A Study on Defense Against Adversarial Examples and Unauthorized Access for Convolutional Neural Networks

March, 2022

April Pyone Maung Maung

Tokyo Metropolitan University

Table of Contents

Chapter 1

Introduction

Artificial intelligence (AI) has emerged into many applications and impacts nearly all aspects of our lives. AI is not just for video games anymore. In all major applications such as search engines, recommendation systems, social networks, etc., AI is lurking in the background. With a tremendous developing speed, many more innovative applications of AI are expected to come. Therefore, reliability is essential for AI-based systems, especially in adversarial settings. This thesis deals with defense against a specific adversarial attack (adversarial examples) and unauthorized access for convolutional neural networks which is a deep learning architecture, which is, in turn, a kind of machine learning, that is mainly used for many approaches in AI.

This chapter provides background and overview of the thesis which covers motivation, issues to be addressed, and contributions of the thesis. The chapter also describes the structure of the thesis with an outline.

1.1 Background

The amount of data produced is growing exponentially. With the abundance of data, machine learning (ML) has entered ubiquitous computing. Many applications nowadays are not conceivable without ML. Smart devices such as smartphones, smartwatches, etc. are now equipped with ML-powered applications. As ML has become a prevalent tool for many applications, it is necessary to investigate how ML techniques perform when they are exposed to adversarial conditions, which is known as *adversarial machine learning* [1, 2].

Particularly, one ML technique, also known as *deep learning* has brought groundbreaking developments in pattern-recognition technology. Some notable examples where deep learning excels are visual recognition [3], natural language processing [4], and

speech recognition [5]. Deep learning learns a representation of data with multiple levels of abstraction from simple non-linear modules [6]. Learning procedure in deep learning is done by the use of neural networks, and the word "deep" refers to multiple layers of neural networks. Different layers in the networks learn different representations; the learned features from low-level layers represent low-level features and the ones from high-level are high-level features [7]. For image classification tasks, the first layer learns the presence or absence of edges at particular orientations and locations in an image, the second layer detects motifs, the third layer may assemble motifs into larger combinations, and so on [6]. Unlike traditional machine learning, these learned features are not handcrafted by human engineers but are automatically discovered by using the backpropagation algorithm in deep learning. The architecture of the neural network and the activation function used in each layer vary from application to application. Usually, convolutional neural networks (CNNs) are used for vision-related tasks and recurrent neural networks (RNNs) are effective for sequential data. This thesis focuses on CNNs.

Traditionally fully connected multi-layer perceptrons (MLPs) are invariant to the order of features, discarding the spatial structure of the pixels. In contrast, CNNs are specifically designed to detect spatial features. Moreover, CNNs are computationally efficient because they require fewer parameters than MLPs, and they are convenient to parallelize in graphics processing units (GPUs). CNNs are inspired by the human visual system and are translation invariant. The earliest layers of CNNs focus on local regions, rather than the contents of the image. Recent advances in CNNs have brought major breakthroughs in computer vision [6]. Impressively, the last ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2017 proved that the image classification accuracy has surpassed the level of human performance (i.e., an error rate of 2.25 %).

The image classification problem is the task of classifying an input image into a class category from a fixed set of categories according to its visual content. The image classification task is one of the core problems in computer vision, and other tasks such as object detection and segmentation can be reduced to image classification. As CNNs excel in exploring the structure of image data, CNN-based architectures are ubiquitous in the field of computer vision and provides state-of-the-art results for the image classification task. Therefore, CNNs have dominated visual recognition systems in many different applications.

Despite the remarkable performance, modern CNN architectures such as residual network (ResNet) are known to be sensitive to small image transformation [8]. In particular, carefully perturbed data points known as adversarial examples are indistinguishable from clean data points but can cause CNNs to make erroneous predictions with high confidence [9, 10]. Adversarial examples have raised security and reliability concerns, since CNNs are deployed in security-critical applications such as autonomous

vehicles, healthcare, and finance. As adversarial examples are an obvious threat, numerous methods for generating adversarial examples (attack methods) and defenses against them have been proposed in the literature [11]. However, there is no defense method that provides a high classification accuracy.

Regardless of being vulnerable to adversarial examples, CNNs are deployed in many practical applications. In addition, trained CNN models are shared online via different platforms such as Tensor Hub [12] for research and development purposes. The shared models can be fine-tuned, and potentially be commercialized and monetized. Moreover, training a production-level CNN model is not trivial, and requires a huge amount of data, efficient algorithms, and fast computing resources (e.g., GPUs). Therefore, a trained model can be regarded as a new intellectual property (IP). It is challenging and demanding to protect trained CNN models from unauthorized and illegal activities such as access control and copyright infringement.

1.2 Thesis Overview

1.2.1 Motivation

This thesis focuses on the security of CNNs in the context of the image classification tasks. In particular, this thesis addresses adversarial examples, investigates attacks and defenses, and develops new solutions to counter adversarial examples from different perspectives in a well-defined threat model. In addition, the solution proposed for the adversarial defense is further extended to protect trained CNN models from unauthorized access as new applications (model access control and model watermarking) in this thesis.

As CNNs are deployed in security-critical applications such as autonomous vehicles, healthcare, and finance, security and safety for such applications are crucial, and under scrutiny. Incorrect decisions made by CNNs can cause serious and dangerous problems. For example, self-driving cars may misclassify a "Stop" sign as "Speed Limit" [13], and face recognition systems may authenticate unauthorized users as authorized ones [14–17].

Adversarial attacks and defenses have entered into an arms race in the literature, and attacks methods are ahead of defense ones. Once defense mechanisms are known to the attacker, adaptive attacks can be carried out [18, 19]. In particular, input transformation-based defenses can be defeated due to obfuscated gradients [18]. Moreover, adversarially trained models (with maximum norm-bounded perturbation) can be attacked by Taxicab norm-bounded adversarial examples [20]. Although certified defenses are attractive, they can be bypassed by generative perturbation [21] or parametric

perturbation (outside of pixel norm ball) [22]. Therefore, state-of-the-art adversarial defenses are not yet satisfactory, and defense against adversarial examples remains an open problem for adversarial machine learning research.

The study of adversarial examples goes beyond security implications. The phenomenon of adversarial examples demonstrates that models are not learning the underlying concepts in a robust manner [23]. Adversarial examples show where models can fail and can be used to understand the models better. In addition, the notion of adversarial examples can measure the gap between humans and machines.

Besides, in general, deep learning models including CNN ones are shared online through platforms such as Model Zoo [24], Azure AI Gallery [25], and Tensor Hub [12] for research and development purposes. A recent security assessment of on-device models from Android applications showed that many mobile applications fine-tuned pre-trained models from Tensor Hub [26]. Since models from such sharing platforms are widely used in real-world applications, it is necessary to properly credit model owners. In addition, training a CNN model is not trivial, and requires a huge amount of data, efficient algorithms, and fast computing resources (e.g., GPUs). Therefore, successful trained CNN models have great business values for potential commercialization. To prevent model thefts and copyright infringement, researchers have proposed model access control [27] and model watermarking [28–34] methods. However, model protection is still in its infancy and there are issues to be addressed such as piracy attacks [32,35,36].

1.2.2 Issues to be Addressed

As motivated above, this thesis identifies the following issues:

- Issue 1: Conventional adversarial defenses reduce classification accuracy, or are completely broken. Once knowledge of defense methods is available, attackers can design adaptive attacks and defeat the defense methods [18, 19]. Figure 1.1 illustrates an image classification scenario where an adversarial attack can be successful even on a model with an adversarial defense. Therefore, state-of-the-art adversarial defenses are not yet satisfactory, and new defenses that can maintain a high classification for both plain images and adversarial examples are desired and demanding.

- Issue 2: As CNN models have great business values, protection of models from unauthorized access has become increasingly important. The consequences of stolen models can lead to not only economic damage, but also other threats such as model inversion attacks [37] and adversarial attacks [9]. A scenario of a stolen model is depicted in Fig. 1.2. There is only one prior work that uses a secret

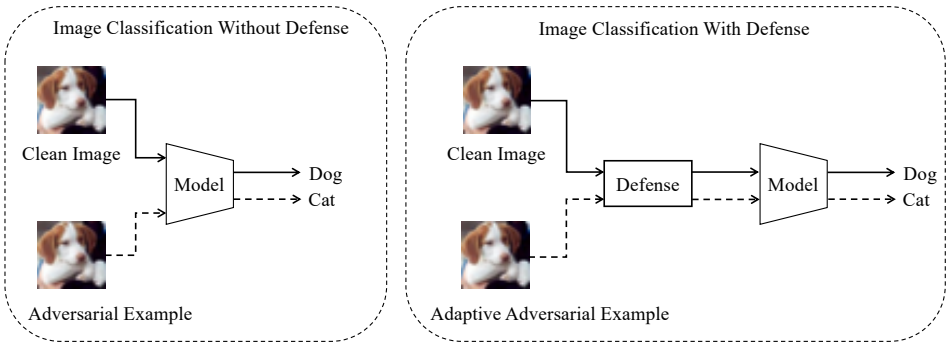

Figure 1.1: Image classification scenario where an adversarial example can fool a model even in the presence of an adversarial defense.

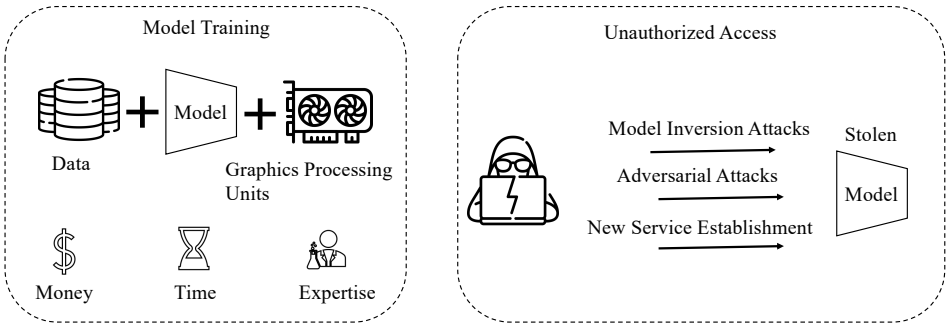

Figure 1.2: Scenario of consequences of stolen model.

perturbation network to protect the functionality of the models [27]. This method slightly reduces classification accuracy and requires training an extra network.

- Issue 3: CNN models are shared online via different platforms for research and development purposes. A recent study shows that many Android applications are fine-tuned from pre-trained models [26]. Therefore it is necessary to properly credit model owners. Although there are many model watermarking methods to address this issue, conventional methods are vulnerable to attacks such as piracy attacks and ambiguity attacks [32, 35, 36]. A scenario of copyright violation of a trained CNN model is shown in Fig. 1.3.

1.2.3 Contributions

To maintain a high classification accuracy whether or not the model is under attack, and to protect the model from unauthorized access, this thesis develops two novel

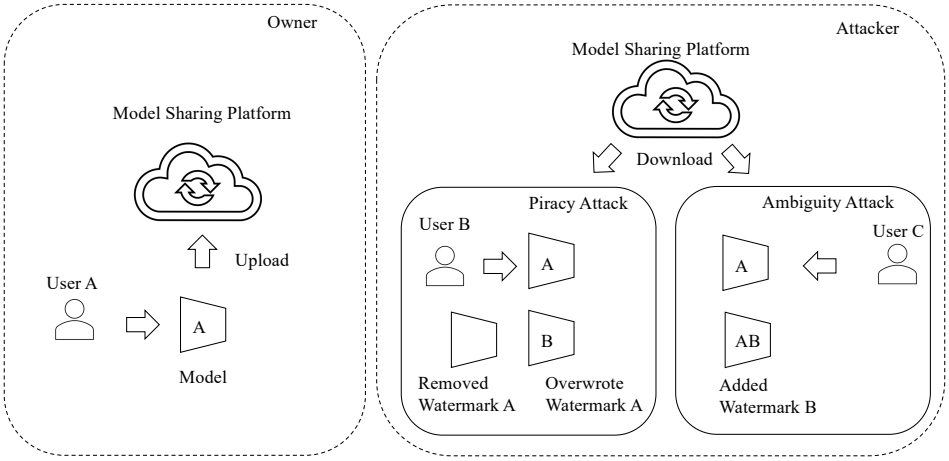

Figure 1.3: Scenario of copyright violation of model where ambiguity and piracy attacks can be carried out.

defense frameworks against adversarial examples and extends the defense technique to model access control and model watermarking applications. All the work here has been published or accepted at peer-reviewed venues. Table 1.1 summarizes the main contributions of this thesis that consists of:

- **Defense by Double Quantization [38] (Chapter 3):** On Issue 1, a novel defense framework in a restricted condition that maintains exactly the same accuracy whether or not the model is under attack is proposed. Assuming only 1–bit images are available to attackers, the defense framework utilizes linear quantization to remove the adversarial noise, and uses dithering to improve classification accuracy. Empirical results were presented comparing with state-of-the-art methods.

- **Defense by Block-Wise Transformation [40] (Chapter 4):** On Issue 1, another novel defense framework that is more general, applicable to 8–bit images, and maintains a high classification accuracy on both plain images and adversarial examples is proposed by following the second Kerckhoffs's cryptographic principle. The defense framework introduces three different transformations by taking inspiration from perceptual image encryption methods. Empirical results on adaptive and non-adaptive attacks were presented comparing with state-of-the-art methods.

- **Model Protection by Block-Wise Transformation [44–46] (Chapter 5):** On Issues 2 and 3, two model access control frameworks and one model wa-

Table 1.1: Contributions of thesis

Method	Assumption	Threat Model	Performance	Conventional Methods
Defense [38]	1–bit Images	ℓ_∞, ℓ_2	Superior	[23, 39]
Defense [40]	Secret Key	$\ell_\infty, \ell_2, \ell_1$	Superior	[41–43]
Model Access Control [44]	Secret Key	Key Estimation, Fine-tuning	Superior	[27]
Model Access Control [45]	Secret Key	Key Estimation, Fine-tuning	Superior	[27, 44]
Model Watermarking [46]	Secret Key	Pruning, Fine-tuning	Similar	[36]

termarking framework are put forward by adopting the defense technique by the block-wise transformation with a secret key. The first model access control framework utilizes the block-wise transformation to input images, and the second one deploys a specific block-wise transformation, pixel shuffling to one or more feature maps of the network (Issue 2). The model watermarking approach employs the negative/positive transformation in a block-wise manner to embed a watermark (Issue 3). Various experiments and relevant attacks were carried out to evaluate the model protection methods and results were presented comparing with state-of-the-art methods in this thesis.

1.3 Thesis Structure

The thesis is structured according to the outline in Fig. 1.4.

Chapter 1 provides a background of the thesis, an overview of the thesis including the motivation, issues to be addressed, and the contributions of the thesis. It also describes the structure of the thesis.

Chapter 2 discusses the security of neural networks in general. Then, it focuses on adversarial examples; it describes threat models in detail, and surveys recent attacks and defenses in a comprehensive way. It also discusses the intellectual property of deep neural networks.

Chapter 3 introduces a novel adversarial defense framework that uses double quantization for a scenario involving restricted 1-bit images. It also discusses previous related work and describes issues. Next, it presents experiments and results in comparison with state-of-the-art methods. In addition, it discusses the justification and limitations of the defense framework. This framework is effective and maintains an identical accuracy whether or not the model is under attack for restricted 1-bit images.

Chapter 4 puts forward a new adversarial defense framework with a secret key that is more general and applicable to 8-bit images. The chapter discusses previous related work and addresses the issue of a low classification accuracy. Three different transformations for the defense are introduced that take inspiration from perceptual image encryption methods. The main idea of the key-based defense is to embed a secret key into the model structure with minimal impact on model performance. Assuming the key stays secret, an attacker will not obtain any useful information on the model, which will render adversarial attacks ineffective. The chapter demonstrates the effectiveness of this defense framework by conducting rigorous experiments and presents the results in comparison with state-of-the-art methods. Models protected by the defense framework were confirmed to be resistant against both adaptive and non-adaptive attacks on different datasets. The chapter also highlights the advantages and limitations of the defense framework.

Chapter 5 adopts the block-wise transformation from Chapter 4 and extends the concept of the secret key to model protection. It introduces two model access control frameworks and one model watermarking framework. It presents experiments and shows the results of performing relevant attacks to verify the effectiveness of the model protection frameworks. It also provides a discussion and comparison with state-of-the-art methods for each model protection framework.

Chapter 6 concludes this thesis by providing a summary of the results in this thesis with concluding remarks and directions for future work.

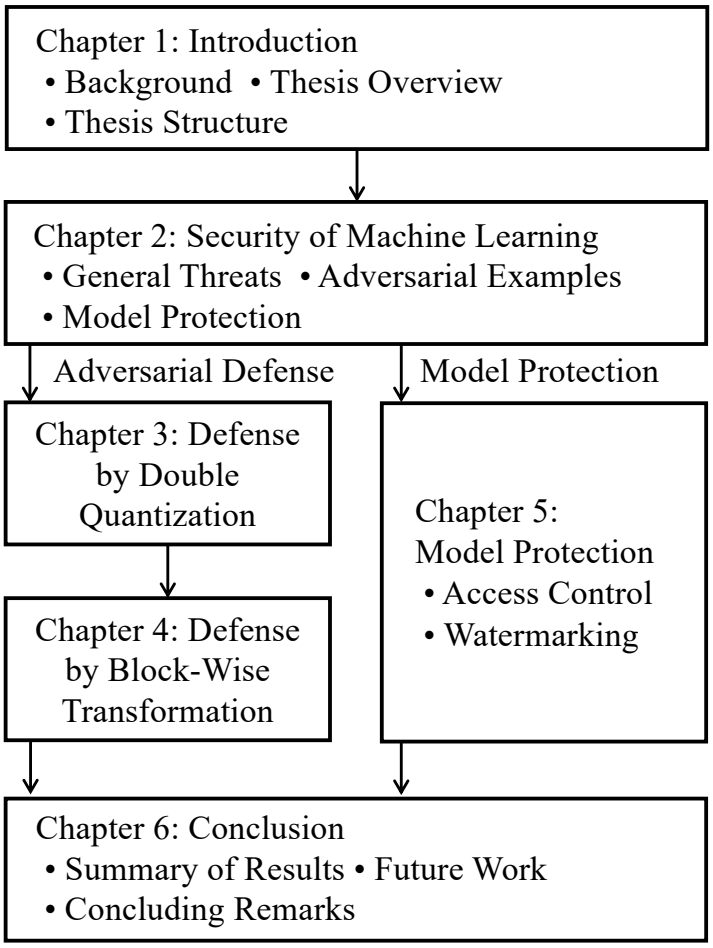

Figure 1.4: Outline of the thesis.

Chapter 2

Security of Machine Learning

While we enjoy groundbreaking advancements in AI, deep neural networks have been proven to be at risk for various attacks such as data corruption, model thefts, and adversarial examples. Therefore, many organizations across different industries pay significat attention to securing their AI systems.

This chapter first describes notations used throughout this thesis. Then, it discusses general threats of machine learning and focuses on adversarial examples that cover threat models, attacks, and defenses. Next, the chapter explains model protection including model access control and model watermarking.

2.1 Notation

The following notations are utilized throughout this thesis.

- h, w, and c are used to denote the height, width, and number of channels of an image.
- The tensor $\mathbf{x} \in [0,1]^{c \times h \times w}$ represents an input color image.
- $f(\cdot)$ denotes a deep convolutional neural network image classifier.
- $\boldsymbol{\theta}$ is a vector of parameters for the network f.
- $\boldsymbol{z}$ denotes a n-dimensional prediction vector, (i.e., $f(\mathbf{x}) = (z_1, z_2, \ldots, z_n)$).
- $C(\cdot)$ denotes the arg-max operation of $f(\mathbf{x})$ (i.e., $C(\mathbf{x}) = \arg\max_i z_i$).
- $\hat{y}$ denotes the predicted label (i.e., $C(\mathbf{x})$).

- y denotes the truth class label of $\mathbf{x}$.
- t denotes the target class label of $\mathbf{x}'$.
- $\mathcal{L}$ and $\mathcal{L}'$ denote loss functions.
- λ is a hyperparameter in the Carlini and Wagner's attack.
- $Z(\cdot)$ denotes the output of all layers except softmax function (i.e., logits).
- $\max(\cdot, \cdot)$ is a function returns the item with the largest value.
- κ denotes a constant to control the attack confidence in Carlini and Wagner's attack.
- β is a regularization parameter in the elastic-net attack.
- $\boldsymbol{\delta}$ denotes an adversarial noise that is added to the clean image $\mathbf{x}$.
- The tensor $\mathbf{x}' \in [0, 1]^{c \times h \times w}$ represents an adversarial example for $\mathbf{x}$, i.e., $\mathbf{x}' = \mathbf{x} + \boldsymbol{\delta}$.
- Δ denotes a allowable perturbation set.
- α is a step size in the iterative gradient-based attack.
- ϵ denotes a perturbation budget.
- $g(\cdot)$ denotes a general defensive transformation.
- K describes a key used in a block-wise transformation.
- $\boldsymbol{v}$ denotes a random permutation vector generated by key K.
- $\boldsymbol{r}$ denotes a random binary vector generated by key K.
- M is a block size of an image used in a block-wise transformation.
- $g(\mathbf{x}, K, M)$ denotes a block-wise transformation of $\mathbf{x}$ with block size M and key K.
- $\hat{\mathbf{x}}$ denotes a block-wise transformed image (i.e., $g(\mathbf{x}, K, M) = \hat{\mathbf{x}}$).
- $d(\cdot, \cdot)$ denotes a distance metric bounded by ℓ_p norm.
- ϵ denotes a perturbation budget for an adversarial example.

- $\mathcal{Q}^1$ denotes a one bit quantizer.
- $\mathbb{Z}^{[0,255]}$ denotes a set of integers from 0 to 255 (i.e., $\{0, 1, \ldots, 254, 255\}$).
- FSD$(\cdot)$ denotes a function that carries out Floyd-Steinberg dithering (FSD).
- s denotes a pixel value that is already being scanned.
- p denotes a current pixel value being scanned.
- $A(\cdot)$ denotes a function that deploys an attack algorithm and generates an adversarial example.
- $\mathbf{x}_{1-\text{bit}}$ denotes an image tensor in 1–bit.
- $\mathbf{x}'_{1-\text{bit}}$ denotes an adversarial example for $\mathbf{x}_{1-\text{bit}}$.
- $\hat{\mathbf{x}}_{1-\text{bit}}$ denotes an image tensor in 1–bit after removing the noise.
- $\lfloor a \rfloor$ denotes the largest integer smaller than a.
- $\mathbf{B}_{(i,j)}$ denotes a block of an image with a dimension of $c \times M \times M$.
- $\boldsymbol{b}_{(i,j)}$ denotes flattened $\mathbf{B}_{(i,j)}$.
- $\boldsymbol{b}'_{(i,j)}$ denotes transformed $\boldsymbol{b}_{(i,j)}$.
- $\mathcal{K}$ denotes a key space of a key-based transformation.
- $\{(\mathbf{x}_i, y_i)\}_i$ denotes a set of training examples (pairs of images and labels).
- $\{(\hat{\mathbf{x}}_i, y_i)\}_i$ denotes a set of training examples (pairs of transformed images and labels).
- $\{\mathbf{u}_i\}_i$ denotes a set of N test images for watermark detection.
- τ denotes a matching rate used for watermark detection.
- th denotes a threshold value for ownership verification.

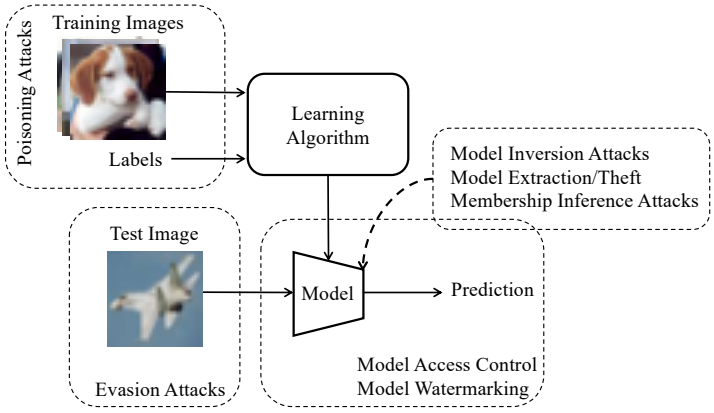

Figure 2.1: Attacks on ML pipeline.

