AUG-ILA: MORE TRANSFERABLE ATTACKS AND THEIR APPLICATION TO ADVERSARIAL TRAINING

by

CHIU WAI YAN

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Master of Philosophy
in Computer Science and Engineering

June 2022, Hong Kong

Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

_____

CHIU WAI YAN

AUG-ILA: MORE TRANSFERABLE ATTACKS AND THEIR APPLICATION TO ADVERSARIAL TRAINING

by

CHIU WAI YAN

This is to certify that I have examined the above M.Phil. thesis

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by

the thesis examination committee have been made.

PROF. DIT-YAN YEUNG, THESIS SUPERVISOR

PROF. DIT-YAN YEUNG, HEAD OF DEPARTMENT

Department of Computer Science and Engineering

18 June 2022

TABLE OF CONTENTS

Title Page i

Authorization Page ii

Signature Page iii

Table of Contents iv

Abstract vi

Chapter 1 Introduction 1

1.1 Background 1

1.2 Transferability of Adversarial Examples 4

1.3 Introduction to Adversarial Training 6

Chapter 2 Related Works 11

2.1 White-box Attacks 11

2.2 Black-box Attacks 14

2.3 Transfer-Based Black-box Attacks 15

2.4 Defenses to Adversarial Attacks 16

2.5 Intermediate Level Attack for Fine-tuning Adversarial Examples 18

Chapter 3 Methods 20

3.1 Augmentation to Intermediate Level Attack 21

3.2 Reverse Adversarial Update on the Clean Example 22

3.3 Attack Interpolation on the Reference Attack 23

Chapter 4 The Effect of Aug-ILA as a Transfer-based Black-box Attack 25

4.1 Effectiveness of Common Image Augmentation Operations 26

4.2 Black-box Transferability of Aug-ILA 27

4.3 Effect of Aug-ILA on Defended Models 32
4.4 Effect of Different Hyper-parameters 33
4.5 Roles of Intermediate Layer Perturbation and Augmentation 36

Chapter 5 The Effect of Aug-ILA in Fast Adversarial Training 38
5.1 Setup 38
5.2 Adversarial Robustness Trained with Aug-ILA 40
5.3 Studies on the Hyper-parameters 43

Chapter 6 Conclusion 48

References 50

Appendix A Hyper-parameters Used in the Baselines 59

Appendix B Effect of Reverse Adversarial Update on the Model Performance 60

AUG-ILA: MORE TRANSFERABLE ATTACKS AND THEIR APPLICATION TO ADVERSARIAL TRAINING

by

CHIU WAI YAN

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

ABSTRACT

An intriguing property of deep neural networks is that adversarial attacks can transfer across different models. Existing methods such as the Intermediate Level Attack (ILA) further improve black-box transferability by fine-tuning a reference adversarial attack, so as to maximize the perturbation on a pre-specified layer of the source model. In this work, we revisit ILA and evaluate the effect of applying augmentation to the images before passing them to ILA. We start by looking into the effect of common image augmentation techniques and exploring novel augmentation with the aid of adversarial perturbations. Based on the observations, we propose Aug-ILA, an improved method that enhances the transferability of an existing attack under the ILA framework. Specifically, Aug-ILA has three main characteristics: typical image augmentation such as random cropping and resizing applied to all ILA inputs, reverse adversarial update on the clean image, and interpolation between two attacks on the reference image. Our experimental results show that Aug-ILA outperforms ILA and its subsequent variants, as well as state-of-the-art transfer-based attacks,

by achieving 96.99% and 87.84% average attack success rates with perturbation budgets $13/255$ (0.05) and $8/255$ (0.03), respectively, on nine undefended models.

Besides, being a strong transfer-based attack, Aug-ILA can also be adopted in adversarial training. We propose a two-phase training scheme which aims to both speed up the training time and also achieve better robustness compared to previous works. Having a pre-training phase using an existing framework, we further employ Aug-ILA to fine-tune the model. Extensive experiments illustrate that Aug-ILA can boost the model robustness up to 5% while the model can still converge in a reasonable time.

CHAPTER 1

INTRODUCTION

1.1 Background

With recent advances, deep neural network (DNN) models have achieved outstanding performance in solving many computer vision tasks, including object recognition, semantic segmentation, image synthesis, object tracking, and many more. Within its wide range of applications, some tasks might require the model to be secure and reliable, for instance, medical imaging analysis, autonomous driving, machine control, etc. On the other hand, DNNs are also sometimes adopted in system security applications such as spam filtering, biometric authentication, etc. In these applications, the consequences of errors might be more severe, highlighting the importance of the reliability and robustness of the model.

A DNN model composes of a sequence of layers, while each layer can be viewed as a mapping function [1]: $f^{\ell}_{\theta_\ell} : \mathbb{R}^M \rightarrow \mathbb{R}^N$, where θ_ℓ is the model parameters with ℓ indicating the index of the layer, M is the input dimension and N is the output dimension. $f^{\ell}_{\theta_\ell}$ can be a combination of linear and non-linear transformations. In other words, it can exhibit properties, or vulnerabilities of both linear and non-linear transformations. The entire model F_θ with the groups of parameters ℓ and depth k can generally be defined by:

$$F_\theta(\boldsymbol{x}) = f^{k}_{\theta_k}(f^{k-1}_{\theta_{k-1}}...(f^{2}_{\theta_2}(f^{1}_{\theta_1}(\boldsymbol{x})))) \tag{1.1}$$

For supervised learning, $\boldsymbol{x}$ is the input data. A classification model predicts correctly when $F_\theta(\boldsymbol{x}) = \boldsymbol{y}$, where $\boldsymbol{y}$ is the class label. The model learns by minimizing a loss function, and updates the parameters through gradient descent, an iterative approximation to alter the parameters such that the object function is minimized. The gradient of the input example, is calculated by backpropagation.

[1] Although some model architectures such as inception module and residual connection can contain branches, the gradient can be propagated in the same way.

Specifically, for image data $\boldsymbol{x}$, we could compute its gradient $\frac{\partial F}{\partial \theta}$ to obtain the proper adjustment to the model's parameters. Oppositely, this also potentially enables a destructive perturbation by computing $\frac{\partial F}{\partial \boldsymbol{x}}$ to update the examples.

The inspiring destructive update was realized quickly. Studies [64, 19] showed that DNNs can be vulnerable to a special input known as adversarial examples or adversarial attacks. For most of the DNN models, adversarial examples can be crafted to degrade their performance significantly. By adding small perturbations to normal images [2], the resulting adversarial examples can consistently fool the model to yield incorrect predictions. Such adversarial perturbations are crafted to be tiny that they are almost imperceptible by humans, yet they can cause apparent fluctuations in the model output. An adversarial example $\boldsymbol{x}' = \boldsymbol{x} + \delta$ working on a classification model F_θ, can be defined by:

$$F_\theta(\boldsymbol{x} + \delta) \neq F_\theta(\boldsymbol{x}) \tag{1.2}$$

One implicit assumption here is that the model predicts the correct class of the input $\boldsymbol{x}$, i.e. $F_\theta(\boldsymbol{x}) = y$. If the model predicts an input incorrectly, the adversarial example of that particular input is not defined.

To constrain the magnitude of the perturbation so that it is hardly distinguishable by humans, the adversarial perturbation is naturally bounded by a perturbation budget ϵ such that $\|\delta\|_p \leq \epsilon$. Common norms include ℓ_0-norm (the number of pixels being changed), ℓ_2-norm (L2 score of the difference) and ℓ_∞-norm (the maximum pixel value changed in an image), that ℓ_∞-norm is used the most frequently on gradient-based attacks since it can better resemble the difference perceived by human vision. A common and simplistic illustration of adversarial attack is shown in Figure 1.1.

[2] Meanwhile, adversarial examples for other data modalities such as audio and test data also exist. In this work, we will only focus on image data.

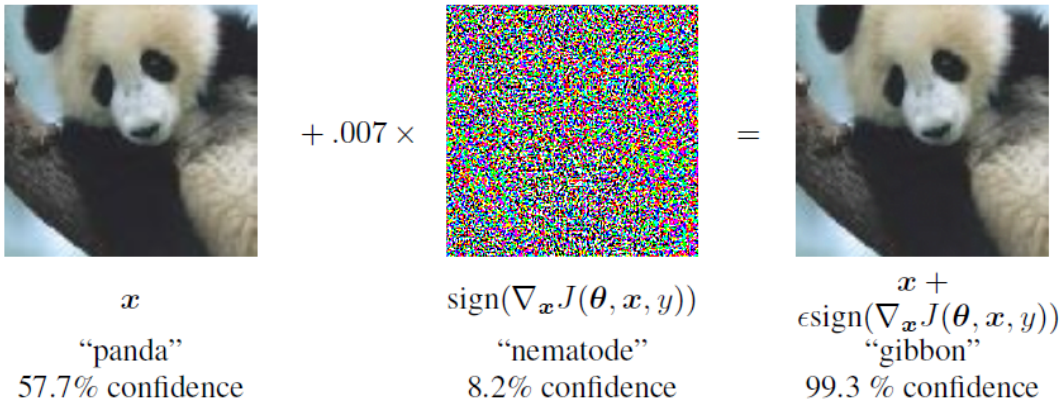


Figure 1.1: An illustration of the effect of adversarial attack in Goodfellow et al. [19]. The attack featured is the FGSM attack introduced in the same paper.

Nilesh et al. [11] first defined the term 'adversarial classification' and viewed classification as a game between a classifier and the adversary. This classifier, however, refers to data-driven models in general. In early works studying adversarial examples in DNNs, the phenomenon was initially believed to be attributed to the non-linearity of the models [65]. Nonetheless, it was later shown that linear behavior in high-dimensional space can sufficiently cause adversarial examples [19]. As the layers get deepened, the adversarial examples amplify to cause a larger influence on the final prediction. Meanwhile, a wide model, which consists of more parameters in each layer, is believed to have better robustness against adversarial examples [84].

In cyber-security, a threat model is used to systematically analyze the target, accessibility, knowledge, etc. possessed by the adversary in order to discover any potential threat and vulnerability in a system. In such context, we assume the attackers solely focus on compromising the *integrity*, which includes the model correctness and performance, of the DNN-based applications in evaluation time [3]. Under the premises, the most common threat models focus on the information occupied by the adversary: the white-box and black-box settings. The white-box setting assumes that one has access to the victim model's internal state, including its gradient, parameters, training dataset, etc. The black-box setting, on the other hand, only allows querying the model with

[3]Other studies such as data poisoning, confidentiality, privacy, etc. will not be discussed in this work.

input but prohibits subsequent access to the model information. While more variations such as the grey-box and no-box settings exist, they are generally not considered in this work.

The effectiveness of adversarial attacks on deep learning models raises concerns in multiple fields especially for security-sensitive applications. In response to this, the research community keeps proposing defenses against adversarial attacks. Since then, the competition between machine learning models and the adversary has progressed for more than a decade. Different attacks and defenses are introduced in detail in multiple survey papers [75, 59].

1.2 Transferability of Adversarial Examples

Another intriguing phenomenon is that adversarial attacks can transfer across different models [49]. If an adversarial example is capable of changing the prediction of a model, the same example can also to some extent change the prediction of other similar models. Among the machine learning models, parameterized models such as regression models and DNNs are relatively sensitive to transfer-based attacks [41]. This makes DNNs especially vulnerable to the adversary. One explanation for the phenomenon of attack transferability is the overlapping decision boundaries shared by different models [41, 14]. In particular, with the same set of training data, different models can end up having similar patterns to determine whether an image belongs to the class, as illustrated in Figure 1.2. With such property, adversaries can generate attacks easily by exploiting existing adversarial examples. Even when the attacker has no access to the internal state of the model (known as the black-box threat model), the transferability can be used either to generate attacks from a surrogate model (another model to replace the original one from being attacked on, also known as substitute model or proxy model) [87] or to provide proper guidance to reduce the number of queries [22].

While many methods set their goals to generate highly transferable attacks [44, 73, 21], some attempt to improve the transferability of a given adversarial example by fine-tuning it, which could gain advantages from the additional information of a created attack. Huang et al. [27] proposed the Intermediate Level Attack (ILA), a technique that takes an existing adversarial example as a reference and fine-tunes it to boost its black-box transferability on other DNN models. Unlike

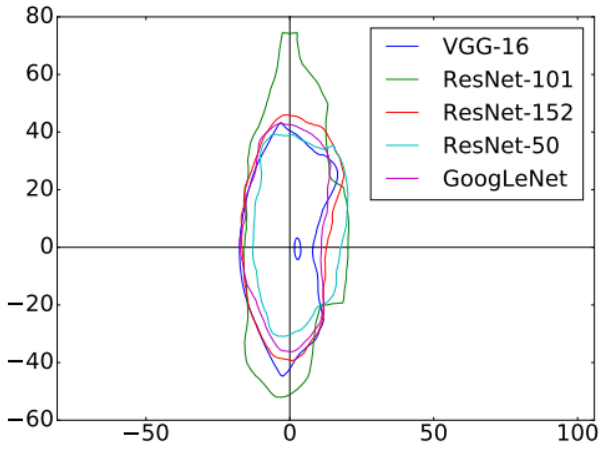


Figure 1.2: An illustration on different models' decision boundaries on a single image performed by Liu et al. [41]. The two axes indicate the perturbation strength (over 255) on the image. The x-axis is the gradient direction of VGG-16, and the y-axis is a random orthogonal direction.

many attacks in which the objective is to maximize the categorical loss, ILA intends to maximize the projection loss of the intermediate feature map discrepancies between attacks. As a result, this finetuning process not only retains the overall damage to the categorical loss, it introduces further discrepancies in the intermediate layers of the models. Therefore, it leads to a higher transferability so that the examples give different activation to other models.

With a strong white-box attack as the reference attack, ILA can achieve remarkable black-box transferability, outperforming previous methods based on direct generation of transfer-based attacks [87, 77]. More specifically, ILA takes three input images as references to fine-tune the attack, including a clean image $\boldsymbol{x}$, a reference attack of the clean image $\boldsymbol{x}'$, and the updated image from the previous iteration $\boldsymbol{x}''$. Both $\boldsymbol{x}$ and $\boldsymbol{x}'$ remain unchanged throughout the fine-tuning process, with $\boldsymbol{x}''$ getting updated every iteration. A simplistic illustration on the relationship between the examples is shown in Figure 1.3. If we view the fine-tuning process of ILA as a generalization task that attempts to make an attack effective for different models, a possible direction for improvement is to increase the diversity of the input references, such as by applying data augmentation.

In this dissertation, we revisit ILA and explore possible room for improvement. A natural way to promote input diversity is to apply data augmentation techniques to the input references. We first evaluate the effect of typical image transformation operations in the context of ILA. Then, we

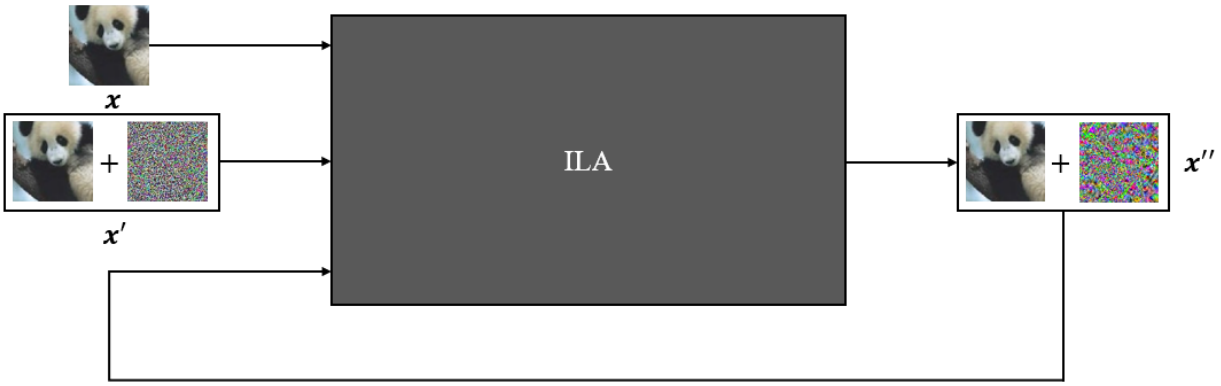


Figure 1.3: A flowchart briefly showing the relationship between the inputs and output of ILA

look into transformations exploiting the adversarial perturbation, and apply them together with the best image transformation found previously. Specifically, we incorporate a mixture of augmentation operations: random image cropping and resizing on all input examples, reverse adversarial update on the clean image, and attack interpolation on the reference attack. Incorporating these adaptations to the original ILA, we propose the Augmented Intermediate Level Attack (Aug-ILA) to further enhance black-box transferability. By fine-tuning simple baseline attacks such as the Iterative Fast Gradient Sign Method (I-FGSM), Aug-ILA not only outperforms ILA and its subsequent variants, but also state-of-the-art transfer-based attacks. A brief overview of the resulting adversarial examples and adversarial perturbations is illustrated in Figure 1.4.

