the CW_L2 adversarial image. In Figure 3-12.b, we can observe that the GR method works more effectively than the GT method when the probabilities of the ground truth label increase steadily with the increment in angle value. In Figure 3-12.c, the

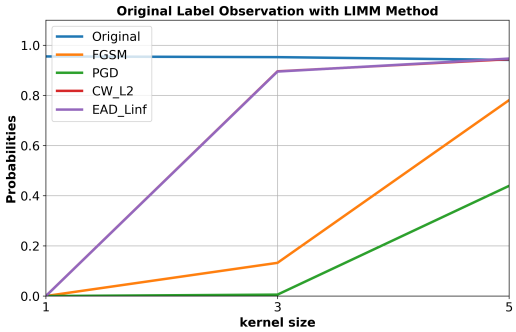
GT&R method achieved slightly better results when the probabilities of the ground truth labels increased close to the baseline. This observation confirms our assumption that adversarial perturbations are weak to our proposed model and by adapting the transform layer in our model, we can propose a powerful automatic identification model for adversarial examples.



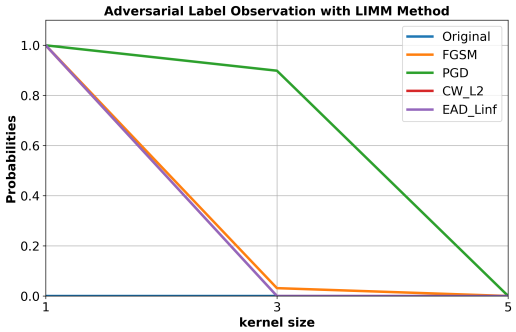
(a) Probabilities of ground truth label with LIMG method



(b) Probabilities of targeted label with LIMG method



(c) Probabilities of ground truth label with LIMM method



(d) Probabilities of targeted label with LIMM method

Figure 3-13: Adversarial examples suffers to our first proposed model (LIMG + LIMM)

Next, this is followed by empirical results using the LIMG and LIMM methods. Figure 3-13 shows the observations on the probabilities of the ground truth and the targeted label. In Figure 3-13.a and 3-13.b, both methods help the classifier to increase the probability of ground truth when we increase the kernel size from (3×3) to (5×5). In contrast, the probability of targeted labels also decreases as the kernel size increases as are depicted in Figure 3-13.b and 3-13.d.

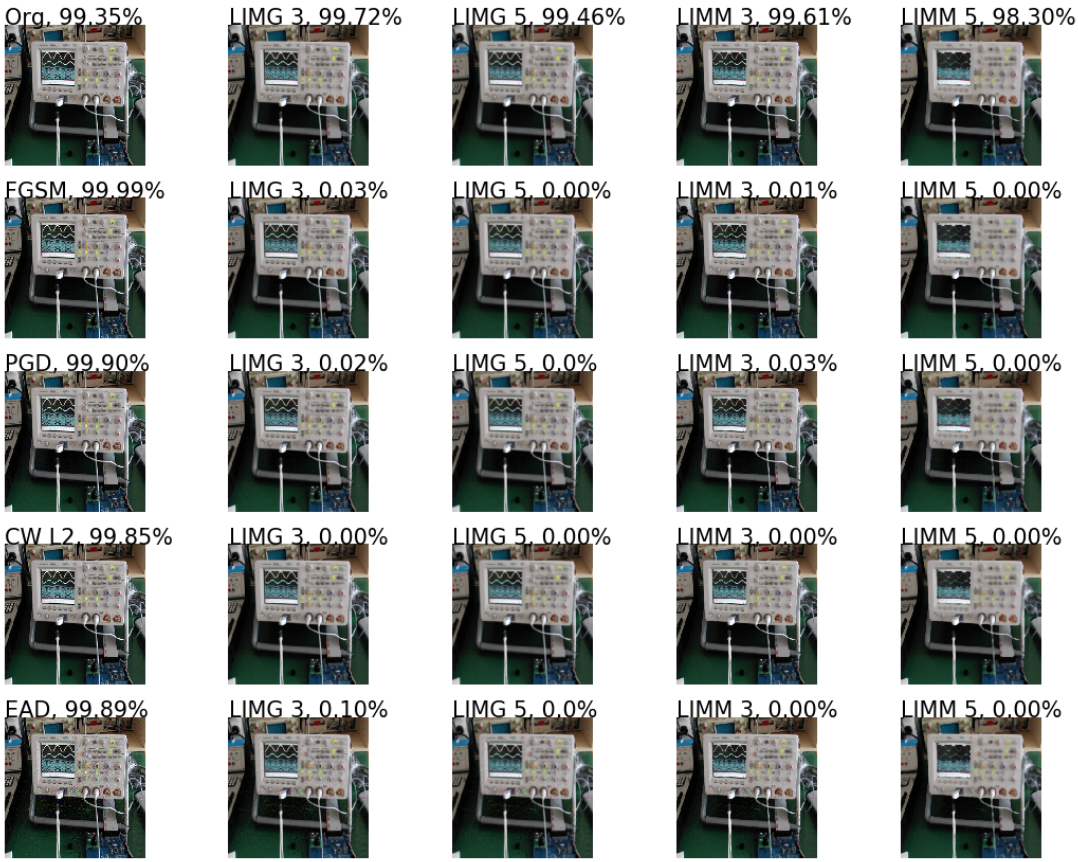


Figure 3-14: Our observation on original *oscilloscope* image and adversarial *ostrich* images that created by FGSM [31], PGD [70], CW_L_2 [14], and EAD [16] methods and when we use LIMG and LIMM method. First row, percentage values illustrate classification rates for the ground truth label. Other rows, percentage values are classification rates for the targeted label.

Figure 3-14 depicts the experimental results with the original image of *oscilloscope* and the targeted label of *ostrich* when the LIMG and LIMM methods are applied. In the first row, the percent values represent the probabilities of the oscilloscope label. In the other rows, the percent values show the probabilities of the ostrich label. Obviously, when the probability of the *oscilloscope* label decreases, the probability of the *ostrich* label increases and vice versa. Figure 3-14 shows both the LIMG and LIMM methods perfectly to remove the adversarial noises with any kernel sizes in Gaussian and Median filters.

Table 3.11: Detection Rate on MNIST dataset

	Our Method						Xu et al. [120]	
	GT	GR	GT&R	LIMG	LIMM	CMTD	Bit-D	Best-Joint
No. Files	2,000	2,000	2,000	2,000	2,000	2,000	2,000	2,000
Threshold	NA	NA	NA	NA	NA	NA	0.0005	0.0029
TP	1,767	1,786	1,775	1,786	1,776	1,799	-	-
TN	174	199	171	199	182	156	-	-
FP	26	1	29	1	18	44	-	-
FN	33	14	25	14	24	1	-	-
Accuracy	0.971	**0.993**	0.973	**0.993**	0.979	0.978	-	-
Precision	0.985	**0.999**	0.984	**0.999**	**0.999**	0.976	-	-
Recall	0.982	0.992	0.986	0.992	0.987	**0.999**	0.903	0.982
F1 score	0.984	0.996	**0.996**	**0.996**	0.988	0.988	-	-

Table 3.11 and 3.12 summarize the empirical results of our models with Xu et al [120]. On the MNIST dataset (Table 3.11), the CMTD method achieves the best detection accuracy rate at 99.9%. However, in general, GT&T and LIMG methods have the better results on F1 score at 99.6% to compare with 98.8% from CMTD method. On the ImageNet dataset (Table 3.12), the CMTD method also has the best performance in detection accuracy rate at 99.9%. But with F1 score, the GT&T method is slightly better than the CMTD method at 97.8%. In general, it is obvious to see that our proposed CMTD method (figure 3-7) has the best result in identifying adversarial images with the rate of 99.9% in both MNIST and ImageNet datasets. The '-' notation in the table means there is no information from Xu et al. The methods we propose do not need to set threshold parameters like Xu's work [120], so the installation and deployment are much faster and more efficient. In addition, we also evaluate our models with other indicators such as accuracy, precision, recall and F1 score that also reached the highest value of 99.6% on MNIST dataset and 97.8% on ImageNet dataset.