2.2 General Threats

In the era of ubiquitous computing (software everywhere), software security itself is a critical research area as new vulnerabilities are found and exploited almost every day. Machine learning (ML) models are also computer software that addresses the tasks such as computer vision, speech recognition, natural language processing, etc. that are extremely hard or impossible to be solved by traditional programming (first-principles approach). In addition to traditional software security, ML suffers from adversarial attacks throughout its whole pipeline (training, model, inference) that can malfunction ML systems. Figure 2.1 depicts attacks on ML pipeline, where poisoning attacks can be carried out on training data, evasions attacks on test data, and the attacks such as model inversion, model extraction/theft, and membership inference attacks can be performed on the model itself. In addition, there are intellectual property issues on the ML models such as model access control and model ownership verification via model watermarking.

As the ML models especially data-hungry deep learning models require a huge amount of data, the curation of training data is often outsourced and automated. This opens a significant security flaw for data poisoning attacks where training data can be manipulated to control the behavior of the ML models [47, 48]. The danger of the poisoning attacks was illustrated by the manipulation of the Tay chatbot [49]. In a recent survey, poisoning is also reported as a top attack that would affect the businesses of the organizations [50].

Trained ML models are also exposed to other privacy and security attacks. Model inversion attacks recover training data from the model [37]. Model parameters can be extracted by prediction queries that can cause intellectual property theft [51]. Another

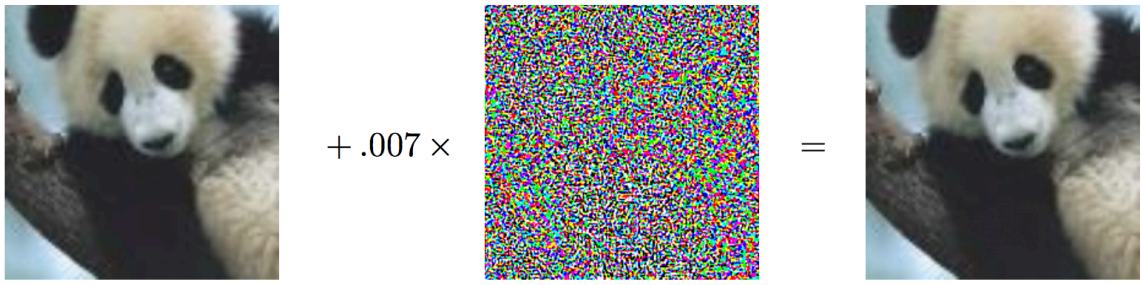

Figure 2.2: Example of adversarial example [53].

privacy attack to determine whether a given data record is in the model's training data, known as membership inference attacks [52]. Moreover, ML models can be abused to generate fake content (known as *deep fakes*), that has a severe societal impact linking to elections, automated trolling, court evidence, etc.

The ML models can also be attacked by manipulating the test input known as evasion attacks. In evasion attacks, carefully perturbed data points known as *adversarial examples* are indistinguishable from clean data points, but they cause ML models to make erroneous predictions [9, 10].

Since the ML models are shared on model-sharing platforms such as Model Zoo [24], Azure AI Gallery [25], and Tensor Hub [12], it is necessary to protect models from copyright infringement and unauthorized access. A recent security assessment of on-device models from Android applications showed that many mobile applications fine-tuned pre-trained models from Tensor Hub [26]. Therefore, the ML models are also prone to copyrights and intellectual property thefts.

This thesis focuses on defending against adversarial examples (i.e., evasion attacks) and model protection for access control, and ownership verification (i.e., model watermarking). Therefore, the following subsections will further elaborate on these topics.

2.3 Adversarial Examples

Researchers have shown that neural networks are vulnerable to imperceptible intentional perturbed data points known as adversarial examples [9, 10], despite the state-of-the-art results. These adversarial examples can cause neural networks to misclassify or force them to classify a targeted class with high confidence. Concretely, Goodfellow et al. show that an input image ("panda") with a small noise vector whose

elements are equal to the sign of the gradients of the loss function with respect to the input on GoogLeNet [54] is predicted as "gibbon" with high confidence, as shown in Fig. 2.2 [53]. Incorrect decisions made by the neural networks can cause serious and dangerous problems. As an example, self-driving cars may misclassify a "Stop" sign as "Speed Limit" [13]. Therefore, adversarial examples have received a significant amount of attention even though security for machine learning began over a decade ago [55].

As adversarial examples are an obvious threat, numerous methods for generating adversarial examples (attack methods) and defenses against them have been proposed in the literature [11]. In this section, threat models for crafting adversarial examples are first described, and selected recent attacks and defenses are surveyed in a brief and comprehensive way.

2.3.1 Threat Models

To evaluate an adversarial defense method, precisely defining threat models is necessary. As described in [56], a threat model includes a set of assumptions such as an adversary's goals, capabilities, and knowledge. Moreover, with the knowledge of a defense mechanism, an adaptive adversary can be mounted to evaluate a defense method.

Let us consider a neural network image classifier $f : [0,1]^{c\times h\times w} \to [0,1]^n$ that takes an image $\mathbf{x} \in [0,1]^{c\times h\times w}$, for a c-channel image of height h and width w, and outputs a n-dimensional prediction vector $\boldsymbol{z} = (z_1, z_2, \ldots, z_n) = f(\mathbf{x})$. Obtaining the most likely label (namely, the predicted label) $\hat{y}$ for $\mathbf{x}$ is done by finding the maximum element of the output vector, i.e., $\hat{y} = C(\mathbf{x}) = \arg\max_i z_i$.[1]

Adversary's Goals

An adversary can construct adversarial examples to achieve different goals when attacking a model, that is, either reduce the performance accuracy (i.e., untargeted attacks) or classify an image as a targeted class (i.e., targeted attacks). Formally, untargeted attacks will cause a classifier f to misclassify an adversarial example $\mathbf{x}'$ (i.e., $C(\mathbf{x}') \neq y$) regardless of its resulting label, and targeted ones will force the classifier to output a targeted label t (i.e., $C(\mathbf{x}') = t$). This thesis focuses on untarget attacks for efficiency in experiments although targeted attacks can be achieved in a similar way.

[1] In this thesis, it is assumed that the vector $\boldsymbol{z}$ always has one maximum element and the argmax operation $\arg\max_i z_i$ returns an integer, although the argmax operation generally returns a set, i.e., indices of the maximum elements.

2.3.2 Adversary's Capabilities

An adversarial example $\mathbf{x}'$ is usually not perceptibly different from the original corresponding example $\mathbf{x}$, but it is misclassified by a classifier [9]. Therefore, an adversary is restricted to modifying an input image $\mathbf{x}$ under some similarity metric d such that $d(\mathbf{x}, \mathbf{x}') \leq \epsilon$, where ϵ is the perturbation distance. As for the metric d, the ℓ_p norm (e.g., ℓ_2 or ℓ_∞) is most often utilized.

There are two common approaches to finding such perturbations under ℓ_p bounded threat models for untargeted attacks. The first approach creates an adversarial example $\mathbf{x}'$ by searching for a small perturbation by maximizing the loss function $\mathcal{L}$ (e.g., the cross-entropy loss), i.e.,

$$\mathbf{x}' = \arg\max_{\hat{\mathbf{x}}} \mathcal{L}(f(\hat{\mathbf{x}}), y) \text{ s.t. } d(\mathbf{x}, \hat{\mathbf{x}}) \leq \epsilon. \tag{2.1}$$

The second approach finds the minimum perturbation possible to generate an adversarial example $\mathbf{x}'$, i.e.,

$$\mathbf{x}' = \arg\min_{\hat{\mathbf{x}}} d(\mathbf{x}, \hat{\mathbf{x}}) \text{ s.t. } C(\hat{\mathbf{x}}) \neq y. \tag{2.2}$$

In addition, there are also other threat models that spatially transform input images such as [57–59].

Regarding ℓ_p-bounded perturbation, threat models do not match real-world applications [60, 61] because there can be various physical conditions (e.g., camera angle, lighting/weather), physical limits on imperceptibility, etc. However, it has been proved that adversarial threats on neural networks remain real [13, 16, 62–64]. In addition, ℓ_p-bounded threat models are crucial for principled deep learning due to their well-defined nature [56]. They are helpful not only for evaluating the robustness of deep learning models but also for understanding them better. It is almost certain that models that are not robust against ℓ_p-bounded attacks will fail in real-world scenarios.

Adversary's Knowledge

We can assume that an adversary may have different levels of knowledge about a target model. According to [56], the adversary's knowledge can be white-box (inner workings of the defense mechanism, complete knowledge of the model and its parameters) or black-box (no knowledge of the model) with varying degrees of access levels (e.g., limited number of queries to the model, etc.). The adversary may also be gray-box (anything in between white-box and black-box).

In some scenarios, we assume that the adversary does not know some secret information about the model. As in the field of cryptography, there can be a small amount

of secret information even in white-box settings if the secret information must be easily replaceable and non-extractable [56]. Some defense methods introduce a secret key to transform the input [40, 41]. In this case, the adversary may or may not know the transformation algorithm, but the secret key is not known to the adversary.

2.3.3 Adaptive Adversaries

To construct strong threat models, adaptive adversaries are necessary where the specific details of the defense methods are adapted. Athalye et al. pointed out that some defenses that seem to defeat iterative optimization-based (white-box) attacks cause obfuscated gradients (a way of gradient masking) [18]. Obfuscated gradients occur under three conditions: (1) shattered gradients (non-existent/incorrect) due to non-differentiable operations or numerical instability, (2) stochastic gradients because of test-time randomness, and (3) vanishing/exploding gradients for very deep computation [18]. These obfuscated gradients can be bypassed by using backward pass differentiable approximation (BPDA), expectation over transformation (EOT), or reparameterization as in [18]. To avoid a false sense of security, adaptive adversaries are critical to evaluate the robustness of a defense [56].

2.3.4 Attacks

Techniques for generating adversarial examples can mainly be divided into four categories: gradient-based, transfer-based, score-based, and decision-based attacks. Figure 2.3 depicts categories of attacks according to the varying degrees of access to a model. As described in Section 2.3.2, attacks methods can also be categorized according to the knowledge available (white-box, black-box, or gray-box). In this section, white-box (gradient-based) attacks and black-box (transfer-based, score-based, and decision-based attacks) ones are summarized as follows.

Gradient-Based Attacks

Gradient-based attacks are known as white-box attacks because they require direct access to a model to compute the gradient of the loss with respect to the input. One of the easy and popular gradient-based attacks is the fast gradient sign method (FGSM) [53] that is performed under an ℓ_∞ norm with a single gradient step by solving Eq. (2.1). The straightforward extended version of FGSM is the basic iterative method (BIM), which involves taking multiple small gradient steps [65]. BIM with multiple random restarts and initialization with uniform random noise is recognized as projected gradient descent (PGD) [23]. The adversarial example, $\mathbf{x}'$ at the $(i+1)^{\text{th}}$ iteration with

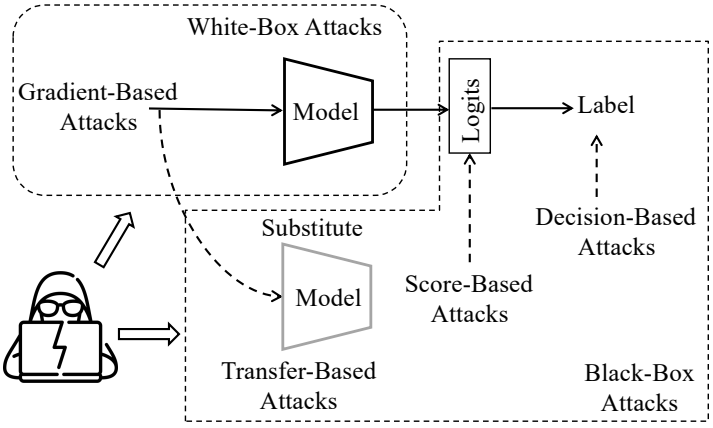

Figure 2.3: Attack categories applicable to a model based on the varying degrees of access to the model.

PGD [23] is

$$\mathbf{x}'_{(i+1)} = \Pi_{\mathbf{x}+\Delta}(\mathbf{x}'_i + \alpha \cdot \text{sign}(\nabla_{\mathbf{x}}\mathcal{L}(f(\mathbf{x}'_i), y))), \tag{2.3}$$

where α is the step size, and $\Pi_{\mathbf{x}+\Delta}$ denotes projection to the allowable set of perturbation, Δ. PGD is known as a powerful first-order adversary.

For solving Eq. (2.2), DeepFool finds the minimum perturbation under an ℓ_2 norm [66]. DeepFool can also be extended to the ℓ_∞ norm. Carlini and Wagner's method (CW) reformulates Eq. (2.2) by using Lagrangian relaxation to find the minimum perturbation under an ℓ_p norm [67]. Specifically, they solve the following optimization,

$$\arg\min_{\boldsymbol{\delta}} \|\delta\|_p + \lambda \cdot \mathcal{L}'(\mathbf{x} + \boldsymbol{\delta}) \text{ s.t. } \mathbf{x} + \boldsymbol{\delta} \in [0,1]^{c \times h \times w}, \tag{2.4}$$

where $\|\cdot\|_p$ is a distance function under ℓ_p norm, λ is a hyperparameter and $\mathcal{L}'$ is a new objective function defined as,

$$\mathcal{L}'(\mathbf{x} + \boldsymbol{\delta}) = \max\left(\max\{Z(\mathbf{x} + \boldsymbol{\delta})_i : i \neq t\} - Z(\mathbf{x} + \boldsymbol{\delta})_t, -\kappa\right), \tag{2.5}$$

where $Z(\cdot)$ denotes the output of all layers except softmax function (i.e., logits), and κ is a constant to control the confidence so that $\mathbf{x} + \boldsymbol{\delta}$ is classified as a target class t. To encourage sparsity in the perturbation, the elastic-net attack (EAD) was proposed to create ℓ_1-oriented adversarial examples [68]. EAD combines ℓ_1 and ℓ_2 regularization, optimizes the following objective:

$$\arg\min_{\boldsymbol{\delta}} \lambda \cdot \mathcal{L}'(\mathbf{x} + \boldsymbol{\delta}) + \beta\|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\delta}\|_2^2 \text{ s.t. } \mathbf{x} + \boldsymbol{\delta} \in [0,1]^{c \times h \times w}. \tag{2.6}$$

CW is also a special case of EAD, where the ℓ_1 regularization parameter, β is set to zero [68].

Transfer-Based Attacks

Transfer-based methods are black-box attacks, and the actual model is not known to attackers. Transferability between models is exploited, and white-box attacks can be applied via a surrogate model. Although transfer-based attacks do not rely on model information, they require information about the training data for the surrogate model. Synthetic data, which are labeled by the targeted model, can be used as a substitute to craft adversarial examples [64, 69]. The momentum iterative method (MIM) integrates the momentum term into the iterative process to stabilize update directions throughout the attack optimization process [70]. To further improve the transferability, the translation-invariant method (TI) optimizes perturbation over an ensemble of translated images [71]. In a similar fashion, the diverse inputs method (DI) applies random transformation on each attack iteration to improve the transferability [72].

Score-Based Attacks

Unlike transfer-based attacks, score-based ones do not need a substitute model. However, attackers need to have access to the output of the model, either a full set of probabilities or the top k (e.g., $k = 5$). Since only the output of a model is available, score-based methods cannot compute gradients analytically as in white-box methods. Therefore, some of the score-based methods estimate gradients on the basis of the query feedback of the targeted model. One method uses natural evolution strategies (NES) to estimate gradients and crafts adversarial examples by observing the output of the model [73]. Another method utilizes zeroth-order optimization to directly estimate gradients of the targeted model [74]. Another strategy to approximate gradients and reliably produce adversarial examples is using simultaneous perturbation stochastic approximation (SPSA) [75]. The prior-guided random gradient-free (P-RGF) method, which can estimate gradients more accurately, generates black-box adversarial examples by using a transfer-based prior and the query information simultaneously [76].

In addition, there are other score-based attacks that do not estimate gradients. $\mathcal{N}$ATTACK learns a probability distribution centered around an input such that a sample drawn from that distribution is likely an adversarial example [77]. The Square Attack (SQUARE), which is based on a random search, generates adversarial examples by modifying a square of the input image at a random position at each iteration [78]. Moreover, the OnePixel attack constructs an adversarial example by modifying one or a few pixels without accessing the weights of a model with differential evolution [79].

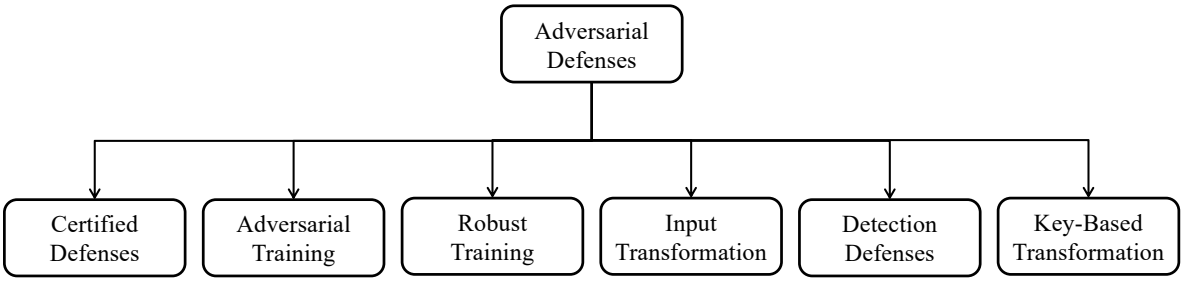

Figure 2.4: Different categories of adversarial defenses.

Decision-Based Attacks

In this category, attack methods are restricted to hard labels only, and they do not have access to class probabilities or scores. Therefore, decision-based methods are more challenging and practical for real-world applications. The Boundary Attack starts from a large adversarial perturbation and seeks to reduce the perturbation based on a random walk along the boundary between the adversarial and non-adversarial region [80]. This random walk-based method is the first method proposed for the decision-based category. One attack method models the hard-label black-box scenario as a real-valued optimization problem that can be solved by any zeroth-order optimization algorithm [81]. To use lesser queries, this problem formulation was also used to directly estimate the sign of gradients in any direction [82]. In a particular application scenario, that of face recognition systems, a decision-based attack method was proposed that uses an evolutionary algorithm to improve the query efficiency [83]. In a particular application scenario of face recognition systems, a decision-based attack method proposed to use an evolutionary algorithm to improve the query efficiency [83].

2.3.5 Defenses

With the development of adversarial attacks, various defense methods have been proposed in the literature. To get a big picture of defenses in different flavors, six categories of defenses (Fig. 2.4): certified defenses, adversarial training, robust training, input transformation, detection, and key-based transformation, are briefly surveyed according to the design nature of the defense. Figure 2.5 shows an overview of the different defense categories, where most defenses are applied during model training. However, input transformation defenses and detection methods can be in and out of the model training. For example, defense by input transformation [39] requires retraining a model in order to adapt the input transformation, and some detection methods like [84] need the defense model to be trained with a modified network.

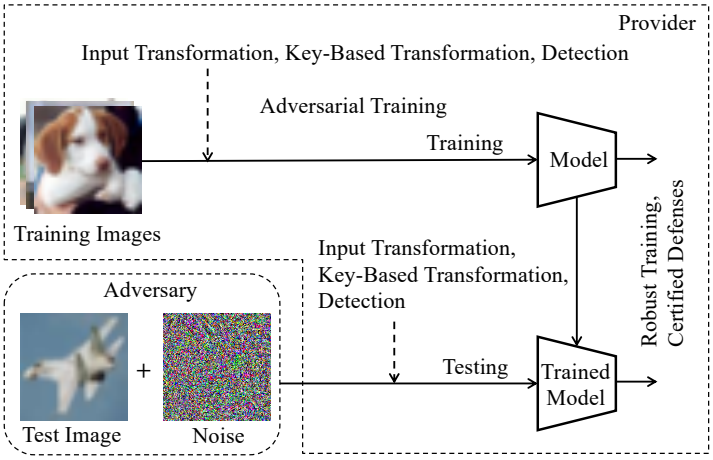

Figure 2.5: Defense categories relative to model training and testing.

Certified Defenses

Certified/provable defenses use formal verification techniques to guarantee that no adversarial examples exist within some bounds. Ideally, these defenses are desired. Inspiring works such as [85–87] proposed provable secure training. Although these methods are attractive, they are not scalable. Some certified defenses have been scaled to a certain degree [88–91], but the guarantees do not match the empirical robustness provided by adversarially trained models, which will be discussed in the next subsection.

Adversarial Training

Current state-of-the-art empirically robust defenses are under the use of adversarial training, which includes adversarial examples in a training set. Adversarial training is a form of regularization that aims to reduce test error [92]. The earliest form of adversarial training is to inject FGSM-based adversarial noise into training data [53]. Since FGSM is not iterative and not robust against iterative attacks such as PGD, FGSM-based training was found to be ineffective [23, 65].

Madry et al. approach adversarial training as a robust optimization problem [23]. Given a classifier network $f_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$ and a dataset $\{(\mathbf{x}_i, y_i)\}_i$, they use a PGD adversary, Δ, under the ℓ_∞ norm to approximate the worst inputs possible in solving the following min-max objective,

$$\arg\min_{\boldsymbol{\theta}} \sum_i \arg\max_{\boldsymbol{\delta} \in \Delta} \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}_i + \boldsymbol{\delta}), y_i), \tag{2.7}$$

where $\Delta = \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_\infty \leq \epsilon\}$.

PGD training is computationally expensive. To make the computation of adversarial training more feasible, "free" adversarial training was proposed in which gradients are computed with respect to the network parameters and the input image on the same backward pass [93]. In addition to "free" adversarial training, "fast" adversarial training was proposed and uses FGSM and standard efficient training tricks [42]. Although FGSM-based adversarial training was dismissed before, it is shown to be effective when random initialization is introduced [42]. Nevertheless, while adversarial training is repeatedly found to be robust against the best-known adversaries [18], the accuracy is still very low compared with non-robust models.

Adversarial training approaches generate perturbation in a supervised way. A recent work [43] proposed an adversarial training method by using an unsupervised method called feature scattering, which involves maximizing the feature matching distance between clean examples and perturbed ones. Although they achieved better performance than conventional methods for several datasets, the ImageNet [94] performance has not yet been tested.

Tsipras et al. show that there is a trade-off between robustness and accuracy, i.e., gaining robustness causes some accuracy loss [95]. In addition, even empirically robust models are susceptible to blind-spot attacks where input images reside in low-density regions of a training data distribution [96]. Therefore, maintaining accuracy and getting adversarial robustness is a growing concern and an ongoing area of research with a high demand.

Robust Training

The defenses in this category change a model's architecture by using a new loss, imposing a special regularization scheme, or altering the final layer for adversarial robustness. A recently proposed defense method uses the Max-Mahalanobis center loss to learn more structured representations, thus enhancing adversarial robustness [97]. Methods that modify the final layer include defensive distillation [98] and DeepCloak [99], which identifies and removes unnecessary features by adding a mask layer before the linear layer (logits layer). Inspired by the contractive autoencoder (CAE), Gu and Rigazio proposed the Deep Contractive Network, which imposes a layer-wise smoothness penalty [100]. Other methods utilize a special regularization scheme for adversarial robustness: Parseval regularization [101], input gradient regularization [102], perturbation-based regularization [103], etc. In short, robust training-based defenses have not been extensively tested with modern convolutional neural networks and different datasets, and it is often recommend to include adversarial training for stronger robustness.

Input Transformation

The methods in this category aim to find a defensive transform $g(\cdot)$ such that

$$C(f(g(\mathbf{x}'))) = C(f(\mathbf{x})). \tag{2.8}$$

The transformation g is designed to reduce the impact of adversarial noise by restricting the space of adversarial examples. The works in this direction utilize various means of transformation such as thermometer encoding [104], image processing-based techniques [105,106], making small changes to pixels with the intent of removing adversarial noise [107], and GAN-based transformation [108]. These input transformation-based defenses are appealing at first due to their higher accuracy. However, they have all been broken because they rely on obfuscated gradients [18]. Accounting for this problem, Raff et al. came up with a defense that uses a number of random different transforms with random parameters [39]. Although their work claims majorly improved accuracy on ImageNet, applying many transforms for each image is computationally expensive and reduces the accuracy when the model is not under attack.