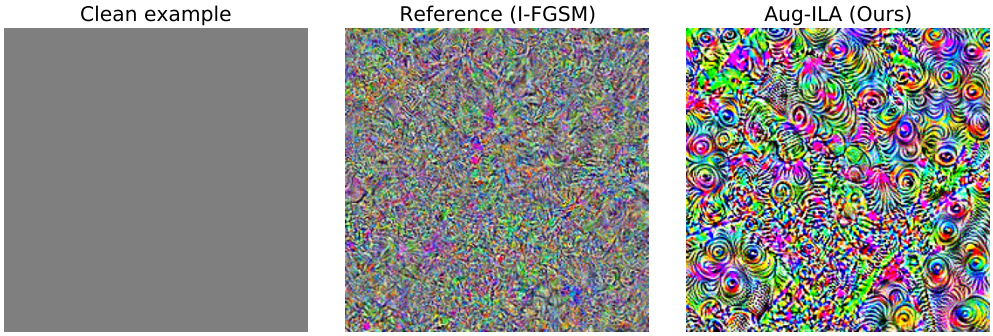
1.3 Introduction to Adversarial Training

Different from standard training, adversarial training [42] intends to train the model with the data which supposes to cause the model wrong prediction. This can be formulated as a min-max game, with an inner (loss) maximzation enclosed by an outer (loss) minimization.

$$\min_{\theta} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}} \left[\max_{\delta\in\mathcal{S}} L(\theta, \boldsymbol{x}+\delta, y) \right] \tag{1.3}$$

From the formulation, L indicates the loss function of the model to train, $\mathcal{S}$ refers to the set of adversarial perturbation within the range ϵ. Since the absolute difference between clean data and

(a) The adversarial examples



(b) The normalized adversarial perturbations

Figure 1.4: A visualization of the clean example $\boldsymbol{x}$, the reference attack $\boldsymbol{x}'$ and Aug-ILA attack $\boldsymbol{x}''$ respectively, together with their corresponding perturbations, with $\epsilon = 8/255$ (0.03).

adversarial examples is tiny, we could directly realize the inner maximization with adversarial examples. In other words, adversarial training is standard training with all the input data replaced by their corresponding adversarial examples. In general, the stronger the attack is (able to change a model's prediction), the more robust the model can be when the attack is used in adversarial training. Overall, the generation of the inner attack, together with the implementation of the outer minimization, are two common directions to improve adversarial training.

For many of the defenses, a phenomenon known as obfuscated gradient [2], as a kind of gradient masking [46], exists to give a false sense of robustness. Obfuscated gradient works by eliminating the gradient of the model in test time, causing most iterative white-box attacks to fail to update properly and hence contributing to its delusion of robustness. Defense schemes relying on obfuscated gradients usually can be circumvented and are vulnerable to black-box attacks which do not make use of the model gradient. According to Athalye et al. [2], the following three characteristics

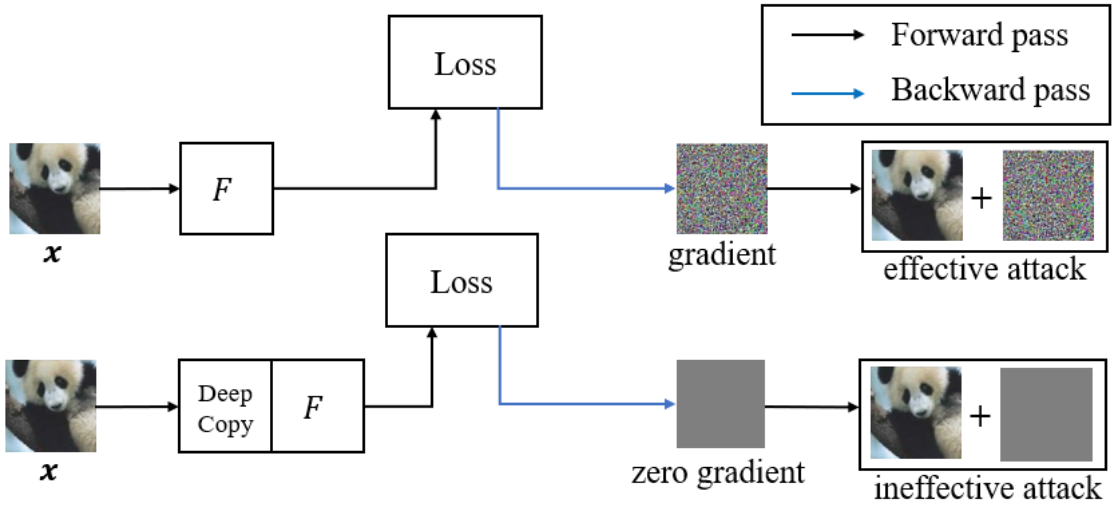


Figure 1.5: A demonstration on how a usual gradient-based white-box attack works on a model without (upper) and with (lower) obfuscated gradient.

can be observed from obfuscated gradient:

1. Shattered gradients, which make the model non-differentiable, or introduce numerical instability, resulting in incorrect gradients.

2. Stochastic gradients, which are usually caused by randomized features of the components. With the model exhibiting different behaviors during attack generation and inference time, the attack fails by incorrectly predicting the gradients.

3. Exploding and vanishing gradients, which are usually caused by multiple iterations of evaluation. The entire evaluation forms an extremely deep network when the procedures are unrolled, resulting in vanishing or exploding gradients. The renowned defensive distillation [50] is an example relying on vanishing gradients with a 'high temperature' softmax in training time followed by a 'cooldown' in evaluation time.

For instance, a deep copy to the input before passing through the model is sufficient to cause obfuscated gradient, specifically shattered gradients, as illustrated in Figure 1.5.

Adversarial training is, however, an example that does not rely on obfuscated gradients. Unlike many of the defenses which compromise white-box attacks by prohibiting the model from leaking

the true gradient information, adversarial training is a method to train the model to be robust and insensitive to most kinds of perturbations. Nevertheless, adversarial training suffers from multiple drawbacks. Due to the iterative nature of a strong attack, training takes much longer than that with clean inputs, causing it impractical for large-scale datasets [60]. Besides, adversarial training suffers from a 'no free lunch' trade-off [67, 68], where an increase in robustness naturally harms the model accuracy. Oppositely, mixing more clean examples with the adversarial ones increases the accuracy but decreases the robustness. On the other hand, overfitting in adversarial training can harm to a very large degree in robustness [55], while a phenomenon known as catastrophic overfitting [72] can occur, which the model ends up in a state that correctly classifying only the training adversarial examples but not the unseen ones. Furthermore, some studies [47, 10] reveal that a slight change in hyper-parameters can cause a huge difference in the final result, raising the difficulty to optimize the model.

To address the inefficiency of adversarial training, which is also one of the most noteworthy drawbacks, Wong et al. proposed a fast framework [72]. Instead of using strong iterative attacks, they discovered using a weak attack with good initialization, named FFGSM, could also achieve comparable robustness to those using strong attacks. In the meantime, the step learning rate scheduling could be replaced by cyclic learning rate scheduling to further reduce the total number of epochs required. In the upcoming sections of this work, we name such framework to be the *Fast* framework. Even though the Fast framework did introduce a breakthrough in terms of efficiency, with advances in traditional adversarial training, the gap of robustness between the standard framework and the Fast framework is still enlarged quickly. This motivates us to have improvements made on top of the original Fast framework.

Incorporating extra features based on a reference attack, Aug-ILA can be a good candidate to fine-tune the FFGSM attack. After demonstrating the effectiveness of Aug-ILA in transfer-based attacks in Chapter 4, we further explore the adversarial robustness of the model when Aug-ILA is used as a fine-tuned attack in adversarial training in Chapter 5. Due to the iterative nature of Aug-ILA, we propose a two-stage model fine-tuning strategy, in which the model is first trained purely under the Fast framework. After that, we slightly fine-tune the model with FFGSM + Aug-ILA for a few epochs. The results suggest that Aug-ILA consistently yields a more robust model within a

reasonable period.

The contributions of this work can be summarized as follows:

- We propose Aug-ILA, a method that fine-tunes a given adversarial example to reinforce its attack transferability. Aug-ILA follows the framework of ILA, but with effective augmentations introduced to its references.
- We identify factors affecting the transfer success rate by extensive experiments and explain the roles of intermediate feature perturbation and augmentation in a transfer-based attack.
- We discover a novel application of Aug-ILA, which can be used to improve the model robustness when it is adopted in adversarial training with the Fast framework.

We will start by discussing previous literature closely related to this work in Chapter 2. In Chapter 3, we will present the details and intuition of Aug-ILA. In Chapter 4, we demonstrate the experimental findings of Aug-ILA as a transfer-based black-box attack. Moreover, in Chapter 5, we present further studies on the result when Aug-ILA is adopted in adversarial training under the Fast approach. Finally, we conclude the dissertation in Chapter 6.

CHAPTER 2

RELATED WORKS

In this chapter, we first review common white-box and black-box adversarial attacks that are closely related to our work, then we investigate defense works as well as variants of adversarial training.

For both white-box and black-box attacks, the attacks can be either non-targeted or targeted. A non-targeted attack is deemed successful as long as the model does not yield the correct prediction. A maximization on the original loss is one of the most common ways to produce the attack. A targeted attack not only causes the model to deviate from its original prediction, but also misguides it to a specific class. It can be achieved by, usually, minimizing the loss on an incorrect class. It can be seen that the same loss can apply to both settings, with the difference between loss maximization and minimization on different target classes.

2.1 White-box Attacks

FGSM. With the intuition of generating attacks through the model gradient, Goodfellow et al. [19] proposed a simplistic but effective attack, known as the Fast Gradient Signed Method (FGSM). The attack generates the adversarial perturbation through a single-step update with the following formulation:

$$\delta = \epsilon \, \text{sign}(\nabla_{\boldsymbol{x}} J(\theta, \boldsymbol{x}, y)) \tag{2.1}$$

where $\nabla_{\boldsymbol{x}} J(\theta, \boldsymbol{x}, y)$ is the gradient vector of the loss with respect to the input example $\boldsymbol{x}$, ϵ is the perturbation budget and θ is the model parameters. The sign function is a function that returns either $+1$ or -1 based on the sign of the input. Although the generation of attacks can be performed quickly, compared to other attacks, FGSM is relatively weak due to the fact that the attack is only iterated once.

PGD. Madry et al. [42] proposed a multi-step variant of the FGSM formulation, forming the Projected Gradient Descent (PGD) attack:

$$\boldsymbol{x}^{t+1} = \text{proj}_{\boldsymbol{x}+\epsilon}(\boldsymbol{x}^t + \alpha \text{ sign}(\nabla_{\boldsymbol{x}} J(\theta, \boldsymbol{x}, y)))$$

where α is the step size, $\text{proj}_{\boldsymbol{x}+\epsilon}$ is a projection function that clips the adversarial example within the perturbation budget and also $[0, 1]$ (the range of a valid image). Being a strong attack, PGD is a good indicator to reflect a model's robustness, and has been widely used as the baseline in different benchmarks [48, 13]. The numbers of iterations vary in different works. In Madry et al.'s experiments [42], 40 and 100 iterations are used in MNIST, 7 and 20 iterations are used in CIFAR-10.

PGD resembles a lot with another attack known as I-FGSM (iterative-FGSM) [33]. While they both have the same update rule, PGD introduces a random initialization of uniform noise around the natural example, which causes more variations in the attack.

C&W's Attack. Apart from exploiting the model gradient, Carlini and Wagner (C&W) proposed an optimization-based attack [6], which makes use of an objective function f with the property $f(\boldsymbol{x} + \delta) \leq 0$ if and only if $F_\theta(\boldsymbol{x} + \delta) = t$ (with t being the target class to manipulate). Then the adversarial perturbation can be obtained by minimizing:

$$\begin{aligned} \arg\min_{\delta} \quad & \|\delta\|_2 \\ \text{s.t. } & f(\boldsymbol{x} + \delta) \leq 0 \\ & \boldsymbol{x} + \delta \in [0, 1]^n \end{aligned} \tag{2.2}$$

With a suitably chosen constant c, the formulation can be derived alternatively as

$$\begin{aligned} \arg\min_{\delta} \quad & \|\delta\|_2 + c \cdot f(\boldsymbol{x} + \delta) \\ \text{s.t. } & \boldsymbol{x} + \delta \in [0, 1]^n \end{aligned} \tag{2.3}$$

Empirical findings show that the attack works well when

$$f(\boldsymbol{x}') = \max(\max(\{Z_i(\boldsymbol{x}') : i \neq t\} - Z_t(\boldsymbol{x}'), -\kappa)) \tag{2.4}$$

where Z_i is the function to the logit output of i-th class without going through softmax and κ is a parameter to enlarge the margin of the logits between clean and adversarial examples.

C&W's attack successfully breaks defensive distillation [50] which was believed to be one of the most robust defense methods at that time. In the meantime, C&W's attack is also able to bypass many of the defenses which rely on obfuscated gradient [5, 2]. In the original implementation of C&W's attack, a binary search is used to search over c for 20 iterations, and for each value of c, 10,000 iterations of gradient descent with Adam optimizer is performed. This setup is not used frequently in practice due to the slow generation of the attacks. Instead, C&W's attack in the literature usually adopts an implementation that directly optimizes Equation (2.4) using PGD within 100 iterations. The new implementation, sometimes called PGD-CW, hardly achieves a good performance compared to the original PGD attack using cross-entropy loss.

AutoAttack. AutoAttack (AA) [10] comprises 4 attacks in sequence, namely non-targeted APGD-CE, targeted APGD-DLR, targeted FAB (Fast Adaptive Boundary) Attack [18] and Square Attack [43]. Only the former three belong to white-box attacks while the Square Attack is a black-box attack with 5000 queries. The proposed APGD attack differs from the original PGD attack by a search for good initial points, as well as introducing a progressively diminishing step size. On the other hand, APGD-DLR is built on the APGD framework but uses the Difference of Logits Ratio (DLR) loss function:

$$\text{Targeted-DLR}_t(\boldsymbol{x}, y) = -\frac{z_y - z_t}{z_{\pi_1} - (z_{\pi_3} + z_{\pi_4})/2} \tag{2.5}$$

where z_i is the logit output of i-th class and π is the ordering of the components of z in descending order. Since APGD consistently outperforms PGD in Croce et al.'s experiments [10] and reveals the weaknesses of many robust models, AutoAttack is more preferred to reliably evaluate a model's robustness [9]. For datasets with more classes, FAB Attack becomes prohibitive in terms of both computation and memory usage. Besides, as Square Attack requires 5000 queries, it causes huge overhead and hence is not feasible for mass evaluations on various models. Therefore, state-of-the-art works [20, 54] sometimes exclude FAB Attack and Square Attack when they adopt AutoAttack as the measure of the model robustness.