Table 3.12: Detection Rate on ImageNet dataset

	Our Method						Xu et al. [120]	
	GT	GR	GT&R	LIMG	LIMM	CMTD	Bit-D	Best-Joint
No. Files	1,800	1,800	1,800	1,800	1,800	1,800	1,800	1,800
Threshold	NA	NA	NA	NA	NA	NA	1.4417	1.2128
TP	1,362	1,390	1,419	1,395	1,434	1,439	-	-
TN	343	337	318	319	294	289	-	-
FP	17	23	42	41	66	71	-	-
FN	78	50	21	45	6	1	-	-
Accuracy	0.947	0.959	**0.995**	0.952	0.960	0.960	-	-
Precision	**0.988**	0.984	0.971	0.971	0.956	0.953	-	-
Recall	0.946	0.965	0.985	0.969	0.996	**0.999**	0.751	0.859
F1 score	0.966	0.974	**0.978**	0.970	0.976	0.976	-	-

3.3.4 Summary for combination of Image Transformation and Filters

In this section, we investigated the defensive strategies for the deep neural networks to the adversarial examples by combining image transform and filters. Based on the assumption and theory framework, we demonstrated the effectiveness of geometric transform and frequency domain in removing adversarial noises. From this observation, we proposed an automated detection system for adversarial examples. We obtained up to 99.9% detection rate on both MNNIST and ImageNet datasets. Our approach is different from related works that we observe the effectiveness of geometric transform and frequency changing on adversarial examples first. Afterward, based on our assumption, theory framework and observation we invented a powerful automatic identification system for distinguishing adversarial examples and legitimate images without threshold setup. Moreover, we explored the best parameters in image transformation and filters for removing the adversarial features. Another important

contribution of our work is that our proposed methods not only defeated adversarial noises but also in some cases our systems regained the ground truth label recognition.

3.4 Evaluating the robustness of adversarial perturbation against Image Filters

The superiority of deep learning performance is threatened by safety issues for itself. Recent findings have shown that deep learning systems are very weak to adversarial examples, an attack form that was altered by the attacker's intent to deceive the deep learning system. There are many proposed defensive methods to protect deep learning systems against adversarial examples. However, there is still lack of principal strategies to deceive those defensive methods. Any time a particular countermeasure is proposed, a new powerful adversarial attack will be invented to deceive that countermeasure. In this study, we focus on investigating the ability to create adversarial patterns in search space against defensive methods that use image filters. Experimental results conducted on the ImageNet dataset with image classification tasks showed the correlation between the search space of adversarial perturbation and filters. These findings open a new direction for building stronger offensive methods towards deep learning systems.

3.4.1 Search Space on Attacking Phase

We consider the white-box targeted attack settings, where the attacker can fully access the model type, model architecture, all trainable parameters, etc., and the adversary aims to change the classifier's prediction to some specific target class. The attackers use available information to identify the feature space where the model is vulnerable or try to find the victim decision boundaries. Then the victim model is exploited by altering a clean input by using adversarial example methods. To create adversarial samples that are misclassified by the machine learning model, an adversary with knowledge of the model's classifier f and its trainable parameters. In this work, we use FGSM [31] method for crafting adversarial examples. We define classifier function $f : \mathbb{R}^n \rightarrow \left[1...k\right]$ that maps image pixel value vectors to a particular label. Then we assume that function f has a loss function $L : \mathbb{R}^n \times \left[1...k\right] \rightarrow \mathbb{R}$. For

an input image $x \in \mathbb{R}^n$ and target label $y \in \Big[1...k\Big]$, our system aims to solve the following optimization problem: $\delta + L(x+\delta, y)$ subject to $x + \delta \in \Big[0,1\Big]^n$, where δ is an adversarial perturbation that we aim to add it to the original image x. We have to note that this function method would yield the solution for $f(x)$ in the case of convex losses, however, the neural networks are non-convex so we end up with an approximation in this case. In this case, we use the output of the second-to-last layer logits for calculating the gradient instead of using the output of the softmax function. Our main purpose to decide the size of adversarial perturbation, it means the search space of adversarial perturbation. We consider the norm operation to determine the size of the adversarial noises. Mathematically, a norm is a total size or length of all vectors in a vector space or matrices. For simplicity, we can say that the higher the norm is, the bigger tha value matrix or vector. Formally, the l_p-norm of vector x is defined as: $\|x\|_p = \sqrt[p]{\sum_i |x_i|^p}$, where $p \in \mathbb{R}$. This is a p^{th}-root of a summation of all elements to the p^{th} power is what we call a norm. The interesting point is even though every l_p-norm is all look very similar to each other, their mathematically properties are very different and thus their application are dramatically different when we use to create the adversarial examples. In this work, we consider three common norm methods: l_1-norm, l_2-norm, and l_∞-norm for evaluating the size of the search space of adversarial perturbation.

$\mathbf{l_1}$-norm. From the definition of l_p-norm, l_1-norm of x is defined as:

$$\|x\|_1 = \sum_i |x_i| \tag{3.31}$$

This norm is quite common among the norm family. It has many name and many forms among various fields, namely Manhattan norm.

$\mathbf{l_2}$-norm. The most popular of all norm is the l_2-norm. It is used in almost everyy field of engineering and science as a whole. l_2-norm is defined as:

$$\|x\|_2 = \sqrt{\sum_i x_i^2} \tag{3.32}$$

l_2-norm is well known as a Euclidean norm, which is used as a standard quantity for measuring a vector difference. If the Euclidean norm is computed for a vector difference, it is known as a Euclidean distance:

$$\|x_1 - x_2\|_2 = \sqrt{\sum_i (x_1^i - x_2^i)^2} \tag{3.33}$$

l_∞-norm. The l_∞-norm is defined as:

$$\|x\|_\infty = \sqrt[\infty]{\sum_i x_i^\infty} \tag{3.34}$$

Let consider the vector X, if x_i is the highest element in vector X, by the property of the infinity itself, we have: $x_i^\infty \approx x_k^\infty \forall i \neq k$, then $\sum_i x_i^\infty = x_k^\infty$. And we have $\|x\|_\infty = \sqrt[\infty]{\sum_i x_i^\infty} = \sqrt[\infty]{x_k^\infty} = |x_k|$. Now we have simple definition of l_∞-norm as: $\|x\|_\infty = max(|x_i|)$.

So our attack phase is denoted as Algorithm 5 by using FGSM. Where x defines the original input, y_{true} defines the ground truth label, y^* is an adversarial label, f is the activation function of machine learning model, ϵ is the maximum adversarial value, l_i defines the norm. In the attacking phase, we set the learning rate for crafting adversarial examples to 0.01 to keeps adversarial noises are as small as possible and the iterative process is repeated 500 times. From the clean images, we will create the targeted output images.

3.4.2 Implementation

The classification task was evaluated on the ImageNet benchmark dataset. We consider the ImageNet dataset [96] that is a very large database designed for use in visual object recognition research. The original ImageNet includes more than 14 million images in 20,000 categories with a typical category, such as "vending machine" or "street sign", consisting of several hundred images. The machine learning model that we use is Google Inception V3 [108] that was trained with 1,000 common categories ImageNet. We randomly select some testing images. By applying FGSM with l_1-norm,

Algorithm 5: Crafting Adversarial Examples with l-norm optimization

input : x, y_{true}, y^*, f, ϵ, l_i
output : x^*
parameter: learning rate = 0.01, epochs = 500

$x \leftarrow x^*$ `// initial adversarial sample`
$\delta_x \leftarrow \vec{0}$ `// initial perturbation factor`
$iter \leftarrow 1$ `// initial iteration counter`
while $\delta_x < \epsilon$ *and* $f(x^*) \neq y^*$ *and* $iter <= epochs$ **do**
    $x^* \leftarrow x + \delta \cdot sign(\bigtriangledown L(y^*|x^*))$
    $\delta_x \leftarrow \text{norm}(l_i)$
    maximize $Loss(y^*|x^*)$ respect to δ_x
    $\delta \leftarrow clip(x^*, x - \epsilon, x + \epsilon)$
    $iter \leftarrow iter + 1$
end
return x^*

l_2-norm, and l_∞-norm to craft adversarial images target to randomly targeted labels.

3.4.3 Results

Intuitively, we use our own images (include pictures of vending machine, computer mouse and keyboard) and an image of oscilloscope from ImageNet dataset for analysis. We randomly selected targeted labels for the creation of adversarial images. By using the FGSM method in combination with l_1-norm, l_2-norm, and l_∞-norm, from each original image we create three different adversarial images.

Figure 3-15 shows the results of creating adversarial images from the original image of the oscilloscope. We find that the deep learning system is easily fooled with adversarial images. We observe the changing of probabilities of the original label before and after using Gaussian and Median filters. In addition, we intuitively observe that adversarial images created with l_1-norm and l_2-norm are easier to defeat than l_∞-norm. With Gaussian kernel size 3×3, the probability of original label is still equal to 0 with adversarial l_∞-norm. It means that Gaussian kernel size 3×3 can not remove the adversarial noises in this case.