As input transformation defenses provide higher accuracy, this thesis also proposed an input transformation defense by enforcing the use of 1–bit images and utilizing quantization in Chapter 3.

Detection

Some defenses are designed to detect adversarial examples so that they can be rejected for classification. Metzen et al. proposed a detection method that trains a binary classification network to distinguish clean data from adversarial examples [109]. Another work by [110] detects adversarial examples by looking at the features in the subspace of deep neural networks. However, it is reported that detection methods can also be bypassed [111]. Another interesting detection method uses a one-to-one encoding scheme from true labels to code vectors to hide input labels from the attackers [84]. Although this approach [84] produces good results on small datasets, it has not been tested on larger datasets.

Key-based Transformation

Defenses with key-based transformation are similar to input transformation-based methods that require transforming an input prior to an inference with a model. However, there are differences: (1) key-based transformation methods have an information advantage, that is, a secret key, and (2) a model is required to be trained by using transformed images.

This category of defenses takes a different direction by taking inspiration from perceptual image encryption techniques such as [112–116]. The main idea is to hide a model's decision from attackers by means of training the model with encrypted images. Taran et al. first introduced a secret block for adversarial defense and proved the idea by taking a pixel shuffling approach (pixel-wise manner) with a secret key with a standard random permutation [41]. Although their method [41] was effective at defending against adversarial examples, it was tested only on small datasets (MNIST [117] and F-MNIST [118]), and clean accuracy significantly dropped on larger datasets such as CIFAR-10 [119] and ImageNet [94]. The reason is that shuffling in a pixel-wise manner causes spatial perceptual information to be lost. As the key-based transformation defense is effective to defend against adversarial examples, Chapter 4 will introduce a method that utilizes a block-wise transformation with a secret key defending against adversarial examples [40]. This method shows good results against white-box attacks assuming that the secret key is kept secret from the attackers.

2.4 Model Protection of Deep Neural Networks

There is no doubt that neural networks provide remarkable performance in many recognition tasks. However, training a successful neural network model is not trivial because it requires a huge amount of data, efficient algorithms, and fast computing resources (e.g., GPU-accelerated computing). For example, the ImageNet (ILSVRC12) dataset [94] contains about 1.28 million images, and training on such a dataset takes days and weeks even on GPU-accelerated machines. In fact, collecting images and labeling them will also consume a massive amount of resources. Moreover, algorithms used in training a model may be patented or have restricted licenses. Therefore, production-level trained models have great business values. Considering the cost of training a model (money, time, expertise), a model should be regarded as a kind of intellectual property (IP).

There are two aspects of IP protection for models: model access control and model watermarking. The former focuses on protecting the functionality of models from unauthorized access, and the latter addresses ownership verification by taking inspiration from digital watermarking.

2.4.1 Model Access Control

As trained models are widely shared and emerged into many applications, model access control has become an increasingly important problem to prevent from economic damage. Model access control ensures that only authorized users can use the model

to full capacity. While distributing a trained model, an illegal party may obtain a model and use it for its own service. One straightforward way of protecting a model from illicit use is to encrypt the trained model weights by the traditional cryptographic methods such as advanced encryption standard (AES). In this case, to be able to use the protected model, rightful users have to decrypt the model. There are millions of parameters in modern deep neural network models, so encrypting/decrypting all the parameters is computationally expensive under the traditional cryptography in general. Besides, once the model is decrypted, it becomes vulnerable for IP thefts.

Another approach for model access control may be to train models via homomorphic encryption or functional encryption so that trained models can work on encrypted parameters without needing the decryption. However, most of model architectures are designed to learn from plaintext and learning from the encrypted data will severely affect the performance. Therefore, researchers have proposed to embed a key to the model's structure by other means. Chapter 5 will introduce new solutions for model access control by a block-wise transformation with a secret key.

2.4.2 Model Watermarking

Digital watermarking technology is widely used to combat copyright infringement for multimedia data [120]. An owner embeds a watermark into multimedia content (such as images, audio, etc.). When the protected content is stolen, the embedded watermark is extracted and used to verify ownership. In a similar fashion, to prevent the illegal distribution of trained models, digital watermarking techniques are used to embed watermarks into proprietary models. There are mainly two scenarios in model watermarking: white-box and black-box.

A model watermarking scenario in white-box settings requires access to model weights for embedding and extracting a watermark. Uchida et al. first proposed a white-box model-watermarking method [28]. A watermark is embedded into one or more layers of model weights by using "an embedding regularizer," which is an additional regularization term in the loss function during training. Similarly, there are other works that follow the use of an additional regularization term as in [29, 30, 32].

Extracting watermarks in white-box settings requires access to the model weights. To overcome this limitation, another model watermarking scenario for black-box settings was proposed, where an inspector observes the input and output of a model in doubt to verify the ownership of the model. In the black-box scenario, adversarial examples are exploited as a backdoor trigger set [31, 121], or a set of training examples is utilized so that a watermark pattern can be extracted from the inference of a model by using a specific set of training examples [32–34]. Therefore, access to the model weights is not required to verify ownership in black-box settings. In this black-box direction,

Chapter 5 will also introduce a new solution for model watermarking by a block-wise transformation with a secret key.

Chapter 3

Adversarial Defense by Quantization

Despite deep neural networks being excellent in many recognition tasks and natural language processing, they are vulnerable towards certain alteration in the input known as adversarial examples [9, 10]. The perturbation in adversarial examples is imperceptible towards humans, but can cause neural networks misclassify or force to classify a targeted class. Adversarial examples can either be physical or digital. It is possible that physical objects can be adversarial examples by default. For example, IMAGENET-A [122] is a dataset of natural adversarial examples with 200 classes. An attacker may distort the physical objects intentionally or synthesize physical adversarial objects [13, 16, 62–64]. When these adversarial objects (naturally or adversarially) are captured by a camera, they become adversarial examples and cause the model to behave differently. There are different methods (as discussed in Chapter 2, Section 2.3.4) to generate adversarial examples in the digital domain on the basics of the knowledge of the model available to the attacker (i.e., white-box or black-box). As there are numerous methods to generate adversarial examples (either physical or digital), the threat of adversarial examples has become a major concern for deep neural networks in security-critical applications such as autonomous vehicles, healthcare, and finance.

Although researchers have proposed various defenses concerning the adversarial attack methods, the performance is not yet satisfactory. Therefore, this chapter reframes the problem of adversarial examples into a specific scenario where only quantized (1–bit) images are available to the attackers. Due to the expensive cost of fast computing resources, models are usually trained in a cloud environment, and the models are deployed as inference services. In such a case, a test image may be potentially intercepted and injected with adversarial noise by an attacker as a man-in-the-middle as shown in Fig 3.1. For example, the test image can be modified by an existing adversarial attack

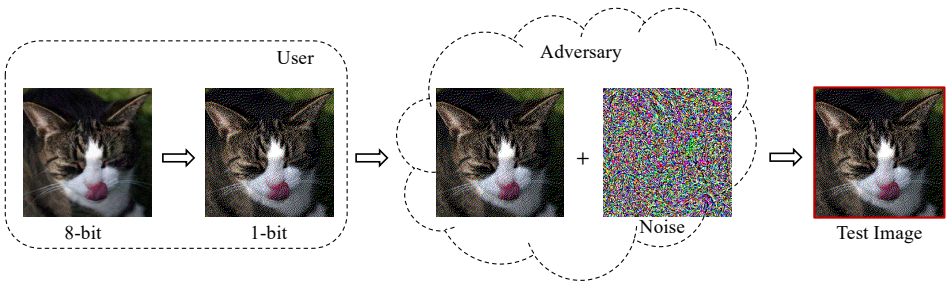

Figure 3.1: Attack scenario on quantized images.

before reaching the model. This chapter considers this particular scenario and enforces the use of quantized images. Therefore, this chapter assumes that original training and test images are in 1–bit, and the attacker does not know the defense mechanism although model weights are available to the attacker.

Although larger bit-depth images are usually preferred for aesthetics, it is not always necessary for interpreting images as we, the human can recognize even black and white images most of the time. The chapter hypothesizes that if adversarial noise is added to 1–bit images, it can be simply removed by linear quantization. However, the use of linearly quantized images can decrease the classification accuracy, and thus, dithering is introduced to improve the accuracy. This chapter aims to achieve the following requirements:

1. High classification for clean images and
2. High classification for adversarial examples.

3.1 Related Work

3.1.1 Previous Input Transformation-Based Defenses

To improve adversarial robustness while maintaining accuracy, many works have attempted to find a defensive transformation. The transformation-based defenses include having several image transformations (bit depth reduction, JPEG compression, total variance minimization) [105], finding a clean input by using a generative adversarial network (GAN) [108], and so on. However, they all have been defeated by adaptive attacks in white-box settings [18]. To reinforce these weak defense methods, Raff, Sylvester, Forsyth, et al. [39] proposed a stronger defense by combining a large number of transforms stochastically. This defense is called "Barrage of Random Transforms"

Figure 3.2: Example of linearly quantized images.

(BaRT) [39]. BaRT is applied in both training and testing phases, and applying many transforms is computationally expensive and reduces the classification accuracy when the model is not under attack.

This chapter focuses on quantization as a mean of adversarial defense. Xu et al. first introduced this concept as feature squeezing [123]. They reported to use 4-bit images because images with bit-depth lower than four introduce some human-observable distortion and neural networks also decrease performance accuracy. Later, Guo et al. proposed adversarial defense by input transformation including quantization and other image processing techniques [105]. Quantization seems an easy and effective way of defending against adversarial examples. Recently, Miyazato et al. proposed a method that searches a q–bit quantized image that maximizes the loss and uses it to backpropagate during the training process [124]. Their method is similar to adversarial training to harden the model by training with q–bit quantized images that maximize the loss instead of training with adversarial examples. However, it was only tested on black-box fast gradient sign method (FGSM) which is an easy adversary. Informed by empirical results in [125], adversarial noise on low bit-depths can be filtered in a condition that the noise is added to the quantized images. Therefore, this chapter presents a defense framework by quantization which enforces the use of 1–bit images.

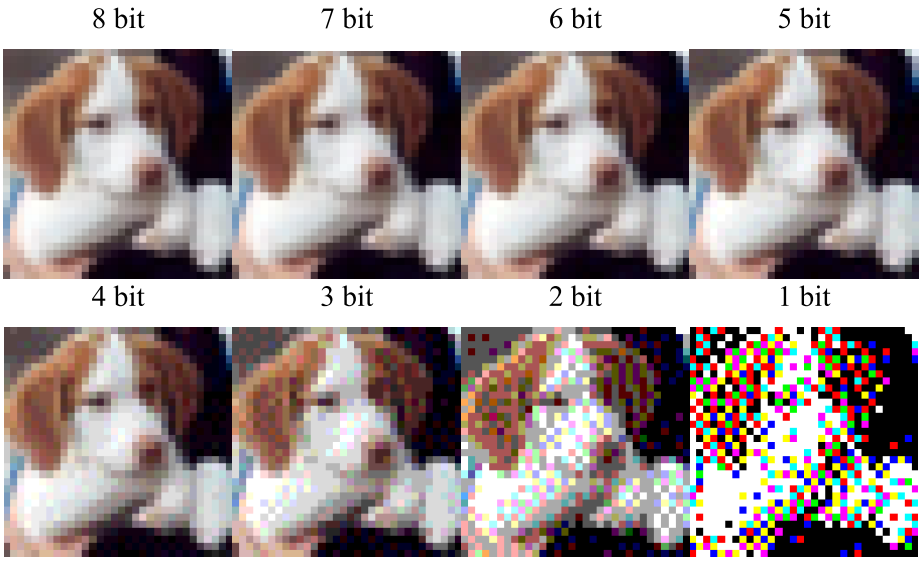

Figure 3.3: Example of dithered images.

3.1.2 Quantization

Quantization is a mapping from a large set of input values to a smaller set of output values. For linear quantization in images, the pixel values are linearly scaled down in accordance with the bit depth. The 1–bit quantizer is defined as:

$$\mathcal{Q}^1 : \mathbb{Z}^{[0,255]} \rightarrow \{0, 255\}.$$

The pixel values in $\{0, 1, \ldots, 254, 255\}$ are mapped to either 0 or 255 in accordance with the closest palette color. Technically, this is a thresholding operation (i.e., values less than 128 are mapped to 0 and those greater than or equal to 128 are to 255). Example of linearly quantized images are shown in Fig. 3.2.

To randomize quantization errors, dithering is usually applied in the quantization. The defense framework in this chapter utilizes the 1–bit Floyd-Steinberg dithering algorithm (FSD) [126]. The following error diffusion filter is applied to distribute the residual quantization error in FSD.

$$\begin{bmatrix} s & s & p & \frac{7}{16} & \cdots \\ \cdots & \frac{3}{16} & \frac{5}{16} & \frac{1}{16} & \cdots \end{bmatrix}$$

The pixel indicated as p in the above filter is the current pixel being scanned, and the pixels indicated as s are already being scanned. The FSD first finds the nearest color

(whether 0 or 255) and calculates the residual error. The error is distributed to the neighbouring pixels by the diffusion filter. For simplicity, Algorithm 1 describes FSD algorithm for one channel image where each pixel is indexed by $\mathbf{x}(i, j)$. Example of dithered images are shown in Fig. 3.3.

Algorithm 1 FSD algorithm

Input: $\mathbf{x}$
Output: $\mathbf{x}_{1-\text{bit}}$

for each $\mathbf{x}(i, j)$ **do** ▷ Scan for each pixel.
  $old \leftarrow \mathbf{x}(i, j)$
  if $old \geq 128$ **then**
    $new \leftarrow 255$
  else
    $new \leftarrow 0$
  end if
  $\mathbf{x}_{1-\text{bit}}(i, j) \leftarrow new$
  $error \leftarrow old - new$
  $\mathbf{x}_{1-\text{bit}}(i + 1, j) \leftarrow \mathbf{x}(i + 1, j) + error * 7/16$
  $\mathbf{x}_{1-\text{bit}}(i - 1, j + 1) \leftarrow \mathbf{x}(i - 1, j + 1) + error * 3/16$
  $\mathbf{x}_{1-\text{bit}}(i, j + 1) \leftarrow \mathbf{x}(i, j + 1) + error * 5/16$
  $\mathbf{x}_{1-\text{bit}}(i + 1, j + 1) \leftarrow \mathbf{x}(i + 1, j + 1) + error * 1/16$
end for each

3.2 Defense Framework by Quantization

The framework of the defense by quantization is depicted in Fig. 3.4, which enforces the use of 1–bit dithered images for training and testing. Both training and testing images are quantized with dithering by the FSD algorithm. The procedure of the framework is detailed as follows.

1. Both training and testing images are first quantized to one bit with dithering by FSD (algorithm 1). This process can be described as

$$\mathbf{x}_{1-\text{bit}} = \text{FSD}(\mathbf{x}). \tag{3.1}$$

   The resulting dithered images are used to train a model.

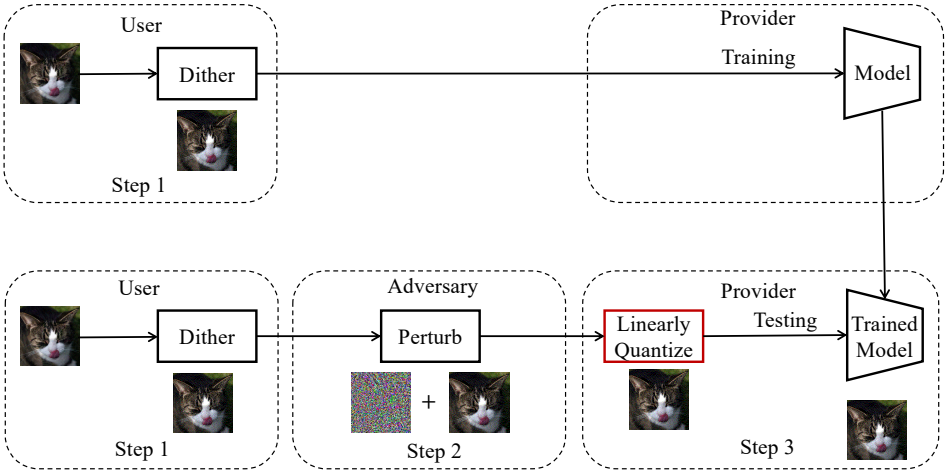

Figure 3.4: Framework of adversarial defense by double quantization.

2. For testing, a testing dithered image from step (1) may be attacked by conventional attacks such as FGSM [53], PGD [23], CW [67], etc. The attacking process is defined as

$$\mathbf{x}'_{1-\text{bit}} = A(\mathbf{x}_{1-\text{bit}}). \tag{3.2}$$

3. Linear quantization is applied to the attacked image (adversarial example) from step (2). The sanitized image is described as

$$\hat{\mathbf{x}}_{1-\text{bit}} = \left\lfloor \frac{\mathbf{x}'_{1-\text{bit}}}{128} \right\rfloor \cdot 255. \tag{3.3}$$

The resulting sanitized image is sent to the model for inference.

3.3 Experiments and Discussion

3.3.1 Experiment Conditions

Datasets

Two datasets, CIFAR-10 [119] and Oxford-IIIT Pet [127] were used to evalute the defense by quantization. CIFAR-10 consists of 60,000 color images (dimension of $32 \times 32 \times 3$) with 10 classes (6000 images for each class) where 50,000 images are for training and 10,000 for testing. The Oxford-IIIT Pet dataset comprises 37 classes of cat and dog breeds with approximately 200 images for each class. There are a total of 7390

high-resolution color images. The dataset is split into two sets: 5912 training images and 1478 test images, and images are resized to $(224 \times 224 \times 3)$ during the experiments.

Networks

Deep residual networks [3] were utilized for evaluating the defense framework by 1–bit double quantization. Experiments were carried out on PyTorch [128] (version 1.1.0) and fastai [129] (version 1.0.54).

For CIFAR-10, ResNet20 without pre-trained weights was trained for 160 epochs with a batch size of 128 and live augmentation (random cropping with padding = 4 and random horizontal flip). The stochastic gradient descent (SGD) optimizer with an initial learning rate of 0.1 was used. A step learning rate scheduler was used with the parameters (lr_steps = 40, gamma = 0.1). The weight_decay and momentum were configured with 0.0001 and 0.9, respectively.

For the Oxford-IIIT Pet dataset, ResNet34 was trained with transfer learning (with ImageNet pre-trained weights), which has been proved to be effective in various visual recognition tasks [130]. The optimizer was AdamW [131], and the parameters were a batch size of 64 and a learning rate in the range of $[1e^{-6}, 1e^{-2}]$. The images were resized into the dimensions of (224×224) and augmented with default augmentation transforms from fastai.

Attack Settings

Three white-box attacks were used to test the defense framework by quantization: fast gradient sign method (FGSM) under ℓ_∞-norm [53], projected gradient descent (PGD) under ℓ_∞-norm [23], and Carlini and Wagner's attack (CW) under ℓ_2-norm [67]. The PGD attack was configured with a noise distance of 8/255, a step size of 2/255, and 20 iterations. The CW attack was set with a learning rate of 0.01, binary search steps of 9, and an initial constant of 0.001 for 20 iterations.

3.3.2 Results

Table 3.1 shows the results of the defense framework by quantization comparing to two state-of-the-art defenses: adversarial training [23], and BaRT [39].

CIFAR-10

The following four models were compared in terms of clean accuracy and accuracy under attacks in Table 3.1.

Table 3.1: Accuracy (%) of models under different attacks for CIFAR-10 and Oxford-IIIT Pet

		CIFAR-10		
Model	Clean	FGSM	PGD	CW
ResNet20	90.71	16.22	0.00	0.21
ResNet20 w/Adv.Train	52.76	42.05	40.95	52.76
ResNet20-BaRT	80.02	80.25	80.21	35.09
ResNet20-dquant	85.28	85.28	85.28	85.28
		Oxford-IIIT Pet		
ResNet34	93.10	6.97	0.00	25.58
ResNet34 w/Adv.Train	3.38	3.38	3.32	2.10
ResNet34-BaRT	83.15	35.25	37.01	57.92
ResNet34-dquant	94.99	94.99	94.99	94.99

- ResNet20: This is the baseline model trained with clean images.
- ResNet20 w/Adv.Train: This model is adversarially trained by PGD training [23], where $\epsilon = 16/255$.
- ResNet20-BaRT: This model with is by the defense, BaRT [39], where the number of transforms $k = 5$. There are ten transform groups in BaRT. In this experiment, only seven of them were used, excluding the zoom group, contrast group, and denoising group. The excluded transform groups caused the loss not a number (NaN) during the experiment. Therefore, these transforms were excluded in this experiment for the CIFAR-10 dataset.
- ResNet20-dquant: This model is trained with 1–bit dithered images by the defense framework of this chapter. One-bit dithering is enforced by the framework because dithering helps improve accuracy and adversarial noise on 1–bit images can be completely removed.

Although the baseline model (ResNet20) has a highest accuracy, the accuracy severely dropped under attacks. ResNet20-dquant achieved identical accuracy (i.e., 85.28 %) whether or not the model was under attack given a condition that test images were 1–bit. The accuracy of ResNet20 w/Adv.Train was 52.76 % under normal conditions (not under attack). In comparison, ResNet20-BaRT model maintained an accuracy of 80 % approximately. To further evaluate the defense models, PGD was run with differnet

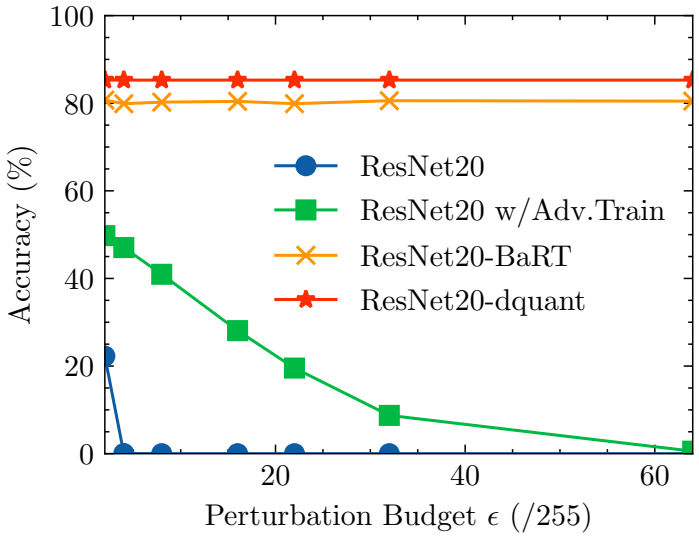

Figure 3.5: Accuracy (%) of models under PGD attack with various noise budgets for the CIFAR-10 dataset.

noise budgets ϵ, and Fig. 3.5 plots the accuracy against varying ϵ. The ResNet20-dquant and ResNet20-BaRT maintained a consistant accuracy and the other models decreased the accuracy as ϵ was increased. From the figure, the defense by quantization is effective even for the large ϵ if the test images are in 1–bit.

Oxford-IIIT Pet

In a similar fashion, the following four models were trained on the Oxford-IIIT Pet dataset.

- ResNet34: This model was trained with clean images; four epochs to the last layer and an additional four epochs to all unfrozen layers and the last layer by using learning rate policy (1cycle) [132].
- ResNet34 w/Adv.Train: This model was trained with adversarial examples generated by PGD (i.e., adversarial training [23]) with $\epsilon = 16$. The model was trained for 4 epochs to the last layer and 30 epochs for all the layers.
- ResNet34-BaRT: This model was trained for four epochs before unfreezing the layers and four epochs more after with BaRT applied images. Here, nine groups of transforms were utilized, excluding the denoising group due to the NaN loss problem.

