A Watermarking-Based Framework for Protecting Deep Image Classifiers Against Adversarial Attacks

by

Chen Sun

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2021

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

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

Abstract

Although deep learning-based models have achieved tremendous success in image-related tasks, they are known to be vulnerable to adversarial examples—inputs with imperceptible, but subtly crafted perturbation which fool the models to produce incorrect outputs. To distinguish adversarial examples from benign images, in this thesis, we propose a novel watermarking-based framework for protecting deep image classifiers against adversarial attacks. The proposed framework consists of a watermark encoder, a possible adversary, and a detector followed by a deep image classifier to be protected.

At the watermark encoder, an original benign image is watermarked with a secret key by embedding confidential watermark bits into selected DCT coefficients of the original image in JPEG format. The watermarked image may then go through possible adversarial attacks. Upon receiving a watermarked and possibly attacked image, the detector accepts it as a benign image and passes it to the subsequent classifier if the embedded watermark bits can be recovered with high precision, and otherwise rejects it as an adversarial example. The embedded watermark is further required to be imperceptible and robust to JPEG re-compression with a pre-defined quality threshold.

Specific methods of watermarking and detection are also presented. It is shown by experiment on a subset of ImageNet validation dataset that the proposed framework along with the presented methods of watermarking and detection is effective against a wide range of advanced attacks (static and adaptive), achieving a near zero (effective) false negative rate for FGSM and PGD attacks (static and adaptive) with the guaranteed zero false positive rate. In addition, for all tested deep image classifiers (ResNet50V2, MobileNetV2, InceptionV3), the impact of watermarking on classification accuracy is insignificant with, on average, 0.63% and 0.49% degradation in top 1 and top 5 accuracy, respectively.

Acknowledgements

First and foremost, I would like to thank my supervisor, Prof. En-Hui Yang, for his continuous guidance and support throughout my master's training. Prof. Yang formed my understanding of what insightful research looks like, and how it makes a difference to the community.

I would like to thank Prof. George Freeman and Prof. Zhou Wang for being my thesis committee members and reading my thesis. Their feedback is invaluable for me to improve my research.

I would like to thank all of my colleagues and friends in Multimedia Lab, especially Dr. Hossam Amer and Dr. Bin Chen, for generously offering their time for discussion.

Finally, I would like to express my deepest appreciation to my parents. They always respect my decisions and provide me with infinity support, both physically and mentally.

Dedication

To my Mom and Dad.

I always hesitate to express my feelings, but you will always be the most important people in my life.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Thesis Motivations and Problem Description

In recent years, Deep Neural Networks (DNNs) have demonstrated tremendous success for many image related tasks, such as image classification and face recognition. Unfortunately, DNNs are also known to be vulnerable to adversarial examples—subtly crafted, but imperceptible modifications of benign inputs which, once fed into DNNs, can lead DNNs to produce incorrect outputs. Specifically, given an original benign image x, a small perturbation can be easily crafted and added to x to generate a modified image x'. The output of a DNN in response to x' will be different from that of the DNN in response to x. Such x' is an adversarial example for x. The existence and easy construction of adversarial examples pose significant security risks to DNNs, especially in safety-critical applications, including face recognition and autonomous driving.

To safeguard DNNs against adversarial attacks, one approach is to build a classifier that distinguishes adversarial examples from natural images. The rationale is that although the adversarial perturbations are imperceptible to human eyes, it may be still possible to design an algorithm to detect their existence. Along this line, several detection-based methods have been proposed [34, 33, 41]. Some detection-based methods focus on finding general intrinsic properties of adversarial examples. Other detection-based methods aim to train classification networks to distinguish adversarial examples from benign images.

Although the detection-based defenses mentioned above are effective, to some extent, against some specific adversarial attacks, they in general have been proven vulnerable to more advanced adaptive adversaries, which have the full knowledge of the DNN to be

secured and the given detection strategy itself. Indeed, in their recent study, Carlini *et al.* examined 10 detection-based defenses proposed in recent years, and designed adaptive adversaries to defeat them all [6]. It seems that there is no general pattern or intrinsic property shared by all adversarial perturbations. The underlying reason is that adversarial examples can be generated in various of ways, and new attacking methods can be designed to avoid a specific pattern. As suggested by many studies [16], adversarial examples are widely distributed in the high-dimensional image space, indicating that finding a property that covers most cases may not be feasible. As such, a more robust and effective defense strategy is desirable. In this thesis, we are going to present such a method.

1.2 Thesis Contribution

In this thesis, we took a radically different approach to build our detector by considering another application scenario. For many real-world applications, from Quality Control cameras in manufacturing [37, 53] to cameras and sensors in self-driving cars, it is reasonable to assume that the original benign image can be processed upon acquisition before it is attacked.

In this thesis, instead of exploring general intrinsic properties of all adversarial examples, we focus on adversarial perturbation itself, which is the malicious modification being made to the benign image by the adversary. Inspired by semi-fragile watermark [11, 15], we propose a radically different approach for adversarial perturbation detection. Specifically, we propose a novel watermarking-based framework for protecting deep image classifiers against adversarial attacks.

The proposed framework consists of a watermark encoder, a possible adversary, and a detector followed by a deep image classifier to be secured. At the watermark encoder, an original benign image is watermarked with a secret key by embedding confidential watermark bits into selected DCT coefficients of the original image in JPEG format. The watermarked image may then go through possible adversarial attacks. Upon receiving a watermarked and possibly attacked image, the detector accepts it as a benign image and passes it to the subsequent classifier if the embedded watermark bits can be recovered with high precision, and otherwise rejects it as an adversarial example. The embedded watermark is further required to be robust to JPEG re-compression with a pre-defined quality threshold. We also present specific methods of watermarking and detection that are specially optimized for adversarial example detection. The proposed framework is independent of adversarial attack methods, and can be applied to protect any image task related DNN.

Our contributions in this thesis are as follows:

- We propose a novel watermarking-based framework for safeguarding image task related DNNs against adversarial attacks.
- Within our proposed framework, specific methods of watermarking and detection are presented.
- Regular adversaries such as Fast Gradient Sign Method (FGSM) [17], Projected Gradient Descent (PGD) [32] and Carlini & Wagner (CW) [7] attacks are modified to work within our proposed framework, and are further extended to attack our watermarking-based detection strategy (i.e., adaptive white-box attacks).
- We show by experiment on a subset of ImageNet validation dataset that our proposed framework along with the presented methods of watermarking and detection is effective against a wide range of advanced attacks (static and adaptive), achieving a near zero (effective) false negative rate for FGSM and PGD attacks (static and adaptive) with the guaranteed zero false positive rate.
- It is shown that for all tested deep image classifiers (ResNet50V2 [21], MobileNetV2 [42], InceptionV3 [49]), the impact of watermarking on classification accuracy is insignificant with, on average, 0.63% and 0.49% degradation in top 1 and top 5 accuracy, respectively.

1.3 Thesis Organization

The remainder of this thesis is organized as follows:

Chapter 2 reviews some core concepts and literature related to this thesis. First, we present an overview of JPEG compression standard. Then we introduce digital watermarking schemes and invoke an invariant property of DCT coefficients from [30]. We describe adversarial attack and defense methods in detail, including their categorization and several representative methods.

Chapter 3 presents the formulation, design principles, and evaluation metrics of the proposed framework.

Chapter 4 presents a specific watermarking and detection algorithm, which is optimized for adversarial perturbation detection. We present the modification to regular adversaries, which make them compatible with our pipeline, and 2 adaptive adversaries as countermeasures against our framework.

Chapter 5 explores the effectiveness of the proposed framework by presenting the experimental results of the evaluation metrics mentioned in chapter 3.

Chapter 6 summarizes the key idea of the thesis and also proposes related future work.

Chapter 2

Background and Literature Review

In this chapter, we will go through several fundamental concepts related to this thesis. Section 2.1 introduces JPEG compression standard, the most widely used standard for both human vision and machine vision. Section 2.2 introduces the concept of watermarking and invokes an invariant property of DCT coefficients. Section 2.3 presents detailed information about adversarial examples, including categorization and 3 specific generation algorithms. Section 2.4 introduces several popular directions of adversarial defenses.

2.1 JPEG Image Compression Standard

JPEG [54] is a lossy image compression standard designed for human vision system. Over the last 30 years, as the amount of image data increased exponentially, JPEG has become the most popular image compression standard. By preserving more low frequency information and discarding more high frequency information, a typical JPEG image achieves 10:1 compression ratio without introducing perceptible distortion to human eyes. JPEG standard is also widely adopted in computer vision datasets and pipelines, e.g., all the images in ImageNet dataset are in JPEG format.

A typical JPEG compression pipeline is shown in Figure 2.1. The key steps of JPEG compression are introduced below:

Color space conversion: A given image is first converted from RGB color space to YCbCr color space, where Y is the luminance (pixel brightness) channel and Cb and Cr are chrominance (pixel color) channels.

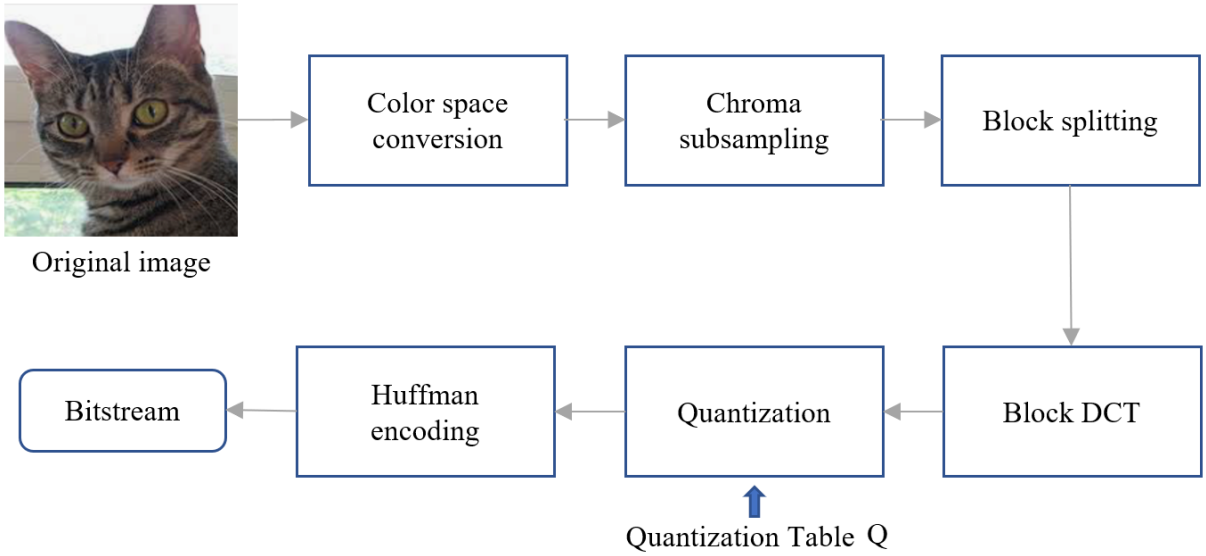


Figure 2.1: Key components of JPEG compression pipeline.

Chroma subsampling: As human's vision system relies mainly on brightness information (Y channel), the other two channels (Cb and Cr channels) were down-sampled to yield greater compression ratio, i.e., each 2×2 pixels are merged into a single pixel.

Block splitting: The smallest unit of a JPEG compressed image is 8×8 block. For each channel, the image is first padded in the right and bottom side so that its size is the next multiple of 8×8, then being divided into 8×8 blocks for further processing.

Block-wise DCT: For each channel, the image is firstly divided into non-overlapping 8×8 blocks. Then, the 64 pixel values of the block are decomposed into 64 frequency components with 2-D discrete cosine transform (DCT), namely DCT coefficients. In what follows, for each $i \in \{0, 1, ..63\}$, we will use $d(i)$ to denote the DCT coefficient at the i-th frequency in zigzag order. Note that a lower index represents a lower frequency.