Figure 3-16 shows the implementation results with the observation on the probabilities of adversarial label. The results demonstrate the probabilities of adversarial

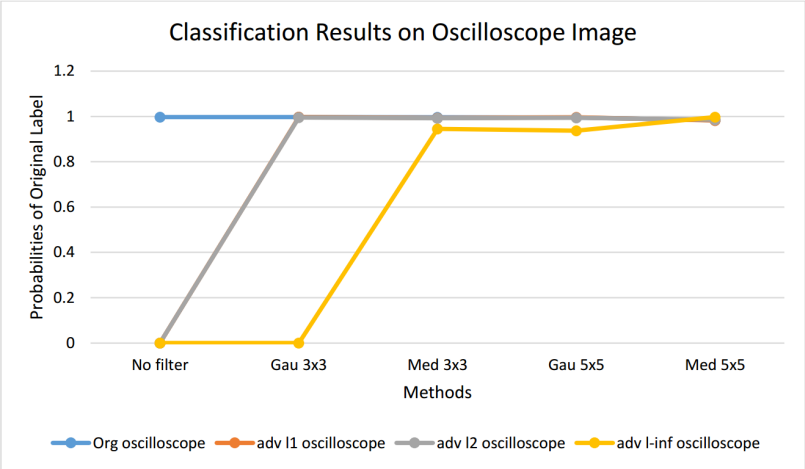


Figure 3-15: Classification Results on Oscilloscope Image with observation on the probabilities of Original Label

label drop from nearly 100% to 0% when we use both Gaussian and Median filter exclude Gaussian kernel size 3×3 with adversarial l_∞-norm.

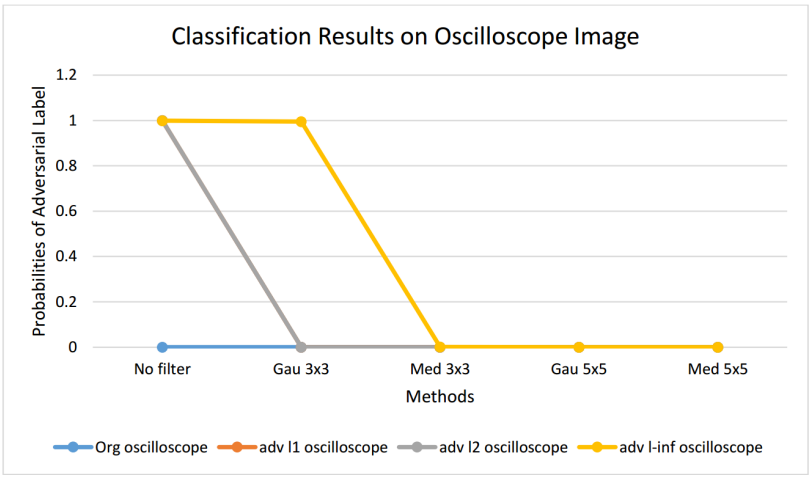


Figure 3-16: Classification Results on Oscilloscope Image with observation on the probabilities of Adversarial Label

Figure 3-17 and Figure 3-18 show the implementation results with image of vending machine. The adversarial vending machine l_∞-norm is the strongest adversarial when both Gaussian and Median can not regain the probabilities of original label.

Figure 3-19 shows the implementation results with images of computer mouse and computer keyboard.

Figure 3-20 shows the results of creating adversarial images from the original

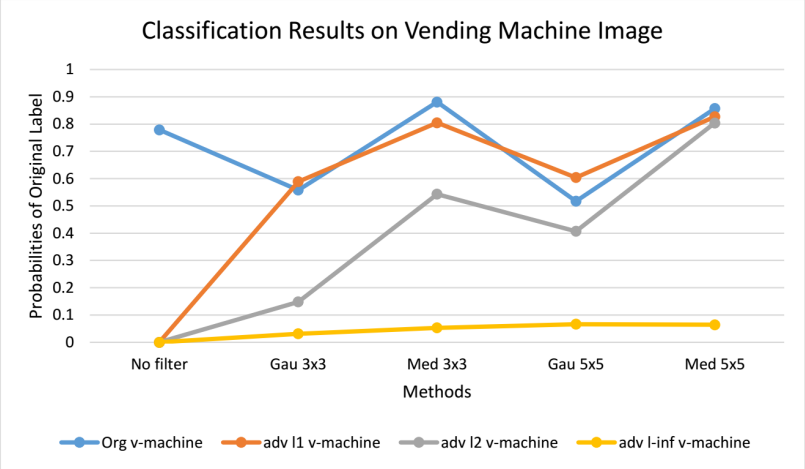


Figure 3-17: Classification Results on Vending Machine Image with observation on the probabilities of Original Label

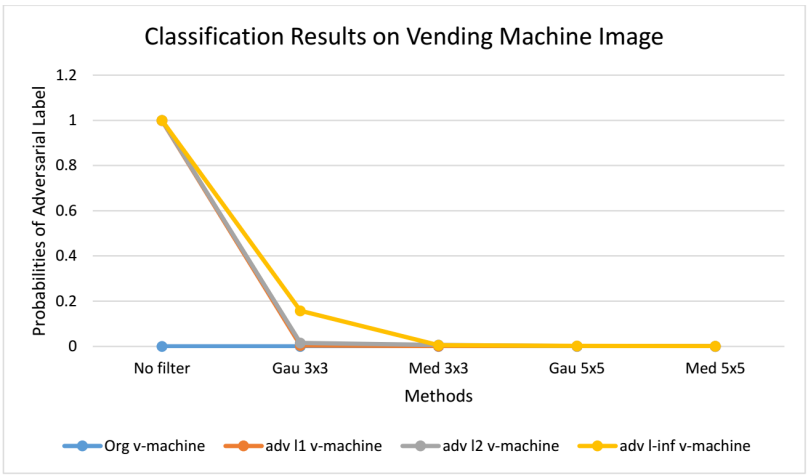
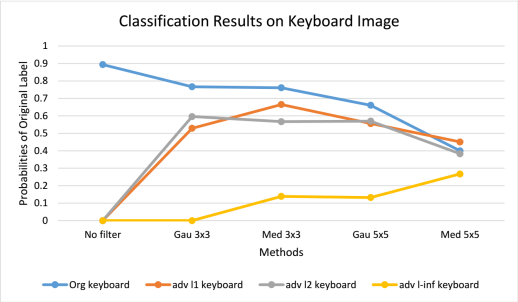


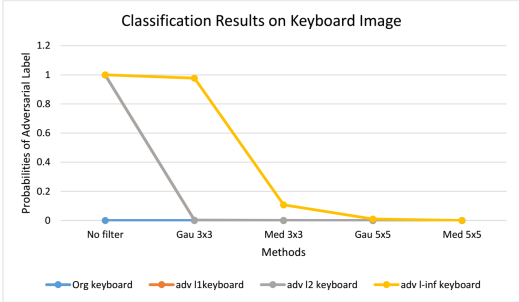
Figure 3-18: Classification Results on Vending Machine Image with observation on the probabilities of Adversarial Label

image of the oscilloscope. We find that the deep learning system is easily fooled with adversarial images. In addition, we intuitively observe that adversarial images created with l_1-norm and l_2-norm are harder to detect than l_∞-norm.

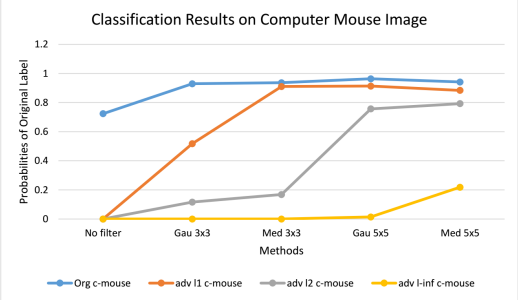
Figure 3-21 shows the experimental results when we use the image filters method on the original image of the oscilloscope. We find that the Gaussian filter reduces classification accuracy more than the median filter. Especially in the case of the median with size filter 3×3 and 5×5, the classification results are better than the original image.