2.2 Black-box Attacks

Black-box threat models can be further split into two sub-categories: score-based (soft label) and decision-based (hard label). In a score-based setting, the output of the model is assumed to be the logit output (score) of each class. In a decision-based setting, the output of the model is solely the final predicted class without any confidence score prompted. The decision-based setting is more challenging and enforced with more restrictions, because less information is available in the model output to be exploited. For both settings, a general pattern to craft the attack is to estimate the model gradient by repeatedly querying the model with varying inputs.

Zeroth-Order Optimization (ZOO) [8] is probably one of the most fundamental forms of black-box attacks with mass queries. In the paper, the attack is computed by stochastic coordinate descent, which estimate the loss with respect to each coordinate (parameter) by symmetric difference quotient $\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}_i} \approx \frac{f(\boldsymbol{x}+h\boldsymbol{e}_i)-f(\boldsymbol{x}-h\boldsymbol{e}_i)}{2h}$, where h is a small constant and $\boldsymbol{e}_i$ is a standard basis vector with only the i-th item being 1. Although the formulation can eventually converge to a strong attack, it requires a huge amount of queries and computation power which drastically grows with the input dimension.

With ZOO, many follow-up works are proposed to reduce the number of queries with different techniques such as feature reduction [3], aids from generative models [69], improved algorithm on boundary search [4, 35, 7], etc. Ilyas et al. [28] adopted bandit optimization over Natural Evolutionary Strategies (NES) with gradient prior, an initialization of gradient-based on both time-dependent factors and data-dependent factors. Compared to the two factors, Yan et al. [79] discovered that the gradient of a surrogate model can be a very good choice of gradient prior, bridging transfer-based attack and query-based attack for the first time. Such discovery turns the spotlight back to transfer-based attacks, in which the attacks can be generated much faster. In the upcoming section, we look into details of black-box attacks that are largely transferred from a surrogate model.

2.3 Transfer-Based Black-box Attacks

Since strong gradient-based white-box attacks do not have high transferability by themselves [34], a number of attempts have been made in the research community to improve transferability apart from increasing the attack strength.

MI-FGSM. [14] first introduced the Momentum Iterative Fast Gradient Sign Method (MI-FGSM) to incorporate momentum to stabilize the update direction in iterative attacks, resulting in more transferable attacks.

DIM. The Diverse Input Method (DI^2-FGSM, or DIM) [77] applies random resizing and zero-padding with a certain probability to the image in each iteration of I-FGSM. Although the processes are simple image transformations, the images generated are able to bypass defenses with extensive image processing.

TIM. Dong et al. [15] pointed out the discretion of attention map between the defended and undefended models and proposed the Translation-Invariant Method (TIM) by attacking an ensemble of translated images. To reduce the computational overhead, it was shown that kernel convolution can be applied to the gradient to achieve similar effects.

SIM and NI-FGSM. Lin et al. [39] introduced the scale-invariant property of DNN and proposed the Scale-Invariant Method (SIM) to attack an ensemble of images with a scaling factor of 0.5 on the pixel values. In the same paper, the Nesterov Iterative Fast Gradient Sign Method (NI-FGSM) was introduced, aiming to escape from poor local maxima with the Nesterov accelerated gradient.

VMI-FGSM and VNI-FGSM. Wang et al. [70] proposed variance tuning. For each iteration, the gradient is adjusted with an expectation of gradient sampled from the image's neighborhood. Combining variance tuning with the composition of previous methods, denoted as the Composite Transformation Method (CTM), they achieved one of the strongest transfer-based black-box attacks on defended models.

A more recent work [74] replaced all input transformations with a DNN model trained to apply distortions that neutralize the adversarial noises. After preparing the denoising model, the attack is

generated with the objective to sustain such distortions. All these methods suggested that adversarial transferability could be improved by augmenting the images or tuning the magnitude of gradient updates.

Alternatively, some recently proposed methods exposed the surprising nature of deep CNN model architectures and utilized them to generate transferable attacks. Wu et al. [73] found that attack transferability can be enhanced by scaling up the gradient in the skip connection and referred to the technique as Skip Gradient Method (SGM). Guo et al. [21] revised the linear property of DNN models and proposed LinBP to propagate backward without considering the non-linear layers. All of the methods surprisingly achieve high attack transferability among common DNN models. One possible reason for the effectiveness is that the repeated stacking of layers causes the model gradient to differ, and getting skipped through some of the layers better aligns the gradients between the surrogate model and the target model.

2.4 Defenses to Adversarial Attacks

Countermeasures for adversarial examples can be categorized into two main strategies [75]: reactive and proactive. Reactive defenses work on mitigating the adversarial examples on a build model without modifying the model itself. Popular works include adversarial detection [17, 80], where anomaly is filtered out, and adversarial purification [62, 81], which attempts to convert any adversarial example back into benign input. Alternatively, proactive defenses attempt to improve the model robustness by altering the model architecture, model parameters, feed-forward pipeline, etc. One of the most effective and famous methods is adversarial training [19, 42]. In this section, we will focus on adversarial training and some additional defenses which are closely related to this dissertation.

Variations to Adversarial Training. First proposed by Goodfellow et al. [19] in 2015, adversarial training referred to model training with both clean and adversarial examples. It targeted to minimize

$$\alpha\ J(\theta, \boldsymbol{x}, y) + (1-\alpha) J(\theta, \boldsymbol{x} + \epsilon\ \text{sign}(\nabla_x\ J(\theta, \boldsymbol{x}, y)) \tag{2.6}$$

The idea is to mix clean examples and adversarial examples into the training data and balance the two with a weighting factor α. As the above equation directly shows, the attack adopted is FGSM proposed in the same paper. Later, Madry et al. [42] refined the method and generalized the loss function to be the min-max form in Equation (1.3) with the attack replaced by maximization of the loss, which can be realized with iterative attacks such as PGD attack.

Apart from traditional adversarial training, a portion of works attempts to improve model robustness by modifying the loss function. Many methods append a term regularizing the feature-based representation in the loss function. Adversarial Logit Pairing (ALP) [29] introduces a regularization term on the logits. Following ALP, Feature Matching [37] was proposed, with the regularization term coupling the high-level feature maps. Some works [82, 53, 16, 1] proposed to regularize the spectral norm, which is the largest singular value of the weight matrices. Some works proposed to enforce regularization in the backward gradient [12, 56], which is sometimes also called Jacobian regularization [24]. Alternatively, TRADES [85] works on the trade-off between robustness and accuracy, with a modified loss function $\max_{X' \in \mathbb{B}(X,\epsilon)} \phi(f(X)f(X')/\lambda)$, where ϕ is a surrogate loss that enhances the theoretical guarantee. MART [71] considers using misclassified clean examples in addition via a misclassification aware regularization such that the loss differentiates correctly classified examples and incorrectly classified examples.

Moreover, ensemble adversarial training [66], which augments a model's training data with adversarial examples attacking another pretrained model, could achieve further robustness level and won the first round in the NIPS 2017 Competition on Defenses against Adversarial Attack [32]. The top 3 submissions in the competition, also being the most popular works, can yield good performance when they are used together with adversarial training or ensemble adversarial training. Hence, they are also frequently compared in subsequent works.

HGD. High-level representation guided denoiser (HGD) [38] was the rank-1 solution in the NIPS 2017 Competition. It was discovered that error amplifies to a large magnitude in the top layers of the model, leading to a wrong prediction. With such observation, the team used the difference between the clean examples and adversarial examples to train a denoising U-Net, in order to suppress the influence of the perturbation. The loss function is defined to be the ℓ_1 norm of the difference of the model outputs: $L = \|F(\boldsymbol{x} + \delta) - F(\boldsymbol{x})\|_1$. Since the denoiser is trained to

recover the logit output, it is said to be guided by the high-level representation.

R&P. The rank-2 solution [76] advised using two randomized image operations, random resizing and random padding. The input images are resized with a random scale applied such that the resized images are within the range [299, 331). Then zero padding is applied in all four directions. After feeding the randomly processed image to an adversarially trained model for classification, it was found that the result was far better than using adversarial training alone.

Rank-3. The rank-3 solution is not documented by its authors. From its source code [1], multiple distortions are applied to the input images. Similar to rank-2 above, the rank-3 submission applies a sequence of random processing, including shearing, translation, zooming and rotation, as linear transformation to the images. Finally, the processed images are tested with an adversarially trained model.

2.5 Intermediate Level Attack for Fine-tuning Adversarial Examples

Huang et al. [27] proposed ILA to fine-tune a reference I-FGSM attack. Three inputs are required to run ILA, namely an input example $\boldsymbol{x}$, an existing adversarial example $\boldsymbol{x}'$, and the output of previous iteration $\boldsymbol{x}''$ ($\boldsymbol{x}'' = \boldsymbol{x}'$ initially). The key idea of ILA is to maximize the intermediate feature discrepancy in the surrogate model, aiming the same to happen in the target model that leads to higher attack success rate. With a function F_l that outputs the feature maps at layer l of a model F, the ILA projection loss minimizes the dot product:

$$L(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{x}'') = -\Delta \boldsymbol{y}_l'' \cdot \Delta \boldsymbol{y}_l' \tag{2.7}$$

where $\Delta \boldsymbol{y}_l'$ and $\Delta \boldsymbol{y}_l''$ are two vectors defined as follows:

$$\Delta \boldsymbol{y}_l' = F_l(\boldsymbol{x}') - F_l(\boldsymbol{x}) \tag{2.8}$$

$$\Delta \boldsymbol{y}_l'' = F_l(\boldsymbol{x}'') - F_l(\boldsymbol{x}) \tag{2.9}$$

ILA immensely raises the black-box transferability in comparison to the original reference attack. Most target models exhibit a significant drop of accuracy even though the attack is generated

[1] https://github.com/anlthms/nips-2017/tree/master/mmd

against a surrogate model. Based on the same framework, [36] proposed another formulation of ILA:

$$\arg\max_{\boldsymbol{x}''} \ (F_l(\boldsymbol{x}'') - F_l(\boldsymbol{x}))^{\mathsf{T}} \boldsymbol{w}^* \tag{2.10}$$

where $\boldsymbol{w}^*$ is a pre-computed parameter vector that directly maps the intermediate-level discrepancies to predict the adversarial loss in the form of regression. This means $\boldsymbol{w}^*$ can be treated as the parameters of another similar model, but much shallower compared to F. Another notable trick of the design is that the computation of $\boldsymbol{w}^*$ involves every feature map discrepancy $F_l(\boldsymbol{x}'_t) - F_l(\boldsymbol{x})$ in each iteration t during the generation of $\boldsymbol{x}'$. We refer to this attack as ILA++ in the rest of this dissertation.

CHAPTER 3

METHODS

Consider the ILA projection loss in Equation (2.7), this is equivalent to maximizing the dot product $\Delta \boldsymbol{y}_l'' \cdot \Delta \boldsymbol{y}_l'$. The idea behind the formulation is to increase the norm without sacrificing the perturbation direction that causes misclassification. Due to the constraint of the perturbation budget, there is hardly room for fine-tuning in the image space. Instead, projection maximization is carried out in the feature space, specifically at layer l (a usual choice of l is 9 for VGG19 and 3-1 for ResNet50) of the model. Note that the projection loss is defined without normalization. Apart from the feature direction that is perturbed, the projection loss also leads to a maximization of the feature discrepancy $\|F_l(\boldsymbol{x}'') - F_l(\boldsymbol{x})\|$. The greater the margin is, the smaller the negative loss turns out to be. As a result, the ILA attack yields a large margin between the feature of the input examples and the feature of fine-tuned adversarial examples, while the dot product induces variation such that the attack no longer overfits a single model.

Huang et al. [27] suggested in order to find the Best Transfer Direction (BTD), we need to find a reference attack $\boldsymbol{x}'$:

$$\arg\max_{\boldsymbol{x}'} \sum_{m \in M} \mathbb{1}[m(\boldsymbol{x}') \neq m(\boldsymbol{x})] \qquad \text{s.t. } \|\boldsymbol{x}' - \boldsymbol{x}\|_\infty \leq \epsilon \tag{3.1}$$

where M is a large set of distinct CNNs. BTD of $\boldsymbol{x}$ is simply the unit vector $\frac{\boldsymbol{x}'-\boldsymbol{x}}{\|\boldsymbol{x}'-\boldsymbol{x}\|_2}$. From the experiments performed in the same work, I-FGSM perturbation in deeper layers of the model exhibits a larger cosine similarity to that of a multi-fooling attack. Therefore I-FGSM is chosen to be the reference attack while the perturbation is performed in the feature space.

However, the procedure of ILA only provides a rough approximation to the BTD, as the baseline I-FGSM essentially differs greatly to the ideal $\boldsymbol{x}'$ which gives the best BTD. It suggests rooms for improvement in augmenting the baselines I-FGSM so that it resembles better to BTD before or during ILA, so that a more transferable attack can be generated via the projection loss.

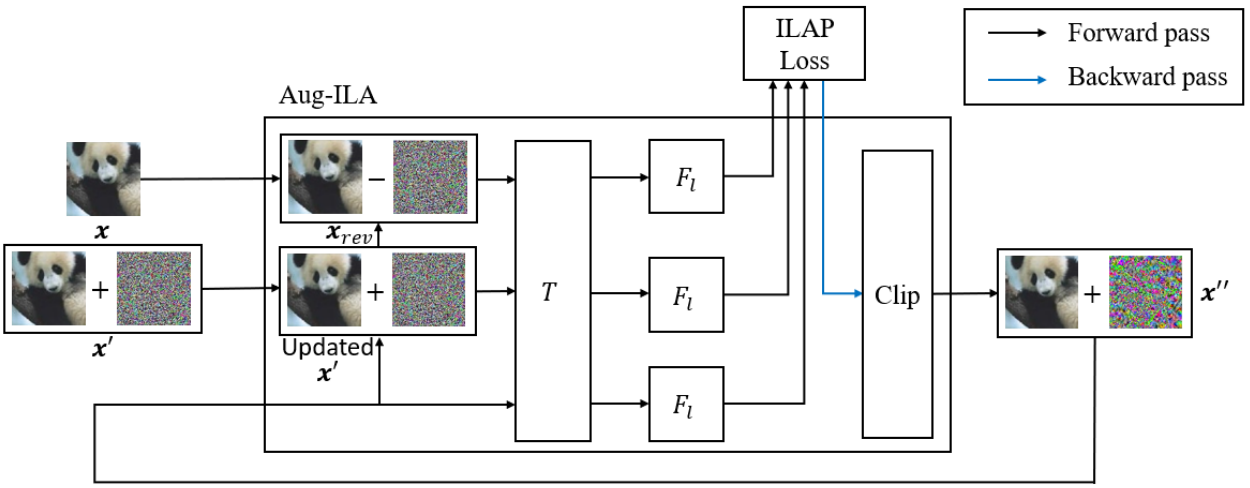


Figure 3.1: A flowchart showing the processes performed in Aug-ILA

3.1 Augmentation to Intermediate Level Attack

Based on what we have observed from the previous works, adversarial attacks tend to exhibit higher transferability when more references are involved. The increase of references can be achieved by data augmentation and reusing the temporal information over multiple iterations. From Equation (2.7), all three ILA input references $\boldsymbol{x}$, $\boldsymbol{x}'$ and $\boldsymbol{x}''$ are images, suggesting the possibility of applying image augmentation like that for representation learning to improve model generalization.

One natural way to augment the images is to apply image transformation to the input before retrieving the intermediate feature maps. Consequently, the function F_l can be substituted by

$$F_l'(\boldsymbol{x}) = F_l(T(\boldsymbol{x})) \tag{3.2}$$

where T can be any image transformation function. However, unlike attacks that enforce a wrong prediction at the logit output, ILA aims to maximize the projection of feature map discrepancies. This requires perfect pixel alignment of $\boldsymbol{x}$, $\boldsymbol{x}'$ and $\boldsymbol{x}''$, as a tiny shift in the image pixel induces misalignment in the feature map. Spatial transformations such as translation, cropping and rotation are required to be applied identically to all three ILA references. Otherwise, the optimization of ILA ends up being trivial in the sense of distorted feature discrepancies, since the feature discrepancies

cannot be properly maximized via the loss function terms $F_l(\boldsymbol{x}') - F_l(\boldsymbol{x})$ and $F_l(\boldsymbol{x}'') - F_l(\boldsymbol{x})$. Such claim is empirically verified in later experiments (Figure 4.1), where the transformation T is not identical for the images.