Quantization: After Block-wise DCT, 64 DCT coefficient will be further quantized to an integer multiple of their corresponding quantization step sizes as follows:

$$D_{QF}(i) = \lfloor \frac{d(i)}{Q_{QF}(i)} \rceil, \tag{2.1}$$

where $D_{QF}(i)$ is the i-th quantized DCT coefficient integer, $\lfloor \cdot \rceil$ denotes the rounding func-

$$\begin{bmatrix} -415 & -33 & -58 & 35 & 58 & -51 & -15 & -12 \\ 5 & -34 & 49 & 18 & 27 & 1 & -5 & 3 \\ -46 & 14 & 80 & -35 & -50 & 19 & 7 & -18 \\ -53 & 21 & 34 & -20 & 2 & 34 & 36 & 12 \\ 9 & -2 & 9 & -5 & -32 & -15 & 45 & 37 \\ -8 & 15 & -16 & 7 & -8 & 11 & 4 & 7 \\ 19 & -28 & -2 & -26 & -2 & 7 & -44 & -21 \\ 18 & 25 & -12 & -44 & 35 & 48 & -37 & -3 \end{bmatrix}$$

(a) DCT coefficients

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

(b) Quantization table Q

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -3 & 4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(c) Quantized DCT coefficients

Figure 2.2: An example of JPEG quantization process.

tion which returns the nearest integer, $Q_{QF}(i)$ is the quantization step size with respect to a certain JPEG quality factor (QF, ranges from 1 to 100), and the quantized DCT coefficient is equal to $D_{QF}(i)Q_{QF}(i)$. A smaller QF corresponds to higher quantization step sizes, which means worse image quality. In theory, any quantization step size other than $Q_{QF}(i)$ can also be used in Equation (2.1). An example of JPEG quantization process is shown in Figure 2.2. Figure 2.2(b) is the default JPEG quantization table for the luminance channel (Y channel). DCT coefficients with high frequency yields greater quantization step size and are compressed more to increase compression ratio while maintaining fine image quality for human vision.

Huffman coding: Finally, quantized DCT coefficients are scanned in zigzag order, producing an 1-D array starting from DC component to highest frequency component. It will go through Run-Length coding and then encoded by Huffman algorithm to produce the final JPEG archive.

2.2 Digital Watermarking

2.2.1 Robustness of Digital Watermarks

A digital watermark in an image is like a bookmark in a book. If the image is being processed, the bookmark may also be changed accordingly. Over the years, watermarks have been widely used for copyright protection and authentication. Based on their robustness, digital watermarking techniques can be classified into three categories:

- **Fragile watermark** A fragile watermark yields low robustness towards modifications, where the watermark will be destroyed after small amount of modification, indicating malicious operation is conducted. Usually, the content of a fragile watermark is not important, however, the presence of the watermark matters, which is considered as the indication of originality.
- **Robust watermark** Just in the opposite way of fragile watermark, A robust watermark should be difficult to remove or damage. Based on specific purposes, robust watermarks are designed to be resistant to a given set of image transformations. If the attacked watermarked image still has a decent quality for the user to use, the watermark should be able to be extracted from it. If the robust watermark cannot be extracted from the attacked image, very likely the attacked watermarked image is in a very bad shape and not usable. In this case, the content of the watermark is usually set to be

ownership information, which provides the host image with Intellectual Property (IP) protection.

- **Semi-fragile watermark** In practice, not all distortion are introduced by malicious operations. For example, unintentional manipulations caused by common image processing operations like JPEG compression won't change the visual meaning of the image. Semi-fragile watermarks are designed to be robust to a set of legal operations and fragile to other operations. As a result, it is commonly used for authentication in specific pipelines.

2.2.2 JPEG-resistant Semi-fragile Watermarking

Attacks modify images and make them carry different visual meanings to the observer. Before the advent of DNNs, fragile or semi-fragile watermark [11], an invisible and removable component of the image that would be distorted after illegal operations, has been invented as one way to protect images from tampering. For these watermarking pipelines, typically a secret algorithm was used to embed watermark into the image before distribution. Therefore, given a copy of the image, the same algorithm was used to detect the presence of the watermark. If the watermark is seriously distorted, this copy will be considered untrustworthy.

In this research thesis, we focus on the setting that JPEG compression as the legal operation and adversarial attacks as the malicious operation. To make the watermark robust to JPEG re-compression, a practical solution is to embed the watermark into DCT domain [11, 15].

To provide theoretical proof for the robustness of these semi-fragile watermarking schemes against JPEG re-compression, we invoke an invariant property of DCT coefficients from [30][Theorem 1].

Lemma 1 (DCT Invariant Property) *If d is an integer multiple of q_0, then for any quantization step size $q < q_0$, quantizing d with q is invertible. That is, d can be fully reconstructed from its quantized value $\lfloor \frac{d}{q} \rceil q$.*

Proof Write d as $d = kq_0$, where k is an integer. It can be verified that

$$\left| d - \lfloor \frac{d}{q} \rceil q \right| \leq \frac{q}{2} < \frac{q_0}{2}. \tag{2.2}$$

Dividing both sides of (2.2) by q_0 yields

$$\left| k - \frac{\lfloor \frac{d}{q} \rceil q}{q_0} \right| < \frac{1}{2}. \tag{2.3}$$

Therefore, given a DCT coefficient d, re-quantizing $\lfloor \frac{d}{q} \rceil q$ with q_0 yields back k and hence d.

2.3 Vulnerability of Deep Models: Adversarial Example

From linear classifies to Deep Neural Networks (DNNs), as the depth and complexity increase, the advancement of machine learning methods comes at the cost of worse interpretability and robustness. Since the seminal work of Szegedy *et al.* [50] and Biggio *et al.* [5] firstly suggested the existence of adversarial examples, various algorithms have been proposed to craft adversarial perturbations—a small and imperceptible noise that, once added to the original image, will change the model's behaviour. In this section, we will introduce the definition and objectives of adversarial examples, along with several representative adversarial attack methods. In addition, this section also covers the categorization of adversarial attack methods from 4 different perspectives.

2.3.1 Formulation of Adversarial Example

Mathematically, Given a trained classifier $\mathcal{C}$ and an benign image x, the objective of an adversarial untargeted attack is to compute an perturbation vector δ such that $\mathcal{C}(x+\delta) \neq \mathcal{C}(x)$ and $D(x, x+\delta) \leq \rho$, where $D(\cdot)$ is a given distance function and $\rho > 0$. Formally, the basic optimization problem to craft adversarial perturbation can be formulated as:

$$\begin{aligned} \text{minimize} \quad & D(x, x+\delta) \\ \text{subject to} \quad & \mathcal{C}(x+\delta) \neq \mathcal{C}(x) \\ & x+\delta \in [0,1]^n, \end{aligned} \tag{2.4}$$

where n is the dimension of the image.

2.3.2 Targeted and Untargeted Attack

Based on the purpose of an adversary, the adversarial attack methods can be divided into 2 categories: targeted attack and untargeted attack.

Targeted attack In most cases, the adversary aims to arbitrarily change the DNN's output in response to an input image. In this case, a attacker-specified targeted class y' is given, and the first constraint in Equation 2.4 is replaced as follows:

$$\mathcal{C}(x + \delta) = y' \tag{2.5}$$

This type of attack is particularly harmful to practical pipelines, as the attacker would make use of the target DNN to force the target pipeline performing a specific behaviour.

Untargeted attack In other cases, the adversary simply wants to damage the functionality of the target DNN, which is made by change the the prediction result in response to $x + \delta$, or minimize the confidence of the original classification result $\mathcal{C}(x)$.

It is worth noting that the targeted attack is strictly harder than untargeted attack, since a successful targeted adversarial example is also a successful untargeted adversarial example. Furthermore, untargeted adversarial example has an considerable chance to make the DNN under attack to produce the second-likely prediction of the benign example because in many cases it requires less distortion so as to minimize the objective function expressed in Equation 2.4. Based on this observation, targeted attack is considered a more important research problem and was paid with more attention.

2.3.3 Adversary's Knowledge

In addition, adversary's knowledge is also critical to attack's performance. There are three different commonly used threat models:

Black-box adversary This type of adversary has no knowledge of either the DNN to be safeguarded or the defense strategy. As such, the choice of attack is limited to a few approaches such as transfer attacks [39] or query-based attacks [2, 9].

Static white-box adversary A static white-box adversary has the full knowledge of the DNN to be secured (including its architecture and parameters), but does not know anything about the defense strategy. Adversarial examples for an image x are generated based on the DNN to be secured, the image x, a targeted output, and the allowed maximum perturbation distance from x.

Adaptive white-box adversary An adaptive white-box adversary is aware of both the DNN to be secured and the defense strategy. Using this perfect knowledge, it could generate the strongest adversarial examples to defeat the defense strategy and cause the DNN to produce incorrect outputs.

2.3.4 Distance Metric

An important goal of adversarial attacks is to introduce as small perturbation to the benign image x as possible. This perturbation term is usually treated as a constraint or used as a penalty term in the objective function. To measure the adversarial distortion, 3 distance metrics are widely-used in the literature, which are L_0, L_2 and L_∞ norms.

The definition of L_p norm is formulated as follows:

$$||\delta|| = (\sum_{i=1}^{n} |\delta_i|^p)^{\frac{1}{p}} \tag{2.6}$$

These 3 L_p norms are selected as the distance metric for their intuitive physical meanings. L_0 distance measures the number of pixels being altered by the attack. L_2 distance measure the Euclidean distance between x and $x+\delta$. L_∞ distance measures the maximum distortion introduced to any of the pixels.

Recall that adversarial examples are expected to be both imperceptible to human eyes and radically different from machine learning perspective, the distance metrics introduced here are really aimed at measuring the human perceptual similarity. Empirically, adversarial examples measured in L_0 distance add large distortion to a small fraction of pixels of an image, which makes the altered pixels conspicuous among its background, and a human eye can easily pick up the difference. In construct, Both L_2 and L_∞ adds small distortion to many pixels, which is better for adversary's purpose.

2.3.5 Gradient-based Attacks

Based on the way how attack algorithms searching for adversarial examples, attack algorithms can be roughly divided into two main categories: gradient-based attack and optimization-based attack. In this subsection, we introduce the concept and development history of gradient-based attacks, along with 3 most popular gradient-based attacks, including FGSM, BIM, and PGD. FGSM and PGD are also used to generate adversarial examples in the experiment of this thesis.

To solve the optimization problem expressed in Equation 2.4, adversarial perturbations are usually calculated based on the gradients of the target DNN with respect to the original image x. In back-propagation, the loss function J is usually defined as the error between the desired output y (usually the ground truth label) and the neural network output in response to a particular input. By consider the input as a constant variable, the calculated loss value is used to determine the gradients of each model parameter θ, which are used to update these parameters in the training process.

The same concept of back-propagation is used in gradient-based attacks to produce a perturbation vector for x. Contrary to the training process, it considers θ constants and x as variables. As a result, the calculated gradients are the partial derivative of the loss function with respect to each input element, e.g., pixel values for RGB images or DCT coefficient values for JPEG images. These gradients are further processed to form different perturbation vectors.

FGSM FGSM is a simple yet effective method that create the adversarial perturbation in one-step [17]. Mathematically, the computation process can be formulated as follows:

$$x_{adv} = x + \epsilon * \text{sign}(\nabla_x J(\theta, x, y)) \tag{2.7}$$

$$\text{sign}(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}, \tag{2.8}$$

where ϵ is the perturbation budget. As $\text{sign}(\nabla_x J(\theta, x, y))$ is a matrix whose absolute value of all elements is 1, the distortion introduced by perturbation vector $\epsilon * \text{sign}(\nabla_x J(\theta, x, y))$ is exactly ϵ in terms of L_∞ norm. The intuitive of introducing the sign() function is to generate the adversarial perturbation with same intensity for each images, and maximize the attack performance under a given L_∞ perturbation budget.