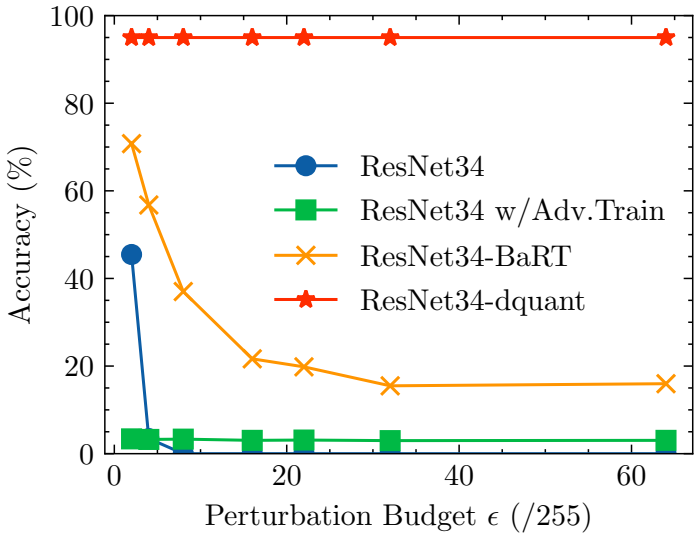

Figure 3.6: Accuracy (%) of models under PGD attack with various noise budgets for the Oxford-IIIT Pet dataset.

- ResNet34-dquant: 1–bit dithered images were used to train this model by the defense frame work of this chapter.

As presented in Table 3.1, with the help of transfer learning, ResNet34-dquant achieved higher accuracy than the baseline model (i.e., 94.99 %). The accuracy of ResNet34 w/Adv.Train was very low because it was only trained for 30 epochs and transfer learning was not useful for this particular case. Although the model, ResNet34-BaRT achieved the accuracy of 83.15 %, the accuracy significaltly dropped under attacks. In contrast, the defense model by quantization (ResNet34-dquant) maintained an identical accuracy whether or not the mode was under attacks. The graph of accuracy versus varying noise budgets ϵ was shown in Fig. 3.6. The ResNet34-dquant with the defense by quantization maintained the exact same accuracy for all noise distances (i.e., $\epsilon \leq 64$). The other models decreased the accuracy as ϵ was increased. Therefore, the experiments confirmed that the defense by quantization is effective if the test images are in 1–bit.

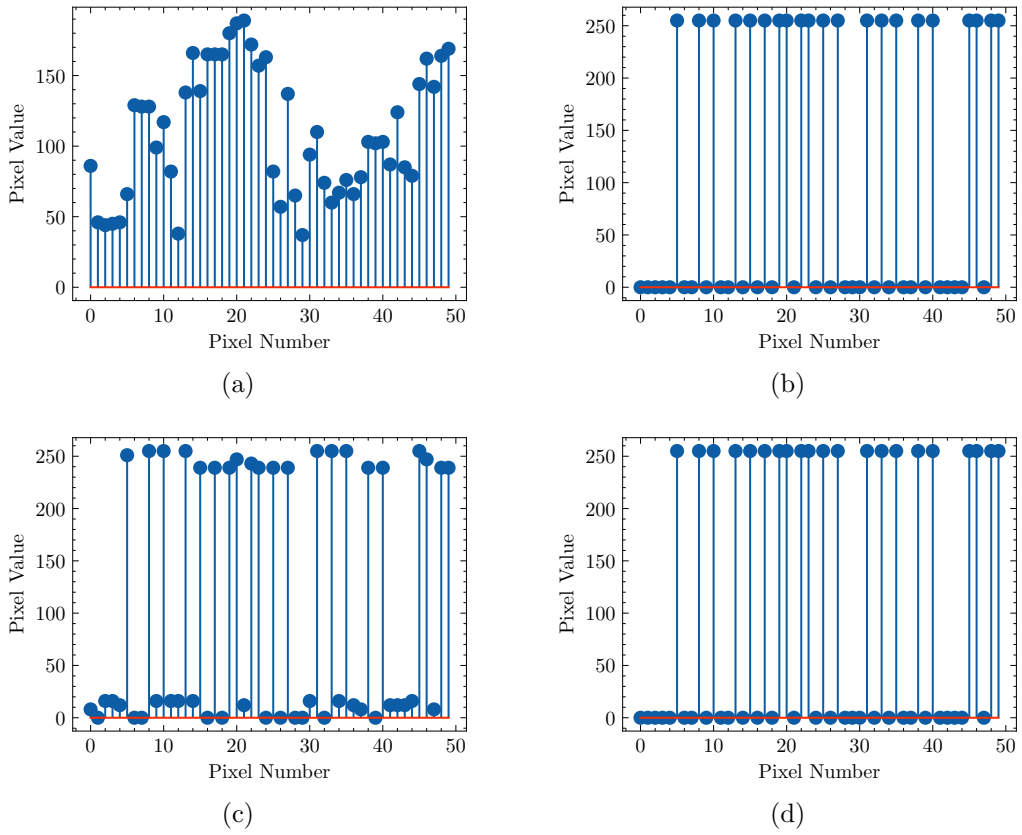

Figure 3.7: Visualization of image signal transform in quantization as a defense. (a) 8-bit image $\mathbf{x}$. (b) 1–bit image $\mathbf{x}_{1-\text{bit}}$. (c) Adversarial example $\mathbf{x'}_{1-\text{bit}}$ with noise distance $\epsilon = 16$. (d) 1–bit image $\hat{\mathbf{x}}_{1-\text{bit}}$ after removing adversarial noise.

3.3.3 Discussion

Justification

Let $\boldsymbol{\delta}$ be $\mathbf{x'}_{1-\text{bit}} - \mathbf{x}_{1-\text{bit}}$. Substituting $\mathbf{x'}_{1-\text{bit}} = \mathbf{x}_{1-\text{bit}} + \boldsymbol{\delta}$ to (3.3), we obtain

$$\mathbf{x'}_{1-\text{bit}} = \left\lfloor \frac{\mathbf{x}_{1-\text{bit}} + \boldsymbol{\delta}}{128} \right\rfloor \cdot 255. \tag{3.4}$$

Since $\mathbf{x}_{1-\text{bit}}(i,j) \in \{0, 255\}$, the following equation holds

$$\mathbf{x'}_{1-\text{bit}} = \mathbf{x}_{1-\text{bit}}, \tag{3.5}$$

under a condition that $|\boldsymbol{\delta}(i,j)| < 128$. This condition is generally satisfied because adversarial noise is usually imperceptible to keep low distortion and the previously

known tested noise distance is 32. Therefore, $\mathbf{x'}_{1-\text{bit}}$ becomes exactly the same as $\mathbf{x}_{1-\text{bit}}$ after applying linear quantization for $\epsilon < 128$.

For ease of understanding, 50 pixels from one test image were extracted, and the change in pixel values in the defense framework was visualized in Fig. 3.7. In the figure, an 8–bit image $\mathbf{x}$ in (a) was dithered to generate a 1–bit $\mathbf{x}_{1-\text{bit}}$ in (b). Next, to form the test image $\mathbf{x'}_{1-\text{bit}}$, an adversarial noise with the distance $\epsilon = 16$ was added as shown in (c). The noise was completely removed by linear quantization as shown in (d) (i.e., the original 1–bit image was reconstructed from the noisy one).

Limitation

The defense by quantization is effective only under the condition that the test images are in 1–bit (i.e., adversarial noise is added to the 1–bit images). However, to defeat the defense by quantization, an adaptive adversary can generate adversarial examples under ℓ_0 projection, which is to control the number of pixels changed. In an experiment, by flipping random 10 % of the pixels in a test image under the gradient direction, the accuracy dropped to 31.72 %. Therefore, the defense by quantization is applicable only for a limited scenario.

3.4 Summary

This chapter narrows down the adversarial space into 1–bit images and considers an adversarial defense by simple linear quantization. The defense by quantization in this chapter also introduces dithering to improve the classification accuracy for the first time. Experiments were carried out for two datasets under three attacks. Empirical results show that the defense by quantization achieved an identical accuracy whether or not the model is under attack. The results suggest that the defense by quantization was effective under the condition that the test images are in 1–bit, and the noise budget is less than 128. However, ℓ_0 attack which controls the number of pixels changed can attack the defense by quantization. Therefore, the defense by quantization works well only under a limited scenario.

Chapter 4

Key-Based Adversarial Defense

As neural networks are vulnerable to adversarial examples [9,10], the security of neural networks for applications such as autonomous vehicles, healthcare, and finance is under scrutiny. In response to the threat of adversarial examples, numerous defenses have been proposed in the literature. Yet, there is no robust model that can maintain a classification accuracy close to a non-protected one.

Most of the defense methods either reduce the classification accuracy or are completely broken. The current state-of-the-art empirically robust defense is adversarial training [23,42,93], but the accuracy of adversarially trained models is almost half lower than that of non-robust models. Other ideal desirable defenses are certified/provable defenses [85–87], but they are not scalable to larger datasets. Some certified defenses have been scaled to a certain degree [88–91], but the accuracy is still not comparable to empirically robust models. Other popular adversarial defenses that seem at first to have a high classification accuracy are input transformation approaches such as in [104–108]. However, they all have been defeated when accounting for obfuscated gradients (a way of gradient masking) [18]. To reinforce these weak defense methods, Raff et al. [39] proposed a stronger defense by combining a large number of transforms stochastically. However, applying many transforms results in a drop in accuracy even when a model is not under attack, and it is computationally expensive.

Recently, new insight into adversarial defense has been gained by taking inspiration from cryptographic principles [40,41,133]. The work by [41] bridges cryptography to adversarial defense, and another method by [40,133] has been inspired by perceptual image encryption methods, which were proposed for privacy-preserving machine learning and encryption-then-compression systems [112–114, 116, 134, 135]. The encryption-inspired adversarial defense (key-based defense) shows that as long as the key is kept secret, conventional gradient-based attacks cannot produce good gradients, thus rendering the attacks ineffective, even when the defense mechanism is known. This chapter aims to

achieve the following requirements:

1. High classification for clean images,
2. High classification for adversarial examples, and
3. Resistance against key estimation attacks.

In this chapter, the key-based defense is described in detail, and empirical results are presented by performing non-adaptive attacks and adaptive attacks in comparison with state-of-the-art defenses. Moreover, the chapter also discusses the advantages and limitations of the key-based defense succinctly.

4.1 Related Work

4.1.1 Previous Encryption-Inspired Defenses

The second Kerckhoffs's cryptographic principle states a system should not require secrecy even if it is exposed to the attacker, but the key should be secret [136]. By following this cryptographic principle, Taran et al. first introduced a defense method that uses a secret block of cryptographic transformation [41]. They proved the idea by using standard random permutation on small datasets (MNIST [117] and F-MNIST [118]) [41]. However, the use of cryptographic means is impractical because cryptography destroys the correlation, and convolutional neural networks work best with correlation in images. This contradiction causes a severe accuracy drop on larger datasets such as CIFAR-10 [119] and ImageNet [94]. To improve the classification accuracy, by taking inspiration from learnable image encryption techniques [115,116], block-wise pixel shuffling with a secret-key was proposed to maintain a high classification accuracy [133]. This chapter generalizes the previous key-based methods [41,133] and introduces other transformations in a block-wise manner to defend against adversarial examples with a secret key.

4.1.2 Learnable Image Encryption

Learnable image encryption (LIE) is to perceptually encrypt images to mainly protect visual information on plain images while maintaining the network ability to learn the encrypted ones for classification tasks. LIE methods are originally proposed for privacy-preserving deep learning, and have two requirements: protecting visual information and maintaining a high classification accuracy under the use of encrypted images. In a block-wise manner, a color image is divided into blocks, and each block is processed by using a

series of encryption with a common key to all blocks [116] or with different keys [115]. In a pixel-wise manner, negative/positive transformation to each pixel and color shuffling across three channels are exploited to produce learnable encrypted images [114, 134]. Another recent method to generate visually protected images is the use of a transformation network which is trained in cooperation with a pre-trained classification model. One such work utilized a generative adversarial network (GAN) [137]. To improve classification accuracy and robustness against various attacks, another approach that uses U-Net has been proposed [138, 139].

The key-based adversarial defense in this chapter is inspired by LIE methods, but targets to achieve a different goal. The defense does not aim to protect visual information on plain images but to achieve a high classification for both clean images and adversarial examples by keeping a secret key, which is used to control the model's decision.

4.2 Defense Framework by Secret Key

The main idea of the key-based defense is to embed a secret key into the model structure with minimal impact on model performance. Assuming the key stays secret, an attacker will not obtain any useful information on the model, which will render adversarial attacks ineffective.

A block-wise transformation with a secret key can be defined as a function. Let g be the transformation which takes input $\mathbf{x} \in [0,1]^{c \times h \times w}$ for a c-channel image of height h and width w, key K, and block size M and produces a transformed image (i.e., $g(\mathbf{x}, K, M) = \hat{\mathbf{x}}$). Here, the size of the input image is assumed to be a square shape (i.e., $h = w$). Figure 4.1 illustrates an overview of the training and inference phases of a classifier f with the key-based defense. Input images are transformed with a secret key prior to training and inference of the model.

4.2.1 Block-Wise Transformation with Secret Key

Key-based transformation g can be implemented in different ways. The defense by [40] introduced three block-wise transformations with a secret key: pixel shuffling (SHF), negative/positive transformation (NEG), and format-preserving Feistel-based encryption (FFX) [140]. Figure 4.2 depicts the process of block-wise transformation [40]. The detailed procedure of the block-wise transformation is as follows.

1. Divide $\mathbf{x}$ into blocks with a size of M such that $\{\mathbf{B}_{(1,1)}, \ldots, \mathbf{B}_{(\frac{h}{M}, \frac{w}{M})}\}$.

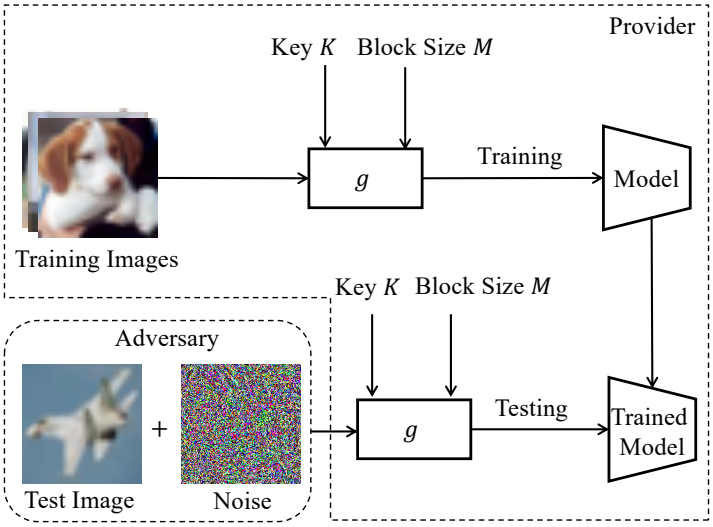

Figure 4.1: Overview of key-based defense framework.

2. Flatten each block tensor $\mathbf{B}_{(i,j)}$ into a vector, $\boldsymbol{b}_{(i,j)} = (b_{(i,j)}(1), \ldots, b_{(i,j)}(c \times M \times M))$.

3. (For SHF)

   - Generate a random permutation vector $\boldsymbol{v}$ with key K, such that $(v_1, \ldots, v_k, \ldots, v_{k'}, \ldots, v_{c \times M \times M})$, where $v_k \neq v_{k'}$ if $k \neq k'$, and $1 \leq v_n \leq c \times M \times M$.
   - Permutate every vector $\boldsymbol{b}_{(i,j)}$ with $\boldsymbol{v}$ as

     $$b'_{(i,j)}(k) = b_{(i,j)}(v_k) \tag{4.1}$$

     to obtain a shuffled vector, $\boldsymbol{b}'_{(i,j)}$ i.e., $(b'_{(i,j)}(1), \ldots, b'_{(i,j)}(c \times M \times M))$.

   (For NEG)

   - Generate a random binary vector $\boldsymbol{r}$ with key K, such that $(r_1, \ldots, r_k, \ldots, r_{c \times M \times M})$, where $r_k \in \{0, 1\}$. To keep the transformation consistent, $\boldsymbol{r}$ is distributed with 50 % of "0"s and 50 % of "1"s.
   - Perform negative/positive transformation on every vector $\boldsymbol{b}_{(i,j)}$ with $\boldsymbol{r}$ as

     $$b'_{(i,j)}(k) = \begin{cases} b_{(i,j)}(k) & (r_k = 0) \\ b_{(i,j)}(k) \oplus (2^L - 1) & (r_k = 1), \end{cases} \tag{4.2}$$

     where $\oplus$ is an exclusive or (XOR) operation, L is the number of bits used in $b_{(i,j)}(k)$, and $L = 8$ to obtain a transformed vector, $\boldsymbol{b}'_{(i,j)}$.

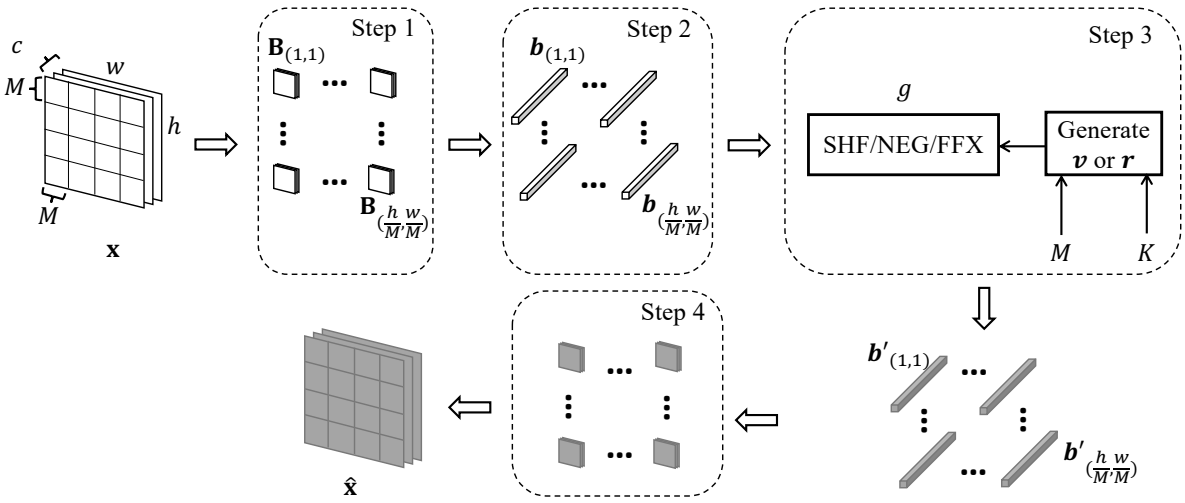

Figure 4.2: Procedure of block-wise transformation with secret key, $g(\mathbf{x}, K, M)$ which takes image $\mathbf{x}$, key K and block size M, and outputs transformed image $\hat{\mathbf{x}}$.

(For FFX)

- Generate a random binary vector $\boldsymbol{r}$ with key K, such that $(r_1, \ldots, r_k, \ldots, r_{c \times M \times M})$, where $r_k \in \{0, 1\}$. To keep the transformation consistent, $\boldsymbol{r}$ is distributed with 50 % of "0"s and 50 % of "1"s.
- Apply FFX to every vector $\boldsymbol{b}_{(i,j)}$ with $\boldsymbol{r}$ as

$$b'_{(i,j)}(k) = \begin{cases} b_{(i,j)}(k) & (r_k = 0) \\ \mathrm{Enc}(b_{(i,j)}(k)) & (r_k = 1), \end{cases} \tag{4.3}$$

  where $\mathrm{Enc}(\cdot)$ is format-preserving Feistel-based encryption [140] configured with an arbitrary password and a length of 3 digits to cover the whole range of pixel values from 0 to 255, to obtain a transformed vector, $\boldsymbol{b}'_{(i,j)}$.

4. Integrate the transformed vectors to form a transformed image tensor $\hat{\mathbf{x}}$.

Notably, the standard random perturbation used in Taran et al.'s method [41] is a special case of SHF with the block size, which is equal to the image size (i.e., $M = w = h$). In addition, different transformations produce different models, which is useful for diversifying models in an ensemble. Moreover, users can have flexibility in choosing a transformation according to desired requirements in a specific application.

4.2.2 Key Space

Typically, defense methods do not have an information advantage over attack methods. Once the defense mechanism is known, an attacker can design adaptive attacks to easily bypass the defense methods [18]. In key-based methods, the key is secret, and it is critical to the strength of the defense method. In this regard, to keep the key secret, it is assumed that transformed images are also secret because the key may be recovered from these images. Therefore, the defensive key-based transformation should be regarded as an intermediate layer, and transformed images should not be saved.

The key spaces of the three transformations from [40] vary depending on block size M. For SHF, the key space is given by

$$\mathcal{K}_{\text{SHF}}(c \times M \times M) = (c \times M \times M)!. \tag{4.4}$$

For NEG and FFX, 50 % of the pixels in each block are inversed/encrypted, and the key controls which pixels are inversed/encrypted. Therefore, their key spaces are the same and written as

$$\mathcal{K}_{\text{NEG/FFX}}(c \times M \times M) = \frac{(c \times M \times M)!}{((c \times M \times M)/2)! \cdot ((c \times M \times M)/2)!}. \tag{4.5}$$

When M is small, brute-force attacks are possible, and attackers can heuristically estimate the key by observing the model accuracy if the model weights are available to the attackers. To overcome this limitation, the work by [141] proposed combining SHF and NEG without degrading the model accuracy. Since there are two transformations in [141], the 50 % constraint on the number of pixels to be transformed is relaxed. Thus, the new combined key space becomes

$$\mathcal{K}_{(\text{SHF + NEG})}(c \times M \times M) = (c \times M \times M)! \times 2^{(c \times M \times M)}. \tag{4.6}$$

4.2.3 Ensemble of Key-Based Models

By following Kerckhoffs's second cryptographic principle, the key-based defense keeps a secret key. Therefore, there is an information advantage over the gradient-based attacks. Assuming the key stays secret, an attacker cannot compute any useful gradients on the model, which will render the existing gradient-based adversarial attacks ineffective. However, an attacker may perform gradient-free attacks (i.e., black-box attacks) without the secret key if the attacker can obtain the probability scores of model prediction.

One straightforward way of defending against such attacks is to form an ensemble. Ensemble methods have been used to improve model predictions in general. There are many forms of an ensemble: voting, bagging, boosting, and stacking. A simple

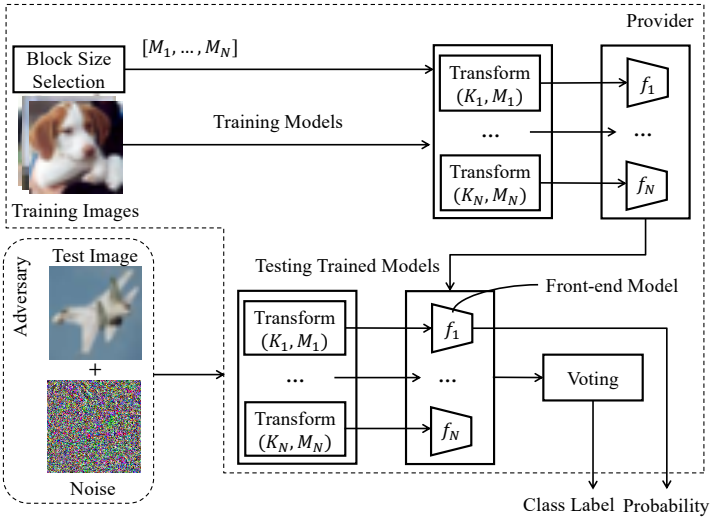

Figure 4.3: Ensemble key-based defense framework [142].

voting ensemble can be formed with models trained by using key-based transformed images [142].

Concretely, an ensemble of N models $\{f_1, \ldots, f_N\}$ is trained by using images transformed by SHF with different keys and block sizes. An overview of the ensemble defense framework is shown in Fig. 4.3. Block sizes $\{M_1, \ldots, M_N\}$ are first selected, and then N models are trained by using images transformed with the selected $\{M_1, \ldots, M_N\}$ and keys $\{K_1, \ldots, K_N\}$. One of the models in the ensemble, f_1, is a front-end model (i.e., public-facing model) that outputs the probability of a prediction. A final class label is determined on the basis of voting prediction results from all models.

When an attacker attacks the public-facing model (f_1), the adversarial noise generated on f_1 is not efficiently transferable to other models in the ensemble. Thus, the voting ensemble is able to defend against score-based black-box attacks. It was confirmed that the ensemble defense [142] achieved more than 87 % accuracy under score-based attacks such as in [78, 143, 144] and a clean accuracy of $\approx$ 95 % for the CIFAR-10 dataset.