Different from augmentation that generalizes model representation, our task is to improve attack transferability, which can be interpreted as generalizing an adversarial example to succeed in attacking more models. Hence, apart from the common image processing methods, we may also explore any transformation associated with the adversarial perturbation. Intuitively, augmentation based on the adversarial perturbation can be viewed as an evaluation of multiple attacks before fine-tuning the final attack. This kind of pixel-wise augmentation can be applied to the images independently without conflicting with most spatial transformations. Algorithm 1 depicts the algorithmic details for applying these transformations together to ILA, where the clip function is to ensure that the first argument is within the range of the second and third arguments. Figure 3.1 is a detailed illustration on the augmentations performed to the inputs.

Algorithm 1 Aug-ILA

Require: $\boldsymbol{x}, \boldsymbol{x}', F_l$, perturbation budget ϵ, step size a, number of iterations n, a sequence of transformation functions T

$\boldsymbol{x}'' \leftarrow \boldsymbol{x}'$

for $i \leftarrow 1$ to n **do**

$\boldsymbol{x}_{\text{rev}} \leftarrow 2\boldsymbol{x} - \boldsymbol{x}'$ ▷ reverse adversarial update

Randomly initialize T ▷ to ensure the image augmentations are aligned

$\Delta \boldsymbol{y}_l' \leftarrow F_l(T(\boldsymbol{x}')) - F_l(T(\boldsymbol{x}_{\text{rev}}))$

$\Delta \boldsymbol{y}_l'' \leftarrow F_l(T(\boldsymbol{x}'')) - F_l(T(\boldsymbol{x}_{\text{rev}}))$

$\boldsymbol{x}'' \leftarrow \boldsymbol{x}'' - a \ \text{sign}(\nabla_{\boldsymbol{x}''}(-\Delta \boldsymbol{y}_l'' \cdot \Delta \boldsymbol{y}_l'))$

$\boldsymbol{x}'' \leftarrow \text{clip}(\boldsymbol{x}'', \boldsymbol{x} - \epsilon, \boldsymbol{x} + \epsilon)$

$\boldsymbol{x}'' \leftarrow \text{clip}(\boldsymbol{x}'', 0, 1)$

$\alpha \leftarrow \frac{\|\Delta \boldsymbol{y}_l''\|_2}{\|\Delta \boldsymbol{y}_l''\|_2 + \|\Delta \boldsymbol{y}_l'\|_2}$

$\boldsymbol{x}' \leftarrow \alpha \boldsymbol{x}'' + (1 - \alpha)\boldsymbol{x}'$ ▷ attack interpolation

end for

return $\boldsymbol{x}''$

3.2 Reverse Adversarial Update on the Clean Example

Let us reconsider Equation (2.8). To perform ILA, we need to obtain the feature map discrepancies between the reference attack $\boldsymbol{x}'$ and the unperturbed image $\boldsymbol{x}$. Although $\boldsymbol{x}$ is correctly classified

by the model, there is no information on how confident the model is and how well the input can represent the ground-truth class. In contrast, if $\boldsymbol{x}$ is not only correctly classified, but it is also done with a high confidence, the discrepancy of the feature maps is expected to intrinsically reflect more useful information about the attack.

In order to boost the confidence of $\boldsymbol{x}$, we propose to add a negative perturbation to it, which is exactly opposite to applying an adversarial attack. If we consider the entire model, this operation is expected to decrease the loss of the classifier, making the classification task easier by emphasizing features that are more significant for the task. Thus, we also expect such input to highly activate the intermediate layers, providing a better guidance for image update in ILA. Our preliminary study on the effect of reverse adversarial update can be found in Appendix B, which verifies the benign behavior brought by the reverse adversarial update. With the reverse adversarial perturbation, as expected, the model gives a lower loss and higher confidence on the correct class.

Instead of crafting a new adversarial attack which would incur extra computational demand, we can extract the perturbation from the existing reference attack $\boldsymbol{x}'$, turning the transformation into

$$T_{\text{adv}}(\boldsymbol{x}) = \boldsymbol{x} - (\boldsymbol{x}' - \boldsymbol{x}) = 2\boldsymbol{x} - \boldsymbol{x}' \tag{3.3}$$

The idea of such an update is similar to DeepDream [45], in which the image is updated towards the models' interpretation of the target class. After the transformation, we look for the feature discrepancies caused by a bi-directional perturbation $\pm\epsilon \operatorname{sign}(\nabla_{\boldsymbol{x}} J(\theta, \boldsymbol{x}, y))$, with the positive side being the adversarial attack and the negative side being the augmentation.

3.3 Attack Interpolation on the Reference Attack

Besides the clean image $\boldsymbol{x}$, the reference attack $\boldsymbol{x}'$ is another factor to remain unchanged throughout all the ILA iterations. Given the image transformation in Equation (3.2), it is unlikely to work with another image transformation unaligned with the other images. Similar to Section 3.2, we instead exploit the adversarial noise to be the augmentation.

According to the empirical results in Huang et al. [27], the stronger the reference attack is, the better ILA performs. This inspires us to strengthen the reference attack during the iterations of

ILA, by interpolating the reference attack itself with a stronger attack. The output of the previous iteration, $\boldsymbol{x}''$, is a good candidate for a strong attack. At iteration t, we set

$$\boldsymbol{x}'_{t+1} = \alpha \boldsymbol{x}''_t + (1-\alpha)\boldsymbol{x}'_t \tag{3.4}$$

where α is a weighting factor in the range $[0, 1]$ to control the proportion of the two attacks. This augmentation is similar to mixup [86], except that we are mixing two adversarial examples, rather than two images from different classes. By interpolating two attacks, we can strengthen the transferability of the reference attack while preserving its original characteristics.

Based on our preliminary findings, setting α to non-trivial values such as 0.5 suffices to yield satisfactory performance. However, a constant value lacks consideration of the behavior of the two attacks. Precisely, we hope to perform an adaptive interpolation depending on the effectiveness of $\boldsymbol{x}''_t$. If $\boldsymbol{x}''_t$ is weak after a single update, α should bias towards preserving the reference attack more than mixing with the new attack. Since ILA focuses on maximizing the linear projection between feature map discrepancies, the norm of the feature map discrepancy can be a good indicator of the performance. Consequently, we set

$$\alpha = \frac{\|\Delta \boldsymbol{y}''_l\|_2}{\|\Delta \boldsymbol{y}''_l\|_2 + \|\Delta \boldsymbol{y}'_l\|_2} \tag{3.5}$$

If we neglect the transformation in Section 3.2 for simplicity, $\Delta \boldsymbol{y}'_l$ and $\Delta \boldsymbol{y}''_l$ will remain the same as what we obtained from Equations (2.8) and (2.9), respectively. Note that the value of α is recomputed in each iteration before interpolation is applied.

A similar approach of reusing previous attacks in the temporal domain can be applied to the reverse adversarial update in Section 3.2. The reference attack $\boldsymbol{x}'$ in Equation (3.3) can be further extended to a collection of $\boldsymbol{x}'_t$ for every iteration t. This is not the only work to consider the temporary values of the reference attack before its convergence. ILA++ [36] also collects all losses computed during the generation of the reference attack, achieving superiority over the standard ILA. Different from [36] which uses past losses (with respect to $\boldsymbol{x}'$) to help update $\boldsymbol{x}''_t$, we make use of the past $\boldsymbol{x}'_t$ directly as an augmentation to enrich the information regarding feature discrepancies under different attacks.

CHAPTER 4

THE EFFECT OF AUG-ILA AS A TRANSFER-BASED BLACK-BOX ATTACK

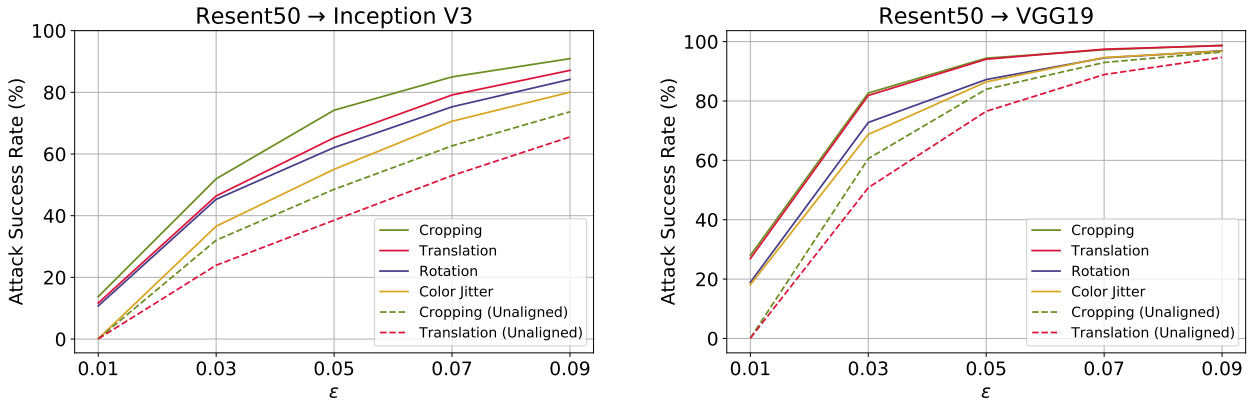
In this section, we start by comparing the effect of different image transformations. After combining the transformations that benefit the most, we evaluate the attack transferability of Aug-ILA in comparison with previous methods based on fine-tuning as well as state-of-the-art transfer-based attack generation methods. Finally, we investigate the selection of parameters and identify influential factors to an attack's transferability.

Setup. Following the line of work in transfer-based attacks, we use a total of nine classification models to evaluate attack transferability, including ResNet50 [23], VGG19 [61], Inception V3 (Inc. V3) [63], Wide ResNet50 2x (WRN) [84], DenseNet161 (DenseNet) [26], ResNeXt101 (ResNeXt) [78], MobileNet V2 (MobileNet) [58], PNASNet5 (PNASNet) [40] and SENet50 (SENet) [25]. Among these models, we use ResNet50, VGG19 and Inception V3 as the source models of the attack. All of the models are pretrained on ImageNet [57], with the model parameters of PNASNet[1] and SENet[2] obtained from public repositories and the remaining from Torchvision [51]. For the choice of the intermediate layer, we opt the layer 3-1 for ResNet50, layer 9 for VGG19, and layer 6a for Inception V3, where the former two have been shown to result in good performance by [36]. Due to the ineffectiveness of transfer-based targeted attacks [41], we mainly consider untargeted attacks with ϵ 0.05 ($\approx 13/255$) and 0.03 ($\approx 8/255$), under the ℓ_∞-norm constraint.

To measure the attack success rate, we randomly sample 5000 images from the ILSVRC2012 validation set with all images being classified correctly by the nine models. That means, the original test accuracy before the attack is 100% for every model. The default number of iterations of I-FGSM is 10 and the attack step size (learning rate) is set to $\max(\frac{1}{255}, \frac{\epsilon}{\text{no. of iterations}})$. To mount a

[1] https://github.com/Cadene/pretrained-models.pytorch

[2] https://github.com/moskomule/senet.pytorch



(a) Transferred from ResNet50 to Inception V3

(b) Transferred from ResNet50 to VGG19

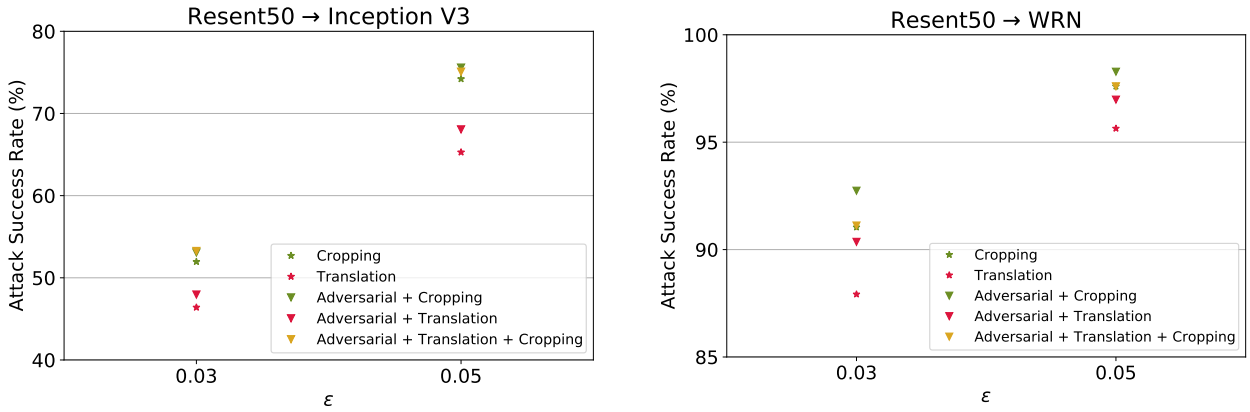
Figure 4.1: Attack success rates of applying translation, cropping, rotation and color jittering as augmentation in ILA.

complete attack, we first run I-FGSM for 10 iterations on the source model, and then pass the example as the reference attack to Aug-ILA for fine-tuning. We use the same model to be the source model of I-FGSM and Aug-ILA. We run Aug-ILA and other fine-tuning methods for 50 iterations in all the experiments.

4.1 Effectiveness of Common Image Augmentation Operations

We select four image transformation operations that are commonly used, including translation, cropping, rotation and color jittering. To test our claim regarding the pixel misalignment causing poor transferability in ILA, we also implement an unaligned version of translation and cropping, such that the transformations with different randomized values are applied to each of the references $\boldsymbol{x}$, $\boldsymbol{x}'$ and $\boldsymbol{x}''$. We then report the transfer success rates on Inception V3 and VGG19 over varying perturbation budget ϵ. The comparison of different transformations is shown in Figure 4.1.

From the result, translation and cropping yield more transferable attacks than the other transformations, especially when the perturbation budget ϵ is small. Under a critical constraint when $\epsilon = 0.01$ ($\approx 3/255$), augmentations without proper alignment fail to transfer at all, with the attack success rate very close to 0. Between translation and cropping, cropping consistently gives better performance. A further evaluation of attack transferability after the incorporation of reverse



(a) Transferred from ResNet50 to Inception V3

(b) Transferred from ResNet50 to WRN

Figure 4.2: Attack success rates of different augmentation, in conjunction with adversarial reverse update (denoted as adversarial). The performance of simple augmentation without reverse adversarial update is also included as baseline.

adversarial update (denoted as 'adversarial') is reported in Figure 4.2, demonstrating that further incorporation of adversarial reverse update leads to higher transferability. Among different combinations, we find that 'adversarial + cropping' stably gives outstanding results, while further complement of translation degrades the transferability. Hence, we opt for random cropping together with reverse adversarial update as the default augmentation of Aug-ILA.

4.2 Black-box Transferability of Aug-ILA

Our previous experiments only evaluate the transformation function T, without interpolation on the reference attack. In this section, we enable all augmentations discussed previously, forming the complete Aug-ILA. We first demonstrate the superiority of Aug-ILA over ILA [27] and ILA++ [36]. We use ResNet50 as the source model and choose layer 3-1 as the intermediate layer for all three fine-tuned attacks. The result is shown in Table 4.1, while the comparison of attacks generated from VGG19 and Inception V3 can be found in Figure 4.2. Moreover, additional experiments with single-step attack (FGSM) are shown in Figure 4.3.