As an untargeted attack, Equation 2.7 aims to minimize the DNN's confidence on y for input x. FGSM can also be extended to a targeted attack by maximizing the DNN's confidence on y' for input x, where y' is the target label. Mathematically, it can be formulated as follows:

$$x_{adv} = x - \epsilon * \text{sign}(\nabla_x J(\theta, x, y')). \tag{2.9}$$

The authors in [17] explain the reason why one-step attack is can be effective by DNN's highly linear nature. Although the design of DNNs involves non-linearity to increase model capacity and to better fit non-linear functions, they behaves in very linear ways with respect to small perturbations. Furthermore, the non-linearity of many models are intentionally decreased to speed up the training process, e.g., implementing Rectified Linear Unit (ReLU)

[36]

$$f(x) = \max\{0, x\} \tag{2.10}$$

or Leaky ReLU [57]

$$f(x) = \max\{0.01x, x\} \tag{2.11}$$

instead of traditional activation functions such as Sigmoid or Tanh. This linearity suggests that this one-step attack that works for linear model should also damage DNNs.

BIM and PGD From the optimization point of view, FGSM can be extended in several ways. In [27], Iterative-FGSM (I-FGSM) was first proposed as Basic Iterative Method (BIM) to increase attack strength by applying FGSM for T iterations with perturbation budget α for each step. In the t th iteration, the update rule of adversarial example is as follows:

$$x_{adv}^0 = x, \tag{2.12}$$

$$x_{adv}^{t+1} = \text{Clip}_{x,\epsilon}\{x + \alpha * \text{sign}(\nabla_x J(\theta, x_{adv}^t, y))\}, \tag{2.13}$$

where $Clip_{x,\epsilon}$ clips the intermediate result x_{adv}^{t+1} into the ϵ-neighbourhood of the original input x so as to fit the overall perturbation budget ϵ. Note that although α, T, ϵ are user-defined hyper-parameters, $\alpha T \geq \epsilon$ are required for the adversarial example to make full use of the perturbation budget.

Going one step further, PGD [32] was extended from BIM to further increase attack strength by adding random initialization. Contrary to Equation 2.12, PGD initializes x_{adv}^0 to a random point inside the ϵ-neighbourhood of the original input x, and restarts the attack algorithm with new starting points until a satisfactory adversarial example is achieved. Under L_∞ constraint, the update rule for PGD is identical to Equation 2.13.

2.3.6 Optimization-based Attacks

The traditional way of adversarial example generation that defining an objective function and perform gradient descent is effective to some extent, however, it only works well on highly linear optimization space. Hence, much powerful optimization tools are eagerly desired to generate stronger adversarial attacks. In this section, we introduce the most representative and popular research work, Carlini and Wagner attacks [7], to address this problem.

Carlini and Wagner attacks In [7], Carlini and Wagner proposed a set of powerful optimization-based attacks that produces effective adversarial examples which distortion

are measured in L_0, L_2 or L_∞ budget, namely C&W-L_0, C&W-L_2 and C&W-L_∞ attacks. The optimization problem of these attacks are formulated as:

$$\min D(x, x+\delta) + c \cdot f(x+\delta) \tag{2.14}$$

subject to

$$x + \delta \in [0,1]^n, \tag{2.15}$$

where $D(\cdot,\cdot)$ denotes the selected distance metric, δ is the adversarial perturbation and $f(\cdot)$ denotes a customized adversarial objective loss that satisfies $f(x+\delta) \leq 0$ if and only if $\mathcal{C}(x+\delta) = t$, i.e., the output of the target DNN $\mathcal{C}$ in response to input $x+\delta$ is the target label t. The constant c is used to balance the trade-off between the distance metric and the loss function. It is worth noting that c is not a user-defined parameter, but is found by binary search so as to meet a sweet point. It is mostly lower bounded by 1×10^{-4} and upper bounded by $+\infty$.

Further, each element of δ, namely δ_i, is substituted by s_i as follows:

$$\delta_i = \frac{1}{2}(\tanh s_i + 1) - x_i. \tag{2.16}$$

As $-1 \leq \tanh s_i \leq 1$, the constraint $x_i + \delta_i \in [0,1]$ always holds. This variable substitution also helps to smooth the optimization process of this attack [7].

In the paper, 7 different objective functions are analyzed and evaluated, in which the best one is given by:

$$f(x_{adv}) = \max(\max\{Z(x_{adv})_i : i \neq t\} - Z(x_{adv})_t, -\kappa), \tag{2.17}$$

where $\max\{Z(x_{adv})_i : i \neq t\}$ is the highest probability for non target class and κ is an user-defined hyper-parameter to encourage an adversarial example that will be classified as target class t with high confidence.

After the optimization problem is defined, an Adam optimizer [25] is employed to optimize over s in each binary search step with different c. In addition, multiple random starting point, the same technique used in PGD, is used to avoid stuck in the optimization process.

C&W-l2 attack yields almost 100% attack success rate on undefended DNNs for MINIST, CIFAR-10 and ImageNet datasets, and was reported to be able to bypass several strong defenses with no or small modifications such as changing loss function [6][22][7]. However, it also has several limitations. First, the computational cost is very expansive

and its runtime is reported to be 10 times slower compared with BIM [7], which makes C&W-l2 attack incompatible for real-time tasks and adversarial training. Second, adversarial examples generated by C&W-l2 attacks are very delicate, any additional distortion, such as random Gaussian noise and high quality JPEG compression can restore the adversarial samples to benign samples [20][12]. Finally, the transferability of C&W-l2 adversarial examples is bad, i.e., adversarial examples for one model is not likely to fool another model, which is desired for black-box attack.

2.4 Adversarial Defense Methods

The existence of adversarial examples poses a serious threat to practical deep learning. Inspired by adversarial examples, researches have been performed to hardening DNNs against adversarial attacks, either to increase the model's robustness with respect to small perturbations, detect adversarial principles based on their intrinsic property, or set obstacles to the optimization process of adversarial example generation. In this section, we present a few representative defense directions, their effectiveness, and countermeasures.

2.4.1 Adversarial Training

Adversarial training is an intuitive and straightforward method against adversarial samples, which attempts to augment the training set with adversarial examples. The objective of adversarial training is described as a min-max problem, which the training procedure is trying to minimize the classification error on the strongest possible adversarial examples. [32][45]. This objective function can be formulated as follows:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{x+\delta\in B(x,\epsilon)} L(\theta, x+\delta, y) \right], \tag{2.18}$$

where $(x, y) \sim \mathcal{D}$ represents training data sampled from distribution $\mathcal{D}$, $L(\cdot)$ denotes the loss function and $B(x, \epsilon)$ is the set of possible examples within allowed perturbation ϵ, defined as $B(x, \epsilon) = \{x + \delta \in [0, 1]^n | D(x, x + \delta) \leq \epsilon\}$. In practice, the inner maximization problem in Equation 2.18 is to find the most effective adversarial example, which is usually crafted by PGD, whereas the outer minimization problem is solved by standard DNN training strategies using Equation 2.18 as its loss function. The resultant DNN is supposed to be resistant to the adversarial attack method used in the inner maximization problem.

PGD searches for a strong adversarial example within the allowed perturbation budget from multiple random starting points and is considered as a universal L_∞ attack [32]. It

is shown by experiment that model trained against PGD is also resistant to most known L_∞ attacks, including FGSM, BIM, and C&W-L_∞, to some degree. As a result, PGD adversarial training [32] has become the most widely acknowledged baseline for adversarial training. Based on this baseline, much research effort is also put on decreasing the computational cost [44] and enhancing adversarial robustness against other types of attacks [51].

Adversarial training is an effective method to increase model's robustness. By augmenting the training set with adversarial examples in each training loop, the trained model behaves much better than the standard trained model when facing adversarial examples. On the MINIST dataset, state-of-art adversarial training provides a robust model with over 90% classification accuracy against 20-step PGD attack, however, this number decreases to around 50% and 40% for CIFAR-10 and ImageNet datasets, respectively. Additionally, while PGD-trained networks are resistant to a wide-range of gradient-based attacks, the trained network is still vulnerable to more advanced C&W-l2 attack. On top of that, the extended training time and decreases training time also decreases practicality.

2.4.2 Randomization-based Approach

Adversarial examples are effective yet delicate, especially for strong adversarial examples with small perturbation. Some random noise added to such adversarial examples is likely to remove their adversarial effect. That is intuitive to understand: most examples sampled around the original benign example x are not adversarial, so the examples sampled around an adversarial example $x + \delta$ should be the same if $D(x, x + \delta)$ is reasonable small. For example, PGD and C&W adversarial perturbations are being constructed precisely, and there is a good chance that can be degraded to random effects under another distortion. In another hand, DNNs are not very sensitive to such random distortion in most cases. Therefore, DNNs equipped with certain randomization measures are likely to perform better on adversarial attacked images.

The first the most intuitive randomization-based approach is random input transformation presented by Xie *et al.*[56]. In their work, the input image x is first processed by random resizing and padding, generating a modified input x', which is consequently fed into the underlying DNN. The pipeline is illustrated in Figure 2.3. [56]. Both the random resizing layer and random padding layer are simple layers that transform the image but in a random manner. The input image x is first resized and then zero-padded to the DNN's input size. The proposed method is tested effective in black-box settings, however, was being broke quickly by Expectation over Transformation (EoT) method [4], in which the

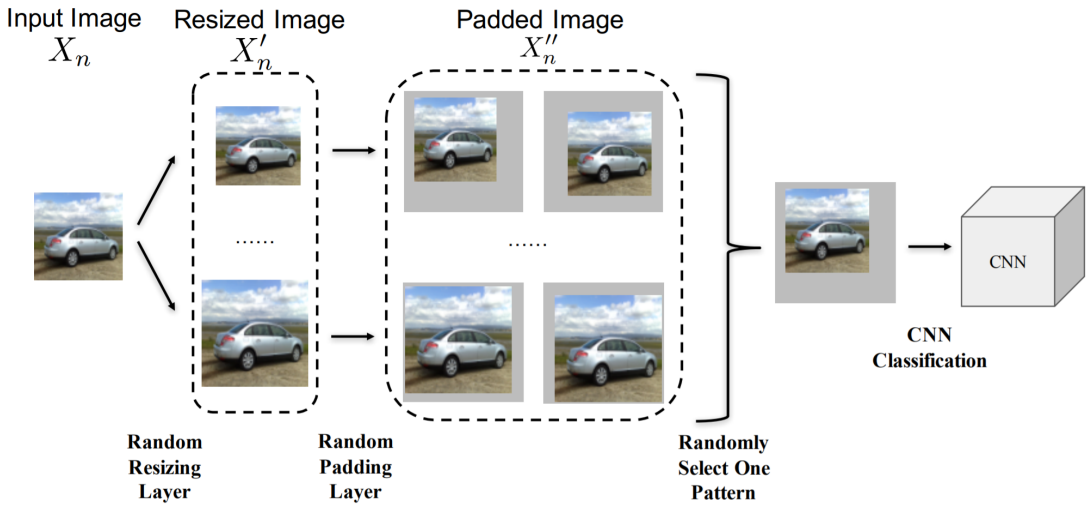


Figure 2.3: Illustration of random resizing and padding pipeline proposed by Xie *et al.*[56]. The input image is first resized then padded with zero in an random manner before feeding into the DNN under protection.

gradient of the new pipeline is approximated by averaging over the gradients of multiple differently resized and padded image.

Apart from directly modify the input image, randomization can also be added to the DNN architecture. Feature pruning is another seemingly promising direction of adversarial defense, in which some of DNN's learned features are 'pruned' thus do not contribute to the model's output. In [14], a subset of features that have relatively smaller activation values are being stochastically selected and pruned for each layer to stabilize the behavior of the model with respect to adversarial examples. To compensate for the pruned features, the remaining activation values are scaled up and normalized. In other feature pruning papers, new criteria to select the features or neurons to be pruned are also proposed, and new mechanisms are used to further hardening its defense. However, an obvious limitation to this line of feature pruning is that it decreases the model capacity, hence the performance on benign examples degenerates. In addition, adversarial attacks against this specific defense have also been proposed and tested effective [4].

2.4.3 Input Transformation and Reconstruction

Different from previously mentioned defense methods that mainly focus on increasing the DNN's robustness, input reconstruction methods aim at filtering out the adversarial perturbation by transforming or reconstructing the input image.