4.3 Threat Models

As described in [56], a threat model includes a set of assumptions such as an adversary's goals, capabilities, and knowledge. This chapter considers untargeted attacks and employs three white-box attacks: projected gradient descent (PGD) under the ℓ_∞-norm [23], Carlini and Wagner's attack (CW) under the ℓ_2-norm [67], and elastic-net

attack (EAD) under the ℓ_1-norm [68].

The adversarial example, $\mathbf{x}'$ at the $(i+1)^{\text{th}}$ iteration with PGD [23] is

$$\mathbf{x}'_{(i+1)} = \Pi_{\mathbf{x}+\Delta}(\mathbf{x}'_i + \alpha \cdot \text{sign}(\nabla_{\mathbf{x}} \mathcal{L}(f(\mathbf{x}'_i), y))), \tag{4.7}$$

where α is the step size, and $\Pi_{\mathbf{x}+\Delta}$ denotes projection to the allowable set of perturbation, Δ. PGD is known as a powerful first-order adversary.

CW reformulates the optimization problem by using Lagrangian relaxation to find the minimum perturbation under an ℓ_p norm [67]. Specifically, they solve the following optimization,

$$\arg\min_{\boldsymbol{\delta}} \|\delta\|_p + \lambda \cdot \mathcal{L}'(\mathbf{x} + \boldsymbol{\delta}) \text{ s.t. } \mathbf{x} + \boldsymbol{\delta} \in [0,1]^{c \times h \times w}, \tag{4.8}$$

where $\|\cdot\|_p$ is a distance function under ℓ_p norm, λ is a hyperparameter and $\mathcal{L}'$ is a new objective function defined as,

$$\mathcal{L}'(\mathbf{x} + \boldsymbol{\delta}) = \max\left(\max\{Z(\mathbf{x} + \boldsymbol{\delta})_i : i \neq t\} - Z(\mathbf{x} + \boldsymbol{\delta})_t, -\kappa\right), \tag{4.9}$$

where $Z(\cdot)$ denotes a softmax function, and κ is a constant to control the confidence so that $\mathbf{x} + \boldsymbol{\delta}$ is classified as a target class t.

To encourage sparsity in the perturbation, EAD generates ℓ_1-oriented adversarial examples [68]. EAD combines ℓ_1 and ℓ_2 regularization, optimizes the following objective:

$$\arg\min_{\boldsymbol{\delta}} \lambda \cdot \mathcal{L}'(\mathbf{x} + \boldsymbol{\delta}) + \beta \|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\delta}\|_2^2 \text{ s.t. } \mathbf{x} + \boldsymbol{\delta} \in [0,1]^{c \times h \times w}. \tag{4.10}$$

CW is also a special case of EAD, where the ℓ_1 regularization parameter, β is set to zero [68].

The chapter also assumes that the inner mechanism of the key-based defense is known to the attackers except for the secret key. Therefore, to evaluate the key-based defense, the three white-box attacks are utilized in both adaptive and non-adaptive manners.

4.4 Experiments and Discussion

4.4.1 Experiment Conditions

Datasets

Two datasets were used: CIFAR-10 [119] and ImageNet [94]. CIFAR-10 consists of 60,000 color images (dimension of $32 \times 32 \times 3$) with 10 classes (6000 images for each

class), where 50,000 images are for training and 10,000 for testing. For data preprocessing, a batch size of 128 and live augmentation (random cropping with a padding of 4 and random horizontal flip) were used for the training set. ImageNet comprises 1.28 million color images for training and 50,000 color images for validation. Images were progressively resized during training, starting with larger batches of smaller images to smaller batches of larger images. Three phases of training were deployed from the DAWNBench top submissions as mentioned in [42]. Phases 1 and 2 resized images to 160 and 352 pixels, respectively, and phase 3 used the entire image size from the training set. The augmentation methods used in the experiment were random resizing and cropping (sizes of 128, 224, and 288 respectively for each phase) and random horizontal flip.

Networks

Deep residual networks [3] were utilized to evaluate the key-based defense. For CIFAR-10, ResNet18 was trained for 200 epochs with efficient training techniques from the DAWNBench top submissions: cyclic learning rates [145] and mixed-precision training [146]. The parameters of the stochastic gradient descent (SGD) optimizer were a momentum of 0.9, weight decay of 0.0005, and maximum learning rate of 0.2. For ImageNet, ResNet50 with pre-trained weights was used according to the training settings from [42] with the removal of weight decay regularization from the batch normalization layers. The network was trained for 15 epochs in total for the ImageNet dataset.

Attack Settings

Three white-box attacks were used to test the key-based defense: PGD under ℓ_∞-norm [23], CW under ℓ_2-norm [67], and EAD under ℓ_1-norm [68]. The PGD attack was configured with a noise distance of 8/255, a step size of 2/255, 50 iterations, and random initialization. Since the evaluation was on untargeted attacks, CW and EAD were configured with a confidence value of 0, learning rate of 0.01, binary search steps of 9, and an initial constant of 0.001 for 1000 iterations for CIFAR-10 and 100 iterations for ImageNet. EAD was set up with the elastic-net (EN) decision rule.

Evaluation Metrics

Two metrics were used to measure the performance of the key-based defense: accuracy (ACC) and attack success rate (ASR). ACC is given by

$$\text{ACC} = \begin{cases} \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(f(\mathbf{x}_i) = y_i) & \text{(clean)} \\ \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(f(\mathbf{x}_i + \boldsymbol{\delta}_i) = y_i) & \text{(attacked)}, \end{cases} \tag{4.11}$$

and ASR is defined as

$$\text{ASR} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(f(\mathbf{x}_i) = y_i \wedge f(\mathbf{x}_i + \boldsymbol{\delta}_i) \neq y_i), \tag{4.12}$$

where N is the number of test images, $\mathbb{1}$(condition) is one if condition is true, otherwise zero, $\{\mathbf{x}_i, y_i\}$ is a test image ($\mathbf{x}_i$) with its corresponding label (y_i), and $\boldsymbol{\delta}_i$ is its respective adversarial noise depending on a specific attack.

4.4.2 Results

Clean Classification Accuracy

Table 4.1 captures the clean accuracy (ACC) of both the standard (no defense) and key-based defense models for the CIFAR-10 and ImageNet datasets (i.e., when the models were not under attack). Different block sizes ($M \in \{2, 4, 8, 16\}$) were used for CIFAR-10, and only $M = 4$ was selected for ImageNet due to the expense of training an ImageNet model. The ACC was calculated for the whole test set (10,000 images for CIFAR-10 and 50,000 images for ImageNet). The trained models are denoted by their defense method and block size in Table 4.1. For example, a model trained by using SHF with a block size of $M = 2$ is indicated as "SHF ($M = 2$)." From Table 4.1, the key-based defense models did not reduce the ACC significantly except for SHF with $M = 8$ and 16 for the CIFAR-10 dataset. The results show that the key-based defense models achieved a high clean accuracy close to that of a non-protected model.

Robustness Against Non-Adaptive Attacks

Models with different block sizes ($M \in \{2, 4, 8, 16\}$) for CIFAR-10 and models with $M = 4$ for ImageNet were attacked by three adversaries bounded by different norm balls: PGD (ℓ_∞) [23], CW (ℓ_2) [67], and EAD (ℓ_1) [68]. The attacks were carried out without considering the defense (i.e., non-adaptive). Table 4.2 summarizes the results for both CIFAR-10 and ImageNet.

CIFAR-10: The ASR was too low for all models under all attacks except for NEG with $M = 8$ and 16. It is interesting to note that PGD could effectively defeat these two NEG models, suggesting that the block size M plays an important role in key-based defenses. However, the models using NEG with $M = 2$ and 4 were resistant against PGD attacks and yielded low ASR values.

ImageNet: All three models had low ASR values for all attacks. Therefore, the key-based defense was effective at defending against non-adaptive attacks even for a large dataset like ImageNet.

Table 4.1: ACC (%) of standard and key-based defense models

Model	ACC (CIFAR-10)	ACC (ImageNet)
Standard (No defense)	95.45	73.70
SHF ($M = 2$)	94.45	–
SHF ($M = 4$)	91.84	72.41
SHF ($M = 8$)	85.12	–
SHF ($M = 16$)	76.22	–
NEG ($M = 2$)	95.32	–
NEG ($M = 4$)	93.41	72.63
NEG ($M = 8$)	91.54	–
NEG ($M = 16$)	92.68	–
FFX ($M = 2$)	93.67	–
FFX ($M = 4$)	92.30	72.18
FFX ($M = 8$)	91.99	–
FFX ($M = 16$)	91.38	–

Since PGD is a stronger adversary from the empirical results, the models with $M = 4$ were further evaluated with PGD with different perturbation budgets (i.e., $\epsilon \in \{2, 4, 8, 16, 22, 32\}$). A graph of ACC versus perturbation budget ϵ is shown in Fig. 4.4. For both datasets, the ACC dropped as the perturbation budget ϵ increased. All the models maintained a high ACC value when $\epsilon = 8/255$. However, the models reduced the ACC for $\epsilon = 32/255$ significantly.

Robustness Against Adaptive Attacks

Three possible adaptive attacks, key estimation, estimation over transformation, and transferred attack, are described, and the results of the attacks are presented as follows.

Key Estimation

Without the correct key or a near-correct key, conventional white-box attacks will not work on the key-based defense. Brute-force attacks may not be feasible if the key space is large enough. A heuristic way of searching for the key is to change the key on the basis of accuracy. In the key-based defense, key K is used to generate a random permutation vector $\boldsymbol{v} = (v_1, \ldots, v_{c \times M \times M})$ for SHF and a random binary vector $\boldsymbol{r} = (r_1, \ldots, r_{c \times M \times M})$ for NEG and FFX. Therefore, the adversary can generate $\boldsymbol{v}'$ or $\boldsymbol{r}'$ with random key K'

Table 4.2: ASR (%) of standard and key-based defense models under non-adaptive attacks

	CIFAR-10			ImageNet		
Model	PGD (ℓ_∞)	CW (ℓ_2)	EAD (ℓ_1)	PGD (ℓ_∞)	CW (ℓ_2)	EAD (ℓ_1)
SHF ($M = 2$)	7.90	0.00	0.00	–	–	–
SHF ($M = 4$)	3.82	0.00	0.00	8.43	0.00	0.20
SHF ($M = 8$)	3.19	0.00	0.00	–	–	–
SHF ($M = 16$)	2.21	0.00	0.00	–	–	–
NEG ($M = 2$)	8.93	0.18	0.09	–	–	–
NEG ($M = 4$)	3.18	0.00	0.00	6.36	0.00	0.00
NEG ($M = 8$)	90.30	2.66	4.28	–	–	–
NEG ($M = 16$)	98.88	5.31	9.31	–	–	–
FFX ($M = 2$)	4.53	0.37	0.28	–	–	–
FFX ($M = 4$)	4.37	0.28	0.00	8.86	0.80	0.00
FFX ($M = 8$)	4.07	0.19	0.09	–	–	–
FFX ($M = 16$)	2.17	0.76	0.09	–	–	–

Algorithm 2 Bubble sort-like Approach

Input: A batch of images, model
Output: $\boldsymbol{v}'$ or $\boldsymbol{r}'$

```
Initialize v' or r' with a random key K'
accuracy ← 0
for i ← 1, ..., c × M × M do
    for j ← 1, ..., (c × M × M) − i do
        Swap v'_j and v'_{j+1} or r'_j and r'_{j+1}
        current_accuracy ← Calculate the accuracy of the model
        if current_accuracy > accuracy then
            accuracy ← current_accuracy
        else
            Swap v'_j and v'_{j+1} or r'_j and r'_{j+1}        ▷ Revert the swap
        end if
    end for
end for
```

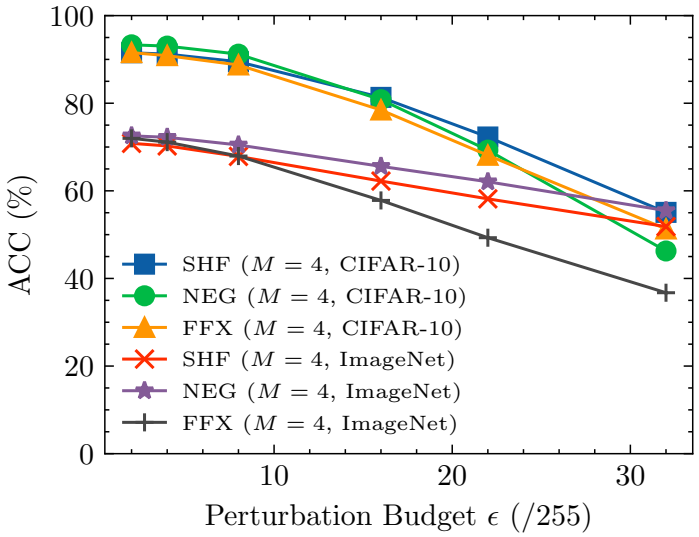

Figure 4.4: ACC of key-based defense with $M = 4$ under PGD attack with various ϵ for both CIFAR-10 and ImageNet.

and modify $\boldsymbol{v}'$ or $\boldsymbol{r}'$ by using the average accuracy over a batch of images as a guide to carry out an adaptive attack. This can be done in a variety of ways. One straightforward way is to run a bubble sort-like algorithm (repeatedly swapping the adjacent elements if the accuracy improves as in Algorithm 2). Another way of estimating the key is finding the correct position of an element in $\boldsymbol{v}'$ or $\boldsymbol{r}'$ by swapping an element with every other element in $\boldsymbol{v}'$ or $\boldsymbol{r}'$ [40] (Algorithm 3) or by swapping every possible pair of elements in $\boldsymbol{v}'$ or $\boldsymbol{r}'$ in accordance with the improvement in accuracy [141] (Algorithm 4).

After estimating $\boldsymbol{v}'$ or $\boldsymbol{r}'$, PGD was deployed to attack the models with $M = 4$. Table 4.3 shows the results of all three algorithms for key estimation attacks, where Algorithm 3 was run with $T = 10$ as reported in [40]. Algorithm 3 found an effective key for NEG with $M = 4$, achieving an ASR of 77.76 %, which was significant. However, the key estimation attacks were not successful for the other two models (i.e., SHF and FFX).

Estimation over Transformation Attack

The Estimation over Transformation Attack (EOT) is effective for estimating gradients in adversarial defenses with randomization as explained in [18]. Instead of taking one step in the direction of gradients $\nabla_{\mathbf{x}} f(\mathbf{x})$, the direction of $\sum_{i=1}^{30} \nabla_{\mathbf{x}} f(\mathbf{x})$ is accounted for to have better gradients. In other words, 30 keys are used to generate adversarial examples under a PGD attack. Experiments results from [40] show that the ASR was

Algorithm 3 Method in [40]

Input: A batch of images, model
Output: $\boldsymbol{v}'$ or $\boldsymbol{r}'$

```
Initialize v' or r' with a random key K'
accuracy ← 0
for t ← 1...T do
    for i ← 1, ..., c × M × M do
        for j ← i + 1, ..., c × M × M do
            Swap v'_i and v'_j or r'_i and r'_j
            current_accuracy ← Calculate the accuracy of the model
            if current_accuracy > accuracy then
                accuracy ← current_accuracy
            else
                Swap v'_i and v'_j or r'_i and r'_j          ▷ Revert the swap
            end if
        end for
    end for
end for
```

Algorithm 4 Method in [141]

Input: A batch of images, model
Output: $\boldsymbol{v}'$ or $\boldsymbol{r}'$

```
Initialize v' or r' with a random key K'
indices ← {(1,2), (1,3), ..., (i,j), ..., (c × M × M − 1, c × M × M)}
accuracy ← 0
for each pair (i, j) in indices do
    Swap v'_i and v'_j or r'_i and r'_j
    current_accuracy ← Calculate the accuracy of the model
    if current_accuracy > accuracy then
        accuracy ← current_accuracy
    else
        Swap v'_i and v'_j or r'_i and r'_j                  ▷ Revert the swap
    end if
end for each
```

Table 4.3: ASR (%) of standard and key-based defense models under adaptive attacks for the CIFAR-10 dataset

Model	Key Estimation			EOT	Transferred Attack
	Bubble	Method [40] ($T = 10$)	Method [141]		
SHF ($M = 4$)	4.17	3.70	3.70	1.70	5.07
NEG ($M = 4$)	3.80	77.76	28.73	1.58	1.49
FFX ($M = 4$)	3.46	3.27	3.09	5.24	6.41

too low for this type of attack (Table 4.3), suggesting the strength of key-based defenses.

Transferred Attack

Since the defense mechanism is known to the attacker, a substitute model can be trained with the attacker's assumed key. Then, the attacker generates adversarial examples with PGD over the substitute model. This type of attack can be successful if the attacker's key is close to the correct secret key. Table 4.3 shows the empirical results from [40] for the transferred attack. The ASR was low, and the adversarial examples from one key-based defense model could not be transferred to another one. However, this type of attack may be more effective if the attacker's key is closer to the correct one.

Comparison with State-of-the-art Defenses

Most of the recent adversarial defenses, especially input transformation-based defenses, have been invalidated by rigorous adaptive attacks [18, 19]. Although adversarial training is repeatedly found effective for conventional attacks like PGD, it has been known to be extremely difficult at the ImageNet scale due to the high computation cost [65]. Recently, fast adversarial training was proposed to overcome such difficulty [42]. The key-based defenses (both standard random permutation (SRP) [41] and block-wise transformation [40]) have been compared with fast adversarial training (Fast AT) [42] and a recent feature scattering-based approach (FS) [43] in terms of accuracy, whether or not the model was under PGD attack with a perturbation budget of $\epsilon = 8/255$.

CIFAR-10: The graph in Fig. 4.5 shows bar charts of the clean and attacked accuracy (i.e., model accuracy under attack) of the defense models. One key-based defense, NEG, achieved the highest accuracy whether or not the model was under attack. However, NEG was vulnerable under key estimation attacks as shown in Table 4.3. Although, the clean accuracies of Fast AT [42] and FS [43] were relatively high

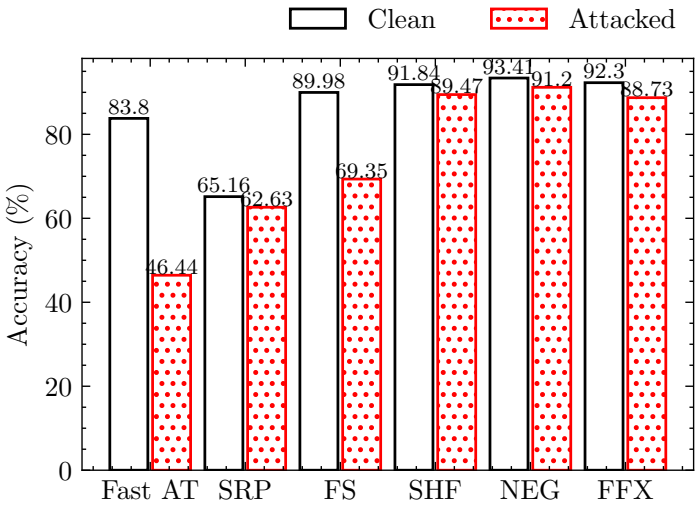

Figure 4.5: Comparison of key-based defenses ($M = 4$) and state-of-the-art defenses under PGD with $\epsilon = 8/255$ for CIFAR-10.

(i.e., 83.80 % and 89.98 %), the attacked accuracy was significantly low (46.44 % and 69.35 %, respectively). The overall accuracy of SRP [41] was low. It can be clearly seen that block-wise transformation provides a better performance accuracy even though the same pixel shuffling operation was applied as in [41]. In short, key-based defenses outperformed Fast AT [42] and FS [43] by having the information advantage of secret keys for the CIFAR-10 dataset.

ImageNet: Similarly, the graph in Fig. 4.6 shows a performance comparison of Fast AT [42] and key-based defense models. The clean accuracy of SRP [41] for ImageNet was significantly low, and therefore, it was excluded from the comparison. The defense by FS [43] was not available for ImageNet, and thus, the comparison was done with Fast AT [42] only. When under attack, Fast AT [42] reduced the accuracy severely (i.e., 11.61 %). From the empirical results, key-based defense models can be scaled to the ImageNet level and maintain a high classification whether or not the models are under attack.

4.4.3 Discussion

Advantages

- Classification Accuracy: Key-based defenses provide almost the same accuracy as non-protected ones. One possible reason is that block-wise transformation

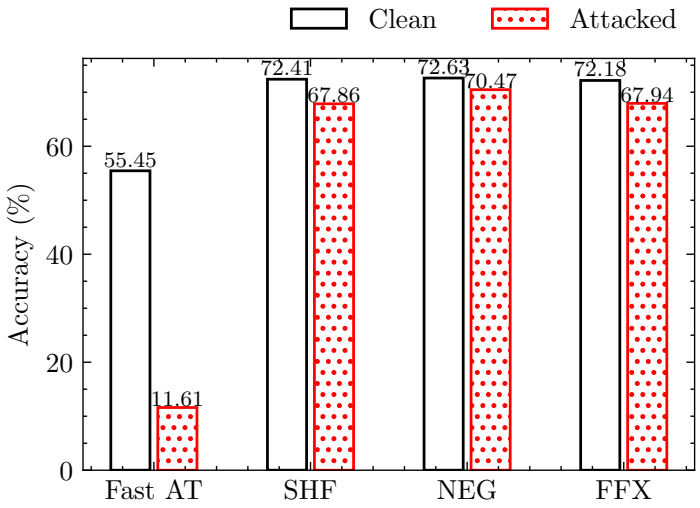

Figure 4.6: Comparison of key-based defenses ($M = 4$) and a state-of-the-art defense under PGD with $\epsilon = 8/255$ for ImageNet.

maintains some correlation between pixels as reported in [44]. Moreover, each block position in an input image is not changed (keeping spatial information) unlike traditional encryption-then-compression (EtC) systems [112].

- Information Advantage for Defenses: Assuming the key stays secret, gradient-based attacks are not effective against key-based defenses. Previous defenses, especially input transformation defenses, are broken on account of obfuscated gradients [18]. In contrast, in the key-based defense, transformed images are not stored and are not available to the attacker. Thus, techniques like Backward Pass Differentiable Approximation (BPDA) [18] are not applicable. Therefore, the key-based defense has an information advantage over such attacks.

- Low Computation Cost: The block-wise operation utilized in the key-based defense can be efficiently implemented with vectorized operations and is available for large-scale systems without any noticeable overheads during training/inference, in contrast to other image processing-based defenses such as [39].

- Application Range: The key-based defense can be expanded for other applications such as model access control [44] and model watermarking [46].

Limitations

- High Perturbation Budget: In general, all adversarial defenses are designed to defend against a certain noise budget. In real-world application scenarios, adversarial noise cannot be controlled. Therefore, to account for a high perturbation budget, the key-based defense alone is not enough, and it should be combined with other detection methods.

- Black-Box Attacks: The key-based defense does not have any information advantage over score-based and decision-based attacks. Therefore, such attacks may successfully defeat the key-based defense in its current form. To overcome this limitation, a simple solution is to add a randomization component to the key-based defense or to form an ensemble. There are many ways of forming an ensemble, such as voting, bagging, boosting, and stacking. A recent work shows that a simple voting ensemble can be formed with models trained by using key-based transformed images to defend against score-based attacks [142].

- Application Scenario: The key-based defense is specifically designed for image classification models. Future research is required for other application scenarios such as semantic segmentation and image retrieval.

4.5 Summary

This chapter presents a key-based defense that has the information advantage over the attacks by following the second Kerckhoffs's cryptographic principle. By keeping the key secret, an attacker can not obtain any useful information on the model, which will render adversarial attacks ineffective. The key-based defense in this chapter utilizes a block-wise transformation with a secret key and is implemented in three different transformations: pixel shuffling (SHF), negative/positive transformation (NEG), and format-preserving Feistel-based encryption (FFX). Experiment results show that the key-based defense maintained a high classification accuracy under gradient-based attacks, and outperformed state-of-the-art adversarial defenses. However, the key-based defense does not have any information advantage over black-box attacks.