In Table 4.1, we can see that Aug-ILA outperforms both ILA and ILA++ for all the models except the source model (ResNet50). The apparent improvement in attack transferability reflects

Table 4.1: Attack success rates of ImageNet adversarial examples on different models, generated from ResNet50 in the untargeted setting.

ϵ	Method	ResNet50*	Inc. v3	WRN	VGG19	PNASNet
13/255	I-FGSM	**100.0%**	25.80%	72.12%	48.34%	29.62%
	I-FGSM + ILA	**100.0%**	51.50%	93.20%	86.60%	65.20%
	I-FGSM + ILA++	99.98%	63.68%	96.36%	90.57%	70.34%
	I-FGSM + Aug-ILA (Ours)	99.78%	**90.02%**	**99.04%**	**98.76%**	**93.18%**
	VMI-CT-FGSM	99.96%	69.66%	91.04%	81.78%	73.38%
	I-FGSM + LinBP + SGM + ILA	**100.0%**	72.46%	98.88%	96.82%	81.82%
8/255	I-FGSM	99.96%	14.88%	41.28%	26.36%	17.40%
	I-FGSM + ILA	99.98%	34.56%	79.88%	66.72%	43.32%
	I-FGSM + ILA++	99.98%	41.44%	87.14%	75.24%	49.18%
	I-FGSM + Aug-ILA (Ours)	99.42%	**66.02%**	**93.92%**	**92.90%**	**75.76%**
	VMI-CT-FGSM	98.28%	47.42%	69.94%	56.46%	48.08%
	I-FGSM + LinBP + SGM + ILA	**100.0%**	42.08%	90.82%	84.08%	53.90%

ϵ	Method	DenseNet	ResNeXt	MobileNet	SENet	Average
13/255	I-FGSM	57.80%	70.90%	54.56%	71.92%	59.01%
	I-FGSM + ILA	85.50%	91.00%	84.00%	91.90%	83.21%
	I-FGSM + ILA++	92.32%	92.30%	89.34%	96.28%	87.90%
	I-FGSM + Aug-ILA (Ours)	**98.40%**	**96.90%**	**98.12%**	98.72%	**96.99%**
	VMI-CT-FGSM	88.50%	82.22%	82.82%	92.64%	84.67%
	I-FGSM + LinBP + SGM + ILA	97.64%	95.96%	95.60%	**98.96%**	93.13%
8/255	I-FGSM	31.42%	41.82%	31.88%	44.00%	38.78%
	I-FGSM + ILA	69.16%	78.48%	67.60%	80.16%	68.87%
	I-FGSM + ILA++	79.08%	78.46%	75.28%	87.64%	74.83%
	I-FGSM + Aug-ILA (Ours)	**91.04%**	**87.76%**	**90.38%**	**93.40%**	**87.84%**
	VMI-CT-FGSM	66.06%	58.02%	60.22%	72.84%	64.15%
	I-FGSM + LinBP + SGM + ILA	85.24%	80.58%	80.56%	92.64%	78.88%

* The source model used to generate the attack.

Table 4.2: Attack success rates of ImageNet adversarial examples on different models, generated from VGG19 in the untargeted setting.

ϵ	Method	ResNet50	Inc. v3	WRN	VGG19*	PNASNet
13/255	I-FGSM + ILA	77.32%	45.42%	74.78%	**99.40%**	70.30%
	I-FGSM + ILA++	79.56%	47.08%	76.04%	99.30%	70.76%
	I-FGSM + Aug-ILA (Ours)	**87.36%**	**66.40%**	**87.00%**	99.02%	**82.34%**
8/255	I-FGSM + ILA	53.78%	25.94%	49.54%	**99.34%**	46.30%
	I-FGSM + ILA++	57.88%	27.94%	52.94%	99.30%	49.48%
	I-FGSM + Aug-ILA (Ours)	**66.22%**	**40.82%**	**64.90%**	97.82%	**60.24%**
ϵ	Method	DenseNet	ResNeXt	MobileNet	SENet	Average
13/255	I-FGSM + ILA	67.38%	61.68%	85.34%	72.90%	72.72%
	I-FGSM + ILA++	68.66%	63.56%	85.60%	74.46%	73.89%
	I-FGSM + Aug-ILA (Ours)	**81.72%**	**80.30%**	**93.06%**	**85.84%**	**84.78%**
8/255	I-FGSM + ILA	40.12%	36.26%	66.52%	49.42%	51.91%
	I-FGSM + ILA++	44.88%	39.70%	68.70%	53.62%	54.94%
	I-FGSM + Aug-ILA (Ours)	**57.38%**	**54.02%**	**77.58%**	**63.32%**	**64.70%**

* The source model used to generate the attack.

the effectiveness of augmentation with respect to both image transformation and adversarial transformation. Another observation worth mentioning is that the attack difficulty increases according to how much the model architectures differ. For example, attacks generated from ResNet (with skip connections) tend to transfer better to models also with skip connections, such as WRN, DenseNet, ResNeXt, comparing to others such as Inception V3 and PNASNet. However, the improvement in transferability brought by Aug-ILA is more significant under such model dissimilarity, as reflected in the large gap of attack success rates between Aug-ILA and the baselines for Inception V3 and PNASNet. We believe such a phenomenon can be attributed to the extensive augmentations that enable Aug-ILA to exploit gradients with better resemblance to different architectures.

To show the generalization capability of Aug-ILA, we conduct experiments with two more datasets, namely CIFAR-10 and CIFAR-100 [31]. Partly following [36], we consider four models: VGG19 with batch normalization [61], Wide ResNet-28-10 (WRN) [84], ResNeXt-29 (ResNeXt) [78] and DenseNet-190 (DenseNet) [26]. The model parameters are obtained from a public repository[3]. For both datasets, we randomly sample 3000 images from the test set that are classified

[3] `https://github.com/bearpaw/pytorch-classification`

Table 4.3: Attack success rates of ImageNet adversarial examples with single-step FGSM as the reference attack, generated from ResNet50 in the untargeted setting.

ϵ	Method	ResNet50*	Inc. v3	WRN	VGG19	PNASNet
13/255	FGSM	80.86%	28.78%	37.08%	46.24%	36.04%
	FGSM + ILA	93.36%	28.56%	51.94%	58.22%	38.48%
	FGSM + ILA++	**97.88%**	43.14%	74.88%	65.92%	44.04%
	FGSM + Aug-ILA (Ours)	96.98%	**57.48%**	**87.20%**	**91.76%**	**67.40%**
8/255	FGSM	84.68%	20.44%	29.00%	30.96%	23.52%
	FGSM + ILA	88.60%	19.48%	38.02%	39.50%	24.04%
	FGSM + ILA++	**97.02%**	29.82%	60.02%	50.64%	30.60%
	FGSM + Aug-ILA (Ours)	89.72%	**32.46%**	**65.44%**	**72.60%**	**40.08%**

ϵ	Method	DenseNet	ResNeXt	MobileNet	SENet	Average
13/255	FGSM	32.48%	30.72%	45.32%	41.42%	42.10%
	FGSM + ILA	39.10%	40.12%	52.42%	52.58%	50.53%
	FGSM + ILA++	65.80%	65.48%	66.12%	74.12%	66.38%
	FGSM + Aug-ILA (Ours)	**79.48%**	**74.20%**	**84.18%**	**83.70%**	**80.26%**
8/255	FGSM	25.10%	22.38%	29.86%	31.70%	33.07%
	FGSM + ILA	28.50%	29.22%	35.76%	39.56%	38.08%
	FGSM + ILA++	50.54%	**50.62%**	52.66%	61.36%	53.70%
	FGSM + Aug-ILA (Ours)	**55.68%**	48.22%	**58.74%**	**62.56%**	**58.39%**

* The source model used to generate the attack.

Table 4.4: Attack success rates of CIFAR-10 adversarial examples on different models, generated from VGG19 in the untargeted setting.

ϵ	Method	VGG19*	WRN	ResNeXt	DenseNet	Average
13/255	I-FGSM	99.54%	67.91%	67.91%	64.00%	74.84%
	I-FGSM + ILA	99.54%	97.85%	98.19%	97.08%	98.16%
	I-FGSM + ILA++	**99.57%**	98.00%	98.17%	97.40%	98.29%
	I-FGSM + Aug-ILA	98.77%	**98.22%**	**98.46%**	**98.06%**	**98.38%**
8/255	I-FGSM	99.32%	55.27%	55.43%	51.64%	51.64%
	I-FGSM + ILA	**99.45%**	88.26%	89.61%	87.15%	91.12%
	I-FGSM + ILA++	99.13%	89.67%	90.13%	87.40%	91.58%
	I-FGSM + Aug-ILA	98.31%	**93.05%**	**93.42%**	**91.45%**	**94.06%**

* The source model used to generate the attack.

Table 4.5: Attack success rates of CIFAR-100 adversarial examples on different models, generated from VGG19 in the untargeted setting.

ϵ	Method	VGG19*	WRN	ResNeXt	DenseNet	Average
13/255	I-FGSM	**99.45%**	48.15%	39.45%	42.41%	57.37%
	I-FGSM + ILA	99.03%	92.73%	88.97%	88.63%	92.34%
	I-FGSM + ILA++	99.10%	**93.13%**	89.53%	89.70%	**92.87%**
	I-FGSM + Aug-ILA	96.08%	92.65%	**90.56%**	**90.69%**	92.50%
8/255	I-FGSM	98.84%	38.91%	31.94%	33.77%	50.87%
	I-FGSM + ILA	**98.91%**	80.77%	72.71%	73.93%	81.58%
	I-FGSM + ILA++	98.53%	81.23%	74.77%	74.60%	82.28%
	I-FGSM + Aug-ILA	95.22%	**84.37%**	**78.33%**	**78.56%**	**84.12%**

* The source model used to generate the attack.

correctly by all four models and we pick VGG19 to be the source model. The experimental results using CIFAR-10 and CIFAR-100 are shown in Table 4.4 and Table 4.5 respectively.

For both CIFAR-10 and CIFAR-100, the attack success rates of the baselines are much higher than that of ImageNet. Although Aug-ILA mostly attains the highest attack success rate among the baselines, we observe that the degree of improvement is not as high as that in Table 4.1. It may be because of the tiny size and low resolution (32×32 for both CIFAR-10 and CIFAR-100) of the images, leading to a reduction in the effect of data augmentation such as random cropping.

The supremacy of Aug-ILA is not limited to methods based on fine-tuning, but also other state-

of-the-art transfer-based attacks. Combination of the Composite Transformation Method (CTM) and variance tuning [70], dubbed VMI-CT-FGSM (or VMI-SI-TI-DI-FGSM more detailedly), exhibits remarkably high transferability among the state-of-the-art methods. On the other hand, Guo et al. [21] found that LinBP works favorably together with SGM [73] and ILA, with their incorporation further advancing transferability to a new level. We reproduce the conjunction of these works, with the hyper-parameters specified in Appendix A. Under our choices of model architecture, VMI-CT-FGSM does not work well to transfer between different model architectures, despite its remarkable performance in attacking defended models. Among the baselines, I-FGSM + LinBP + SGM + ILA achieves the best transferability between undefended models, while it is still slightly outperformed by Aug-ILA.

4.3 Effect of Aug-ILA on Defended Models

In this section, we evaluate the effect of Aug-ILA on defended models. We consider three robust models using ensemble adversarial training [66], including an ensemble of 3 adversarially trained Inception V3 (Inc-v3$_{ens3}$), an ensemble of 4 adversarially trained Inception V3 (Inc-v3$_{ens4}$), and an ensemble of 3 adversarially trained Inception-ResNet-v2 (IncRes-v2$_{ens}$). On top of that, we also test the attacks against three defenses: HGD [38], R&P [76] and NIPS-r3[4], which are the top 3 defenses in the NIPS 2017 adversarial competition. For all three defenses, we adopt the default models and hyper-parameters used in their corresponding repositories.

We ported the implementation of Aug-ILA into the same experimental setup used in previous works [14, 15, 70], which includes 1000 sampled test images and the pretrained parameters of the models. Then we compare the baselines, including I-FGSM, MI-FGSM, MI-CT-FGSM, NI-CT-FGSM, and VNI-CT-FGSM, with ILA and Aug-ILA inserted. We pick Inception V3 as the source model, and all attacks are performed with perturbation size $\epsilon = 16/255 (\approx 0.0627)$. For ILA and Aug-ILA, we finetune the attack for 50 iterations by choosing the layer 6a in Inc-V3$_{adv}$, an adversarially trained Inception V3 model. The attack success rate against each of the defenses is reported in Table 4.6. From preliminary results, ILA cannot improve the attack success rate on the

[4] https://github.com/anlthms/nips-2017/tree/master/mmd

defended models, so it is removed in the experiments with stronger attacks. Nevertheless, Aug-ILA is able to improve the attack success rate of most of the attacks, except VNI-CT-FGSM.

Table 4.6: Attack success rates of ImageNet adversarial examples, generated from Inception V3 in the untargeted setting.

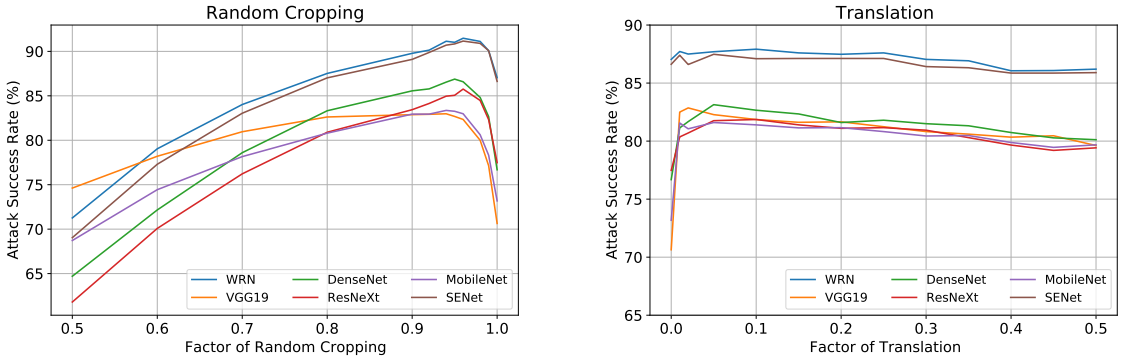
Attack	Inc-v3$_{ens3}$	Inc-v3$_{ens4}$	IncRes-v2$_{ens}$	HGD	R&P	NIPS-r3
I-FGSM	12.1%	10.9%	5.80%	2.70%	4.00%	8.30%
I-FGSM + ILA	10.1%	10.6%	5.00%	0.10%	2.50%	7.90%
I-FGSM + Aug-ILA	43.0%	33.4%	24.0%	35.1%	23.6%	14.3%
MI-FGSM	14.1%	13.0%	6.60%	4.60%	5.00%	8.30%
MI-FGSM + ILA	14.1%	12.5%	6.40%	0.70%	3.60%	5.40%
MI-FGSM + Aug-ILA	44.2%	36.3%	24.6%	33.6%	25.3%	28.5%
MI-CT-FGSM	65.5%	62.1%	45.5%	56.6%	44.5%	52.5%
MI-CT-FGSM + Aug-ILA	69.0%	65.8%	49.8%	62.1%	51.6%	65.0%
NI-CT-FGSM	58.8%	54.4%	40.0%	49.2%	38.0%	46.1%
NI-CT-FGSM + Aug-ILA	66.8%	62.4%	44.5%	55.8%	46.5%	52.8%
VNI-CT-FGSM	79.1%	77.4%	65.3%	72.7%	63.5%	70.8%
VNI-CT-FGSM + Aug-ILA	75.6%	73.2%	58.3%	68.4%	81.6%	64.0%

4.4 Effect of Different Hyper-parameters

In the previous sections, translation and random cropping are found to be the two most effective image augmentations to be applied to the ILA references. We hope to evaluate the best values for the shifting proportion and cropping size. We test different values of the shifting factor between 0 to 0.5 inclusively for translation. Similarly, we also test with the size factor of random cropping between 0.5 and 1.0 inclusively. The experiments for the augmentation factors are reported in Figure 4.3. The result indicates that proper hyper-parameters for the transformations contribute greatly to attack transferability, with the best value of 0.05 for translation and 0.95 for random cropping.