Instead of directly feeding the suspicious input image x to the DNN, $\hat{x}$ Given an suspicious image x, these type of adversarial defense methods attempt to cleanse x by a function $G(\cdot)$, producing a reconstructed image $\hat{x} = G(x)$. $\hat{x}$ is then fed into the underlying DNN instead of the original input x. Previous works point out three directions to build such a function $G(\cdot)$. The first direction attempts to apply traditional image processing techniques to either distort the adversarial perturbation or using lossy operations to filter out small perturbations [20]. The second direction construct $G(\cdot)$ based on Generative Adversarial Networks [41][47]. Especially, a generator is trained to learn the distribution of the benign dataset, and trying to project an adversarial example to its benign form. The third direction is based on auto-encoder architectures [33] [24], which attempts to learn the manifold of benign examples. In [33], the encoder compares the input example with the learned manifold, whereas the decoder reforms the input example such that its output lies on the data manifold. In addition, the encoder alone can be used as an adversarial example detector by comparing the distance between the input example and the learned data manifold.

In general, this kind of defense does not require any modifications to the underlying DNN, thus can be introduced to existing pipelines as an additional module. However, adaptive adversaries with perfect knowledge of the defense can still easily bypass these defenses. Since these defenses are close to adding another detector, either another neural network or simple image processing layers, to the beginning of the DNN under protection, it makes sense that adversarial examples can fool the integrated model.

2.4.4 Detection-based Methods

Detection is another intuitive approach against adversarial examples by directly discarding them from the dataset. Further, we introduce a few representative defense-based methods from 3 popular directions.

The first direction of detection-based method is to train a secondary classifier that distinguishes adversarial examples from benign examples. Authors in [34] propose a simple neural network that takes an image as input and returns a binary digit as an indication

of benignity. The network is further trained on a dataset that each image is labeled as a benign example and adversarial example,

Secondly, authors in [19] proposed an adversarial re-training approach, in which a new class is introduced to the classification as the adversarial class, and the adversarial examples in the training set are labeled as this new class. Therefore, the DNN is trained to detect the adversarial examples by classifying them into the adversarial class. The authors claim that adversarial examples and benign examples are drawn from different distributions, and thus can be detected by statistical tests. The modification of the DNN is also designed based on this prior knowledge.

Lastly, another statistical detection approach aims to detect the intrinsic properties of adversarial examples. To this end, detection based on Principle Components Analysis (PCA) [23] [29] and distributional detection [19] are proposed. At a high level, these methods attempt to find a common pattern of changes in statistical properties after an image is attacked.

All of the mentioned detection methods are effective to a certain extent under static settings. Given that adversarial examples can easily fool a single DNN, it is reasonable that they are fool another detection network at the same time. For the statistical detection approaches, it is hard to find an intrinsic property that applies for all adversarial examples, given that the pattern of adversarial perturbation can be easily modified by the adversary.

Chapter 3

Watermarking-based Framework Overview and Design Principles

3.1 Overview

The key idea of our work is to use semi-fragile watermarking for adversarial perturbation detection. In this chapter, we introduce our proposed framework consists of a few components in Section 3.2. Then, we introduce our design principles and evaluation metrics for the watermarking method in Section 3.3.

3.2 Framework Design

As shown in Figure 3.1, our proposed watermarking-based adversarial defense framework consists of a watermark encoder, a possible adversary, and a detector followed by the DNN $\mathcal{C}$ to be secured. Consider an original image x. Instead of directly exposing x to an adversarial environment, we use the watermark encoder ϕ to convert x into $x_{wm} = \phi(x, S)$, a watermarked version of x, by embedding watermark bits into x with a secret key, where S denotes both the secret key and embedded watermark bits. Note that S is kept in secret, i.e., the adversary does not have access to it. Consequently, x_{wm} may undergo adversarial attacks in the adversarial environment or some allowed legal operation before being passed to the detector. Let φ denote an adversarial attack algorithm and g denote the allowed legal operation. The image received by the detector would be either x_{wm}, $\varphi(x_{wm})$, or $g(x_{wm})$. With the help of S, the detector will further distinguish the watermarked and

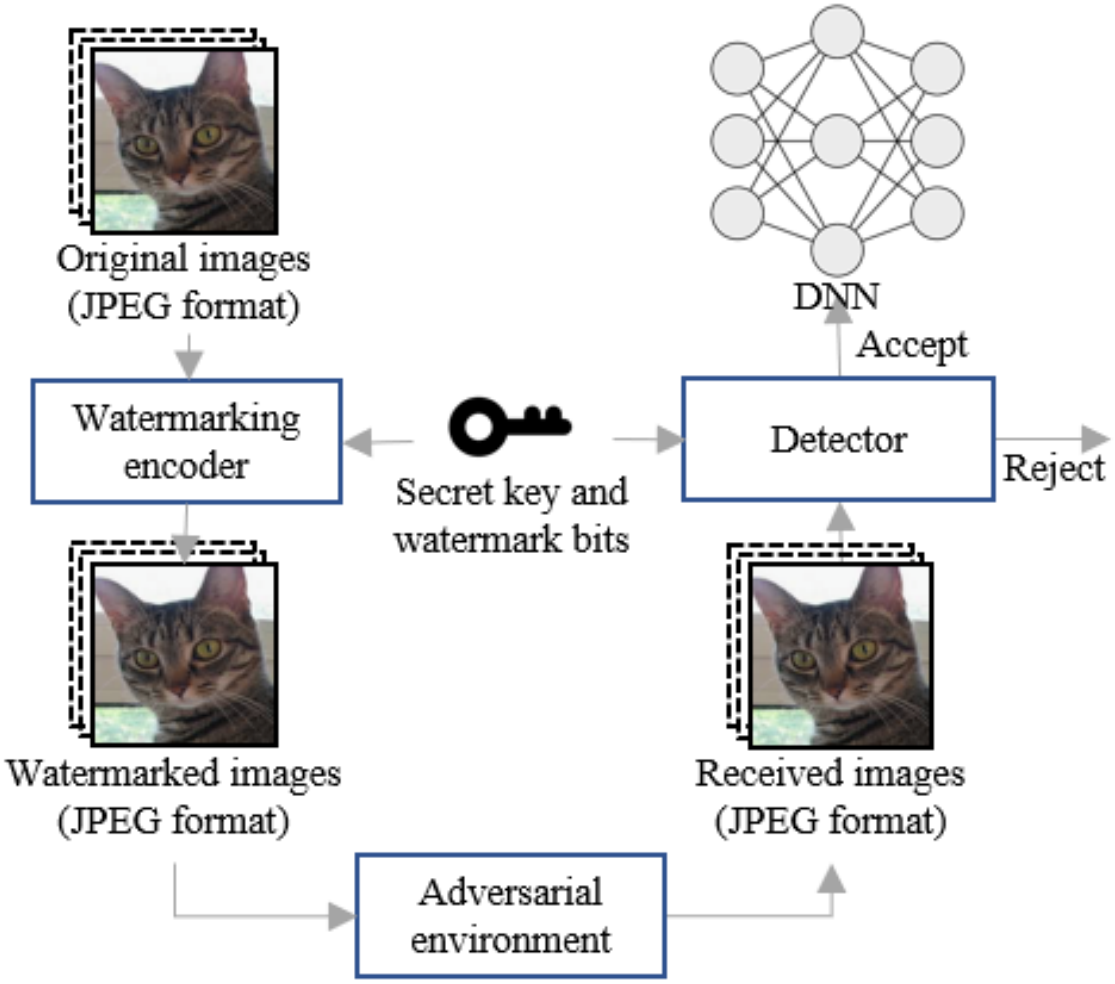


Figure 3.1: Illustration of the watermarking-based adversarial defense framework. Original images are first watermarked with a secret key and then possibly attacked by adversaries. Watermarked and possibly attacked images are accepted by the detector only if the embedded watermark bits are recovered with high precision.

attacked image $\varphi(x_{wm})$ from benign images x_{wm} and $g(x_{wm})$. The received image will be fed into $\mathcal{C}$ only if it is accepted as a benign image by the detector.

As one of the most commonly-used formats for images, JPEG standard is also widely adopted in computer vision datasets and pipelines, e.g., the ImageNet dataset [13]. Since high quality JPEG compression is often acceptable or even required in practical applications, we consider JPEG re-compression with $QF \geq 50$ as the legal operation g in our framework hereafter.

3.3 Watermarking Method Design Principles and Evaluation Metrics

There are two main objectives for our framework. First, the embedded watermark should not visibly distort the original image x, nor degrade $\mathcal{C}$'s performance significantly. Second, the images accepted by the detector should be harmless to $\mathcal{C}$. Formally, the performance metrics of the proposed framework are listed below:

- **Classification accuracy on x_{wm}** Watermarking should not significantly degrade the classification accuracy of $\mathcal{C}$. We evaluate the performance degradation of $\mathcal{C}$ by comparing the Top-1 and Top-5 accuracy on the original dataset and watermarked dataset.
- **Watermarking distortion** The watermarking distortion is evaluated using PSNR between x and x_{wm}.
- **False positive rate** If the watermarked image x_{wm} is not attacked, it should be accepted by the detector as a benign image with high probability. The false positive rate is defined as the percentage of x_{wm} that are rejected by the detector as attacked images. As we shall see later, for our proposed methods of watermarking and detection, the false positive rate is guaranteed to be 0.
- **Robustness to high quality JPEG re-compression** The embedded watermark should be robust to high quality JPEG re-compression with $QF \geq 50$, i.e., the detector should not reject $g(x_{wm})$ so as to produce a false positive case. The robustness against high quality JPEG re-compression is evaluated using the JPEG re-compression false positive rate (JRFPR), defined as the percentage of $g(x_{wm})$ that are rejected by the detector as attacked images.

- **Detection rate** The detection rate is defined as the percentage of attacked images $\varphi(x_{wm})$ that are accepted by the detector as benign images. It reflects the watermark's sensitivity to adversarial attacks.

- **Effective false negative rate** The effective false negative rate is defined as the percentage of attacked images $\varphi(x_{wm})$ that are simultaneously accepted by the detector and also successfully cause the DNN $\mathcal{C}$ to produce outputs different from those corresponding to x_{wm}. The rationale for introducing this metric is to report harmful adversarial examples only: although the adversary may bypass occasionally the detector by decreasing the perturbation budget, the strength of the resulting adversarial example will also decrease. Thus, if $\varphi(x_{wm})$ and x_{wm} share the same prediction result with respect to $\mathcal{C}$, $\varphi(x_{wm})$ is indeed harmless and should not be considered as an effective false negative case.

Chapter 4

Methods of Watermarking, Detection, and Adversarial Attacks

4.1 Overview

We now describe how the watermark encoder, detector, and adversaries are designed within our proposed framework. We firstly explain our motivation of choosing watermark embedding positions in Section 4.2, and then describe the specific methods of watermarking and detection in Section 4.3 and Section 4.4. Lastly, we introduce the design of adversaries that adapt to our framework in Section 4.5.

4.2 Watermark Embedding Positions

Adversarial attacks introduce different amounts of perturbation to different frequency components of an image. As suggested in [46, 52], there are in general more perturbations in low frequency bands than in high frequency bands, at least for ImageNet models. This motivates us to embed watermark bits into DCT coefficients at low frequencies since large perturbations at low frequencies will likely destroy watermark bits embedded therein once the watermarked image is attacked.