Chapter 5

Model Protection by Secret Key

Training a production-level neural network model requires a significant amount of resources such as a huge amount of data, powerful computing hardware, and efficient algorithms. Therefore, trained models are a new form of intellectual property (IP), and have great business value. In order to protect trained models from economic damage, it is necessary to protect the IP of trained models. There are two aspects of IP protection for CNN models: access control and watermarking.

Access control of trained models prevents unauthorized users from illegal usage of the models and piracy. Unlike typical digital content such as audio, image, video, etc., neural network models are not static, but functional. Therefore, a stolen model can be exploited for business gains from its competitors. Model access control addresses this issue by ensuring only authorized users can use the model to its full capacity.

Another form of protecting trained models is through watermarking inspired by digital watermarking methods. To combat copyright infringement, researchers have proposed to embed a watermark into a neural network model, and the embedded watermark is used to verify the ownership of the model. Model watermarking methods focus on ownership verification and do not aim to protect the functionality of the model.

In order to prevent potential economic loss and to properly credit the model owners, model protection methods (both access control and watermarking) are increasingly important and demanding. This chapter presents two model access control frameworks and one model watermarking framework by a secret key in the following sections. This chapter aims to achieve the following requirements for model access control:

1. High classification accuracy for authorized users,
2. Low classification accuracy for unauthorized users, and
3. Resistance against key estimation attacks and fine-tuning attacks.

For model watermarking, this chapter aims to achieve the following requirements:

1. High classification accuracy for clean images,
2. High classification accuracy for transformed images, and
3. Resistance against pruning and fine-tuning attacks accounting for piracy and ambiguity attacks.

5.1 Related Work

5.1.1 Previous Model Access Control Methods

Chen and Wu first proposed a model protection method against unauthorized access inspired by adversarial examples [27]. Their method utilized an anti-piracy transform module which is a secret perturbation network in such a way that the secret perturbation is crucial to the model's decision to control the access of the model [27]. In other words, only the rightful users who have access to the secret perturbation can use the model properly. However, this method [27] requires training a perturbation network along with the classification network so that the optimal perturbation can be learned. In addition, the classification accuracy of the method by [27] slightly drops compared to non-protected models under the same training settings.

5.1.2 Previous Model Watermarking Methods

There are two types of model watermarking methods: white-box and black-box.

White-box approaches require access to model weights for embedding and detecting watermarks in a model. These methods use an embedding regularizer, which is an additional regularization term in a loss function during training [28–30]. A recent study [35] showed that these regularizer-based methods can be attacked. Another paper [32] highlighted that if watermarks are independent of a model's performance, they are vulnerable to ambiguity attacks [147] where two watermarks can be extracted from a protected model, causing confusion regarding ownership. Therefore, they introduced passports and passport layers [32]. However, a recent paper [36] pointed out that ownership verification can be broken by using reverse-engineered secret passport weights. Accordingly, these white-box approaches are not practical in real-world applications such as online services because access to the model weights from a plagiarized party is not supported.

In black-box approaches, watermarks are extracted by observing the input and output of a model. A study in [34] introduced a black-box method by using adversarial examples. Another study in [121] implanted a backdoor in a model so that a watermark can be triggered through the backdoor. Generally, in black-box approaches, a special set of training examples is used so that watermarks are extracted from the inference of a model [31–34]. Li et al. pointed out that backdoor attack-based methods can be defeated by existing backdoor defenses (e.g. [148]), and most of the existing methods are not robust enough against piracy attacks, where a verifiable watermark is injected into a model while maintaining the model's accuracy as described in [36]. Therefore, Li et al. proposed a method called "null embedding", which embeds a pattern into a model's decision process during the model's initial training [36]. However, the effectiveness of their method has not been confirmed yet on large networks such as residual networks [3], which are widely used for image classification tasks.

Similar to the work by Li et al., this chapter introduces a model watermarking framework that uses learnable transformed images with a secret key, in which the original watermark cannot be removed by piracy attacks.

5.1.3 Block-Wise Transformation with Secret Key

Model protection methods (both access control and watermarking) in this chapter adopt the block-wise transformation with a secret key from the key-based adversarial defense in Chapter 4, Section 4. Three transformations: pixel shuffling (SHF), negative/positive transformation (NEG), and format-preserving Feistel-based encryption (FFX) from the adversarial defense (Chapter 4) are employed for a new application, model protection in this chapter. However, there are a few differences as follows:

- For NEG and FFX, the restriction of 50 % of the pixels in each block are flipped or encrypted, is relaxed. Therefore, the key spaces of these two transformations (NEG and FFX) have become enlarged as

$$\mathcal{K}_{\mathrm{NEG/FFX}}(c \times M \times M) = 2^{(c \times M \times M)}. \tag{5.1}$$

- Model protection methods utilize various transformation choices (both single and combined transformations). In contrast, the adversarial defense uses only one or two transformations.

5.2 Model Access Control by Secret Key

There are two variations of model access control frameworks by the block-wise transformation with a secret key: input transformation and feature map transformation. For

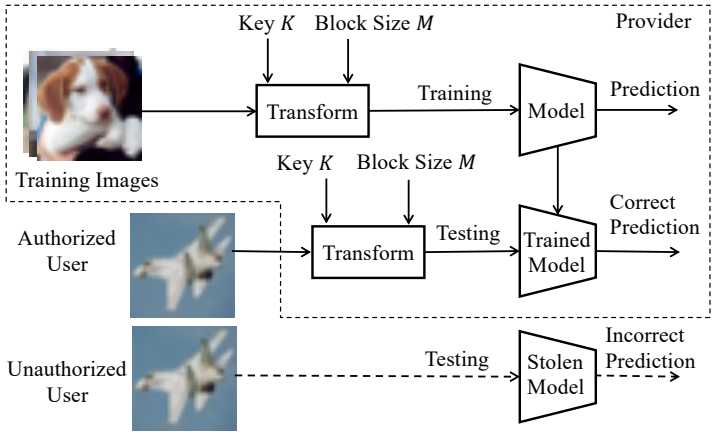

Figure 5.1: Access control framework by input transformation with secret key.

the first one, the block-wise transformation (SHF, NEG, FFX, or any combination of the three) is applied to input images, while the second one applies only SHF to one or more feature maps of the model.

5.2.1 Input Transformation

Figure 5.1 depicts the framework of model access control by input transformation with a secret key. In this framework, input images are transformed by using block-wise transformation with key K and block size M before training or testing a model. To test a trained model, test images are also transformed with the same key K and block size M. Here, there can be more than one transformation, therefore K can be a set of keys depending on the number of transformations. When the model is stolen, unauthorized users cannot get correct predictions in the absence of the secret key as shown in Fig. 5.1.

5.2.2 Feature Map Transformation

Instead of transforming input images, one or more feature maps in the network are transformed by a block-wise transformation (SHF) with secret key K, and block size M. Figure 5.2 illustrates the access control framework by feature map transformation with a secret key. The transformation, SHF is applied to feature maps in the network (e.g., ResNet-18), where the modified network is trained by using secret key K as shown in Fig. 5.2. The model predicts a test image correctly only for authorized users with secret key K, and the model provides incorrect predictions for unauthorized users when the model is stolen. Accordingly, the stolen model cannot be used to its full capacity when secret key K is not available.

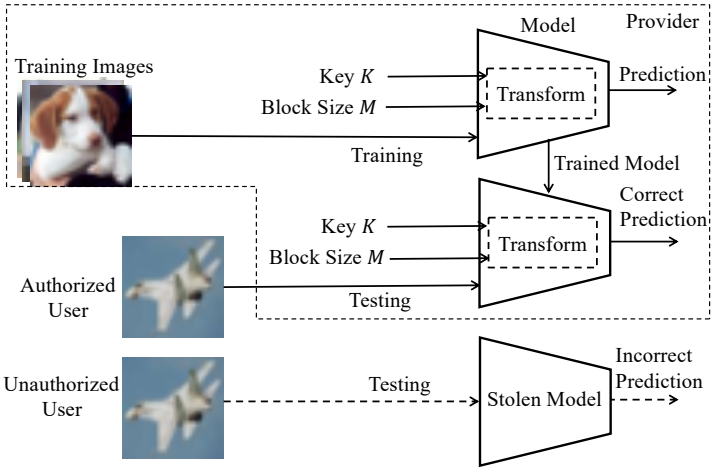

Figure 5.2: Access control framework by feature map transformation with secret key.

5.3 Model Watermarking by Secret Key

For model watermarking, the block-wise transformation, NEG is utilized because NEG provides a higher classification accuracy, and maintains a stronger correlation between adjacent pixel values. An overview of image classification with the watermarking framework is depicted in Fig. 5.3. To embed a watermark into a model, the model is trained with both clean images and images transformed by using secret key K and block size M. Such trained models learn to classify both plain images and transformed ones. This property is exploited to verify the ownership of models.

5.3.1 Watermark Embedding

A pattern caused by the transformation with key K serves as a watermark in the proposed method. To embed the watermark in a model, the model is trained by using both plain and transformed images. Let $\{(\mathbf{x}_i, y_i)\}_i$ be a set of training examples (pairs of images and corresponding labels). Algorithm 5 shows the watermark embedding process during training. Every image $\mathbf{x}_i$ is transformed by NEG ($g(\cdot, \cdot, \cdot)$) with key K to obtain a transformed images $\hat{\mathbf{x}}_i$. Model f is trained by using both $\{(\mathbf{x}_i, y_i)\}_i$ and $\{(\hat{\mathbf{x}}_i, y_i)\}_i$.

5.3.2 Watermark Detection

To detect embedded watermarks, a statistical watermark-extraction method is used during model inference. Let $\{\mathbf{u}_i\}_i$ be a set of test images. Every image $\mathbf{u}_i$ is transformed

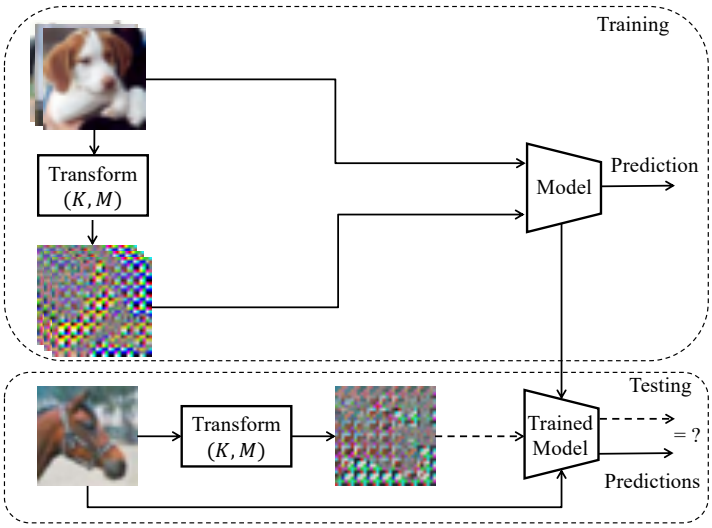

Figure 5.3: Model watermarking framework by block-wise transformation with secret key.

Algorithm 5 Watermark Embedding

Input: $\{(\mathbf{x}_i, y_i)\}_i, K, M$
Output: f

1: **for each** $\mathbf{x}_i$ **do**
2: $\quad \hat{\mathbf{x}}_i \leftarrow g(\mathbf{x}_i, K, M)$
3: **end for each**
4: $f \leftarrow \textsc{Train}\ (\{(\mathbf{x}_i, y_i)\}_i)$
5: $f \leftarrow \textsc{Train}\ (\{(\hat{\mathbf{x}}_i, y_i)\}_i)$

with the same key K and block size M to obtain $\hat{\mathbf{u}}_i$. Notably, $\{\mathbf{u}_i\}_i$ is not a special pre-defined trigger set unlike conventional methods, so it can be a set of any test images within a classifier's distribution. In a typical image-classification scenario, f takes a test image ($\mathbf{u}_i$) and outputs a vector of unnormalized log probabilities (i.e., logits) as $f(\mathbf{u}_i)$. In accordance with this scenario, the class label of $\mathbf{u}_i$ is obtained by the argmax operation denoted as $C(f(\mathbf{u}_i))$.

The watermark is detected by matching between $C(f(\mathbf{u}_i))$ and $C(f(\hat{\mathbf{u}}_i))$, and is defined as

$$\tau = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(C(f(\mathbf{u}_i)) = C(f(\hat{\mathbf{u}}_i))), \tag{5.2}$$

where N is the number of test images, and $\mathbb{1}$(condition) is a value of one if the condition

is satisfied, otherwise a value of zero.

To verify the ownership of a model, an inspector needs to set a threshold th. By using th, the watermark detection process is carried out as in Algorithm 6. If τ is greater than th, the ownership verification is successful.

Algorithm 6 Watermark Detection

Input: $f, \{\mathbf{u}_i\}_i, K, M, th$
Output: Successful or Unsuccessful

```
for each u_i do
    û_i ← g(u_i, K, M)
end for each
Calculate τ as in Eq. (5.2)
if τ > th then
    Successful
else
    Unsuccessful
end if
```

5.4 Threat Models

A threat model includes a set of assumptions such as an attacker's goals, knowledge, and capabilities. An attacker may steal a model to achieve different goals. The attacker's goal is assumed to be able to make use of a stolen model. Therefore, the attacker may estimate a key or fine-tune the model in order to remove the key. Model protection methods in this chapter assume that the attacker obtains a clone of the model and a small subset of the training dataset. There are two common ways of modifying models: fine-tuning and pruning.

Fine-tuning (transfer learning) [149] trains a model on top of pre-trained weights. Since fine-tuning alters the weights of a model, an attacker may use fine-tuning as an attack to overwrite a protected model with the intent of forging a key.

Deep neural network models are often over-parameterized and contain millions of parameters. These giant models cannot be directly deployed on devices with limited resources such as smartphones, digital assistants, and embedded systems. Therefore, pruning techniques such as in [150–153] are used to compress the models by removing unimportant connections or neurons without losing accuracy. In this chapter, parameter pruning is carried out by zeroing out weight values on the basis of the lowest ℓ_1-norm

(i.e., to prune the weights that have the smallest absolute values) as in [28] to the investigate the model's reaction to protection after pruning.

5.5 Experiments and Discussion

5.5.1 Experiment Conditions

Datasets

Two datasets were used to evaluate the model protection methods: CIFAR-10 [119] and ImageNet [94]. CIFAR-10 consists of 60,000 color images (dimension of $32 \times 32 \times 3$) with 10 classes (6000 images for each class) where 50,000 images are for training and 10,000 for testing. For data preprocessing, a batch size of 128 and live augmentation (random cropping with a padding of 4 and random horizontal flip) on a training set. ImageNet comprises 1.28 million color images for training and 50,000 color images for validation. Images were progressively resized during training starting with larger batches of smaller images to smaller batches of larger images. Three phases of training from the DAWNBench top submissions as mentioned in [42] was deployed. Phases 1 and 2 resized images to 160 and 352 pixels, respectively, and phase 3 used the entire image size from the training set. The augmentation methods used in the experiment were random resizing and cropping (sizes of 128, 224, and 288 respectively for each phase) and random horizontal flip.

Networks

Deep residual networks [3] were utilized for evaluating the key-based defense. For CIFAR-10, ResNet18 was trained for 200 epochs with efficient training techniques from the DAWNBench top submissions: cyclic learning rates [145] and mixed-precision training [146]. The parameters of the stochastic gradient descent (SGD) optimizer were a momentum of 0.9, weight decay of 0.0005, and maximum learning rate of 0.2. For ImageNet, ResNet50 with pre-trained weights was used according to the training settings from [42] with the removal of weight decay regularization from batch normalization layers. The network was trained for 15 epochs in total for the ImageNet dataset.

Attack Settings

It is assumed that a subset of dataset D' is available to the attacker. Experiments were carried out on the size of D' (i.e., $|D'| \in \{100, 500, 1000, 5000\}$). The protected models were fine-tuned with D' for 30 epochs whether to remove or overwrite the key.

5.5.2 Results for Model Access Control

Classification Accuracy (Input Transformation)

Table 5.1 summarizes classification performance for protected models by input transformation method for two different datasets (CIFAR-10 and ImageNet). The models were trained by using images transformed by various transformations (both single and combined transformations) with different block sizes (i.e., $M \in \{2, 4, 8, 16\}$). The trained models are named after the shorthand of the respective transformations. For example, the model trained by using images transformed by SHF transformation is denoted as SHF, that by SHF and NEG as SHF + NEG, and so on. The models were evaluated under three conditions: with correct key K, with incorrect key K', and with plain images (without any transformation). The results for incorrect key K' were averaged over 1000 random keys. Overall, NEG achieved the highest accuracy when correct key K was given. However, under the use of bigger block sizes such as 8 or 16, access control performance was weak for the CIFAR-10 dataset (i.e., high accuracy for incorrect key K' and plain images). The models with combined transformations such as SHF + NEG and SHF + FFX decreased the classification accuracy, compared with those with NEG and FFX, when using $M = 8$ or 16. The advantage of a combined transformation is that it can increase the key space, but it slightly reduces the classification accuracy. In short, the access control method by input transformation provides high accuracy for correct key K, and low accuracy for incorrect key K' and plain images.

Key Estimation Attack (Input Transformation)

The Algorithm 4 from Chapter 4 was utilized to estimate a key by observing the accuracy improvement over a batch of images. The resulting estimated key K' was used to evaluate the classification performance of the protected models. Table 5.2 captures the classification accuracy of protected models ($M = 4$) for both correct key K and estimated key K'. The estimated keys were not good enough to provide a reasonable accuracy except for FFX, which replaces a pixel value with a random value by using a password, so the encrypted pixel value contains almost no information. Therefore, the location of the un-encrypted pixel values plays an important role in the model's decision-making process, as the encrypted pixel values are not important. This property helps an attacker to effectively find a good key when performing key estimation attacks (Chapter 4 Algorithm 4). In contrast, in the other two transformations (SHF and NEG), transformed pixel values have some information, and both positions of unencrypted pixels and pixel values are important. Therefore, the indication to search for a good key was difficult for SHF and NEG compared to FFX.

Table 5.1: Accuracy (%) of protected models and baseline model for two datasets. Best results are in **bold**.

	Model	CIFAR-10 Correct (K)	CIFAR-10 Incorrect (K')	CIFAR-10 Plain	ImageNet Correct (K)	ImageNet Incorrect (K')	ImageNet Plain
$M = 2$	SHF	94.76	36.36	31.43	73.00	46.57	40.35
	NEG	**95.32**	18.44	13.91	**73.04**	6.53	0.98
	FFX	93.80	15.69	38.84	72.43	**0.12**	0.23
	SHF + NEG	94.50	20.65	**11.79**	72.90	4.87	1.12
	SHF + FFX	93.02	15.30	19.60	72.30	0.47	0.19
	SHF + NEG + FFX	92.82	**14.03**	18.69	71.96	0.16	**0.18**
$M = 4$	SHF	92.58	20.23	27.77	72.41	13.06	32.98
	NEG	**93.41**	12.67	**12.17**	**72.63**	0.68	0.36
	FFX	92.29	18.38	37.06	72.17	0.15	**0.15**
	SHF + NEG	92.37	12.11	12.35	72.15	0.21	0.25
	SHF + FFX	90.71	12.31	20.75	71.96	0.14	0.17
	SHF + NEG + FFX	90.50	**10.60**	13.10	71.68	**0.12**	0.16
$M = 8$	SHF	86.40	17.00	14.42	70.85	1.25	11.74
	NEG	91.54	71.35	79.51	**71.83**	0.26	0.12
	FFX	**92.00**	47.07	37.25	71.46	0.30	**0.09**
	SHF + NEG	86.47	12.16	14.75	71.14	0.19	0.86
	SHF + FFX	86.01	11.81	15.20	70.77	0.11	0.14
	SHF + NEG + FFX	85.49	**10.23**	**10.31**	70.18	**0.10**	0.11
$M = 16$	SHF	77.24	10.57	13.36	67.03	0.23	4.22
	NEG	**92.68**	88.27	89.00	**70.19**	0.97	5.52
	FFX	91.38	72.91	29.35	69.24	2.07	0.14
	SHF + NEG	77.52	10.66	11.70	67.50	0.11	0.18
	SHF + FFX	76.28	10.20	12.79	63.75	0.16	0.13
	SHF + NEG + FFX	75.78	**10.00**	**9.92**	63.43	**0.09**	**0.12**
	Baseline	95.45 (Not protected)			73.70 (Not protected)		

Table 5.2: Accuracy (%) of protected models ($M = 4$) under key estimation attack

Model	Correct (K)	Estimated (K')
SHF	92.58	25.66
NEG	93.41	37.44
FFX	92.29	80.97
SHF + NEG	92.37	14.53
SHF + FFX	90.71	15.04
SHF + NEG + FFX	90.50	11.00

Fine-tuning Attack (Input Transformation)

As described in Section 5.5.1, fine-tuning attacks were performed to overwrite the correct key with a subset of the dataset, D'. Table 5.3 shows the results of fine-tuning attacks for the CIFAR-10 dataset. Although the accuracy improved with respect to the size of D', it was still lower than the performance of the correct key K. Therefore, the results show that the compromised models were not as good as the original models. As a result, the attacker is not able to use the model to full capacity.

Table 5.3: Accuracy (%) of protected models by input transformation under fine-tuning attacks

Model	Correct (K)	$\|D'\| = 100$	$\|D'\| = 500$	$\|D'\| = 1000$
SHF	92.58	12.69	38.33	46.73
NEG	93.41	10.57	37.25	47.41
FFX	92.29	10.15	32.30	40.52
SHF + NEG	92.37	14.03	37.50	46.36
SHF + FFX	90.71	12.54	46.28	55.11
SHF + NEG + FFX	90.50	11.15	39.59	48.13

Comparison with State-of-the-art Methods (Input Transformation)

The comparison was made for the model with NEG ($M = 4$) and the state-of-the-art passport protected model, Scheme $\mathcal{V}_1$ [32] in terms of classification accuracy with/without correct key/passports, overheads, network modification and key management for the CIFAR-10 dataset. Scheme $\mathcal{V}_1$ [32] was not trained and tested using the same settings as NEG model because the network in $\mathcal{V}_1$ was modified with passport layers and the

hyperparameters were based on the modified network. In contrast, the NEG model used a standard ResNet18 and was trained with cyclic learning rates [145] and mixed precision training [146].

Table 5.4 summarizes the comparison. In terms of accuracy when the correct key/passport was given, the accuracy of $\mathcal{V}_1$ was slightly higher than that of NEG at 1.21 %. However, it was confirmed that if block size $M = 2$ was used, NEG achieved higher accuracy than $\mathcal{V}_1$ (i.e., 95.32 %). When estimated incorrect key was given, the accuracy of NEG significantly dropped. In contrast, when reverse-engineered (i.e., estimated) passports were used, the accuracy of $\mathcal{V}_1$ was high (70 %).

In terms of overhead, $\mathcal{V}_1$ modifies a network with additional passport layers; therefore, it introduces a training and inference overhead for both datasets. The overheads in [32] are based on the relative recorded time taken as mentioned in the paper by the original authors. In contrast, the model with NEG does not have any noticeable overhead, and there is no modification in the network. Moreover, the block-wise transformation in the proposed model protection can be efficiently implemented with vectorized operations; therefore, pre-processing with the block-wise transformation does not cause any noticeable overheads in both training and testing.

From a key management perspective, $\mathcal{V}_1$ requires a trained model to generate passports, and the model with NEG does not need any model to generate keys. Therefore, the key management of NEG is simple and straightforward.

Table 5.4: Comparison of protected model NEG by input transformation and state-of-the-art passport-protected model

Model	Correct K / Passports	Estimated K' / Passports	Training Overhead	Inference Overhead	Network Modification	Key Management
NEG ($M = 4$)	93.41	37.44	Negligible	Negligible	No	Easy
Scheme $\mathcal{V}_1$ [32]	94.62	70.00	15–30% [32]	10 % [32]	Yes	Difficult

Classification Accuracy (Feature Map Transformation)

Different models were trained by applying SHF to different feature maps in ResNet-18. All models in the experiments used a block size M of 2. The trained models models were evaluated under three conditions for transformation: with correct key K, with incorrect key K', and without applying the transformation.