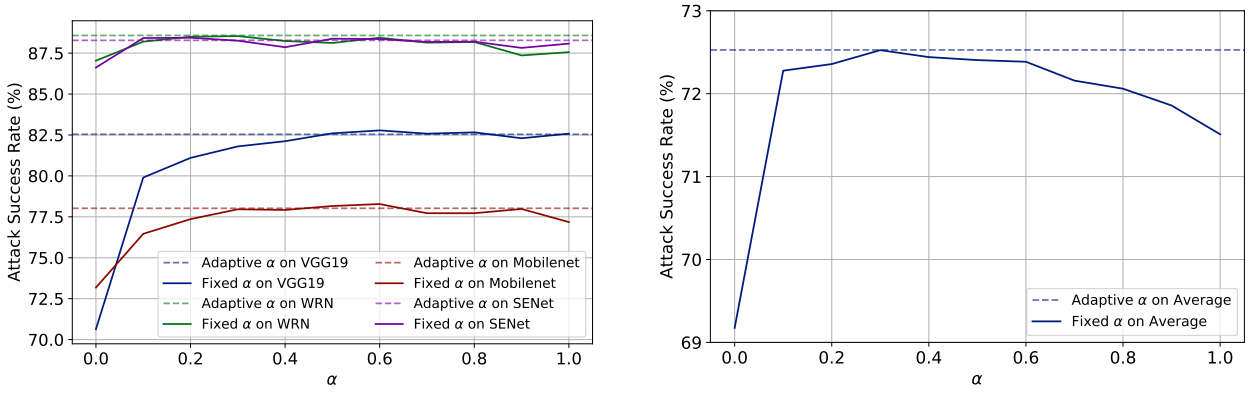
Next, we study the effect of α used for attack interpolation, with fixed values between 0 and 1, in addition to the adaptive choice computed from the norm of the feature map discrepancy. The result is shown in Figure 4.6. Although a non-trivial value of α suffices to improve the performance,

the adaptive selection using Equation (3.5) results in optimal or sub-optimal attack transferability in most cases.



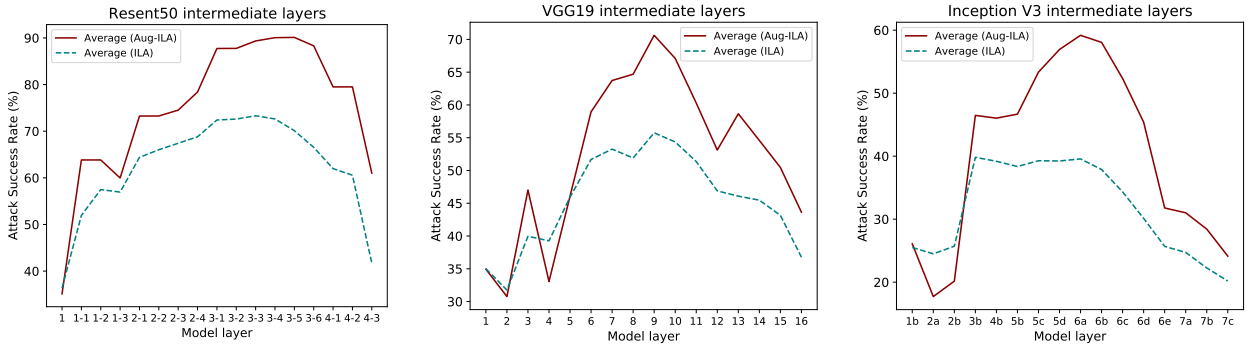
(a) Random cropping **(b)** Translation

Figure 4.3: Attack success rates of different hyper-parameters for random cropping and translation, using ResNet50 as the source model with $\epsilon = 0.03 (\approx 8/255)$.



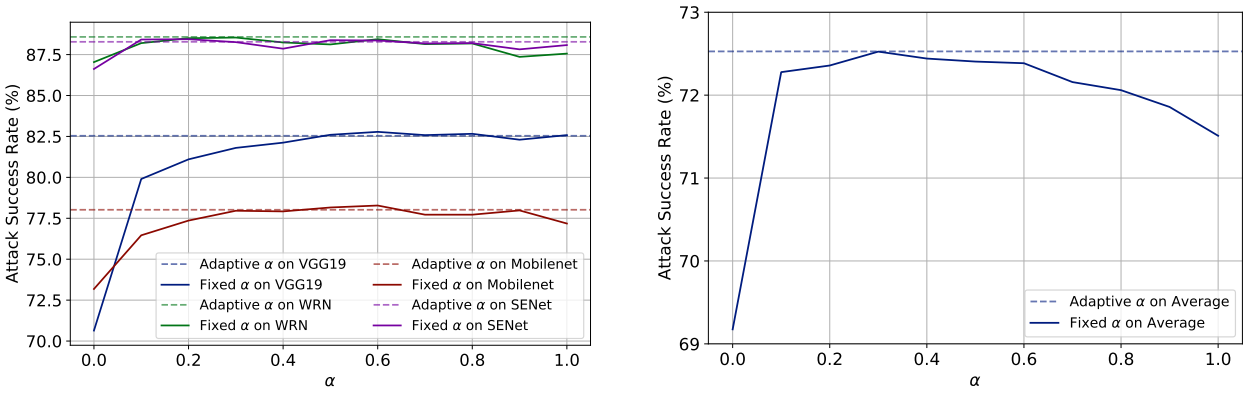
(a) VGG19, WRN, MobileNet and SENet **(b)** Average of the 8 models (excluding the source model)

Figure 4.4: Attack success rates over different values of α, using ResNet50 as the source model with $\epsilon = 0.03$.



(a) ResNet50 **(b)** VGG19 **(c)** Inception V3

Figure 4.5: Attack transferability over different choices of the intermediate layer for ResNet50, VGG19 and Inception V3. The dotted curve shows the performance of ILA without augmentation. '3-1' in ResNet50 refers to the first residual block of the third meta-block. The naming of layers of Inception V3 follows the PyTorch convention.



(a) VGG19, WRN, MobileNet and SENet **(b)** Average of the 8 models (excluding the source model)

Figure 4.6: Attack success rates over different values of α, using ResNet50 as the source model with $\epsilon = 0.03$.

Another important factor is the intermediate layer where ILA is performed. We apply ILA and Aug-ILA at each layer of the source models, and plot the attack success rates in Figure 4.5. The overall fluctuation pattern is similar, but Aug-ILA results in peaks that are more protruded, by virtue of the various augmentations. However, we also observe that for shallow layers, Aug-ILA turns out to have limited performance, sometimes even inferior to its baseline. While [27] interpreted the tendency above in terms of the linearity of the decision boundary, we offer our explanation from

the perspective of intermediate feature perturbation.

4.5 Roles of Intermediate Layer Perturbation and Augmentation

Transfer-based attacks can be viewed as using one model's gradient to estimate other models' gradients. However, it was shown that the gradient directions of different models are almost orthogonal to each other [41]. Locally, each layer of the target model can cause a *weakening effect* to the attack, assembling to lead to the gradient orthogonality. The deeper the attack goes, the weaker its influence becomes. For an attack on the intermediate feature, since it transits through fewer layers, both forward and backward, it is weakened to a lesser degree. One problem is, if the attack only perturbs the intermediate feature without considering the remaining layers, it is also less capable of altering the model decision, getting demoted to an essentially weaker attack. Therefore, in the choice of the ILA layer, one that is either too shallow or too deep decays the attack transferability.

With the same insight, three factors can be identified for black-box transferability:

1. Perturbation strength
2. Similarity between the predicted gradient and the actual gradient
3. Difference in architecture between the source model and the target model.

The first two factors increase transferability while the last one hinders the attack to transfer. Factor 1 is determined by the perturbation budget, but it can be improved implicitly by ILA, such that the intermediate perturbation is also strengthened. However, factor 1 does not dominate the attack transferability, since the victim model does not necessarily have any resembling architecture to the source model. Such difference, which is factor 3, induces the *weakening effect* on the attack. We will show that Aug-ILA accomplishes a good balance between factors 1 and 2, leading to a less severe *weakening effect* from factor 3.

Since ILA maximizes the projection of feature discrepancies, the norm of the feature discrepancy should also be larger (factor 1). We visualize the L2 norm of $F_l(\boldsymbol{x}'') - F_l(\boldsymbol{x})$ at different layers in ResNet50 and compare with different attacks in Figure 4.7. Also, we estimate factor 2 by computing the cosine distance between the model gradient suggested by the attack ($\boldsymbol{x}'' - \boldsymbol{x}$) and

the gradient direction of the clean image ($\nabla_{\boldsymbol{x}} J(\theta, \boldsymbol{x}, y)$), with the result shown in Table 4.7.

One interesting finding is that the L2 norm of ILA is actually stronger than that of Aug-ILA in shallow layers, which agrees with the poor transferability of Aug-ILA when applied at the first few layers in Figure 4.5. However, after the selected layer in ILA, the perturbation norm of Aug-ILA surpasses ILA and the situation continues until the output of the model. In the meantime, despite the strengthened feature perturbation, ILA causes the gradient direction to be more orthogonal (worse for transferability) to the actual gradient direction of the model, comparing to I-FGSM. Aug-ILA, nevertheless, slightly improves the gradient estimation without significant loss in perturbation strength. With a stronger perturbation strength together with a better predicted model gradient, Aug-ILA facilitates its comeback for the perturbation in deeper layers, and therefore its resulting attack is more transferable than the baselines.

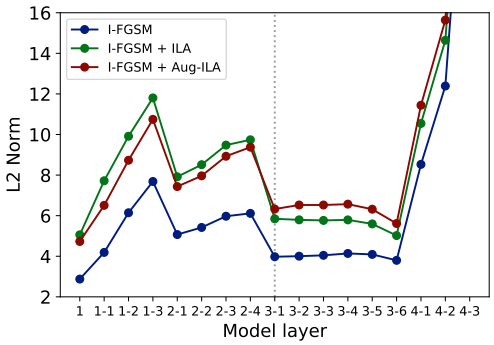


Figure 4.7: L2 norm of $F_l(\boldsymbol{x}'') - F_l(\boldsymbol{x})$ at different layers of ResNet50.

Source	Target	Cosine distance		
		I-FGSM	ILA	Aug-ILA
ResNet50	VGG19	0.0110	0.0067	0.0089
	Inception V3	0.0016	0.0010	0.0016
Inception V3	ResNet50	0.0024	0.0013	0.0017
	VGG19	0.0042	0.0029	0.0026
VGG19	ResNet50	0.0049	0.0031	0.0035
	Inception V3	0.0016	0.0013	0.0016

Table 4.7: Cosine distance between $\boldsymbol{x}'' - \boldsymbol{x}$ and $\nabla_{\boldsymbol{x}} J(\theta, \boldsymbol{x}, y)$. A larger cosine distance refers to a closer prediction of the gradient direction.

CHAPTER 5

THE EFFECT OF AUG-ILA IN FAST ADVERSARIAL TRAINING

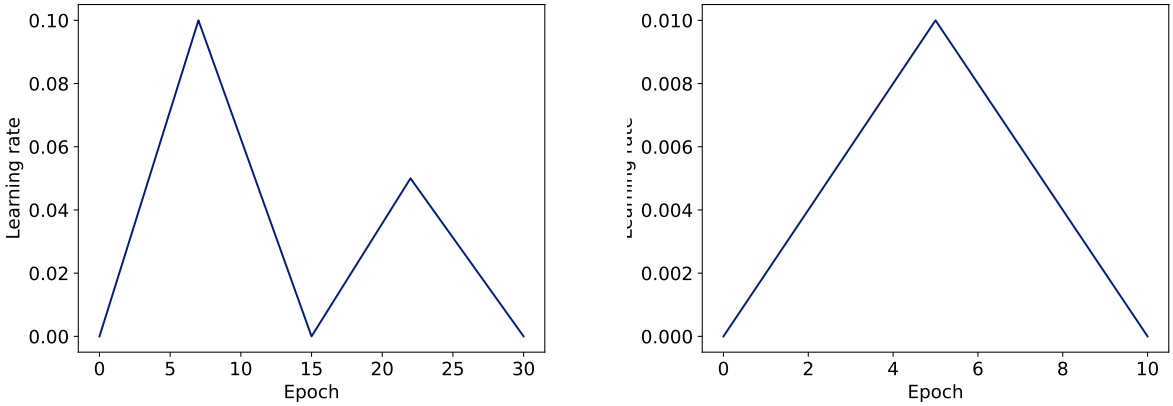
Adversarial training can be costly in terms of time due to iterative updates of the adversarial examples. In Equation (1.3), the inner maximization of the loss is where the attack is generated. Madry et al. [42] suggested using 40 and 7 iterations of PGD attack to generate the adversarial examples on MNIST and CIFAR-10 respectively. However, a 7-iteration PGD attack induces 7 extra forward and backward passes during training, resulting in an extremely slow training condition.

5.1 Setup

To make the training more efficient, we adopt the Fast [72] [1] training scheme, which replaces the computational heavy PGD with a special single-step attack: FFGSM. Different from FGSM which has no initialization, FFGSM initializes from a uniformly random perturbation within the range $[-\epsilon, \epsilon]$. With such a subtle modification, the performance of adversarial training using the new attack can be on par with that of using PGD, which greatly speeds up the training process without sacrificing robustness.

The whole training process is divided into two phases, the pre-training phase and the fine-tuning phase. In the pre-training phase, we run the Fast training scheme for 30 epochs using SGD with momentum. Cyclic learning rate, which scales the learning rate up before reaching a peak and scales down, is used as the scheduler with a peak learning rate of 0.1. A plot of the explicit learning rate over time is shown in Figure 5.1. In the fine-tuning phase, we load the pre-trained models and fine-tune the model with FFGSM + Aug-ILA for 10 more epochs, with a peak learning rate of 0.01 and the remaining hyper-parameters same as the pre-training phase. To speed up the attack

[1] We only discuss the algorithmic aspect of the framework. Thereby, hardware adaptations such as mixed precision will be neglected.



(a) Learning rate over epochs in the pre-training phase

(b) Learning rate over epochs in the fin-tuning phase

Figure 5.1: A plot of the explicit learning rate over epochs under our chosen cyclic learning rate scheduler.

generation, instead of using the iterative I-FGSM, we switch to single-step FFGSM, and refine the reference attack with an n-iteration ($n = 4$ as searched in later experiments) Aug-ILA attack. CIFAR-10 and CIFAR-100, are used as the datasets. We use the same set of hyper-parameters to train on the two datasets. The models are then evaluated against clean examples, PGD-10 attack and also AutoAttack [10] on the corresponding validation datasets. All implementations are from Torchattacks [30]. In addition to the original settings, following Gowal et al.'s recommendation [20, 54], model weight averaging [52] with $\tau = 0.999$ is applied to the training process and CutMix [83] is used as the data augmentation. We opt for ResNet18 [23] and WideResNet28-10 (WRN-28-10) [84], a 10-times wider variant of ResNet, to be our source model. For the extra model used in Aug-ILA, the two models are used interchangeably. That is, for training ResNet18, WRN-28-10 is the surrogate model and vice versa. Note that this setup is the same as Section 4.

Moreover, the training time of the models is also recorded. All experiments are performed with an Nvidia GeForce RTX3090. To retrieve the time and model performance reliably, we repeat the same settings three times with different random seeds, and report the mean and the standard deviation of the trials.