To determine possible DCT coefficients where watermark bits can be embedded into, we performed an coefficient-wise perturbation analysis for 100,000 JPEG blocks in Luminance channel with respect to FGSM with $\epsilon = 8$. We collected the values of additive adversarial

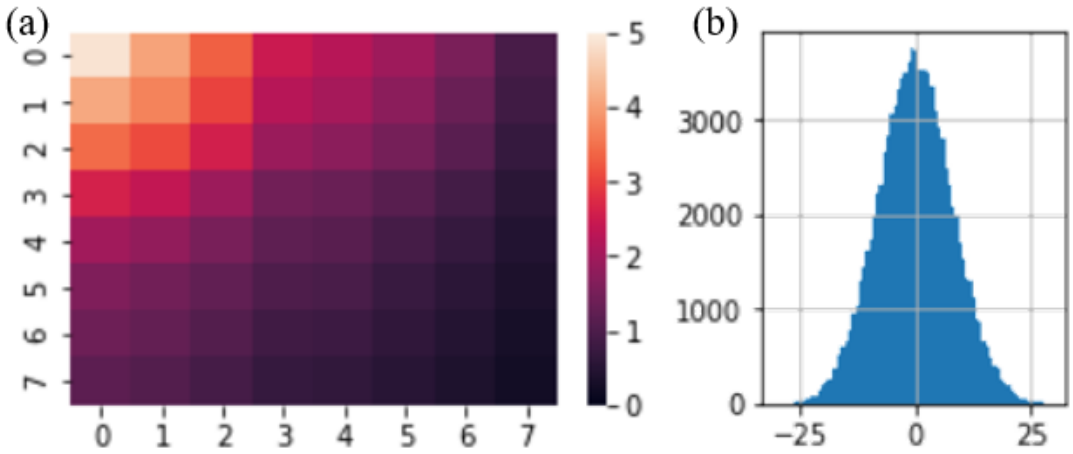


Figure 4.1: Coefficient-wise perturbation analysis for FGSM with $\epsilon = 8$ from the values of additive adversarial perturbations in 64 DCT coefficients of 100,000 JPEG blocks: (a) the standard derivation of perturbations per DCT coefficient; (b) distribution of perturbations at DC coefficient (DCT coefficient on the top-left corner). Note that the mean value of perturbations per DCT coefficient is more or less zero, and the perturbation energy is largely concentrated on low DCT frequencies.

perturbation in 64 DCT coefficients of these JPEG blocks. The standard derivation of perturbations per DCT coefficient is shown in Figure 4.1 (a). As can be seen from Figure 4.1 (a), the perturbation energy introduced by FGSM is largely concentrated on low frequencies, particularly the first 16 DCT coefficients in zigzag order. Also shown in Figure 4.1 (b) is the distribution of perturbations at DC, which is more or less a zero-centered Gaussian distribution. Based on this analysis, we shall select the first 16 DCT coefficients in zigzag order as possible embedding positions.

4.3 Watermarking Method

Our methods of watermarking, adaptive attacks, and detection applies the DCT Invariant Property Lemma 1 several rounds. Figure 4.2 (a) sketches the pipeline of our watermarking method. In our method, the secret information S is divided into three parts: the secret key, the watermark bits, and a special reference switch bit. The secret key is used for DCT coefficient selector to determine the embedding positions. The reference switch bit along with the key is used to determine a reference bit. With reference to the reference bit, the watermark bits are first differentially encoded and then embedded into the Least Significant Bit (LSB) of the selected DCT coefficients after the latter is further quantized with the respective quantization step size from the quantization table corresponding to $QF = 50$. Formally, for each 8×8 JPEG block, the watermark embedding process consists of the following 4 steps:

DCT coefficient selector The first step is to randomly select 5 DCT coefficients from 16 possible embedding positions. This selector requires a key of length $\lfloor \log_2 \binom{16}{5} \rceil$ for each block. Among the 5 selected DCT coefficients, the first one is used as a reference to determine, along with the special reference switch bit, the reference bit. The other 4 selected DCT coefficients are used for watermark embedding, namely the embedding positions. There are 4 watermarking bits per block, one for each embedding position. Therefore, the length of S per 8×8 JPEG block is $\lfloor \log_2 \binom{16}{5} \rceil + 5$.

Quantization of selected DCT coefficients Denote the DCT coefficient at the watermark embedding position i as $d(i)$. As the second step, we quantize $d(i)$ with the quantization step size $Q_{50}(i)$ from the quantization table corresponding to $QF = 50$ as follows:

$$D_{50}(i) = \lfloor \frac{d(i)}{Q_{50}(i)} \rceil. \tag{4.1}$$

Watermark embedding Given a watermark bit w to be embedded into the embedding position i, we differentially encode w with respect to the reference bit to derive an

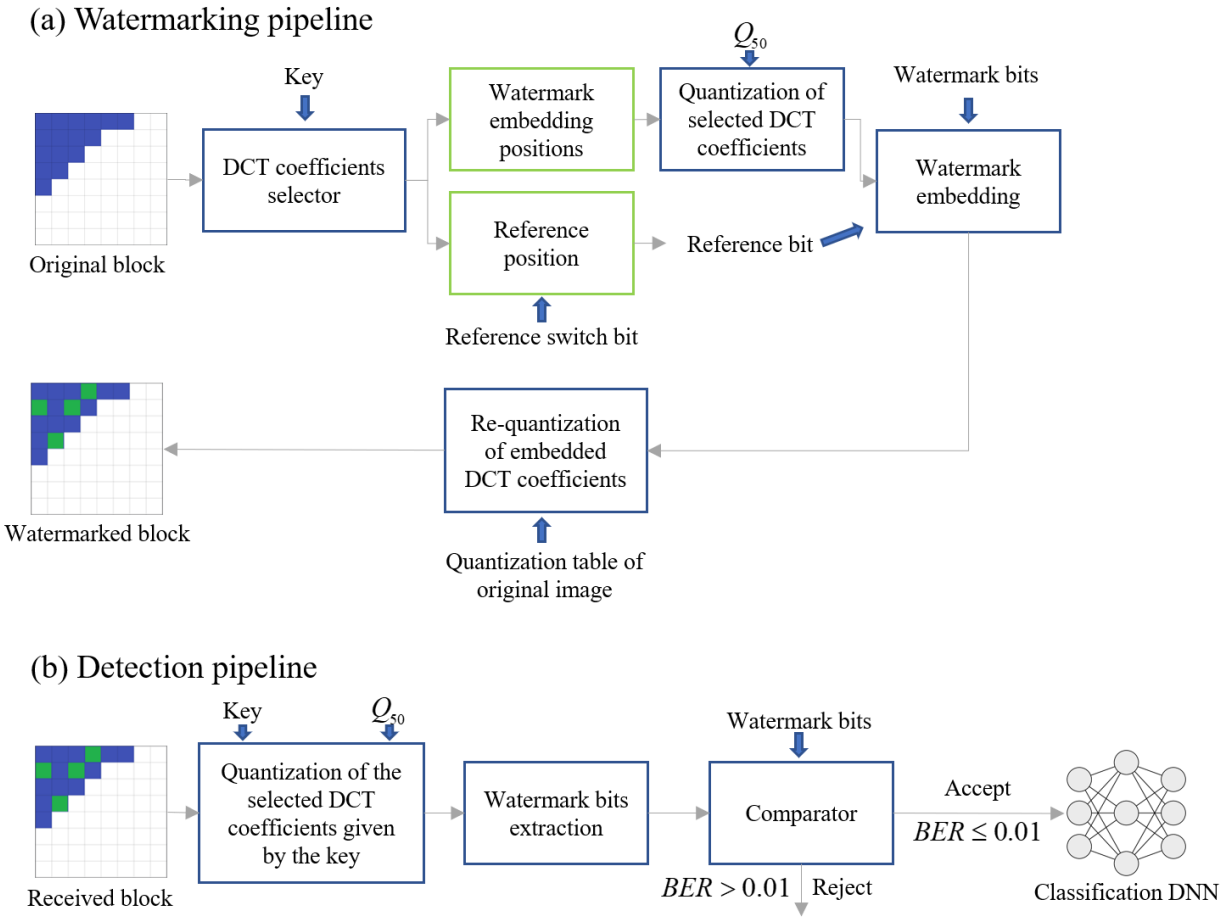


Figure 4.2: Illustration of the pipelines of watermarking and detection methods. The blue part of the original block represents the 16 possible embedding positions. The green part of the watermarked block represents the selected watermark embedding positions.

embedding value E. The bit w is then embedded into the position i by embedding E into the LSB of $D_{50}(i)$ as follows:

$$D_{50}^{wm}(i) = 2\lfloor D_{50}(i)/2 \rfloor + E, \tag{4.2}$$

where $D_{50}^{wm}(i)$ is the embedded DCT coefficient integer.

The calculation of E is another important operation in our method. Let $d(j)$ be the selected reference DCT coefficient. To determine the reference bit r for this block, we first quantize $d(j)$ with the quantization step size $Q_{50}(j)$ and then select r to be the second or third last bit of the quantized coefficient integer, depending on whether the special reference switch bit s is 0 or 1. Formally,

$$r = \lfloor \lfloor \frac{d(j)}{Q_{50}(j)} \rceil / 2^{s+1} \rfloor \bmod 2. \tag{4.3}$$

Finally, the embedding value E for the watermark bit w is determined by

$$E = r \oplus w. \tag{4.4}$$

The underlying rationale for embedding E rather than w directly is to prevent the adversary from directly accessing the watermark bits through the LSB of $D_{50}^{wm}(i)$. Otherwise, even though the selected watermark embedding positions are not known to the adversary, the adversary can iterate over all possible embedding positions and keep the corresponding watermark bits consistent with the watermarked image so as to bypass the detector without significantly deviating from desired adversarial examples (see Section 4.5 for more details).

Re-quantization of embedded DCT coefficient Finally, the embedded DCT coefficient needs to be re-quantized using the quantization table of the original image so as to keep the consistency of quantization. The watermarked and re-quantized DCT coefficient with respect to the original quantization step size $D_{ori}^{wm}(i)$ can be obtained from the following process:

$$D_{ori}^{wm}(i) = \lfloor D_{50}^{wm}(i) * Q_{50}(i)/Q_{ori}(i) \rceil, \tag{4.5}$$

where $Q_{ori}(i)$ is the quantization step size at position i in the quantization table of original image. Here we assume that the original quantization step size $Q_{ori}(i)$ is strictly less than $Q_{50}(i)$, which is the case in many applications.

Repeat this procedure for all JPEG blocks in the Luminance channel. The resulting image will be the watermarked image. Pseudo code of our watermarking method is shown in Algorithm 1. It took a single 4.5 GHz CPU approximately 0.3 second to compute (in Python) an watermarked image for ImageNet dataset.

Algorithm 1: Method of Watermarking.

Data: Original image; Secret key;
Result: Watermarked image;
Load the Luminance channel of the original image as Y;
Load Q_{50};
for *each* 8×8 *block in* Y **do**
    Determine the 4 watermark embedding positions and 1 reference position given by the key;
    Determine the reference bit r from the reference position using the reference switch bit s;
    for *each watermark embedding position* **do**
        Determine the embedding value E by differentially encoding its watermark bit with respect to the reference bit r;
        Quantizing the DCT coefficient using the quantization step size in the corresponding location of Q_{50} ;
        Embedding E into the LSB of the quantized DCT coefficient;
        Re-quantize the DCT coefficient using the quantization step size with respect to the original image;

4.4 Detection Method

Figure 4.2 (b) illustrates the pipeline of our detection method. Given a received block, which could be a watermarked block, a watermarked and attacked block, or a watermarked and JPEG re-compressed block, the detection method has three main steps:

Quantization with Q_{50} Determine the 5 selected embedding positions from the key. Quantize the DCT coefficients at those determined positions with Q_{50} to compute the respective $\hat{D}_{50}(i)$, in a way similar to Eq. (4.1). If the received block is not attacked, nor re-compressed, it follows from the DCT Invariant Property that $\hat{D}_{50}(i)$ will be equal to $D_{50}^{wm}(i)$.

Watermark bits extraction For each watermark embedding position, take the LSB of $\hat{D}_{50}(i)$ as the estimation $\hat{E}$ of E. Also, compute the estimation $\hat{r}$ of the reference bit r from the reference position according to Eq. (4.3) with $d(j)$ replaced by $\hat{d}(j)$. The extracted watermark bit $\hat{w}$ corresponding to the watermark bit w is then computed as

$$\hat{w} = \hat{E} \oplus \hat{r}. \tag{4.6}$$

Comparator For all the watermark embedding positions, we compare the extracted watermark bits with the original watermark bits and compute the Bit Error Rate (BER). The BER represents the percentage of embedded DCT coefficients that are significantly distorted and a larger BER means more distortion is added to the watermarked image. We empirically set a BER threshold of 0.01 to distinguish attacked images from benign images. The received image will be accepted by the detector only if it yields a $BER \leq 0.01$.