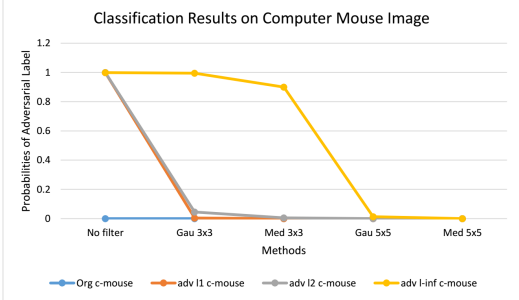
(a) Observation on the probabilities of Original Keyboard Label



(b) Observation on the probabilities of Adversarial Label



(c) Observation on the probabilities of Original Computer Mouse Label

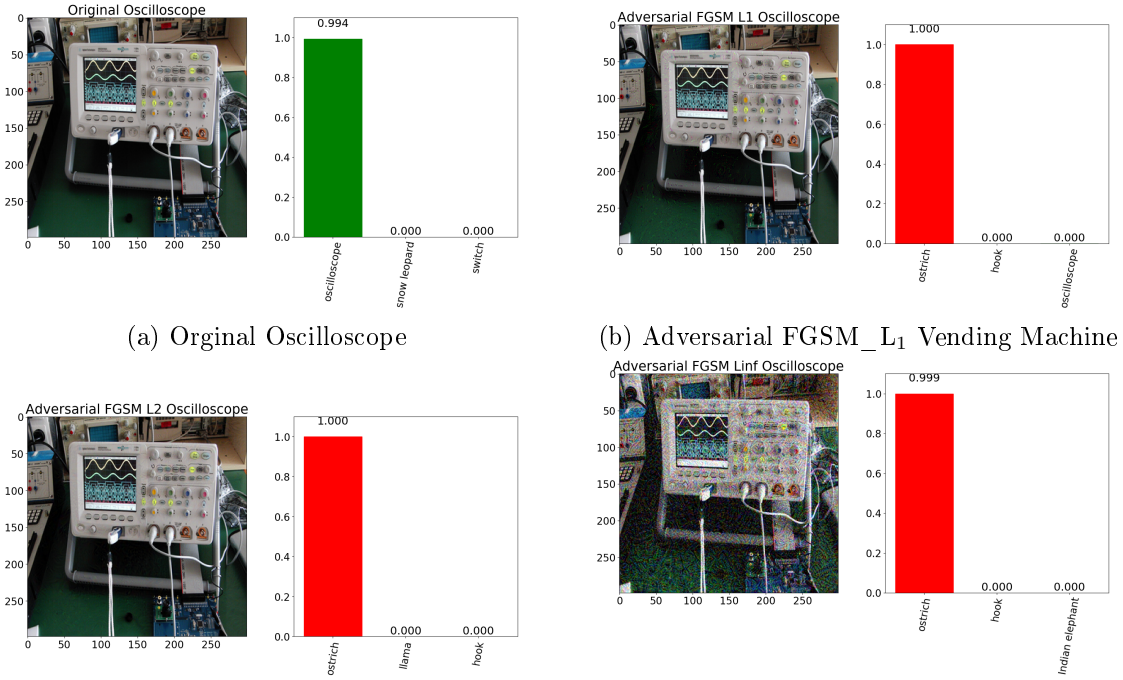


(d) Observation on the probabilities of Adversarial Label

Figure 3-19: Classification Results on Keyboard and Computer Mouse Images

Similar to the original image, we also apply image filter methods to adversarial images. Figure 3-22 shows classification results on adversarial images created by the FGSM method in combination with l_1-norm. Figure 3-23 illustrates classification results on adversarial images created by the FGSM method in combination with l_2-norm. We observed that Gaussian kernel size 3x3 could not restore identity to ground truth label on adversarial image with l_2-norm. The probability for vending machine label is only 14.8%. Meanwhile, the median filter still works effectively in removing adversarial noises. Figure 3-24 shows classification results on adversarial images created by the FGSM method in combination with l_∞-norm. We observed that Gaussian kernel size 3×3 could not eliminate the effect of adversarial noise with l_∞-norm on deep learning system classification. Gaussian 5×5 gives better results, but the label with the highest probability of identification is "tabacco shop". The Median filter removes adversarial noises but cannot help the deep learning system correctly identify ground truth labels.

Table 3.13 shows experimental results on oscilloscope, vending machine (v-machine),



(a) Orginal Oscilloscope

(b) Adversarial FGSM_L_1 Vending Machine

(c) Adversarial FGSM_L_2 Vending Machine

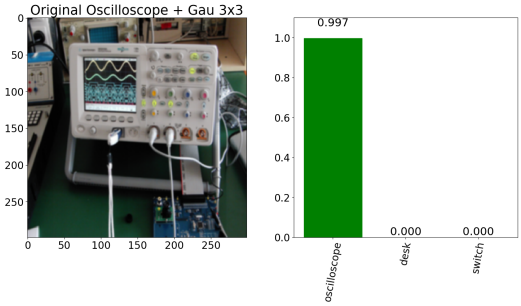
(d) Adversarial FGSM_L_∞ Vending Machine

Figure 3-20: Adversarial Oscilloscope (targeted class: Ostrich Sign)

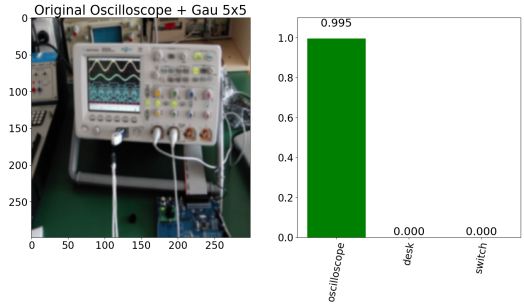
computer mouse (c-mouse) and keyboard sets. This result shows us a large correlation between norm operations in search space of adversarial examples. It is clear that for the l_∞-norm, the Gaussian (3×3, 5×5) and the Median (3×3) methods are more difficult to completely eliminate adversarial noises based on the l_1 and l_2 norm. Median (5×5) still proved superior in removing adversarial noises in all settings.

3.4.4 Summary for Evaluating the robustness of adversarial perturbation against Image Filters

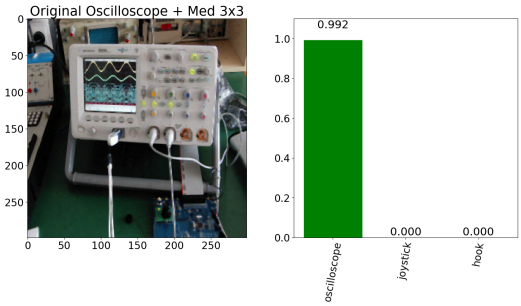
In this section, we focus on investigating the connection between the search space of adversarial examples and the defense based on the frequency domain. Our empirical results demonstrate that the FGSM method in combination with l_∞-norm produces the strongest adversarial examples. In this case, both the Gaussian and the Median filters are unable to restore identification to the ground truth label. However, when using l_∞-norm to create adversarial examples, we also significantly reduce the quality of the original image compared to using l_1 and l_2 norm. In terms of similarities with



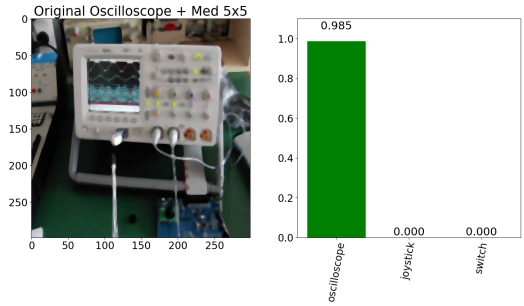
(a) Orginal Oscilloscope with Gaussian (3×3)



(b) Orginal Oscilloscope with Gaussian (5×5)



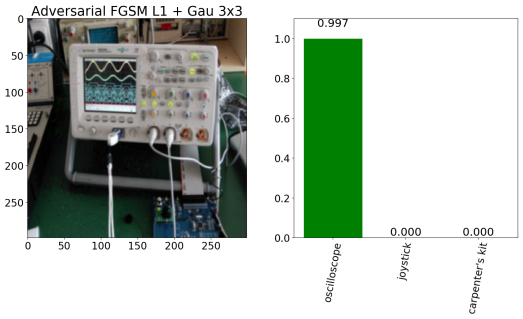
(c) Orginal Oscilloscope with Median (3×3)



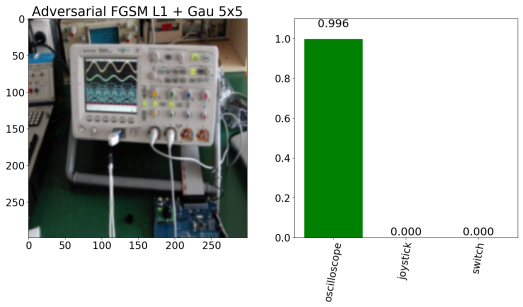
(d) Orginal Oscilloscope with Median (5×5)

Figure 3-21: Original Oscilloscope with Image Filters

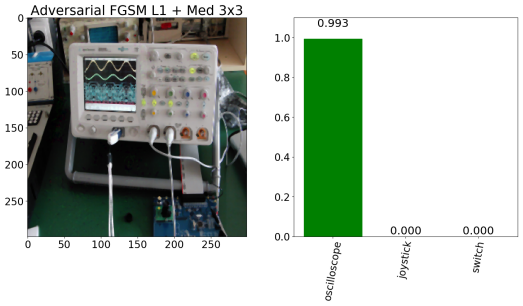
the original image, l_1 and l_2 norm produce much better adversarial examples than l_∞ norm.