Table 3.1 summarizes the results of protected models by feature transformation comparing with the ones by input transformation, where the classification accuracy for incorrect key K' was averaged over 100 random keys. The key space for all models is also presented in Table 3.1. The model trained by transforming the feature map of the initial convolution is indicated as "Initial Conv", that of the first group of residual blocks as "Layer 1", the second as "Layer 2", and so on. Experiment results show that the accuracy of the protected models by feature transformation is almost the same as that of non-protected models (i.e., baseline). Moreover, feature transformation significantly increased the key space, and maintained a higher classification accuracy for correct key K, a lower classification accuracy for incorrect key K' and without transformation. Therefore, the models by feature transformation outperformed the ones by input transformation.

Table 5.5: Accuracy (%) and key space of protected models by feature transformation comparing with ones by input transformation and baseline model

	Model	Key Space	Accuracy (K)	Accuracy (K')	Accuracy (without transformation)
Feature†	Initial Conv ($M = 2$)	256!	94.83	10.74	9.94
	Layer 1 ($M = 2$)	256!	95.38	9.64	10.08
	Layer 2 ($M = 2$)	512!	95.16	10.64	6.55
	Layer 3 ($M = 2$)	1024!	95.39	10.16	10.22
	Layer 4 ($M = 2$)	2048!	95.21	11.36	1.30
Input*	SHF ($M = 2$)	12!	94.76	36.55	31.43
	NEG ($M = 2$)	2^{12}	95.32	19.40	13.91
	FFX ($M = 2$)	2^{12}	93.80	14.67	38.84
	SHF ($M = 4$)	48!	92.58	20.15	27.77
	NEG ($M = 4$)	2^{48}	93.41	12.50	12.17
	FFX ($M = 4$)	2^{48}	92.29	18.45	37.06
	Baseline	95.45 (Not protected)			

Input* = Input Transformation and Feature† = Feature Transformation

Table 5.6: Accuracy (%) of protected models by feature transformation under key estimation attack comparing with previous protected models

Model	Correct (K)	Estimated (K')
Initial Conv ($M = 2$)	94.83	20.17
Layer 1 ($M = 2$)	95.38	16.35
Layer 2 ($M = 2$)	95.16	23.58
Layer 3 ($M = 2$)	95.39	50.13
Layer 4 ($M = 2$)	95.21	89.30
SHF* ($M = 4$)	92.58	25.66
NEG* ($M = 4$)	93.41	37.44
FFX* ($M = 4$)	92.29	80.97

* indicates the model by the input transformation method.

Key Estimation Attack (Feature Map Transformation)

Similar to the access control method by input transformation, the Algorithm 4 from Chapter 4 was also deployed to estimate a key by observing the accuracy improvement over a batch of images. The resulting estimated key K' was used to evaluate the classification performance of the protected models.

Table 5.6 captures the classification performance of the protected models comparing with the ones by input transformation under the key estimation attack. Note that the previous models with $M = 4$ were used for the comparison because the key space of the previous models for $M = 2$ is relatively small. From the table, the accuracy of estimated key K' for Layer 3 and 4 are 50.13 % and 89.30 % respectively. Interestingly, although the key space of Layer 3 and 4 was larger, key estimation attacks found a good key to provide a reasonable accuracy. However, the estimated keys were not good enough for the other models. Therefore, the protected models by feature transformation provided better resistance against key estimation attacks except the models, Layer 3 and 4.

Fine-tuning Attack (Feature Map Transformation)

For feature transformation method, there are many feature maps in the network, and the attacker may not know the location of the transformed feature map. Therefore, it is more natural to carry out fine-tuning attacks to remove the key from the model, rather than overwriting the key. The models were fine-tuned according to the settings in Section 5.5.1 with a subset of the dataset, D'. Table 5.7 shows the results of fine-tuning attacks for the protected models comparing with the ones by input transformation.

Table 5.7: Accuracy (%) of protected models under fine-tuning attacks comparing with models by input transformation

Model	Original	$\|D'\| = 100$	$\|D'\| = 500$	$\|D'\| = 1000$
Initial†	94.83	18.47	55.38	69.42
Layer 1	95.38	22.37	66.90	78.54
Layer 2	95.16	21.75	66.38	74.94
Layer 3	95.39	20.84	74.59	80.99
Layer 4	95.21	87.43	73.28	94.63
SHF*	94.76	33.89	70.69	78.01
NEG*	95.32	16.84	58.17	75.00
FFX*	93.80	44.32	75.45	80.86

Initial† stands for "Initial Convolution".

* indicates the model by the input transformation method.

Although the accuracy improved with respect to the size of D', it was still lower than the performance of the correct key K except for Layer 3 and 4. Overall, the model "Initial Conv" provided better robustness against fine-tuning attacks than any other models.

Comparison with State-of-the-art Methods (Feature Map Transformation)

Since underlying mechanisms of the access control method by [27], which uses a perturbation network, and the method by feature transformation are different, it is difficult to directly compare them. To make a high-level comparison, the anti-piracy method [27] was implemented in the same training settings as in Section 5.5.1 for the CIFAR-10 dataset. Then, the models were compared in terms of authorized accuracy (i.e., with correct transformation/perturbation), unauthorized accuracy (i.e., without transformation/perturbation), and the core method used in the two models (Table 5.8). The main difference is that the method by feature transformation uses a block-wise transformation with a secret key and the anti-piracy method [27] utilizes a perturbation network. In terms of classification performance, the method by feature transformation achieves a higher authorized accuracy, which is close to baseline accuracy, and a lower unauthorized accuracy than that of the anti-piracy method [27].

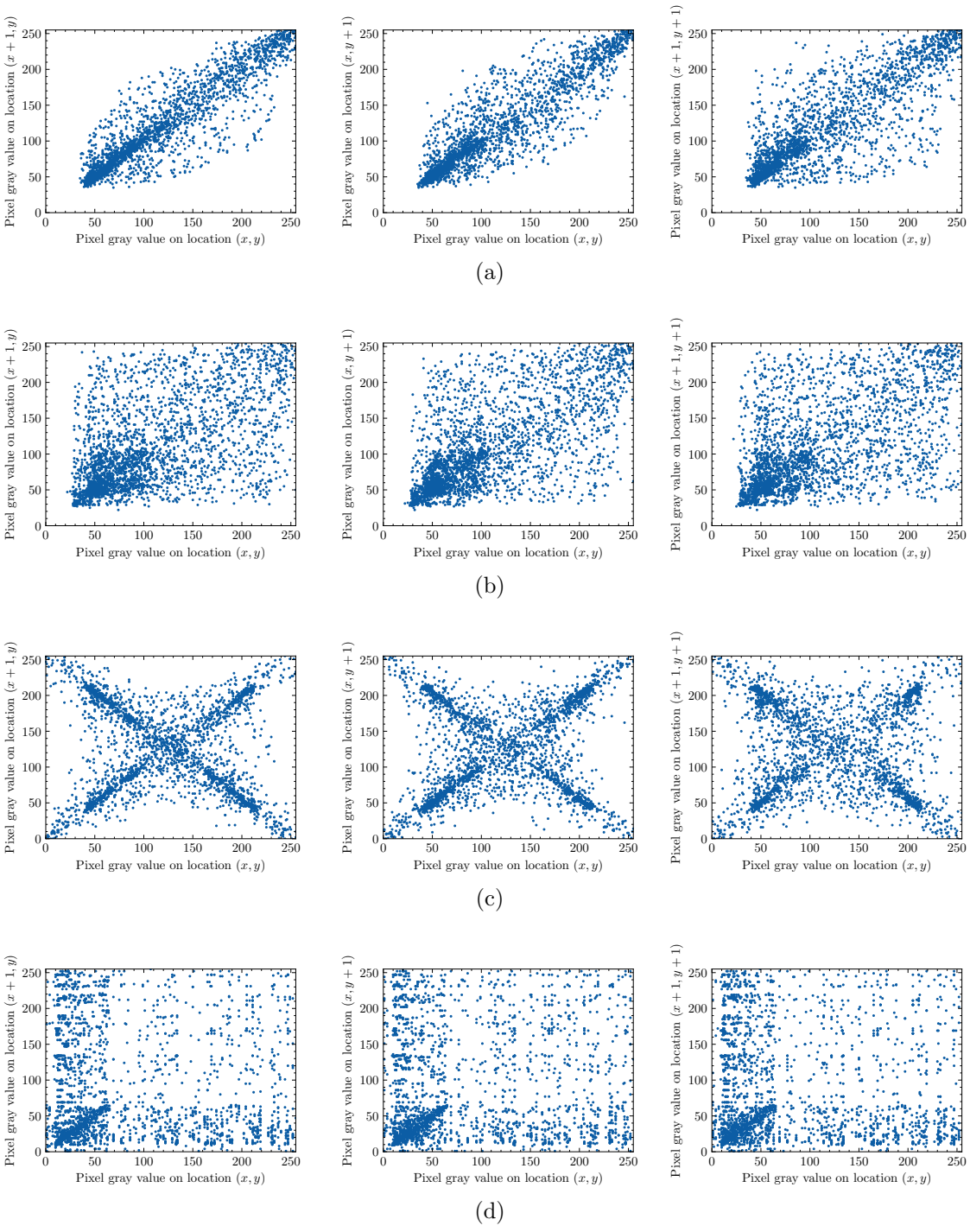

Figure 5.4: Horizontal, vertical and diagonal correlation test results for (a) plain image and images transformed by (b) SHF, (c) NEG and (d) FFX ($M = 4$).

Table 5.8: Comparison of proposed protected model and state-of-the-art anti-piracy model [27]

Model	Authorized Accuracy	Unauthorized Accuracy	Method
Initial Conv	94.83	9.94	Block-Wise Transformation
Anti-piracy [27]	92.89	14.21	Perturbation Network
Baseline	95.45	95.45	Non-protected

5.5.3 Discussion for Model Access Control

Image Correlation Analysis (Input Transformation)

To gain insights into the classification performance of block-wise transformed images, adjacent pixel correlation tests were carried out on a test image, "dog," in the horizontal, vertical, and diagonal directions as shown in Fig. 5.4. From the figure, all transformations were confirmed to maintain some correlation between pixels differently. The pixel correlation distribution of the image transformed by SHF was similar to that of the plain image. For NEG and FFX, the pixel correlation distributions were different from that of the plain image. In addition, the pixel correlation of FFX was slightly weak due to the use of FFX, compared with the other ones, so this property might have caused a lower accuracy than those of the other transformations in Table 5.1. Accordingly, there is some correlation between pixels in the transformed images, so block-wise transformations can achieve a high classification accuracy.

Key Sensitivity Test (Input Transformation)

Key sensitivity tests were carried out for models with $M = 4$ and 8 on the CIFAR-10 dataset. The key sensitivity is defined as the difference in accuracy between the correct key and the modified key (i.e., the correct key with a small change), which is given by

$$\text{Key Sensitivity} = \text{ACC} - \text{ACC}', \tag{5.3}$$

where ACC is the classification accuracy with a correct key, and ACC$'$ is that with a key that has a small change from the correct key.

To make a small change, two random elements were swapped in the correct key for SHF, and one element in the correct key was flipped for NEG and FFX (i.e., "0" to "1" and "1" to "0"). Table 5.9 shows the result of the key sensitivity tests, where the values in the table were averaged over $c \times M \times M$ times to cover changes in the

different positions of the keys. From the results, it is observed that low key sensitivity values reflected a higher accuracy for the incorrect keys, and high ones corresponded to a lower accuracy for the incorrect ones, as shown in Table 5.1. The key sensitivity in the table gives some insights into the difference among transformations.

Table 5.9: Key sensitivity of various transformations with $M = 4$ and 8

Model	$M = 4$	$M = 8$
SHF	1.79	0.31
NEG	9.68	0.11
FFX	5.17	0.29
SHF + NEG	3.80	0.59
SHF + FFX	5.15	0.67
SHF + NEG + FFX	5.65	0.72

Key Improvement (Input Transformation)

When $M = 2$, the key space for the block-wise transformations is relatively small, so brute force attacks are possible. To improve the key space, there are two ways: (1) to use a larger block size such as 8×8, 8×4, etc. and (2) to use a combined transformation such as SHF + NEG or SHF + FFX. Note that a value of M affects not only the key space but also the classification accuracy and key sensitivity. Accordingly, users are requested to find a good trade-off among them.

Selection of Transformations (Input Transformation)

Classification accuracy and model protection performance depend on the type of transformation and block size M. We recommend the following selection of transformations accordingly. When a higher classification accuracy is required, NEG or a combined transformation such as SHF + NEG or SHF + FFX with a small block size M is recommended. When higher protection performance is preferred, a larger M with SHF or a combined transformation is suitable.

Limitations (Input Transformation)

The access control method by input transformation focuses on image classification tasks because the three encryption methods are designed for image classification tasks. When these encryption methods are applied to other tasks such as image segmentation and

image retrieval, the performance may drop compared with that of using plain images. Therefore, the access control method in this chapter is limited to image classification tasks, and novel image transformations are expected to be designed for applying other tasks.

Difference between Input and Feature Transformation

Although both feature transformation and input transformation utilize a block-wise transformation with a secret key, only SHF is applicable to the feature transformation because SHF does not change pixel values. The access control method by feature transformation outperformed the one by input transformation in terms of classification accuracy and key space. However, the models trained by transforming feature maps of later layers in the network such as Layer 3 and 4 were vulnerable towards key estimation and fine-tuning attacks. Therefore, transforming the feature maps of earlier layers in the network is suitable for the stronger access control performance.

Both access control methods (input transformation and feature transformation) are limited to image classification tasks. Further research is required to apply these methods in other tasks such as semantic segmentation and image retrieval.

5.5.4 Results for Model Watermarking

Classification Accuracy

The models were trained with the watermarking framework under five block sizes (i.e., $M \in \{2, 4, 8, 16, 32\}$). The trained models were evaluated under three conditions: using plain images (plain), using transformed images with correct key K, and using transformed images with incorrect key K'. The matching rates, τ and τ' were calculated to verify the ownership for correct key K and incorrect key K' respectively.

Table 5.10 summarizes the results obtained under the above conditions. The models with a small block size such as $M = 2$ and 4 performed better in detecting watermarks than the ones with $M = 8$, 16, and 32. The models trained by transformed images with bigger block sizes are sensitive towards the key, thus they may cause ownership confusion. The baseline model did not have a watermark and therefore, the τ value was low. Since models with $M = 2$ and 4 maintained a high classification accuracy when correct key K was used, and the accuracy severely dropped when incorrect key K' was given, the models with $M = 2$ and 4 were further evaluated against fine-tuning and pruning attacks.

Table 5.10: Classification Accuracy (%) and τ (%) of watermarked models and baseline model.

Model	Plain	Correct (K)	τ	Incorrect (K')	τ
$M = 2$	92.74	93.43	95.87	10.53	10.26
$M = 4$	92.99	92.24	94.20	15.55	15.75
$M = 8$	93.52	87.25	89.18	73.40	75.00
$M = 16$	93.71	89.26	90.50	82.21	83.87
$M = 32$	93.88	89.00	91.08	85.51	87.78
Baseline	95.45	11.34	11.43	12.02	12.12

Table 5.11: Classification Accuracy (%) and τ (%) of watermarked models under fine-tuning attacks.

	① $\lvert D' \rvert = 100$			② $\lvert D' \rvert = 500$			③ $\lvert D' \rvert = 5000$		
Model	Accuracy	τ	τ'	Accuracy	τ	τ'	Accuracy	τ	τ'
$M = 2$	89.44	93.64	13.26	83.59	88.84	31.93	86.37	87.11	84.26
$M = 4$	91.79	93.46	16.50	87.50	89.90	23.14	82.62	71.24	69.15

Fine-tuning Attack

In order to embed a new watermark, fine-tuning attacks were carried out according to the settings in Section 5.5.1 with a subset of the dataset, D'. Table 5.11 captures the results of fine-tuning attacks; model accuracy after fine-tuning, matching rate τ for the original correct key K, and matching rate τ' for new key K'. In any of the cases, fine-tuning attacks impaired the model accuracy, and τ was greater than τ'. Therefore, empirical results show that the original watermark was not overwritten by the new watermark in the model watermarking framework by block-wise transformation, suggesting robustness against piracy attacks.

Pruning Attack

As described in Section 5.4, the pruning attack was carried out with the intent of removing the watermark from the model. In experiments, classification accuracy and matching rate τ under different pruning rates were observed. Figures 5.5 and 5.6 show graphs of accuracy and τ against pruning rates. From the figure, after pruning more than 60 %, both the accuracy and τ dropped. Therefore, the results suggest that

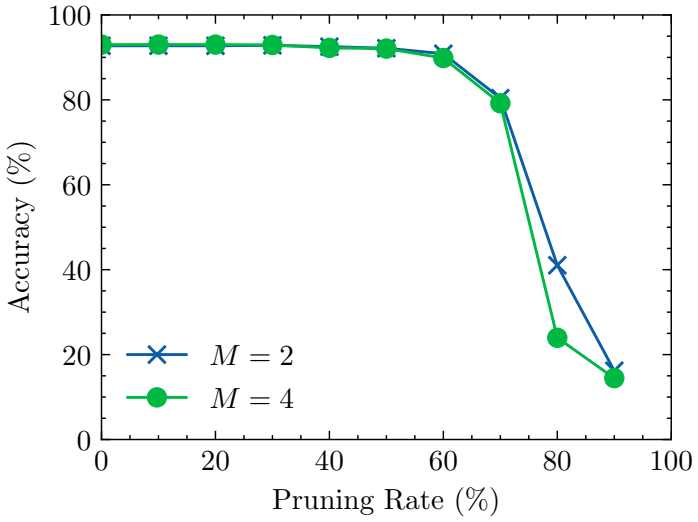

Figure 5.5: Classification accuracy under pruning attacks.

the watermark cannot be removed without impairing the model accuracy by pruning attacks.

Comparison with State-of-the-art Methods

Table 5.12 provides a high-level overview of state-of-the-art model watermarking methods in black-box settings. Embedding and verification methods vary from method to method. Most of the existing methods [31,32,34,121] are not robust to piracy attacks as described in [36]. In contrast, the watermark patterns used in the method by block-wise transformation and Li et al.'s method [36] are directly dependent on model's accuracy. Therefore, piracy attacks will deteriorate model's performance, and the original watermark detection is still stronger than the pirated one. Note that the work in [36] was evaluated only on a small convolutional network, and the method by block-wise transformation was tested on a residual network with 18 layers (ResNet18). Therefore, the effectiveness of the method by block-wise transformation was confirmed for a modern convolutional neural network architecture.

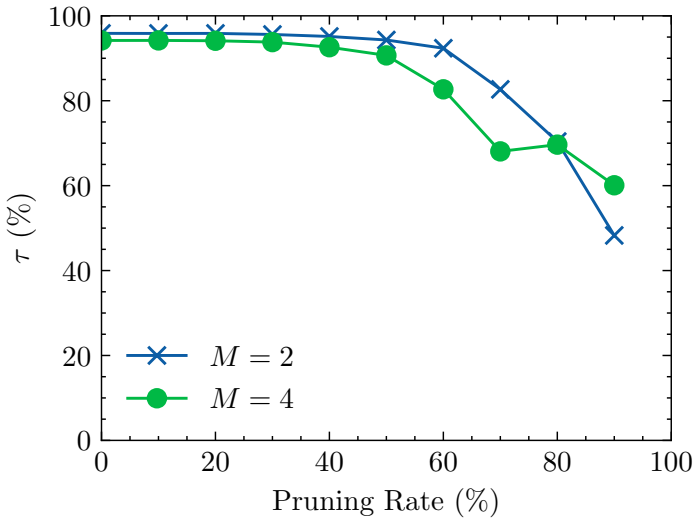

Figure 5.6: Matching rate τ under pruning attacks

Table 5.12: High-level comparison with state-of-the-art black-box model watermarking methods

Model	Embedding Method	Verification Method	Piracy Resistance
Adi et al. [121]	Backdoor	Trigger Set	No
Merrer et al. [34]	Adversarial Examples	Trigger Set	No
Zhang et al. [31]	Watermarked Examples	Trigger Set	No
Fan et al. [32]	Passport Layers + Trigger Set	Passports + Trigger Set	No
Li et al. [36]†	Null Embedding + Trigger Set	Watermark Accuracy + Trigger Set	Yes
Block-wise‡	Learnable Image Transformation	Watermark Detection Accuracy	Yes

† Evaluated on a small convolutional neural network. ‡ Evaluated on ResNet18.

5.5.5 Discussion for Model Watermarking

Unlike model access control methods by input transformation and feature transformation, NEG is applied for model watermarking because NEG maintains a high classification accuracy and strong correlation between adjacent pixel values. In the watermarking framework by block-wise transformation, the original watermark in a model cannot be removed, and adding a new watermark to the model will decrease the model's accuracy. The framework uses a secret key to verify ownership. Therefore, a special trigger set with pre-defined labels for detecting a watermark is not required in the watermarking framework by the block-wise transformation.

The block size plays an important role in the watermarking method by block-wise transformation. Only small block sizes such as 2 and 4 are applicable for the CIFAR-10 dataset. In addition, the watermarking method is also limited to classification models.

5.6 Summary

This chapter presents three model protection methods: two model access control methods and one model watermarking method. All methods utilize block-wise transformations with secret keys. The access control method by input transformation employs different transformations (SHF, NEG, and FFX) in both single and a combined manner. The one with the feature map transformation uses only SHF, and the watermarking framework adopts NEG. Experiments with different possible attacks such as key estimation, fine-tuning, and pruning were carried out to verify the effectiveness of the model protection methods. Results show that the access control method with feature map transformation is superior to that with input transformation in terms of classification accuracy and key space. In addition, the watermarking framework with the block-wise transformation is piracy-resistant and maintains a high classification accuracy.

Chapter 6

Conclusion

In the previous chapters, two adversarial defenses against adversarial examples and one model protection solution that covers both model access control, and model watermarking have been presented. In this chapter, the results of these three approaches are summarized. The chapter also describes future work and concluding remarks.

6.1 Summary of Results

6.1.1 Adversarial Defense by Quantization

This defense considers a restricted scenario where only 1–bit images are exposed to attackers. Due to this condition, adversarial noise generated in 1–bit images can be completely removed by simple linear quantization. To improve the classification accuracy, dithering was also used in this defense framework. The linear quantizer guarantees that original 1–bit test images will be restored regardless of adversarial noise distance, and, therefore, this defense maintains identical accuracy whether or not the model is under attack. The results show that this defense achieves comparable accuracy, 85.28 % on the CIFAR-10 and 94.99 % on the Oxford-IIIT Pet datasets against three state-of-the-art adversaries with even a previously untested maximum adversarial distance of 64.

6.1.2 Key-Based Adversarial Defense

This defense utilizes a block-wise transformation with a secret key as a pre-processing technique for both training and testing a model. Three different transformations: pixel shuffling, negative/positive transformation, and format-preserving Feistel-based encryption were introduced in this defense framework. The results show that the key-based

defense achieved more than 90 % accuracy for both clean images and adversarial examples. Under PGD attack with different perturbation budgets, by having information advantage as a secret key, the key-based defense outperformed the state-of-the art adversarial defenses with the CIFAR-10 and ImageNet datasets.

6.1.3 Model Protection by Secret Key

Access Control by Input Transformation

The block-wise transformation from the adversarial defense was adopted to protect the functionality of a model from unauthorized access. The performance accuracy of a protected model was close to that of a non-protected model when the key was correct, and it dropped drastically when an incorrect key was given, suggesting that a protected model is not usable when the model is stolen. This access control method is also applicable to large datasets like the ImageNet dataset, which has never been tested by previous model-protection methods. Moreover, the access control method by input transformation does not introduce any overhead in both training and inference time. It is also robust against fine-tuning attacks in which the adversary has a small subset of a training dataset to adapt a new forged key and key estimation attacks.

Access Control by Feature Map Transformation

Instead of applying a transformation to the input images, this access control method directly applies a block-wise transformation, pixel shuffling with a secret key to feature maps in the network. As a result, this access control method not only improves the classification accuracy but also increases the key space substantially. The performance accuracy of a protected model by this access control method was close to that of a non-protected model when the key was correct, and it dropped drastically when an incorrect key was given, suggesting the model cannot be used to its full capacity for unauthorized users. Experiments results show that the access control method by feature map transformation outperformed the previous access control methods in terms of classification accuracy and robustness against key estimation attacks and fine-tuning attacks.