Finally, after the received image is accepted, its quantized version after the step 1 above is presented to the classification DNN.

4.5 Adversary Design

Regular adversaries usually work with valid RGB images with fixed image size. That is, input images to a regular adversarial attack algorithm normally take integer-valued pixel intensities and also have their size equal to the input size of the classification DNN. In order for regular adversaries to be able to work within our proposed framework, certain modifications are necessary. Below we first describe how to modify regular adversaries to their advantage so that they can work with JPEG images with various resolutions, and then further extend them to attack our detection strategy itself.

4.5.1 Modifications of Regular Adversaries

A regular adversary is modified via the following key steps:

No pixel rounding in JPEG decoding A regular adversary is modified to take JPEG images directly as their inputs. However, in the process of decoding a JPEG image into its RGB pixel intensities, real-valued pixel values will be kept without any rounding. This avoids possible damages caused by rounding on watermark bits.

Adaptation to various resolutions In our framework, it is necessary for the adversary to provide adversarial examples with the same size as the image to be attacked. To this end, we integrate the resizing process into the classification model as the front layer, which resizes the image to the model's input size. The adversary can then directly add adversarial perturbations into the image through either gradient-based attacks or optimization-based attacks for the integrated model. The resulting attacked images are adversarial examples for the integrated model, and after resizing, are also adversarial examples for the original model.

JPEG encoding Finally, the attacked images, after being JPEG compressed using the same quantization table as in the original image, are adversarial examples produced by the modified regular adversary.

4.5.2 Design of Type 1 Adaptive Adversary

The JPEG encoding step in a modified regular adversary may weaken its attack strength, especially for C&W-l2 adversary. To eliminate the negative impact of JPEG encoding on attack strength, we apply JPEG-resistant method proposed by Shin *et al.* [48] to strengthen modified regular adversaries including PGD, FGSM, and CW-l2, resulting our Type 1 adaptive adversaries.

In our pipeline, an static adversarial example is first JPEG encoded by the adversary for re-distribution, then JPEG decoded to RGB image for inference. To avoid the alleviation of adversarial effect, the adversarial constraint expressed in Equation 2.4 needs to be updated as follows:

$$\mathcal{C}(\textbf{Decoding}(\textbf{Encoding}(x+\delta))) \neq \mathcal{C}(\textbf{Decoding}(\textbf{Encoding}(x))). \tag{4.7}$$

Given the original trained network $\mathcal{C}$, we define $\mathcal{C}'(x) = \mathcal{C}'(\text{Decoding}(\text{Encoding}(x)))$. $\mathcal{C}'$ is constructed by adding JPEG encoding and decoding process as 2 layers to the beginning of $\mathcal{C}$. The encoding layer takes the original RGB image as input and output blocks of DCT coefficients, whereas the decoding layer takes the DCT coefficients as input and outputs the reconstructed RGB image. The task of Type 1 adaptive adversary is then defined as generating adversarial perturbation δ such that $x + \delta$ is an successful adversarial example for the integrated network $\mathcal{C}'$.

The JPEG compression as adversarial defense can be interpreted as an obfuscated gradient method [3]. Most of adversarial attacks rely on calculating the gradient, which is the partial derivative of the loss function with respect to each input elements. However, the

quantization process in JPEG encoding is a non-differentiable operation, which restricts the feasibility of these attacks. To address this problem, a common solution is Backward Pass Differentiable Approximation (BPDA) [3], which attempts to calculate the gradient by computing the forward pass normally and computing the backward pass using a differentiable approximation of $\mathcal{C}'$.

To build such approximation, each step in JPEG encoding and decoding process should be re-formulated to a form that is suitable for neural networks and optimization. In addition, the JPEG encoding and decoding process should be implemented as two TensorFlow layers which only involves differentiable operations defined by TensorFlow itself. As a result, the differentiation process when computing the backward pass will be automatically handled by TensorFlow. We referred to the method proposed by Shin *et al.* [48] to create these two customized layers, as the key steps of JPEG formulated as follows:

- **Color space conversion** The color space conversion from RGB to YCbCr is expressed as matrix multiplication as follows:

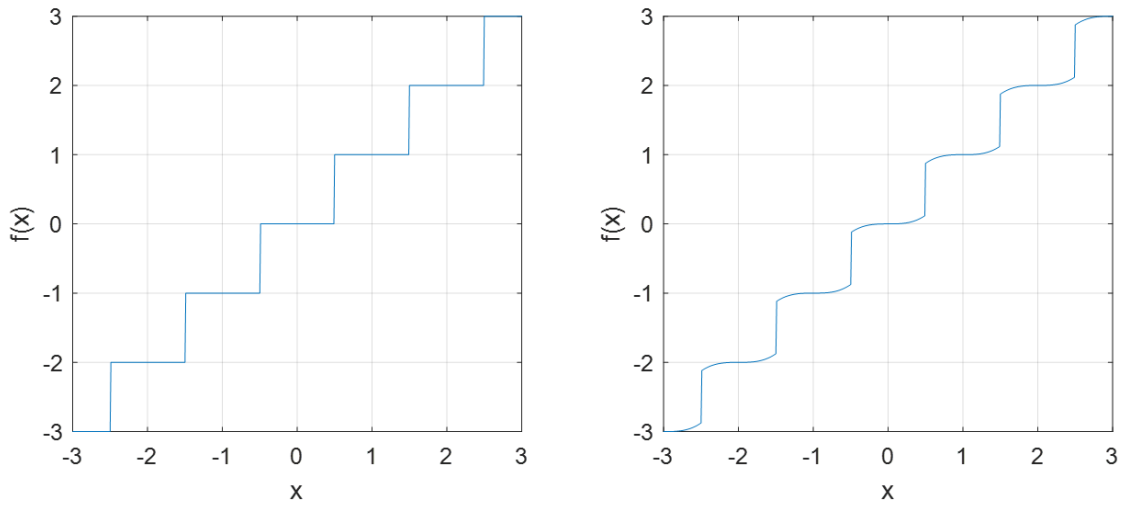
$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & -0.418688 & -0.081312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \tag{4.8}$$

- **Chroma subsampling** Observe that down-sampling Cb and Cr channels by merging every 2×2 region on the image is the same as performing a 2×2 average pooling with a stride of 2. This step is implemented by calling the pooling function pre-defined by TensorFlow.

- **Block splitting** Block splitting is involves 2 parts: padding the channels to the next multiple of the block size and dividing JPEG blocks. The first part is implemented by the padding function in TensorFlow by padding zeros to the bottom side and right side of the image, whereas the second part is implemented by matrix reshaping.

- **Block DCT** DCT itself is an differentiable transformation. It is then implemented by matrix operation and re-arranged to 8×8 JPEG blocks by zigzag scanning [48].

- **Block quantization** The quantization step expressed in Equation 2.1 involves 2 steps: perform an element-wise division over the DCT coefficients and rounding of these quantized coefficients. As a function of DCT coefficient $d(i)$, $D_{QF}(i) = \lfloor \frac{d(i)}{Q_{QF}(i)} \rceil$ has derivative 0 nearly everywhere, which is not suitable for generating adversarial examples. Authors



(a) Standard rounding function (b) Approximated rounding function

Figure 4.3: Comparison between standard rounding function and approximated rounding function.

in [48] proposed to use an approximation $\lfloor \cdot \rceil_{approx}$ instead of the standard rounding function $\lfloor \cdot \rceil$. The approximation is defined as follows:

$$\lfloor x \rceil_{approx} = \lfloor x \rceil + (x - \lfloor x \rceil)^3 \tag{4.9}$$

As illustrated in The difference between $\lfloor \cdot \rceil_{approx}$ and $\lfloor \cdot \rceil$ is compared in Figure 4.3. $\lfloor \cdot \rceil_{approx}$ is increasing monotonically, and has positive derivatives nearly everywhere, which is helpful for the gradient descent used in adversarial attack methods.

It is worth noting that the lossless compression part, including Run-Length coding and Huffman coding, are not implemented in this approximated JPEG because they are does not change the content of the image. After the encoding layer is finished, the decoding layer is constructed by reversing all these operations and their order.

4.5.3 Design of Type 2 Adaptive Adversary

Our watermark detection method only takes care of the embedded watermark bits and the reference bits of the received watermarked image, which are lower binary digits of

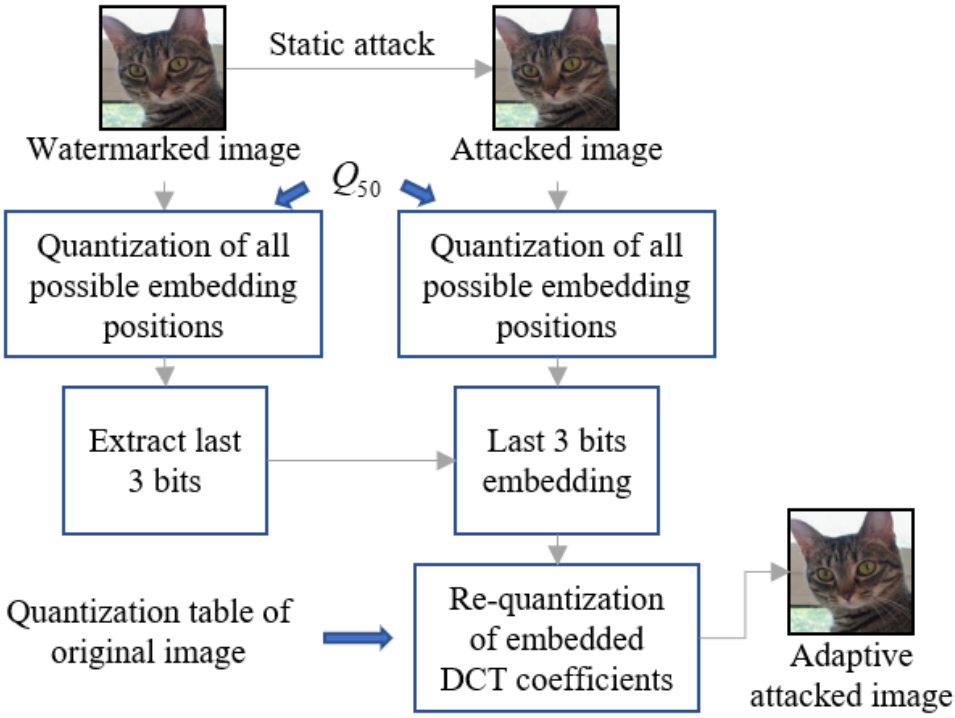


Figure 4.4: Illustration of the last step in a Type 2 adaptive adversary, which is a replacement of the JPEG encoding step in a modified regular adversary.

embedded DCT coefficients. As a result, there is still possible for an adaptive adversary to produce adversarial examples that only modifies the higher binary bits, while maintaining the lower ones. Thus, the watermark in the adversarial example is still retained, and the adversarial example will be passed to the DNN under protection.

Taking the advantage of the complete knowledge of our watermarking and detection methods, we define the Type 2 Adaptive Adversary, who is able to completely bypass the detector. It replaces the last JPEG encoding step in a modified regular adversary by the pipeline shown in Figure 4.4. The DCT Invariant Property guarantees that after both the watermarked image and the adaptive attacked image are quantized with quantization table Q_{50}, the quantized DCT coefficient integer of the DCT coefficient in the adaptive attacked image and its counterpart in the watermarked image have the same last three bits at each possible embedded position. As we shall see later, although this type of adaptive adversary completely bypasses the detector, it does not necessarily cause harm to the DNN to be secured.

Chapter 5

Experiment Result

5.1 Overview

In this chapter, we demonstrate the effectiveness of our watermarking-based framework against both static and adaptive adversarial attacks. The detailed experiment setup is introduced in Section 5.2.

We firstly show the effect of watermarking on classification accuracy and image quality in Section 5.3, followed by its robustness to JPEG re-compression 5.4. Then, we evaluate the effectiveness of our defense against a wide range of adversaries in the rest of this chapter.