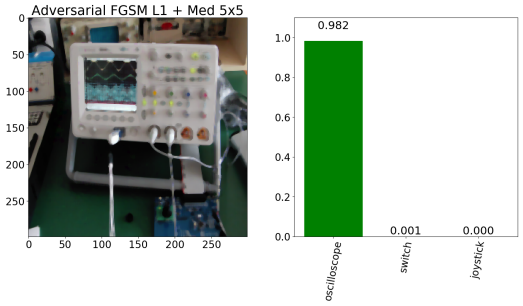
(a) Adversarial Oscilloscope with Gaussian (3×3)



(b) Adversarial Oscilloscope with Gaussian (5×5)



(c) Adversarial Oscilloscope with Median (3×3)

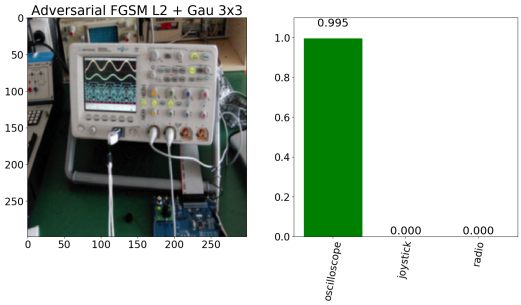


(d) Adversarial Oscilloscope with Median (5×5)

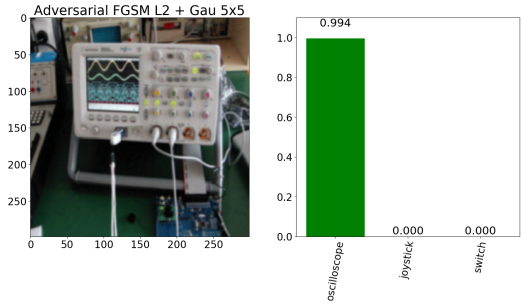
Figure 3-22: Adversarial FGSM L_1 Oscilloscope (targeted class: Ostrich) with Image Filters

Table 3.13: Implementation Results

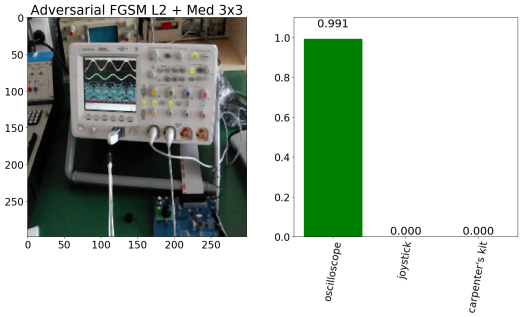
Input	No filter		Gau 3x3		Med 3x3		Gau 5x5		Med 5x5	
	OL	AL	OL	AL	OL	AL	OL	AL	OL	AL
Org oscilloscope	0.997	0	0.997	0	0.996	0	0.995	0	0.983	0
adv l_1 oscilloscope	0	1	0.997	0	0.993	0	0.996	0	0.982	0
adv l_2 oscilloscope	0	1	0.995	0	0.992	0	0.994	0	0.985	0
adv l_∞ oscilloscope	0	1	0	**0.995**	0.945	**0.001**	0.937	0	0.997	0
Org v-machine	0.779	0	0.558	0	0.881	0	0.517	0	0.857	0
adv l_1 v-machine	0	0.999	0.589	0.004	0.805	0.001	0.604	0.001	0.827	0
adv l_2 v-machine	0	1	0.148	0.015	0.543	**0.006**	0.407	0.001	0.804	0
adv l_∞ v-machine	0	1	0.031	**0.157**	0.053	0.005	0.066	**0.002**	0.064	0
Org keyboard	0.894	0	0.767	0	0.761	0	0.661	0	0.4	0
adv l_1 keyboard	0	0.999	0.529	0.002	0.665	0	0.556	0	0.451	0
adv l_2 keyboard	0	0.999	0.596	0.002	0.567	0.001	0.57	0	0.383	0
adv l_∞ keyboard	0	1	0	**0.977**	0.139	**0.107**	0.132	**0.01**	0.267	0
Org c-mouse	0.724	0	0.93	0	0.937	0	0.964	0	0.924	0
adv l_1 c-mouse	0	0.999	0.518	0.004	0.911	0	0.914	0	0.884	0
adv l_2 c-mouse	0	1	0.116	0.045	0.168	0.005	0.757	0	0.793	0
adv l_∞ c-mouse	0	0.999	0	**0.995**	0	**0.9**	0.014	**0.013**	0.218	0



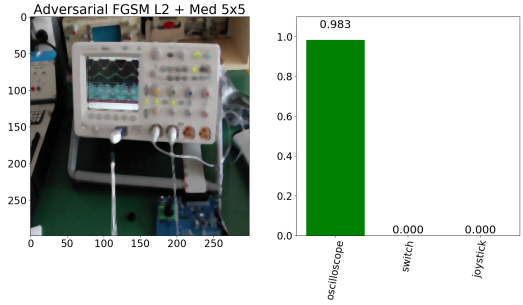
(a) Adversarial Oscilloscope with Gaussian (3×3)



(b) Adversarial Oscilloscope with Gaussian (5×5)

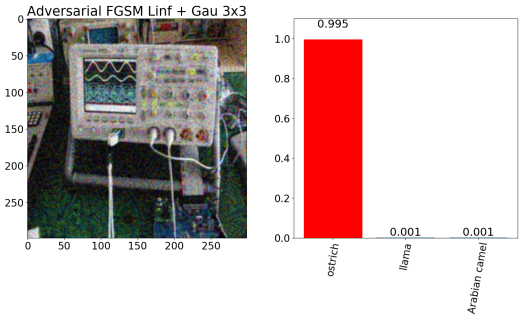


(c) Adversarial Oscilloscope with Median (3×3)

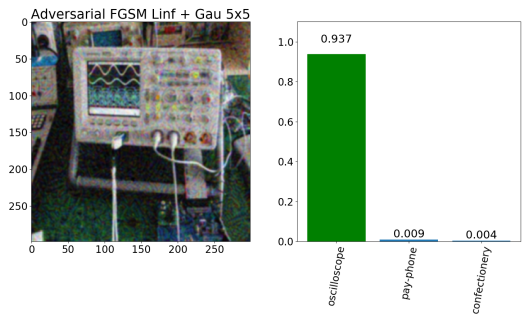


(d) Adversarial Oscilloscope with Median (5×5)

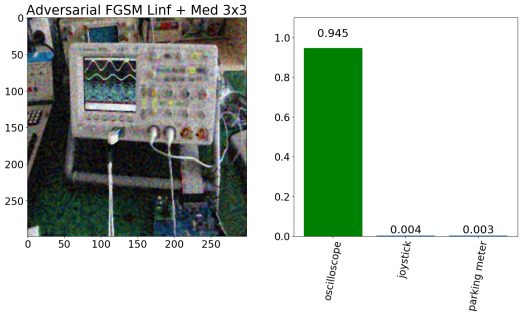
Figure 3-23: Adversarial FGSM L_2 Oscilloscope (targeted class: Ostrich) with Image Filters



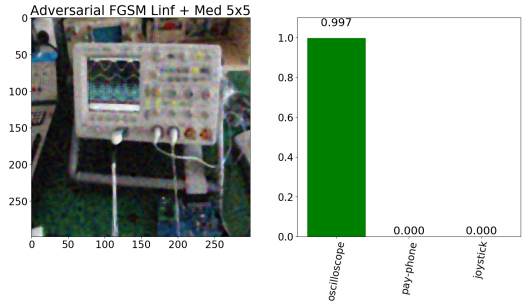
(a) Adversarial Oscilloscope with Gaussian (3×3)



(b) Adversarial Oscilloscope with Gaussian (5×5)



(c) Adversarial Oscilloscope with Median (3×3)



(d) Adversarial Oscilloscope with Median (5×5)

Figure 3-24: Adversarial FGSM L_∞ Oscilloscope (targeted class: Ostrich) with Image Filters

Chapter 4

Conclusion and Future Works

4.1 Conclusion

Adversarial examples (AE) is emerging as one of the key topics in the field of security for AI-based systems. With very little modification of the input data, AE can deceive and completely change the prediction of the output from the AI system while the human is not able to realize the difference between AE and the original data. AE is not only exploited on image classification systems but also appears in natural language processing systems, speech recognition or object detection systems. Because of these major security issues of AE, I focus on how to make the AI system more secure, able to identify and distinguish between adversarial and legitimate patterns. My research mainly focuses on building detection and distinguishing systems between AE and legitimate images in image classification tasks. My main research directions include: (1) AE detection system based on the spatial domain; (2) AE detection system based on the frequency domain; (3) Evaluation of the robustness of AE against image filters.