5.2 Adversarial Robustness Trained with Aug-ILA

An initial comparison of the robustness of the models is shown in Table 5.2. With the Fast framework, the speed of the training process can be boosted to converge quickly. For wide networks like WRN28-10, the training can terminate much faster than the standard adversarial training using PGD attack, while achieving comparable or even better model robustness. In contrast, standard adversarial training with multi-step PGD and other state-of-the-art methods might induce further overhead to the training time. However, one observable drawback of such a method is that the clean accuracy can be slightly decayed. A plot featuring the training time and resulting accuracy of different models is shown in Figure 5.2. As an extension to the Fast framework, fine-tuning with Aug-ILA raises the robustness and slightly worsens the accuracy, with a tiny increase in training time. Despite the trade-off between the two factors, increasing robustness is much more difficult than increasing the clean accuracy, especially without sacrificing the computational efficiency. For instance, Cutmix + DDPM [54], the work with Rank-1 robustness in RobustBench [9], takes a huge amount of synthetic data in order to have a potent improvement in robustness. Such augmentation requires almost doubling the training set and the total number of epochs, leading to an extremely inefficient adversarial training. Besides, with a better training efficiency, unexpected behaviors such as catastrophic overfitting are less likely to occur, which enables a more stable convergence during adversarial training.

Before presenting more experiments, we would like to address a fundamental question regarding the two phases:

Is it necessary to split the training into two phases? Is it possible to train the model with FFGSM + Aug-ILA from scratch?

While it is possible to train the model in one single phase, it might not be the most efficient way. As the generation of Aug-ILA requires multiple updates, repeating the attacks with a large number of epochs slows the overall training process. Alternatively, a 30-epoch pre-training using FFGSM attack can roughly make the model converge within an hour. A plot of accuracy over time of the models and a comparison of the convergence rate are shown in Figure 5.3 and Figure 5.4 respectively. From the figures, when we train the model using FFGSM + Aug-ILA (shorthanded as

Table 5.1: Accuracy comparison of CIFAR-10 trained on ResNet18 and WRN28-10.

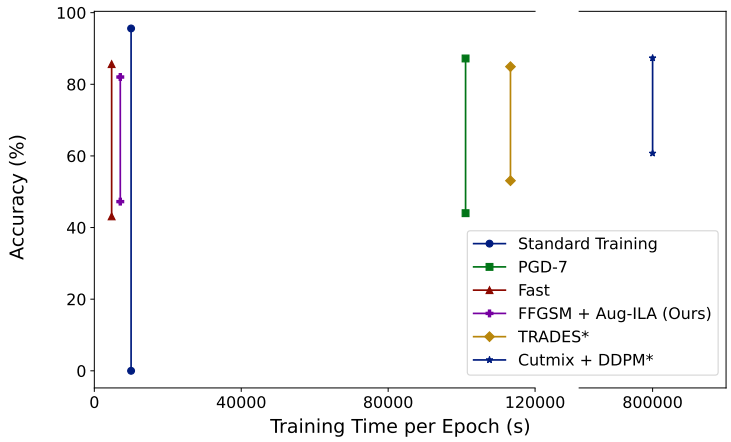
Model	Attack	Epochs	Time (s)	Accuracies		
				Clean	PGD-10	AutoAttack
ResNet18	-	120	3899±287	94.73%±0.47%	00.00%±0.00%	00.00%±0.00%
	PGD-7	120	25532±1101	84.72%±0.32%	44.76%±0.61%	42.04%±0.17%
	FFGSM	30	1868±550	81.83%±0.42%	44.44%±1.32%	40.64%±1.64%
	Aug-ILA (Ours)*	10	4113±63	78.73%±0.33%	47.18%±0.26%	42.58%±0.09%
	Aug-ILA (Ours)*#	10	4050±127	75.11%±0.55%	**49.69%**±0.30%	**43.42%**±0.79%
WRN28-10	-	120	10010±31	95.59%±0.10%	00.00%±0.00%	00.00%±0.00%
	PGD-7	120	101079±15660	87.21%±0.51%	47.42%±0.49%	44.01%±0.37%
	FFGSM	30	4714±28	85.63%±0.69%	46.68%±0.47%	43.12%±0.03%
	Aug-ILA (Ours)*	10	2406±59	84.59%±0.59%	47.63%±0.60%	43.42%±0.79%
	Aug-ILA (Ours)*#	10	2355±66	82.03%±0.07%	**53.45%**±0.20%	**47.24%**±0.20%

* fine-tuned model in phase 2.
# with Cutmix and model weight averaging

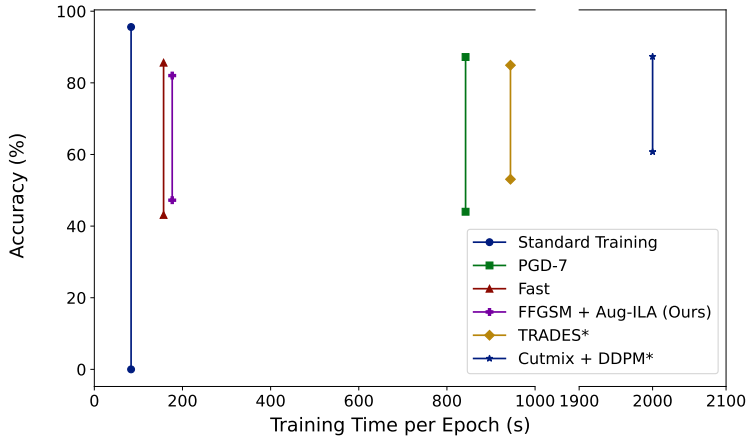
Table 5.2: Accuracy comparison of CIFAR-100 trained on ResNet18 and WRN28-10.

Model	Attack	Epochs	Time (s)	Accuracies		
				Clean	PGD-10	APGD-10
ResNet18	-	120	3937±587	74.91%±0.64%	00.00%±0.00%	00.00%±0.00%
	PGD-7	120	23893±2083	56.40%±0.66%	20.32±0.72%	19.07%±0.47%
	FFGSM	30	1700±126	57.18%±0.89%	22.42%±0.62%	21.54%±0.80%
	Aug-ILA (Ours)*#	10	4124±153	51.80%±0.16	**27.36%**±0.10%	**27.06%**±0.03%
WRN28-10	-	120	11538±2495	77.29%±2.19%	00.00%±0.00%	00.00%±0.00%
	PGD-7	120	71245±264	58.60%±2.21%	21.32%±1.24%	19.94%±1.07%
	FFGSM	30	4722±11	61.75%±0.14%	24.45%±0.13%	22.98%±0.55%
	Aug-ILA (Ours)*#	10	2397±29	57.28%±0.61%	**29.83%**±0.37%	**28.57%**±0.42%

* fine-tuned model in phase 2.
# with Cutmix and model weight averaging

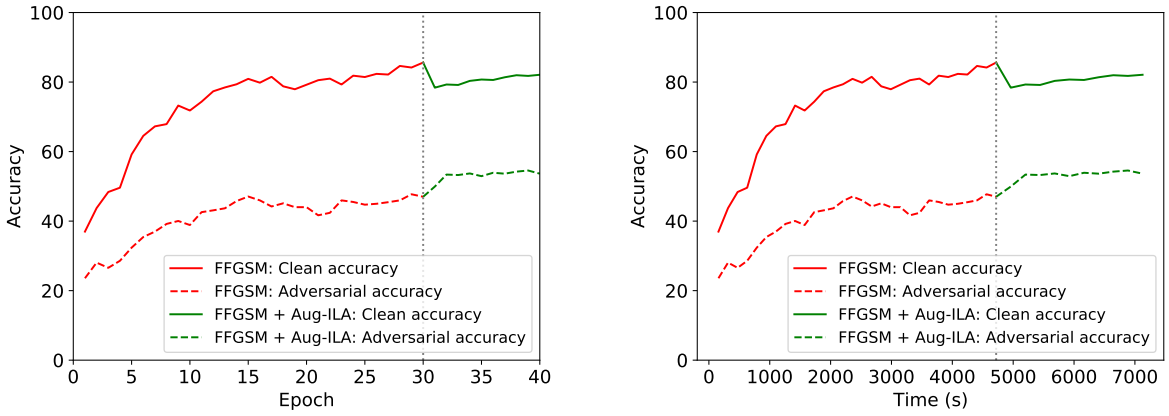


(a) Accuracy over total training time



(b) Accuracy over training time per epoch

Figure 5.2: A comparison of different methods on the accuracy (upper point), robustness against AutoAttack (lower point) and training time. The experiments are performed with WRN28-10 trained on CIFAR-10. The data of the methods marked with * are collected from RobustBench [9], with the training time roughly approximated in accordance to the data size and number of backward passes.



(a) Accuracy over epoch

(b) Accuracy over time

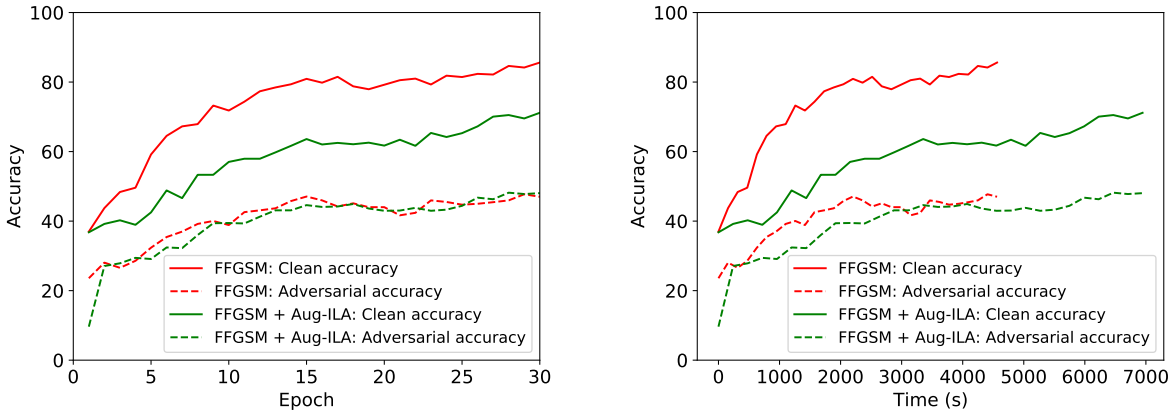
Figure 5.3: The overall convergence rate in the two phases, with phase 1 in red and phase 2 in green.

Aug-ILA) for 30 epochs, although the robust accuracy appears to be stably converging, the clean accuracy exhibits an apparent decay in comparison to that of the pre-trained model. Besides, the training time is almost twice longer, taking around 7200 seconds to complete 30 epochs. Balancing both the convergence rate as well as the training efficiency, we decide to split the training into two phases, such that the FFGSM + Aug-ILA attack can be performed in an effective manner.

5.3 Studies on the Hyper-parameters

In this section, we explore the results when different hyper-parameters are used. Specifically, four main hyper-parameters will be searched: the surrogate model, the combination of perturbation budgets (ϵ) used, the number of iterations of Aug-ILA and the augmentations enabled in Aug-ILA.

We first experiment with the surrogate model used in Aug-ILA. The idea of using Aug-ILA in adversarial training can be viewed similarly to ensemble adversarial training [66], as extra models are introduced in strengthening the robustness of the target model. One difference is that the main reason for adopting a surrogate model in Aug-ILA is to enhance the attack in the aspect of black-box transferability. Therefore, the surrogate model chosen is important since incompatible surrogate models can worsen the robustness of the resulting model. We further explore two more



(a) Accuracy over epoch

(b) Accuracy over time

Figure 5.4: A comparison between the convergence rate of phase 1 pre-training with FFGSM and FFGSM + Aug-ILA.

Table 5.3: Accuracy comparison on Aug-ILA with different surrogate models.

Model	Surrogate Model	Time (s)	Accuracies		
			Clean	PGD-10	AutoAttack
ResNet18	WRN28-10	4050±127	75.11%±0.55%	49.69%±0.30%	43.42%±0.79%
	WRN34-10	5038±39	75.33%±0.28%	49.87%±0.18%	44.54%±0.09%
	VGG19BN	1231±45	76.27%±0.14%	49.61%±0.09%	43.33%±0.05%
WRN28-10	ResNet18	2355±66	82.03%±0.07%	53.45%±0.20%	47.24%±0.20%
	WRN34-10	6018±22	81.53%±0.18%	53.90%±0.95%	47.06%±0.02%
	VGG19BN	2024±10	83.40%±0.17%	52.93%±0.30%	45.70%±0.23%

surrogate models, with a deeper variant of wide ResNet: WRN34-10, and also a DNN with different architecture: VGG19 [61] with batch normalization. The result is reported in Table 5.5.

From Table 5.5, one notable observation is the training time can vary much when different surrogate models are used. It is because Aug-ILA involves multiple incomplete forward passes and one backward pass for each of its iterations. With the surrogate model getting deeper and wider, it takes more time to complete both the forward and backward passes. Therefore, the model attributes can be another crucial factor of a fast training scheme. Although VGG19 consistently speeds up the training time, the resulting model is the least robust among all options. For example,

Table 5.4: Accuracy comparison with different combinations of ϵ_1 and ϵ_2.

Model	Surrogate Model	ϵ_1	ϵ_2	Time (s)	Accuracies		
					Clean	PGD-10	AutoAttack
ResNet18	WRN28-10	8/255	0/255	590±16	79.16%±0.27%	47.10%±0.30%	42.71%±0.17%
		2/255	6/255	4140±6	83.14%±0.51%	36.89%±0.38%	28.47%±0.60%
		4/255	4/255	4134±10	81.38%±0.07%	47.00%±0.46%	40.56%±0.68%
		6/255	2/255	4109±3	79.97%±0.10%	48.66%±0.38%	42.58%±0.46%
ResNet18	WRN28-10	12/255	0/255	590±17	74.67%±0.24%	49.65%±0.13%	43.48%±0.15%
		8/255	2/255	4141±20	77.28%±0.13%	49.76%±0.33%	43.60%±0.21%
		8/255	4/255	4149±15	75.20%±0.59%	49.89%±0.46%	43.64%±0.10%
		8/255	6/255	4149±3	72.94%±0.29%	49.26%±0.20%	43.23%±0.09%

using the WRN28-10 as the trained model, its accuracy against AutoAttack is around 1.5% worse than that of the remaining surrogate models. On the other hand, a deeper and wider surrogate model (WRN34-10) hardly results in improvement but a significant drawback of the training time. Therefore, we believe ResNet18 and WRN28-10 are good default options for surrogate models on CIFAR-10, while VGG19 or other small models can be adopted when time is a more vital factor to be considered.

Next, we explore the values of ϵ. Since two attacks were stacked in sequence during the fine-tuning phase, we have control over two perturbation budgets, one for FFGSM and one for Aug-ILA. We denote them as ϵ_1 and ϵ_2 correspondingly. In Chapter 4, we always assume $\epsilon_1 = \epsilon_2$ since the target is to transfer the attack, where the strongest Aug-ILA attack can lead the transferability to its best. In the setup of adversarial training, since a transferable attack does not necessarily lead to a strong attack, setting ϵ_2 too large might not benefit the robustness as much as it does for the black-box transferability. The details with different combinations of ϵ_1 and ϵ_2 is shown in Table 5.4. In the upper part of the table, the total perturbation budget $\epsilon_1 + \epsilon_2 = 8/255$, and in the lower part of the table, the total perturbation budget is not restricted. To differentiate the effect of Aug-ILA, we further insert the results of fine-tuning with pure FFGSM (where $\epsilon_2 = 0/255$).