5.2 Experiment Setup

We evaluated our proposed defense against adversarial examples on a subset of ImageNet ILSVRC 2012 validation dataset, which was formed by randomly choosing 1,000 images from the whole validation dataset. All selected images are classified correctly before and after watermarking by *ResNet50V2* [21]. (Otherwise, a new image would be selected and tested until this condition was satisfied.) Our adversarial images were produced by attacking *ResNet50V2*, a pre-trained DNN obtained from *Keras* [10]. Three representative targeted adversarial attack methods FGSM, PGD and C&W-l2 were selected. They were implemented with the reference implementations from the *CleverHans* package [38], which were slightly modified to accommodate the modifications mentioned in Section 4.5. The

DNN	Top1	Top1 wm	Top5	Top5 wm
ResNet50V2	67.00%	66.51%	87.81%	87.43%
MobileNetV2	70.85%	69.63%	89.80%	89.01%
InceptionV3	76.85%	76.66%	93.30%	93.00%

Table 5.1: Top-1 and Top-5 accuracy before and after watermarking for three pre-trained DNNs from *Keras* [10]. Note that the classification accuracy may be different from that reported by the original work due to different pre-processing methods.

targets for targeted attacks were randomly selected. The parameters selected for these attacks are described below:

- For FGSM and PGD, the adversarial perturbations are computed subject to an L_∞ constraint and the parameter ϵ controls the magnitude of maximum perturbation per pixel. To evaluate our framework under different perturbation levels, we employed targeted FGSM attacks with $\epsilon = 2, 4$ and 8, as well as targeted PGD attacks with $\epsilon = 8$. The selected parameters are commonly used values in other studies [12].
- For C&W-l2, the adversarial perturbations are optimized under L_2 constraint. Its hyper-parameter κ specifies the confidence that the adversarial image is misclassified by the target DNN, and also controls the amount of perturbations. The smaller κ, the smaller perturbations. Since small perturbations are difficult to be detected, we employed targeted C&W-l2 attacks with $\kappa = 0$ to evaluate our proposed defense strategy in the worst case scenario. It is worth noting that this is the worst case scenario for C&W-l2 attack in terms of parameter κ.

5.3 Classification Accuracy and PSNR

Table 5.1 shows the top-1 and top-5 accuracy before and after watermarking of ResNet50V2, MobileNetV2, InceptionV3 over the entire ImageNet ILSVRC 2012 validation dataset. The impact of watermarking on classification accuracy is indeed insignificant with, on average, 0.63% and 0.49% degradation in top 1 and top 5 accuracy, respectively. The PSNR between the original and watermarked images is found to be 39.34 ± 1.13 dB. As shown in Figure 5.1, the watermark distortion is imperceptible, and significantly less than the adversarial perturbation.

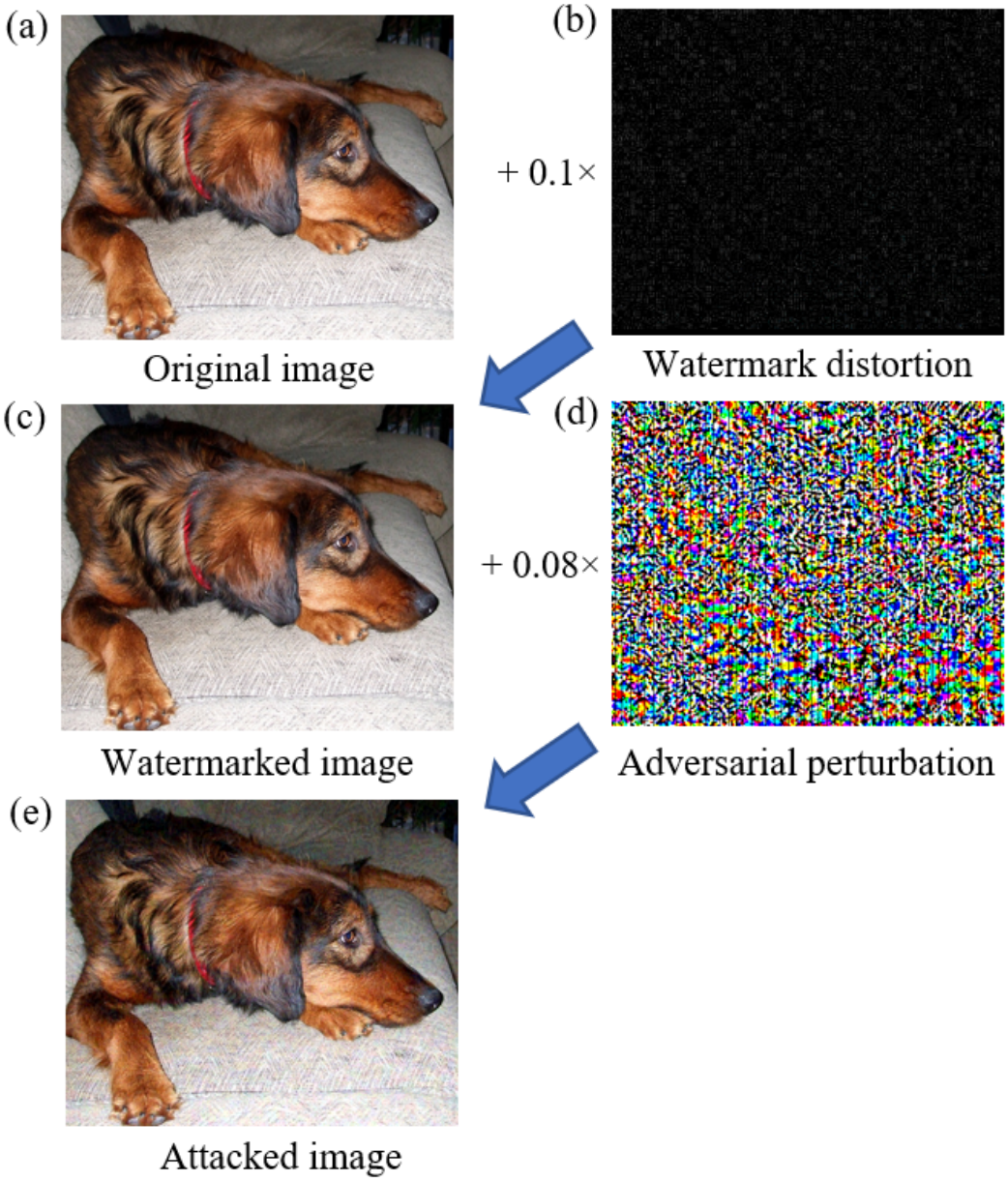


Figure 5.1: An example of watermarking and adversarial perturbation: (a) the original image; (b) watermark distortion (amplified 10 times); (c) the watermarked image; (d) the adversarial perturbation generated by FGSM with $\epsilon = 8$; and (e)the FGSM attacked image.

Rounds of JPEG Re-compression	1	2	3	5
Average BER	0.00067	0.0082	0.021	0.038

Table 5.2: Average BER for different rounds of JPEG re-compression with random QF.

Rounds of JPEG Re-compression	1	2	3	4	5
Average BER	0	0	0.009	0.040	0.044

Table 5.3: Average BER for different rounds of JPEG re-compression with descending QF.

5.4 Robustness to JPEG Re-compression

In our framework, the watermarking and detecting method are designed to be robust to JPEG re-compression. With multiple rounds of re-compression, the error will gradually accumulate and eventually endanger the classification of benign examples. To evaluate the watermarking robustness to JPEG re-compression and justify our choice of BER threshold, two sets of experiments are conducted.

In the first set of experiment, we simulated the situation in daily use, where watermarked images are compressed by multiple rounds of JPEG re-compression with each QF randomly selected from $[50, 100)$ before feeding to the detector. Table 5.2 reports the respective average BER of the detector in each case. It is clear from Table 5.2 that our watermarking method is indeed very robust to one or two rounds of high quality JPEG re-compression. The results in Table 5.2 also justify empirically our choice of BER threshold 0.01.

As proved in 1, if the secondary JPEG re-compression uses a greater QF than the previous one, the quantized DCT coefficient can be precisely reconstructed. As a result, in the second set of experiments, the choice of QFs used in different rounds of JPEG re-compression are changed to be descending order. The QF for the first round is fixed to be 90, and the QF used in the following round is decreased by 10 compared with the previous round. As a result, a maximum of 5 additional JPEG re-compression are performed, where the corresponding QFs are 90, 80, 70, 60 and 50, respectively. The average BER results are shown in Table 5.3. As shown in Table 5.3, the BER is almost zero with QFs greater than 70, indicating strong robustness. As the number of rounds increases and QF decreases, the error quickly accumulates and results in a near 0.044 BER after

Overall, the selected BER threshold 0.01 is good for 2 rounds of JPEG re-compression

Metric	FGSM $\epsilon = 2$	FGSM $\epsilon = 4$	FGSM $\epsilon = 8$	PGD $\epsilon = 8$
Detection rate	99.7%	100.0%	100.0%	100.0%
EFNR	0.2%	0.0%	0.0%	0.0%

Table 5.4: Detection rate and effective false negative rate in the case of static white-box FGSM and PGD attacks.

Metric	FGSM $\epsilon = 2$	FGSM $\epsilon = 4$	FGSM $\epsilon = 8$	PGD $\epsilon = 8$
Detection rate	99.8%	100.0%	100.0%	100.0%
EFNR	0.1%	0.0%	0.0%	0.0%
Detection rate	0.0%	0.0%	0.0%	0.0%
EFNR	0.1%	0.0%	0.4%	1.5%

Table 5.5: Detection rate and effective false negative rate in the case of adaptive white-box FGSM and PGD attacks: top for type 1 adaptive adversary; bottom for type 2 adaptive adversary.

with random QF or 3 rounds of JPEG re-compression with descending QF. In most pipelines, it is of little chance for image data to be compressed multiple times.

5.5 Combating FGSM and PGD Attacks

Table 5.4 and Table 5.5 show the detection rate and effective false negative rate in the case of static FGSM and PGD attacks, and in the case of adaptive FGSM and PGD attacks, respectively. Clearly, the detector can effectively detect adversarial perturbations introduced by static and type 1 adaptive FGSM and PGD attacks. Although type 2 adaptive FGSM and PGD can bypass the detector completely by design, they are nonetheless harmless to the subsequent classification DNN with near zero EFNR. The quantization process with Q_{50} along with forcing the last three bits at each possible embedding position to be the same as those of the counterpart in the watermarked image essentially undoes the adversarial perturbation.

For type 2 adaptive, we also conducted a case study using PGD, in which $\epsilon = 64$. The increased perturbation does help to construct a successful adversarial example, however,

	Static	Type 1	Type 2
Detection rate	34.1%	38.4%	0%
EFNR	4.7%	25.3%	0.0%

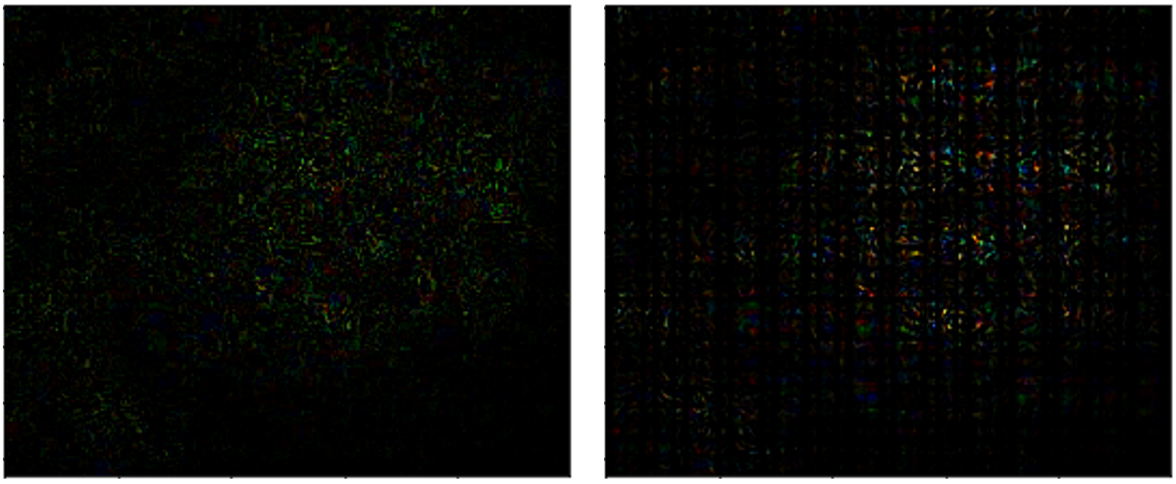
Table 5.6: Detection rate and EFNR in the case of static, type 1 adaptive, and type 2 adaptive C&W-l2 attacks.

this amount of perturbation is way too large to be imperceptible to human eyes and severely damaged the visual content of the image. Hence, in this case, our defense makes generation of adversarial examples much harder and significantly increased the desired perturbation budget.