The first is the spatial domain. Image processing systems based on deep learning such as image classification, image segmentation or object recognition are often trained to overcome changes in spatial structure by using data augmentation techniques. In addition, those deep learning systems are designed with multiple layers with multiple operations and different settings to help the system learn more important features. These properties help deep learning systems can produce very impressive

results in their own tasks. However, most of the training process in deep learning systems assumes that the data distribution is constant and it only focuses on fine-tuning the system trainable parameters and weights. This makes it possible for attackers to take advantage of AEs based on reusing or re-simulating parameters and deep learning structure and then editing them toward to a desired target. It is important that the attack methods are based on the loss function optimization and are not concerned with the spatial structure of the image. However deep learning systems have been trained to work with spatial variation. Based on this analysis, I take advantage of the spatial weaknesses of AE to build an AE identification system based on affine transformations. Our detection system based on affine transformation (see Table. 3.8 on MNIST dataset, Table. 3.7 on ImageNet dataset) mostly defeat many state-of-the-art AE attack methods.

The second one is the frequency domain. During the creation of the AE, the attack methods unintentionally created pixel values that far different from the neighboring areas. Therefore, AEs have been unintentionally created on the high digital currency in the image. Based on this observation and analysis, we design and build automated detection systems (see Table 3.9 for MNIST dataset and 3.10 for ImageNet dataset) for identifying AE and legitimate images based on the frequency domain. And the experimental results have proved my system to produce very satisfactory results on popular data sets such as MNIST and ImageNet datasets. Moreover, the combination of transformation and filter achieve the best detection results to compare with other methods (Table 3.11 for MNIST dataset and 3.12 for ImageNet dataset). From the implementation results, we found that the combination of transformation and filters (the CMTD method - see Figure 3-7) achieves the best detection accuracy rate in both MNIST and ImageNet datasets.

The third one is how to evaluate the robustness of AE against image filters. In the context of adversarial inputs at test time, we observed several effective attack algorithms but few strong countermeasures. Any time a particular countermeasure is invented, a new strong and powerful attack will be proposed to devastate that countermeasure. We can explain adversarial examples in current machine learning models

as the result of unreasonably linear extrapolation but do not know what will happen when we fix this particular problem; it may simply be replaced by another equally annoying category of vulnerabilities. The vastness of the set of all possible inputs to a machine learning model seems to be cause for pessimism. These questions may be addressed empirically, by actually playing out the competition between new attacks and new countermeasures are developed. So there is a very important question that is how to measure the robustness of a machine learning algorithm. If we can answer this question, we confidence to determine a particular machine learning algorithm is strong or weak to AE. We evaluated the robustness of AE to a defensive system based on the frequency domain. We investigated the FGSM method in combination with l_1, l_2 and l_∞-norm to create AE. After that, we evaluate the robustness of the AE against image filters. The implementation results (Table 3.13) show the AE with l_∞-norm is the strongest AE but image filters (Gaussian and Median) still defeat most of AE noises in many settings.

4.2 Future Works

Adversarial machine learning is at a turning point. In the context of adversarial inputs at test time, we observed several effective attack algorithms but few strong countermeasures. Any time a particular countermeasure is invented, a new strong and powerful attack will be proposed to devastate that countermeasure. Can we expect this situation to continue indefinitely? Or can we expect the defender to eventually gain a fundamental advantage? We can explain adversarial examples in current machine learning models as the result of unreasonably linear extrapolation but do not know what will happen when we fix this particular problem; it may simply be replaced by another equally annoying category of vulnerabilities. The vastness of the set of all possible inputs to a machine learning model seems to be cause for pessimism. On the other hand, one may hope that as classifiers become more robust, it could become impractical for an attacker to find input points that are reliably misclassified by the target model, particularly in the black-box setting. These questions may be

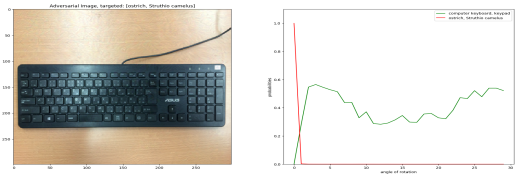
addressed empirically, by actually playing out the competition between new attacks and new countermeasures are developed. From our proposed countermeasure by using image transformation and filtering, there are still unclear that will not have new attacks to overcome them. So far, almost adversarial attacks and defense strategies were proposed based heavily on heuristic and lack of concrete theory. Besides, the gap between machine learning capability and the human brain still remains very large. An arbitrary picture that humans may be easily recognized and give the right answer but it still keeps challenges for a recognition system from different viewpoints, illustrations, luminance, and noises. So there is a very important question that is how to measure the robustness of a machine learning algorithm. If you can answer this question, you can confidence to determine a particular machine learning algorithm is strong or weak to adversarial example. Let consider a multi-classification classifier with K classes and d features where one has a classification function $f : \mathbb{R}^d \rightarrow \mathbb{R}^K$ and a sample x is classified via $f(x)$. We call a classifier robust at sample x if with small changes of the original input do not alter the final decision. Suppose that the classifier outputs class c for input x, that is $f_x(x) > f_j(x)$ for $j \neq c$. The problem of generating an adversarial sample $x + \delta$ such that the classifier decision changes, can be formulated as:

$\underset{\delta \in \mathbb{R}^d}{min}$, s.th. $\underset{l \neq c}{max} f_l(x + \delta) \geq f_c(x + \delta)$ and $x + \delta \in C$,

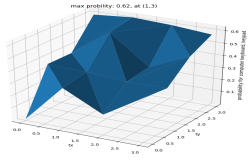
where C is a constraint set specifying certain requirements on the adversarial sample $x+\delta$. However, in computer vision, with small unintended changes $x+\theta$ should not change the output classification. So the idea here to evaluate the robustness of the machine learning system is that how to calculate the probability Ψ that the value $\delta \notin \theta$. If a machine learning system has a large probability Ψ that means it is a strong system to adversarial examples and vice versa.

Appendix A

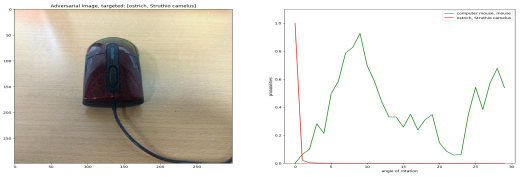
Figures



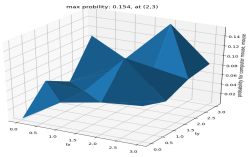
(a) Coordinate Transformation on Adversarial "Keyboard" Image (targeted class is ostrich)



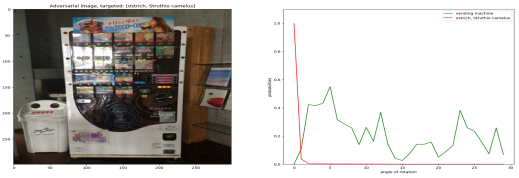
(b) Geometric Translation on Adversarial "Keyboard" Image (targeted class is ostrich)



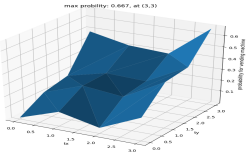
(c) Coordinate Transformation on Adversarial "C-Mouse" Image (targeted class is ostrich)



(d) Geometric Translation on Adversarial "C-Mouse" Image (targeted class is ostrich)

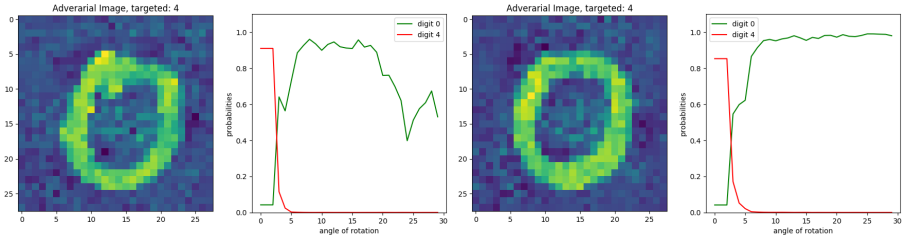


(e) Coordinate Transformation on Adversarial "V-Machine" Image (targeted class is ostrich)



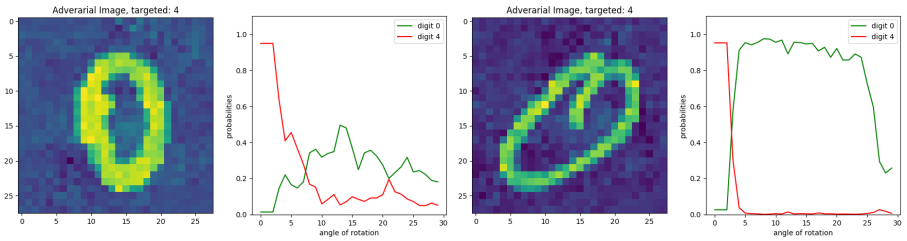
(f) Geometric Translation on Adversarial "V-Machine" Image (targeted class is ostrich)

Figure A-1: Coordinate transformation - Geometric translation on the Adversarial images ("Keyboard", "C-Mouse", "V-Machine").