Model Watermarking

The block-wise transformation from the adversarial defense was also extended to model watermarking scenarios. This method embeds a watermark pattern in a model by using learnable transformed images and allows us to remotely verify the ownership of the model. This was achieved by training a model with both plain and transformed

images. The results of experiments show that such a watermarking method maintains a high classification accuracy, and watermarks in this method could not be overwritten by piracy attacks, i.e., fine-tuning with a subset of dataset. In addition, this watermarking method was also robust against pruning attacks when parameters were pruned up to 60 %.

6.2 Future Work

As future work, there are room for improvement in the key-based defense, and new ideas shall be investigated and explored.

- The key-based defense in its current form does not have an information advantage over black-box attacks. Although a simple voting ensemble was proposed to overcome this limitation, different ensembles with different transformations and block sizes will be explored.
- In the thesis, only analysis in pixel space was done, analysis in feature space should be further explored. For example, Fréchet inception distance (FID) can be used to compare differently transformed images and plain ones.
- As this thesis has experimented to transform input images and feature maps, one of the future works is label encoding with a key.
- Convolutional neural networks are sensitive to spatial features and therefore, block scrambling drops accuracy significantly. To be able to use block scrambling, visual transformers [154] will be pursued.

6.3 Concluding Remarks

Regarding the solutions provided to defend against adversarial examples and unauthorized access in this thesis, a few extra key points are drawn as follows.

- Quantization may not be ideal for real-world applications because 8–bit images are most widely used, and the attacking scenario is limited.
- The key-based defense is effective for having information advantage as a secret key and maintains a high classification accuracy. It opens a new direction to control a model with a key.

- The results presented in this thesis are based on residual networks and a few datasets. To further confirm the performance of the key-based defense, more networks and datasets need to be tested. However, experiments in this thesis showed that both the key-based defense and model access control method were scalable to the ImageNet level.
- This thesis shows that adversarial defense and model protection are closely related. Therefore, the concept of a secret key is useful for developing unified solutions in securing deep learning models.
- With enough data and compute, fine-tuning attacks can be successful. Therefore, this thesis considered the cost of the attack should be lower than training a new model and assumed the attacker has only a small subset of the dataset.
- All in all, the solutions in this thesis can be potentially applied to real-world applications and open new ideas for future research.

References

[1] Y. Vorobeychik and M. Kantarcioglu, "Adversarial machine learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 12, no. 3, pp. 1–169, 2018.

[2] A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar, *Adversarial machine learning*. Cambridge University Press, 2018.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[5] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*, 2013, pp. 6645–6649.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[7] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[8] A. Azulay and Y. Weiss, "Why do deep convolutional networks generalize so poorly to small image transformations?" *Journal of Machine Learning Research*, vol. 20, no. 184, pp. 1–25, 2019.

[9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014.

[10] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2013, pp. 387–402.

[11] K. Warr, *Strengthening deep neural networks: making AI less susceptible to adversarial trickery*. O'Reilly Media, 2019.

[12] "Tensorflow hub is a repository of trained machine learning models," https://www.tensorflow.org/hub, 2021.

[13] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1625–1634.

[14] K. Xu, G. Zhang, S. Liu, Q. Fan, M. Sun, H. Chen, P.-Y. Chen, Y. Wang, and X. Lin, "Adversarial t-shirt! evading person detectors in a physical world," in *European Conference on Computer Vision*. Springer, 2020, pp. 665–681.

[15] S. Komkov and A. Petiushko, "Advhat: Real-world adversarial attack on arcface face id system," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 819–826.

[16] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1528–1540.

[17] N. Guetta, A. Shabtai, I. Singh, S. Momiyama, and Y. Elovici, "Dodging attack using carefully crafted natural makeup," *arXiv:2109.06467*, 2021. [Online]. Available: https://arxiv.org/abs/2109.06467

[18] A. Athalye, N. Carlini, and D. A. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, 2018, pp. 274–283.

[19] F. Tramèr, N. Carlini, W. Brendel, and A. Madry, "On adaptive attacks to adversarial example defenses," *arXiv:2002.08347*, 2020. [Online]. Available: https://arxiv.org/abs/2002.08347

[20] Y. Sharma and P.-Y. Chen, "Attacking the madry defense model with l_1-based adversarial examples," *arXiv:1710.10733*, 2017. [Online]. Available: https://arxiv.org/abs/1710.10733

[21] O. Poursaeed, I. Katsman, B. Gao, and S. Belongie, "Generative adversarial perturbations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4422–4431.

[22] H.-T. D. Liu, M. Tao, C.-L. Li, D. Nowrouzezahrai, and A. Jacobson, "Beyond pixel norm-balls: Parametric adversaries using an analytically differentiable renderer," in *International Conference on Learning Representations*, 2019.

[23] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.

[24] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 675–678.

[25] "Azure ai gallery," https://gallery.azure.ai/, 2021.

[26] Y. Huang, H. Hu, and C. Chen, "Robustness of on-device models: Adversarial attack to deep learning models on android apps," *arXiv:2101.04401*, 2021. [Online]. Available: https://arxiv.org/abs/2101.04401

[27] M. Chen and M. Wu, "Protect your deep neural networks from piracy," in *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2018, pp. 1–7.

[28] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding watermarks into deep neural networks," in *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, 2017, pp. 269–277.

[29] H. Chen, B. D. Rouhani, and F. Koushanfar, "Deepmarks: A digital fingerprinting framework for deep neural networks," *arXiv:1804.03648*, 2018. [Online]. Available: http://arxiv.org/abs/1804.03648

[30] B. D. Rouhani, H. Chen, and F. Koushanfar, "Deepsigns: A generic watermarking framework for IP protection of deep learning models," *arXiv:1804.00750*, 2018. [Online]. Available: http://arxiv.org/abs/1804.00750

[31] J. Zhang, Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, and I. Molloy, "Protecting intellectual property of deep neural networks with watermarking," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018, pp. 159–172.

[32] L. Fan, K. Ng, and C. S. Chan, "Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks," in *Advances in Neural Information Processing Systems*, 2019, pp. 4716–4725.

[33] S. Sakazawa, E. Myodo, K. Tasaka, and H. Yanagihara, "Visual decoding of hidden watermark in trained deep neural network," in *2nd IEEE Conference on Multimedia Information Processing and Retrieval*, 2019, pp. 371–374.

[34] E. L. Merrer, P. Pérez, and G. Trédan, "Adversarial frontier stitching for remote neural network watermarking," *Neural Computing and Applications*, vol. 32, no. 13, pp. 9233–9244, 2020.

[35] T. Wang and F. Kerschbaum, "Attacks on digital watermarks for deep neural networks," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 2622–2626.

[36] H. Li, E. Wenger, B. Y. Zhao, and H. Zheng, "Piracy resistant watermarks for deep neural networks," *arXiv:1910.01226*, 2019. [Online]. Available: https://arxiv.org/abs/1910.01226

[37] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1322–1333.

[38] M. AprilPyone, Y. Kinoshita, and H. Kiya, "Adversarial robustness by one bit double quantization for visual classification," *IEEE Access*, vol. 7, pp. 177 932–177 943, 2019.

[39] E. Raff, J. Sylvester, S. Forsyth, and M. McLean, "Barrage of random transforms for adversarially robust defense," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6528–6537.

[40] M. AprilPyone and H. Kiya, "Block-wise image transformation with secret key for adversarially robust defense," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2709–2723, 2021.

[41] O. Taran, S. Rezaeifar, and S. Voloshynovskiy, "Bridging machine learning and cryptography in defence against adversarial attacks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.

[42] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," in *International Conference on Learning Representations*, 2020.

[43] H. Zhang and J. Wang, "Defense against adversarial attacks using feature scattering-based adversarial training," in *Advances in Neural Information Processing Systems*, 2019, pp. 1831–1841.

[44] M. AprilPyone and H. Kiya, "A protection method of trained CNN model with a secret key from unauthorized access," *APSIPA Transactions on Signal and Information Processing*, vol. 10, p. e10, 2021.

[45] ——, "A protection method of trained cnn model using feature maps transformed with secret key from unauthorized access," in *2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2021, accepted.

[46] ——, "Piracy-resistant DNN watermarking by block-wise image transformation with secret key," in *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, 2021, pp. 159–164.

[47] M. Goldblum, D. Tsipras, C. Xie, X. Chen, A. Schwarzschild, D. Song, A. Madry, B. Li, and T. Goldstein, "Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses," *arXiv:2012.10544*, 2020. [Online]. Available: https://arxiv.org/abs/2012.10544

[48] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 19–35.

[49] J. Wakefield, "Microsoft chatbot is taught to swear on Twitter," https://www.bbc.com/news/technology-35890188, 2016.

[50] R. S. S. Kumar, M. Nyström, J. Lambert, A. Marshall, M. Goertzel, A. Comissoneru, M. Swann, and S. Xia, "Adversarial machine learning-industry perspectives," in *2020 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2020, pp. 69–75.

[51] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 601–618.

[52] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18.

[53] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015.

[54] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[55] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, pp. 317–331, 2018.

[56] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. J. Goodfellow, A. Madry, and A. Kurakin, "On evaluating adversarial robustness," *arXiv:1902.06705*, 2019. [Online]. Available: http://arxiv.org/abs/1902.06705

[57] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "Exploring the landscape of spatial robustness," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1802–1811.

[58] Y. Song, R. Shu, N. Kushman, and S. Ermon, "Constructing unrestricted adversarial examples with generative models," *arXiv:1805.07894*, 2018. [Online]. Available: https://arxiv.org/abs/1805.07894

[59] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song, "Spatially transformed adversarial examples," *arXiv:1801.02612*, 2018. [Online]. Available: https://arxiv.org/abs/1801.02612

[60] L. Engstrom, D. Tsipras, L. Schmidt, and A. Madry, "A rotation and a translation suffice: Fooling CNNs with simple transformations," *arXiv:1712.02779*, 2017. [Online]. Available: http://arxiv.org/abs/1712.02779

[61] J. Gilmer, R. P. Adams, I. J. Goodfellow, D. Andersen, and G. E. Dahl, "Motivating the rules of the game for adversarial example research," *arXiv:1807.06732*, 2018. [Online]. Available: http://arxiv.org/abs/1807.06732

[62] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *International Conference on Machine Learning*, 2018, pp. 284–293.

[63] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *International Conference on Learning Representations*, 2017.

[64] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017, pp. 506–519.

[65] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *International Conference on Learning Representations*, 2017.

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

[67] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy*, 2017, pp. 39–57.

[68] P. Chen, Y. Sharma, H. Zhang, J. Yi, and C. Hsieh, "EAD: elastic-net attacks to deep neural networks via adversarial examples," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, 2018, pp. 10–17.

[69] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv:1605.07277*, 2016. [Online]. Available: https://arxiv.org/abs/1605.07277

[70] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9185–9193.

[71] Y. Dong, T. Pang, H. Su, and J. Zhu, "Evading defenses to transferable adversarial examples by translation-invariant attacks," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4312–4321.

[72] C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, and A. L. Yuille, "Improving transferability of adversarial examples with input diversity," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2730–2739.

[73] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in *International Conference on Machine Learning*, vol. 80. PMLR, 2018, pp. 2142–2151.

[74] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 15–26.

[75] J. Uesato, B. O'Donoghue, P. Kohli, and A. van den Oord, "Adversarial risk and the dangers of evaluating against weak attacks," in *International Conference on Machine Learning*, vol. 80. PMLR, 2018, pp. 5032–5041.

[76] S. Cheng, Y. Dong, T. Pang, H. Su, and J. Zhu, "Improving black-box adversarial attacks with a transfer-based prior," in *Advances in Neural Information Processing Systems*, 2019, pp. 10 934–10 944.

[77] Y. Li, L. Li, L. Wang, T. Zhang, and B. Gong, "NATTACK: learning the distributions of adversarial examples for an improved black-box attack on deep neural networks," in *International Conference on Machine Learning*, vol. 97. PMLR, 2019, pp. 3866–3876.

[78] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square Attack: a query-efficient black-box adversarial attack via random search," in *European Conference on Computer Vision*, 2020, pp. 484–501.

[79] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[80] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," in *International Conference on Learning Representations*, 2018.

[81] M. Cheng, T. Le, P.-Y. Chen, H. Zhang, J. Yi, and C.-J. Hsieh, "Query-efficient hard-label black-box attack: An optimization-based approach," in *International Conference on Learning Representations*, 2019.

[82] M. Cheng, S. Singh, P. H. Chen, P.-Y. Chen, S. Liu, and C.-J. Hsieh, "Sign-OPT: A query-efficient hard-label adversarial attack," in *International Conference on Learning Representations*, 2020.

[83] Y. Dong, H. Su, B. Wu, Z. Li, W. Liu, T. Zhang, and J. Zhu, "Efficient decision-based black-box adversarial attacks on face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7714–7722.

[84] P. Zhao, Z. Fu, Q. Hu, J. Wang *et al.*, "Detecting adversarial examples via key-based network," *arXiv:1806.00580*, 2018. [Online]. Available: https://arxiv.org/abs/1806.00580

[85] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," in *International Conference on Learning Representations*, 2018.

[86] K. Dvijotham, R. Stanforth, S. Gowal, T. A. Mann, and P. Kohli, "A dual approach to scalable verification of deep networks." in *UAI*, vol. 1, no. 2, 2018, p. 3.

[87] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5286–5295.

[88] H. Salman, J. Li, I. P. Razenshteyn, P. Zhang, H. Zhang, S. Bubeck, and G. Yang, "Provably robust deep learning via adversarially trained smoothed classifiers," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 289–11 300.

[89] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T. A. Mann, and P. Kohli, "On the effectiveness of interval bound propagation for training verifiably robust models," *arXiv:1810.12715*, 2018. [Online]. Available: http://arxiv.org/abs/1810.12715

[90] M. Mirman, T. Gehr, and M. T. Vechev, "Differentiable abstract interpretation for provably robust neural networks," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, 2018, pp. 3575–3583.

[91] E. Wong, F. Schmidt, J. H. Metzen, and J. Z. Kolter, "Scaling provable adversarial defenses," in *Advances in Neural Information Processing Systems*, 2018, pp. 8400–8409.

[92] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural computation*, vol. 7, no. 2, pp. 219–269, 1995.

[93] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. P. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" in *Advances in Neural Information Processing Systems*, 2019, pp. 3353–3364.

[94] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[95] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," in *International Conference on Learning Representations*, 2019.

[96] H. Zhang, H. Chen, Z. Song, D. Boning, inderjit dhillon, and C.-J. Hsieh, "The limitations of adversarial training and the blind-spot attack," in *International Conference on Learning Representations*, 2019.

[97] T. Pang, K. Xu, Y. Dong, C. Du, N. Chen, and J. Zhu, "Rethinking softmax cross-entropy loss for adversarial robustness," in *International Conference on Learning Representations*, 2020.

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

[99] J. Gao, B. Wang, Z. Lin, W. Xu, and Y. Qi, "Deepcloak: Masking deep neural network models for robustness against adversarial samples," in *ICLR (Workshop)*, 2017. [Online]. Available: https://arxiv.org/abs/1702.06763

[100] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," in *ICLR (Workshop)*, 2015. [Online]. Available: http://arxiv.org/abs/1412.5068

[101] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, "Parseval networks: Improving robustness to adversarial examples," in *International Conference on Machine Learning*. PMLR, 2017, pp. 854–863.

[102] A. S. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[103] Z. Yan, Y. Guo, and C. Zhang, "Deep defense: Training DNNs with improved adversarial robustness," in *Advances in Neural Information Processing Systems*, vol. 31, 2018, p. 417–426.

[104] J. Buckman, A. Roy, C. Raffel, and I. Goodfellow, "Thermometer encoding: One hot way to resist adversarial examples," in *International Conference on Learning Representations*, 2018.

[105] C. Guo, M. Rana, M. Cissé, and L. van der Maaten, "Countering adversarial images using input transformations," in *International Conference on Learning Representations*, 2018.

[106] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. L. Yuille, "Mitigating adversarial effects through randomization," in *International Conference on Learning Representations*, 2018.

[107] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," in *International Conference on Learning Representations*, 2018.

[108] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-GAN: Protecting classifiers against adversarial attacks using generative models," in *International Conference on Learning Representations*, 2018.

[109] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *International Conference on Learning Representations*, 2017.

[110] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *arXiv:1703.00410*, 2017. [Online]. Available: http://arxiv.org/abs/1703.00410

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

[112] T. Chuman, W. Sirichotedumrong, and H. Kiya, "Encryption-then-compression systems using grayscale-based image encryption for jpeg images," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 6, pp. 1515–1525, June 2019.

[113] W. Sirichotedumrong and H. Kiya, "Grayscale-based block scrambling image encryption using YCbCr color space for encryption-then-compression systems," *APSIPA Transactions on Signal and Information Processing*, vol. 8, p. e7, 2019.

[114] W. Sirichotedumrong, Y. Kinoshita, and H. Kiya, "Pixel-based image encryption without key management for privacy-preserving deep neural networks," *IEEE Access*, vol. 7, pp. 177 844–177 855, 2019.

[115] K. Madono, M. Tanaka, M. Onishi, and T. Ogawa, "Block-wise scrambled image recognition using adaptation network," in *Workshop on Artificial Intelligence of Things (AAAI-WS)*, 2020.

[116] M. Tanaka, "Learnable image encryption," in *2018 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, 2018, pp. 1–2.

[117] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[118] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv:1708.07747*, 2017. [Online]. Available: http://arxiv.org/abs/1708.07747

[119] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[120] M. D. Swanson, M. Kobayashi, and A. H. Tewfik, "Multimedia data-embedding and watermarking technologies," *Proceedings of the IEEE*, vol. 86, no. 6, pp. 1064–1087, 1998.

[121] Y. Adi, C. Baum, M. Cissé, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *27th USENIX Security Symposium*, 2018, pp. 1615–1631.

[122] D. Hendrycks, K. Zhao, S. Basart, J. Steinhardt, and D. Song, "Natural adversarial examples," *arXiv:1907.07174*, 2019. [Online]. Available: https://arxiv.org/abs/1907.07174

[123] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *arXiv:1704.01155*, 2017. [Online]. Available: https://arxiv.org/abs/1704.01155

[124] S. Miyazato, X. Wang, T. Yamasaki, and K. Aizawa, "Reinforcing the robustness of a deep neural network to adversarial examples by using color quantization of training image data," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 884–888.

[125] M. AprilPyone, Y. Kinoshita, and H. Kiya, "Filtering adversarial noise with double quantization," in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Nov 2019, pp. 1745–1749.

[126] R. W. Floyd and L. Steinberg, "An Adaptive Algorithm for Spatial Greyscale," *Proceedings of the Society for Information Display*, vol. 17, no. 2, pp. 75–77, 1976.

[127] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. Jawahar, "Cats and dogs," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3498–3505.

[128] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.

[129] J. Howard *et al.*, "fastai," https://github.com/fastai/fastai, 2018.

[130] S. Kornblith, J. Shlens, and Q. V. Le, "Do better imagenet models transfer better?" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2661–2671.

[131] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv:1711.05101*, 2017. [Online]. Available: https://arxiv.org/abs/1711.05101

[132] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay," *arXiv:1803.09820*, 2018. [Online]. Available: https://arxiv.org/abs/1803.09820

[133] M. AprilPyone and H. Kiya, "Encryption inspired adversarial defense for visual classification," in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 1681–1685.

[134] W. Sirichotedumrong, T. Maekawa, Y. Kinoshita, and H. Kiya, "Privacy-preserving deep neural networks with pixel-based image encryption considering data augmentation in the encrypted domain," in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 674–678.

[135] K. Kurihara, S. Imaizumi, S. Shiota, and H. Kiya, "An encryption-then-compression system for lossless image compression standards," *IEICE Transactions on Information and Systems*, vol. 100, no. 1, pp. 52–56, 2017.

[136] A. Kerckhoffs, "La cryptographie militaire," *Journal des sciences militaires*, pp. 5–38, 1883.

[137] W. Sirichotedumrong and H. Kiya, "A gan-based image transformation scheme for privacy-preserving deep neural networks," in *2020 28th European Signal Processing Conference (EUSIPCO)*, 2020, pp. 745–749.

[138] H. Ito, Y. Kinoshita, and H. Kiya, "Image transformation network for privacy-preserving deep neural networks and its security evaluation," *arXiv:2008.03143*, 2020. [Online]. Available: https://arxiv.org/abs/2008.03143

[139] ——, "A framework for transformation network training in coordination with semi-trusted cloud provider for privacy-preserving deep neural networks," in *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Dec 2020, pp. 1420–1424.

[140] M. Bellare, P. Rogaway, and T. Spies, "Addendum to "the ffx mode of operation for format-preserving encryption"," *A parameter collection for enciphering strings of arbitrary radix and length, Draft 1.0, NIST*, 2010.

[141] M. AprilPyone and H. Kiya, "An extension of encryption-inspired adversarial defense with secret keys against adversarial examples," in *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2020, pp. 1369–1374.

[142] ——, "Ensemble of models trained by key-based transformed images for adversarially robust defense against black-box attacks," *arXiv:2011.07697*, 2020. [Online]. Available: https://arxiv.org/abs/2011.07697

[143] J. Uesato, B. O'Donoghue, P. Kohli, and A. van den Oord, "Adversarial risk and the dangers of evaluating against weak attacks," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, 2018, pp. 5032–5041.

[144] Y. Li, L. Li, L. Wang, T. Zhang, and B. Gong, "NATTACK: learning the distributions of adversarial examples for an improved black-box attack on deep neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, 2019, pp. 3866–3876.

[145] L. N. Smith and N. Topin, "Super-convergence: Very fast training of residual networks using large learning rates," *arXiv:1708.07120*, 2017. [Online]. Available: http://arxiv.org/abs/1708.07120

[146] P. Micikevicius, S. Narang, J. Alben, G. F. Diamos, E. Elsen, D. García, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, "Mixed precision training," *arXiv:1710.03740*, 2017. [Online]. Available: http://arxiv.org/abs/1710.03740

[147] S. Craver, N. D. Memon, B. Yeo, and M. M. Yeung, "Resolving rightful ownerships with invisible watermarking techniques: limitations, attacks, and implications," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 4, pp. 573–586, 1998.

[148] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 707–723.

[149] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[150] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 243–254, 2016.

[151] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *International Conference on Learning Representations*, 2016.

[152] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015, pp. 1135–1143.

[153] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *International Conference on Learning Representations*, 2017.

[154] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

List of Figures

List of Tables

Acknowledgement

I would like to express my gratitude to:

- Prof. Hitoshi Kiya who is my supervisor. He has guided and encouraged me throughout the study. He continually and persuasively conveyed a spirit of adventure in regard to research. Without his supervision and constant help, this thesis would not have been possible.

- Asst. Prof. Sayaka Shiota who is like my second supervisor. She has helped me in building my career. Her support and encouragement has motivated me for my Ph.D. research.

- Prof. Nobutaka Ono and Prof. Masaaki Fujiyoshi, who gave feedback for my research at the end of every semester. Their valuable comments have improved my research in this thesis.

- Prof. Naoyuki Kubota and Prof. Masayuki Tanaka who are reviewers of this thesis. Their kind feedback has helped me improve this thesis.

- Dr. Yuma Kinoshita and all members in Kiya-Shiota laboratory, who have helped me in many different ways from administrative things to research.

- Mrs. Kimiko Fukushima who is the secretary of Kiya-Shiota laboratory. She has helped me with all the paper works throughout my Ph.D. course.

- Mrs. Mayu Abe who used to be the secretary in the international office of Tokyo Metropolitan University. She has helped me even before I came to Japan for getting visa and administrative things. She is like a family member and her hospitality makes me feel home in Japan.

- Ms. Sayaka Koseki who is my close friend. She has helped me in many different ways throughout my study.

- My mother who always encourages me for higher education, and always tell me education is the key.
- All people who have helped me in any way throughout my Ph.D. course.

I would like to to express the deepest appreciation to:

- Tokyo Metropolitan Government and Tokyo Metropolitan University for granting me Tokyo Human Resources Fund for City Diplomacy scholarship. Without this support, it would not be possible to pursue my Ph.D.