5.6 Combating C&W-l2 Attack

C&W-l2 attack [7] provides strong adversarial examples with high confidence under a tight perturbation budget. The light perturbation increased the difficulty of detecting it. However, as suggested in [20, 12], adversarial perturbation generated by C&W-l2 method is fragile to JPEG compression. That is, JPEG compression can effectively filter out the adversarial perturbation and recover the original classification result. The results for both static and adaptive white-box C&W-l2 attacks are shown in Table 5.6. The relatively-low detection rate of static C&W-l2 attack suggests that the watermarked coefficients are barely distorted after quantization. On the other hand, converting adversarial examples to JPEG format also significantly decreases the effectiveness of the static attack, resulting in a low effective false negative rate.

With the JPEG-resistant feature, type 1 adaptive C&W-l2 yielded a higher effective false negative rate at 25.3%. It also increased the distortion required and resulted in a higher detection rate. A comparison of C&W-l2 adversarial perturbation generated static and type 1 adaptive adversaries are shown in Figure 5.2. In Figure 5.2(b), it is clear that the adversarial perturbation is mostly distributed to some DCT coefficients of each blocks, showing a apparent division of blocks. This helps the adversarial perturbation to be preserved after DCT quantization process. In addition, both adversarial perturbation shown in Figure 5.2 concentrates on similar regions. The type 1 adaptive adversary introduces more distortion measure in L_2norm to the benign example, resulting in a slightly increased detection rate, however, the improvement of its attack success rate also increased the EFNR to 25.3%. In terms of type 2 adaptive adversary, both detection rate and EFNR



(a) Static adversarial perturbation (b) Adaptive adversarial perturbation

Figure 5.2: Comparison of adversarial perturbation generated by adaptive and static C&W-l2 attack (amplified 250 times).

are 0% for the same reason described in Section 5.5.

To combat C&W-l2 attack, an effective way is to decrease the BER threshold of the detector. Table 5.7 shows the detection rate and effective false negative rate in the case of type 1 adaptive C&W-l2 attacks for different BER thresholds. When the BER threshold is set to 0.0025, the detection rate increases significantly from 38.4% to 85.1%, whereas the effective false negative rate decreases significantly from 25.3% from 5.8%. The improved performance is at the cost of watermarking robustness to multiple rounds of high quality JPEG re-compression. With the BER threshold at 0.0025, our watermarking method is

BER threshold	0.0025	0.005	0.0075	0.01
Detection rate	85.1%	67.5%	54.6%	38.4%
EFNR	5.8%	12.3%	16.7%	25.3%

Table 5.7: Detection rate and effective false negative rate in the case of type 1 adaptive C&W-l2 attacks for different BER thresholds.

very robust to only one round of JPEG re-compression.

Chapter 6

Conclusion and Future work

6.1 Conclusion

This research thesis presents a watermarking-based framework for adversarial example detection. Most previous detection-based methods are dedicated to finding general intrinsic properties of the adversarial sample compared to benign examples, so as to classify the input image, however, most of them cannot resist adaptive adversarial attacks if their defense method is known to the adversary. In this thesis, we focus on the adversarial perturbation itself and present a semi-fragile watermarking scheme, in which a watermark is embedded to the benign image upon acquisition, then released to an adversarial environment. Before the watermarked image is fed into the DNN under protection, a detector will check the watermarking bits of the watermarked image, and pass the image to the DNN only if the watermarking bits can be recovered with high precision. As a result, we can effectively monitor possible adversarial attacks during image re-distribution.

Specific methods of semi-fragile watermarking and detection are also presented. When designing the watermarking method, four design principles are considered:

- Watermark should be invisible to human eyes;
- Watermark should not introduce too much distortion or degrade classification accuracy of the neural network;
- Watermark should be robust to pre-defined legal operations, which is defined as high-quality JPEG compression in our study;

- Watermark should be severely distorted after adversarial attacks.

Based on statistical properties of both adversarial perturbation and JPEG compression, we determined the first 16 DCT coefficients of each JPEG block as possible embedding positions. Among them, 4 positions are randomly selected and embedded with a image-independent watermark bit. Both the selection of embedding position and the value of watermark bit is determined by a secret key, which is blind to the adversary.

The proposed method is evaluated on a subset of ImageNet validation dataset against a wide range of advanced attacks (FGSM, PGD and C&W-l2 attacks). Two types of adaptive adversaries are also introduced and evaluated. It has been shown by experiment that the framework is effective against a wide range of advanced attacks (static and adaptive), achieving a near zero (effective) false negative rate with the guaranteed zero false positive rate. At the same time, the impact of watermarking on classification accuracy of DNNs is insignificant.

6.2 Future Work

This work can be further proved in several ways. In the following subsections, we discuss some of these possible improvements.

6.2.1 Reconstruct Adversarial Examples to Benign Images using Watermarking Information

Once the watermark is extracted from the received image, not only we can calculate the BER, but also the regions where the watermark is distorted can be determined. Adversarial perturbation generally focus on edges and textures of the image for the reason that these regions contributes the most to the model's output. If we can eliminate the adversarial perturbations in these regions or make better use of the rest of the image, we can partially restore the value of the image. With the help of the watermark, we can localize such regions with adversarial perturbations. therefore, we can mask or reconstruct these regions so as to make use of the images again. For example, for each JPEG blocks, we determine this block as perturbed block if any one of the four watermark bits is distorted. Afterwards, these perturbed blocks can be reconstruct by several possible ways, e.g. low quality JPEG compression or GAN-based image completion [8] [58].

6.2.2 Reduce the Amount of Blocks That Need To Be Watermarked

In our proposed the method, each JPEG blocks are treated equally with 4 watermarking bits embedded. In this case, all the blocks contributes equally to the BER. This is consistent with JPEG compression, however, can be optimized for adversarial attacks.

Given a perturbation budget, most adversarial attack methods tends to put more perturbation in the important regions, which normally are edges and textures of the main objects. In addition, with the same amount of perturbation, these blocks make more difference to the model's output. Thus, if we only deploy our watermark in these regions, the sensitivity of the watermark can be increased, whereas the watermarking distortion can be decreased. There are quite a few possible ways to determine such important regions as a saliency map or producing regions of interest (ROI), including Grad-CAM [43] and bounding boxes used in Faster R-CNN [40].

6.2.3 Justify the Selection of Hyper-Parameters

In this thesis, several empirical choices were made in the methodology, including possible embedding positions, the number of watermark bits per block, and the acceptable BER threshold. Although their influence on the overall performance against different adversarial attacks is partially examined and discussed, it is important to justify the basis of these choices by conducting comparative experiments.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] Moustafa Alzantot, Yash Sharma, Supriyo Chakraborty, Huan Zhang, Cho-Jui Hsieh, and Mani B Srivastava. Genattack: Practical black-box attacks with gradient-free optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1111–1119, 2019.

[3] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018.

[4] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018.

[5] 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.

[6] 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.

[7] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[8] Vaishnav Chandak, Priyansh Saxena, Manisha Pattanaik, and Gaurav Kaushal. Semantic image completion and enhancement using deep learning. In *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–6. IEEE, 2019.

[9] Jianbo Chen and Michael I Jordan. Boundary attack++: Query-efficient decision-based adversarial attack. *arXiv preprint arXiv:1904.02144*, 2(7), 2019.

[10] François Chollet et al. Keras. https://keras.io, 2015.

[11] Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalker. *Digital watermarking and steganography*. Morgan kaufmann, 2007.

[12] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Siwei Li, Li Chen, Michael E Kounavis, and Duen Horng Chau. Shield: Fast, practical defense and vaccination for deep learning using jpeg compression. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 196–204, 2018.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[14] Guneet S Dhillon, Kamyar Azizzadenesheli, Zachary C Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Anima Anandkumar. Stochastic activation pruning for robust adversarial defense. *arXiv preprint arXiv:1803.01442*, 2018.

[15] Anna Egorova and Victor Fedoseev. Semi-fragile watermarking for jpeg image authentication: A comparative study. In *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–6. IEEE, 2019.

[16] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018.

[17] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[18] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1994.

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

[20] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[22] 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.

[23] Dan Hendrycks and Kevin Gimpel. Early methods for detecting adversarial images. *arXiv preprint arXiv:1608.00530*, 2016.

[24] Uiwon Hwang, Jaewoo Park, Hyemi Jang, Sungroh Yoon, and Nam Ik Cho. Puvae: A variational autoencoder to purify adversarial examples. *arXiv preprint arXiv:1903.00585*, 2019.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] Donald Knuth. *The TEXbook*. Addison-Wesley, Reading, Massachusetts, 1986.

[27] Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. Adversarial examples in the physical world, 2016.

[28] Leslie Lamport. *LATEX — A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

[29] Xin Li and Fuxin Li. Adversarial examples detection in deep networks with convolutional filter statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5764–5772, 2017.

[30] Ching-Yung Lin and Shih-Fu Chang. Semifragile watermarking for authenticating jpeg visual content. In *Security and Watermarking of Multimedia Contents II*, volume 3971, pages 140–151. International Society for Optics and Photonics, 2000.

[31] Ching-Yung Lin and Shih-Fu Chang. A robust image authentication method distinguishing jpeg compression from malicious manipulation. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(2):153–168, 2001.

[32] 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.

[33] 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, 2017.

[34] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017.

[35] 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*, pages 2574–2582, 2016.

[36] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[37] Ridvan Ozdemir and Mehmet Koc. A quality control application on a smart factory prototype using deep learning methods. In *2019 IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT)*, volume 1, pages 46–49. IEEE, 2019.

[38] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang, Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg, Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, and Rujun Long. Technical report on the cleverhans v2.1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2018.

[39] 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, 2017.

[40] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[41] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018.

[42] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[43] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[44] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *arXiv preprint arXiv:1904.12843*, 2019.

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

[46] Yash Sharma, Gavin Weiguang Ding, and Marcus Brubaker. On the effectiveness of low frequency perturbations. *arXiv preprint arXiv:1903.00073*, 2019.

[47] Shiwei Shen, Guoqing Jin, Ke Gao, and Yongdong Zhang. Ape-gan: Adversarial perturbation elimination with gan. *arXiv preprint arXiv:1707.05474*, 2017.

[48] Richard Shin and Dawn Song. Jpeg-resistant adversarial images. In *NIPS 2017 Workshop on Machine Learning and Computer Security*, volume 1, 2017.

[49] 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.

[50] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[51] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[52] Yusuke Tsuzuku and Issei Sato. On the structural sensitivity of deep convolutional networks to the directions of fourier basis functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 51–60, 2019.

[53] Javier Villalba-Diez, Daniel Schmidt, Roman Gevers, Joaquín Ordieres-Meré, Martin Buchwitz, and Wanja Wellbrock. Deep learning for industrial computer vision quality control in the printing industry 4.0. *Sensors*, 19(18):3987, 2019.

[54] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[55] Jun Xiao, Zhiqiang Ma, Bai Lin, Jiaorao Su, and Ying Wang. A semi-fragile watermarking distinguishing jpeg compression and gray-scale-transformation from malicious manipulation. In *2010 IEEE Youth Conference on Information, Computing and Telecommunications*, pages 202–205. IEEE, 2010.

[56] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*, 2017.

[57] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[58] Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. Pluralistic image completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1438–1447, 2019.