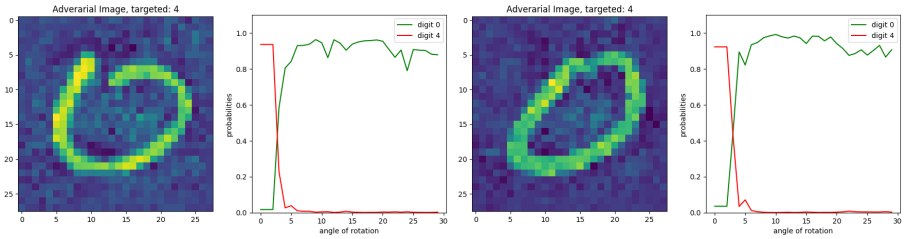
(a) digit 0 - index: 4457

(b) digit 0 - index: 5584



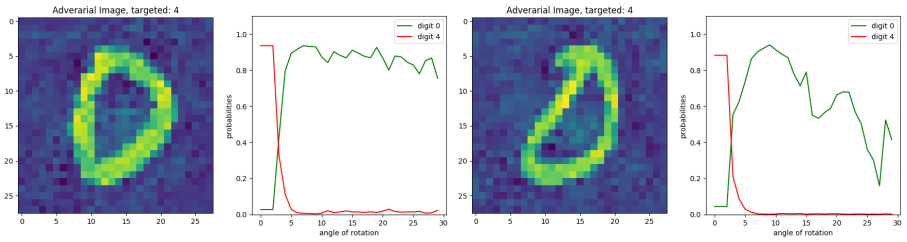
(c) digit 0 - index: 1438

(d) digit 0 - index: 1558



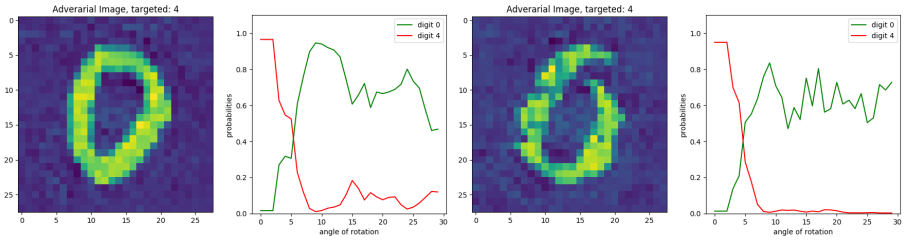
(e) digit 0 - index: 7031

(f) digit 0 - index: 9962



(g) digit 0 - index: 297

(h) digit 0 - index: 5259



(i) digit 0 - index: 4070

(j) digit 0 - index: 4515

Figure A-2: MNIST - Coordinate Transformation with $\alpha \in [0, 30]$. Index numbers identify the index of images in MNIST training dataset. The figure illustrates the adversarial perturbation is defeated by Coordinate Transformation in all experiments.

Bibliography

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

[2] Naveed Akhtar, Jian Liu, and Ajmal Mian. Defense against universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3389–3398, 2018.

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

[4] Hyrum S Anderson, Jonathan Woodbridge, and Bobby Filar. Deepdga: Adversarially-tuned domain generation and detection. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, pages 13–21. ACM, 2016.

[5] Anurag Arnab, Ondrej Miksik, and Philip HS Torr. On the robustness of semantic segmentation models to adversarial attacks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition CVPR*, pages 888–897, 2018.

[6] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. Drebin: Effective and explainable detection of android malware in your pocket. In *Ndss*, volume 14, pages 23–26, 2014.

[7] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International Conference on Machine Learning*, pages 284–293, 2018.

[8] Yonatan Belinkov and James Glass. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72, 2019.

[9] Arjun Nitin Bhagoji, Daniel Cullina, Chawin Sitawarin, and Prateek Mittal. Enhancing robustness of machine learning systems via data transformations. In *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, pages 1–5. IEEE, 2018.

[10] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

[11] Battista Biggio, B Nelson, and P Laskov. Poisoning attacks against support vector machines. In *29th International Conference on Machine Learning ICML*, pages 1807–1814. ArXiv e-prints, 2012.

[12] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations ICLR*, 2018.

[13] 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*, pages 3–14, 2017.

[14] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (S&P 2017)*, pages 39–57. IEEE, 2017.

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

[16] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[17] Tarang Chugh, Kai Cao, and Anil K Jain. Fingerprint spoof buster: Use of minutiae-centered patches. *IEEE Transactions on Information Forensics and Security*, 13(9):2190–2202, 2018.

[18] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International Conference on Machine Learning (ICML)*, 2018.

[19] Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, Deepak Verma, et al. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108. ACM, 2004.

[20] Guneet S. Dhillon, Kamyar Azizzadenesheli, Jeremy D. Bernstein, Jean Kossaifi, Aran Khanna, Zachary C. Lipton, and Animashree Anandkumar. Stochastic activation pruning for robust adversarial defense. In *International Conference on Learning Representations ICLR*, 2018.

[21] Harris Drucker and Yann Le Cun. Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*, 3(6):991–997, 1992.

[22] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. A study of the effect of jpg compression on adversarial images. In *the International Society for Bayesian Analysis (ISBA 2016) World Meeting*, 2016.

[23] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, 2018.

[24] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition CVPR*, 2018.

[25] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

[26] Ugo Fiore, Alfredo De Santis, Francesca Perla, Paolo Zanetti, and Francesco Palmieri. Using networks for improving classification effectiveness in credit card fraud detection. *Information Sciences*, 2017.

[27] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE, 2018.

[28] Thomas Gebhart and Paul Schrater. Adversary detection in neural networks via persistent homology. *arXiv preprint arXiv:1711.10056*, 2017.

[29] Justin Gilmer, Luke Metz, Fartash Faghri, Sam Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres. In *International Conference on Learning Representations ICLR*, 2018.

[30] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[31] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations ICLR*, 2015.

[32] Gaurav Goswami, Nalini Ratha, Akshay Agarwal, Richa Singh, and Mayank Vatsa. Unravelling robustness of deep learning based face recognition against adversarial attacks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[33] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

[34] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial perturbations against deep neural networks for malware classification. *arXiv preprint arXiv:1606.04435*, 2016.

[35] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial examples for malware detection. In *European Symposium on Research in Computer Security*, pages 62–79. Springer, 2017.

[36] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. In *International Conference on Learning Representations ICLR*, 2018.

[37] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

[38] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. Adversarial example defense: Ensembles of weak defenses are not strong. In *11th {USENIX} Workshop on Offensive Technologies ({WOOT} 17)*, 2017.

[39] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.

[40] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[41] Sungeun Hong, Woobin Im, Jongbin Ryu, and Hyun S Yang. Sspp-dan: Deep domain adaptation network for face recognition with single sample per person. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 825–829. IEEE, 2017.

[42] Hossein Hosseini, Yize Chen, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. Blocking transferability of adversarial examples in black-box learning systems. *arXiv preprint arXiv:1703.04318*, 2017.

[43] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning ICML*, pages 2142–2151, 2018.

[44] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems NeurIPS 2015*, pages 2017–2025, 2015.

[45] Guoqing Jin, Shiwei Shen, Dongming Zhang, Feng Dai, and Yongdong Zhang. Ape-gan: Adversarial perturbation elimination with gan. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3842–3846. IEEE, 2019.

[46] Jinkyu Kim and John Canny. Interpretable learning for self-driving cars by visualizing causal attention. In *Proceedings of the IEEE international conference on computer vision ICCV*, pages 2942–2950, 2017.

[47] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[48] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1885–1894. JMLR. org, 2017.

[49] Jascha Kolberg, Marta Gomez-Barrero, Sushma Venkatesh, Raghavendra Ramachandra, and Christoph Busch. Presentation attack detection for finger recognition. In *Handbook of Vascular Biometrics*, pages 435–463. Springer, 2020.

[50] Jernej Kos, Ian Fischer, and Dawn Song. Adversarial examples for generative models. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 36–42. IEEE, 2018.

[51] Rajesh Kumar, Zhang Xiaosong, Riaz Ullah Khan, Jay Kumar, and Ijaz Ahad. Effective and explainable detection of android malware based on machine learning algorithms. In *Proceedings of the 2018 International Conference on Computing and Artificial Intelligence*, pages 35–40. ACM, 2018.

[52] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *International Conference on Learning Representations ICLR*, 2017.

[53] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations ICLR*, 2017.

[54] Ruggero Donida Labati, Enrique Muñoz, Vincenzo Piuri, Roberto Sassi, and Fabio Scotti. Deep-ecg: Convolutional neural networks for ecg biometric recognition. *Pattern Recognition Letters*, 126:78–85, 2019.

[55] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[56] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[57] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

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

[59] J Li, S Ji, T Du, B Li, and T Wang. Textbugger: Generating adversarial text against real-world applications. In *26th Annual Network and Distributed System Security Symposium*, 2019.

[60] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4208–4215. AAAI Press, 2018.

[61] Bin Liang, Hongcheng Li, Miaoqiang Su, Xirong Li, Wenchang Shi, and Xiaofeng Wang. Detecting adversarial image examples in deep networks with adaptive noise reduction. *arXiv preprint arXiv:1705.08378*, 2017.

[62] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition CVPR*, pages 1778–1787, 2018.

[63] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3756–3762. AAAI Press, 2017.

[64] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *International Conference on Learning Representations ICLR*, 2017.

[65] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision CVPR*, pages 3730–3738, 2015.

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

[67] Jiajun Lu, Hussein Sibai, Evan Fabry, and David Forsyth. No need to worry about adversarial examples in object detection in autonomous vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition CVPR*, 2018.

[68] Chunchuan Lyu, Kaizhu Huang, and Hai-Ning Liang. A unified gradient regularization family for adversarial examples. In *2015 IEEE International Conference on Data Mining*, pages 301–309. IEEE, 2015.

[69] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[70] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[71] 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*, pages 135–147. ACM, 2017.

[72] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. In *International Conference on Learning Representations (ICLR)*, 2017.

[73] Jan Hendrik Metzen, Mummadi Chaithanya Kumar, Thomas Brox, and Volker Fischer. Universal adversarial perturbations against semantic image segmentation. In *Proceedings of the IEEE International Conference on Computer Vision ICCV*, pages 2774–2783. IEEE, 2017.

[74] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[75] Vahid Mirjalili and Arun Ross. Soft biometric privacy: Retaining biometric utility of face images while perturbing gender. In *2017 IEEE International joint conference on biometrics (IJCB)*, pages 564–573. IEEE, 2017.

[76] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. Adversarial training methods for semi-supervised text classification. In *International Conference on Learning Representations ICLR*, 2017.

[77] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[78] Andreas Mogelmose, Mohan Manubhai Trivedi, and Thomas B Moeslund. Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1484–1497, 2012.

[79] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5115–5124, 2017.

[80] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition CVPR*, pages 1765–1773, 2017.

[81] 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 CVPR*, pages 2574–2582, 2016.

[82] James Newsome, Brad Karp, and Dawn Song. Polygraph: Automatically generating signatures for polymorphic worms. In *2005 IEEE Symposium on Security and Privacy (S&P'05)*, pages 226–241. IEEE, 2005.

[83] Linh Nguyen, Sky Wang, and Arunesh Sinha. A learning and masking approach to secure learning. In *International Conference on Decision and Game Theory for Security*, pages 453–464. Springer, 2018.

[84] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

[85] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[86] Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. Crafting adversarial input sequences for recurrent neural networks. In *MILCOM 2016-2016 IEEE Military Communications Conference*, pages 49–54. IEEE, 2016.

[87] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (S&P 2016)*, pages 582–597. IEEE, 2016.

[88] Roberto Perdisci, David Dagon, Wenke Lee, Prahlad Fogla, and Monirul Sharif. Misleading worm signature generators using deliberate noise injection. In *2006 IEEE Symposium on Security and Privacy (S&P'06)*, pages 15–pp. IEEE, 2006.

[89] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and captions. In *Advances in neural information processing systems*, pages 2352–2360, 2016.

[90] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 7263–7271, 2017.

[91] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems NeurIPS 2015*, pages 91–99, 2015.

[92] Yafeng Ren and Donghong Ji. Neural networks for deceptive opinion spam detection: An empirical study. *Information Sciences*, 385:213–224, 2017.

[93] Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[94] Andras Rozsa, Manuel Günther, Ethan M Rudd, and Terrance E Boult. Are facial attributes adversarially robust? In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 3121–3127. IEEE, 2016.

[95] Andras Rozsa, Manuel Günther, Ethan M Rudd, and Terrance E Boult. Facial attributes: Accuracy and adversarial robustness. *Pattern Recognition Letters*, 2017.

[96] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision IJCV*, 115(3):211–252, 2015.

[97] Swami Sankaranarayanan, Arpit Jain, Rama Chellappa, and Ser Nam Lim. Regularizing deep networks using efficient layerwise adversarial training. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[98] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, pages 6103–6113, 2018.

[99] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing*, 307:195–204, 2018.

[100] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition.

In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1528–1540. ACM, 2016.

[101] Sijie Shen, Ryosuke Furuta, Toshihiko Yamasaki, and Kiyoharu Aizawa. Fooling neural networks in face attractiveness evaluation: Adversarial examples with high attractiveness score but low subjective score. In *2017 IEEE Third International Conference on Multimedia Big Data (BigMM)*, pages 66–69. IEEE, 2017.

[102] Richard Shin and Dawn Song. Jpeg-resistant adversarial images. In *NeurIPS 2017 Workshop on Machine Learning and Computer Security*, 2017.

[103] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[104] Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramer, Atul Prakash, and Tadayoshi Kohno. Physical adversarial examples for object detectors. In *12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18)*, 2018.

[105] Lance Spitzner. Honeypots: Catching the insider threat. In *19th Annual Computer Security Applications Conference, 2003. Proceedings.*, pages 170–179. IEEE, 2003.

[106] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 2019.

[107] Yanmin Sun, Andrew KC Wong, and Mohamed S Kamel. Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(04):687–719, 2009.

[108] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[109] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations ICLR*, 2014.

[110] Pedro Tabacof, Julia Tavares, and Eduardo Valle. Adversarial images for variational autoencoders. In *Advances in neural information processing systems NeurIPS Workshop 2016*, 2016.

[111] Rohan Taori, Amog Kamsetty, Brenton Chu, and Nikita Vemuri. Targeted adversarial examples for black box audio systems. In *2019 IEEE Security and Privacy Workshops (SPW)*, pages 15–20. IEEE, 2019.

[112] Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: adversarial patches to attack person detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019.

[113] Shixin Tian, Guolei Yang, and Ying Cai. Detecting adversarial examples through image transformation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[114] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations ICLR*, 2018.

[115] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[116] Xuyu Wang, Lingjun Gao, Shiwen Mao, and Santosh Pandey. Deepfi: Deep learning for indoor fingerprinting using channel state information. In *2015 IEEE wireless communications and networking conference (WCNC)*, pages 1666–1671. IEEE, 2015.

[117] Jonathan Wu, Prakash Ishwar, and Janusz Konrad. Two-stream cnns for gesture-based verification and identification: Learning user style. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 42–50, 2016.

[118] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2018.

[119] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is feature selection secure against training data poisoning? In *International Conference on Machine Learning ICML*, pages 1689–1698, 2015.

[120] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21*, 2018.

[121] Qingchen Zhang, Laurence T Yang, Zhikui Chen, and Peng Li. A survey on deep learning for big data. *Information Fusion*, 42:146–157, 2018.

[122] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition CVPR*, pages 4480–4488, 2016.

[123] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision ICCV*, 2017.

[124] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.

[125] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856. ACM, 2018.

My Publications

Refereed journal papers:

[2] Dang Duy Thang and Toshihiro Matsui, "Adversarial Examples Identification in an End-to-end System with Image Transformation and Filters", IEEE ACCESS (SCIE Q1, IF: 4.098). Accepted.

[1] Dang Duy Thang and Toshihiro Matsui, "Search Space of Adversarial Perturbations Against Image Filters", International Journal of Advanced Computer Science and Applications (ESCI, IF: 1.324), Vol. 11 No 1, pages 11-19, 2020.

Refereed conference papers:

[2] Dang Duy Thang and Toshihiro Matsui, "A Label-based Approach for Automatic Identifying Adversarial Examples with Image Transformation", In Proceedings of the Seventh International Symposium on Computing and Networking (CANDAR'19), Nagasaki, Japan, November 26-29, pages 112-120. IEEE, 2019 (acceptance rate 14.9%).

[1] Dang Duy Thang and Toshihiro Matsui, "Automated Detection System for Adversarial Examples with High-Frequency Noises Sieve", In Proceedings of the 11th International Symposium on Cyberspace Safety and Security (CSS), Guangzhou, China, December 1-3, pages 348-362. Springer, 2019 (acceptance rate 26%).

Non-reviewed papers:

[3] Dang Duy Thang, Taisei Kondo, Toshihiro Matsui, "A Label-based System for Detecting Adversarial Examples by Using Low Pass Filters", Computer Security Symposium, Nagasaki, Japan, October 21-24, pages 1356-1363, 2019.

[2] Taisei Kondo, Dang Duy Thang, Toshihiro Matsui、"Creation of Adversarial Example Attack that is Difficult to Protect by LPF", Computer Security Symposium, Nagasaki, Japan, October 21-24, pages 1364-1369, 2019.

[1] Dang Duy Thang, Toshihiro Matsui, "White-box attack on Google Machine Learning system", Poster Session, the 13th International Workshop on Security, Tohoku University, Japan, 2018. Best Poster Award