In the upper part of Table 5.4, when $\epsilon_1 < \epsilon_2$, the robustness ends up being worse than that in the pure FFGSM settings. In contrast, when $\epsilon_1 \geq \epsilon_2$, the model starts to outperform the pure FFGSM setting. In the lower part, we can observe a general pattern of the 'no free lunch' trade-off, where

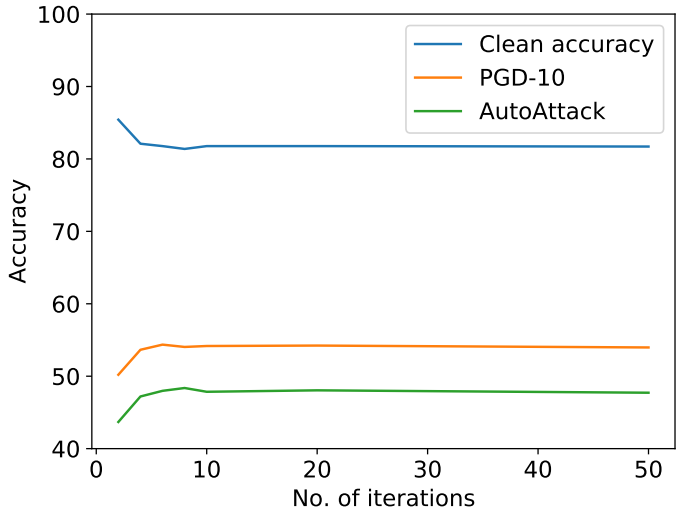


Figure 5.5: Accuracy comparison with different iterations of Aug-ILA used in the fine-tuning phase.

the accuracy drops while the robustness increases. Balancing the two factors, we believe setting $\epsilon_1 = 8/255$ and $\epsilon_2 \in [2/255, 4/255]$ are good default options.

As the efficiency is also one of the important factors in consideration, the number of iterations of Aug-ILA should not be too large. However, unlike FFGSM, a single iteration of Aug-ILA could not yield good robustness. Therefore, we search and report the performance of different numbers of iterations in Figure 5.5, with a WRN28-10 model trained on CIFAR-10. Although Aug-ILA requires 10 iterations to be transferable enough as an attack, in adversarial training, a small number such as 4 is sufficient to yield a robust model. Further increasing the number of iterations does not exhibit any apparent effect on the model performance.

Finally, we test the effect of different components of Aug-ILA. Recall that Aug-ILA consists of three extra augmentations, namely image processing (cropping), reverse adversarial update and attack interpolation. We perform an ablation study with each of the components removed and examine the robustness of the model. The result is shown in Table 5.5. From the table, both cropping and reverse update contribute to defending against different types of attacks, while the attack interpolation is the most trivial, albeit slightly worsening the robustness, among the three augmentations. One possible reason is that since both the number of Aug-ILA iterations and ϵ_2 are small,

Table 5.5: Accuracy comparison with each component of Aug-ILA removed.

Model	Method	Time (s)	Accuracies		
			Clean	PGD-10	AutoAttack
WRN28-10	FFGSM + Aug-ILA	2327±50	**82.08%**±0.14%	53.50%±0.26%	47.24%±0.20%
	w/o reverse update	2491±88	80.44%±0.40%	51.48%±0.56%	43.79%±0.42%
	w/o attack interpolation	2391±92	81.86%±0.23%	**53.87%**±0.27%	**47.33%**±0.07%
	w/o cropping	2479±71	81.11%±0.21%	53.10%±0.23%	46.79%±0.08%

the magnitude of the interpolation is not sufficient to lead to improvement of robustness. Besides, integrating three augmentations also improves the clean accuracy. It may be because the conjunction of augmentations slightly weakens the FFGSM attack but brings additional feature information from the surrogate model, which results in a rise in both the clean accuracy and robustness.

From all the experiments above, we can conclude that Aug-ILA is a suitable option to further boost the model robustness with a Fast framework. Most of the components in Aug-ILA lead to benefits except the distribution of perturbation size should be adjusted such that $\epsilon_1 > \epsilon_2$. Also, we highly recommend using a shallow and thin model to be the surrogate model (such as ResNet18) to speed up the overall training time. In the meantime, despite Aug-ILA's excellence in fine-tuning models within a few epochs, it is not suitable to be used in any pre-training works due to its poor convergence rate and also the worsened efficiency compared to FFGSM.

CHAPTER 6

CONCLUSION

In this thesis, we presented Aug-ILA, which applies extra augmentation to the input references in the ILA framework. We evaluated different image transformations to be applied, and selected random cropping to be our base augmentation. The transformation has to be applied consistently to all the input images in order to align the values of the intermediate feature maps. Moreover, we applied two more augmentations exploiting the adversarial perturbation, namely, reverse adversarial update and attack interpolation across iterations. By incorporating all the augmentations introduced, Aug-ILA not only outperforms ILA and its variants, but also the combination of state-of-the-art methods for transfer-based attacks. We have conducted extensive studies to study the effects of different hyper-parameters. Finally, we provided explanations of the effect of augmentation in terms of the weakening effect on the perturbation strength. All of the experimental results suggest that Aug-ILA can be a strong transfer-based black-box attack given a suitable reference attack such as I-FGSM.

Moreover, we also performed an extensive study on the effect of Aug-ILA in adversarial training. We proposed a two-phase fast training framework that uses FFGSM in the pre-training phase and FFGSM + Aug-ILA in the fine-tuning phase. Multiple experiments suggest that such a framework successfully improves the model robustness without sacrificing the training efficiency. Nevertheless, the new framework could not be exempted from the 'no free lunch' trade-off between the accuracy and robustness, that the clean accuracy still marginally drops while the robustness rises. While the current state-of-the-art works in adversarial training advocate applying extensive data augmentation or even extra datasets which induces extra training time, we believe an efficient training framework is still of high importance. We hope the proposal of Aug-ILA and its application in fast adversarial training could shift the perspective from tedious adversarial training to a more efficient training or fine-tuning mechanism, such that the methods are considerate enough to be used in real-world datasets.

Although the two-fold excellence of Aug-ILA reveals the possibility of integrating data augmentation into generating adversarial attacks, Aug-ILA only consists of simple transformation operations such as cropping. One potential improvement is to consider more sophisticated augmentations such as randomized or automatic data augmentation. With the success of adopting Cutmix and auto-encoders as shown in previous works, it is possible that a combination of these data augmentation techniques might better benefit the effectiveness of Aug-ILA. These extensions and modifications, however, will be marked as future studies.

REFERENCES

[1] Rahnama Arash, Andre T. Nguyen, and Raff Edward. Robust design of deep neural networks against adversarial attacks based on lyapunov theory. In *CVPR*, 2020.

[2] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, 2018.

[3] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. Practical black-box attacks on deep neural networks using efficient query mechanisms. In *ECCV*, 2018.

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

[5] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *AISec*, 2017.

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

[7] Jianbo Chen, Michael I. Jordan, and Martin J. Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *IEEE Symposium on Security and Privacy (SP)*, 2020.

[8] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui 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.

[9] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *ArXiv*, abs/2010.09670, 2020.

[10] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020.

[11] Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 99–108, 2004.

[12] Jakubovitz Daniel and Giryes Raja. Improving dnn robustness to adversarial attacks using jacobian regularization. In *ECCV*, 2018.

[13] Yinpeng Dong, Qi-An Fu, Xiao Yang, Tianyu Pang, Hang Su, Zihao Xiao, and Jun Zhu. Benchmarking adversarial robustness on image classification. In *CVPR*, 2020.

[14] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *CVPR*, 2018.

[15] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *CVPR*, 2019.

[16] Farzan Farnia, Jesse Zhang, and David Tse. Generalizable adversarial training via spectral normalization. In *ICLR*, 2019.

[17] Reuben Feinman, Ryan R. Curtin, Saurabh Shintre, and Andrew B. Gardner. Detecting adversarial samples from artifacts. In *ICML*, 2017.

[18] Croce Francesco and Hein Matthias. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *ICML*, 2020.

[19] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2015.

[20] Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. In *ICLR*, 2020.

[21] Yiwen Guo, Qizhang Li, and Hao Chen. Backpropagating linearly improves transferability of adversarial examples. In *NeurIPS*, 2020.

[22] Yiwen Guo, Ziang Yan, and Changshui Zhang. Subspace attack: Exploiting promising subspaces for query-efficient black-box attacks. In *NeurIPS*, 2019.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[24] Judy Hoffman, Daniel A. Roberts, and Sho Yaida. Robust learning with jacobian regularization. *ArXiv*, abs/1908.02729, 2019.

[25] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

[26] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[27] Qian Huang, Isay Katsman, Horace He, Zeqi Gu, Serge Belongie, and Ser-Nam Lim. Enhancing adversarial example transferability with an intermediate level attack. In *ICCV*, 2019.

[28] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. In *ICLR*, 2019.

[29] Harini Kannan, A. Kurakin, and Ian J. Goodfellow. Adversarial logit pairing. *ArXiv*, abs/1803.06373, 2018.

[30] Hoki Kim. Torchattacks: A pytorch repository for adversarial attacks. *ArXiv*, abs/2010.01950, 2020.

[31] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2012.

[32] Alexey Kurakin, Ian Goodfellow, Samy Bengio, Yinpeng Dong, Fangzhou Liao, Ming Liang, Tianyu Pang, Jun Zhu, Xiaolin Hu, Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren,

Alan Yuille, Sangxia Huang, Yao Zhao, Yuzhe Zhao, Zhonglin Han, Junjiajia Long, Yerkebulan Berdibekov, Takuya Akiba, Seiya Tokui, and Motoki Abe. Adversarial attacks and defences competition. Technical report, ArXiv, 2018.

[33] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *ICLR*, 2017.

[34] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *ICLR*, 2017.

[35] Huichen Li, Xiaojun Xu, Xiaolu Zhang, Shuang Yang, and Bo Li. QEBA: query-efficient boundary-based blackbox attack. In *CVPR*, 2020.

[36] Qizhang Li, Yiwen Guo, and Hao Chen. Yet another intermediate-level attack. In *ECCV*, 2020.

[37] Zhuorong Li, Chao Feng, Jianwei Zheng, Minghui Wu, and Hongchuan Yu. Towards adversarial robustness via feature matching. In *NeurIPS*, 2020.

[38] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *CVPR*, 2018.

[39] Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E. Hopcroft. Nesterov accelerated gradient and scale invariance for adversarial attacks. In *ICLR*, 2020.

[40] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan L. Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, 2018.

[41] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *ICLR*, 2017.

[42] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[43] Andriushchenko Maksym, Croce Francesco, Flammarion Nicolas, and Hein Matthias. Square attack: a query-efficient black-box adversarial attack via random search. In *ECCV*, 2020.

[44] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *CoRR*, abs/1610.08401, 2016.

[45] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Deepdream - a code example for visualizing neural networks. Technical report, Google Research, 2015.

[46] Papernot Nicolas, McDaniel Patrick, Goodfellow Ian, Jha Somesh, Celik Z. Berkay, and Swami Ananthram. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, page 506–519, New York, NY, USA, 2017. Association for Computing Machinery.

[47] Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. Bag of tricks for adversarial training. In *ICLR*, 2021.

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

[49] Nicolas Papernot, P. Mcdaniel, and I. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *ArXiv*, abs/1605.07277, 2016.

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

[51] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy,

Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 2019.

[52] Izmailov Pavel, Podoprikhin Dmitrii, Garipov Timur, Vetrov Dmitry, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *UAI*, 2018.

[53] Haifeng Qian and Mark N. Wegman. L2-nonexpansive neural networks. In *ICLR*, 2019.

[54] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A. Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. Fixing data augmentation to improve adversarial robustness. In *NeurIPS*, 2021.

[55] Leslie Rice, Eric Wong, and J. Zico Kolter. Overfitting in adversarially robust deep learning. In *ICML*, 2020.

[56] Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *AAAI*, 2018.

[57] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[58] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[59] Alex Serban, Erik Poll, and Joost Visser. Adversarial examples on object recognition: A comprehensive survey. *ACM Computing Surveys*, 53(3), June 2020.

[60] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *NeurIPS*, 2019.

[61] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[62] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *ICLR*, 2018.

[63] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[64] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

[65] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

[66] Florian Tramer, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *ICLR*, 2018.

[67] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. There is no free lunch in adversarial robustness (but there are unexpected benefits). Technical report, DeepAI, 2018.

[68] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *ICLR*, 2019.

[69] Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *AAAI*, 2019.

[70] Xiaosen Wang and Kun He. Enhancing the transferability of adversarial attacks through variance tuning. In *CVPR*, 2021.

[71] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *ICLR*, 2020.

[72] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *ICLR*, 2020.

[73] Dongxian Wu, Yisen Wang, Shu-Tao Xia, James Bailey, and Xingjun Ma. Skip connections matter: On the transferability of adversarial examples generated with resnets. In *ICLR*, 2020.

[74] Weibin Wu, Yuxin Su, Michael R. Lyu, and Irwin King. Improving the transferability of adversarial samples with adversarial transformations. In *CVPR*, 2021.

[75] Yuan Xiaoyong, He Pan, Zhu Qile, and Li Xiaolin. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9), September 2019.

[76] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. In *ICLR*, 2018.

[77] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan Yuille. Improving transferability of adversarial examples with input diversity. In *CVPR*, 2019.

[78] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[79] Ziang Yan, Yiwen Guo, and Changshui Zhang. Subspace attack: Exploiting promising subspaces for query-efficient black-box attacks. In *NeurIPS*, 2019.

[80] Xuwang Yin, Soheil Kolouri, and Gustavo K Rohde. Gat: Generative adversarial training for adversarial example detection and robust classification. In *ICLR*, 2020.

[81] Jongmin Yoon, Sung Ju Hwang, and Juho Lee. Adversarial purification with score-based generative models. In *ICML*, 2021.

[82] Yuichi Yoshida and Takeru Miyato. Spectral Norm Regularization for Improving the Generalizability of Deep Learning. *ArXiv*, abs/1705.10941, 2017.

[83] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019.

[84] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

[85] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In *ICML*, 2019.

[86] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

[87] Wen Zhou, Xin Hou, Yongjun Chen, Mengyun Tang, Xiangqi Huang, Xiang Gan, and Yong Yang. Transferable adversarial perturbations. In *ECCV*, 2018.

APPENDIX A

HYPER-PARAMETERS USED IN THE BASELINES

We reproduce the algorithms of variance tuning, CTM (DIM, TIM, SIM), MI-FGSM and NI-FGSM according to the repository released by Wang et al. [70], and test the attacks under our experimental settings. The list of hyper-parameters used in the experiments is shown in Table A.1, which are the default values in their corresponding papers. For the implementation of SGM and LinBP, we directly use the official implementation [21].

Table A.1: Hyperparameters used in the baselines.

Method	Hyper-parameter	Value
MI-FGSM	α	1.0
NI-FGSM	α	1.0
DIM	Probability	0.5
	Upscale ratio	1.1
TIM	Kernel size	7×7
SIM	Scale copies	5 ($i = 0, 1, 2, 3, 4$)
Variance Tuning	N	20
	β	$1.5/255$
SGM	λ	0.5
LinBP	I-FGSM iteration	300
	Layer	'3-1' for ResNet50

APPENDIX B

EFFECT OF REVERSE ADVERSARIAL UPDATE ON THE MODEL PERFORMANCE

We would like to verify the intuition of using reverse adversarial update, specifically, whether in practice such operation can decrease the loss and increase the model confidence. Therefore, we randomly sample 5000 images, including those that are misclassified by the model. Then we apply I-FGSM10 in the form of reverse adversarial update to the images, and pass them back to the model for classification. We record the loss and apply softmax to the logits to obtain the confidence, and report the average result of all 5000 images. The result is summarized in Table B.1.

Table B.1: Changes in loss and confidence after applying reverse adversarial update on the images for ResNet50.

Model	Image	Loss	Confidence	Accuracy
ResNet50	Clean	0.9755	0.7913	75.08%
	Reversely updated	0.3070	0.8847	92.20%
Inception V3	Clean	1.1165	0.7339	76.44%
	Reversely updated	0.3525	0.8296	94.60%
VGG19	Clean	1.1431	0.7419	70.54%
	Reversely updated	0.5756	0.8998	89.98%

Furthermore, we also inspect the class activation map (CAM) of the source model. Figure B.1 visualizes some of the results. For some images (the first two), updating reversely helps the model focus on more features. For the remaining (the latter two), adversarial reverse update does not exhibit a significant change of intermediate layers' contribution to the ground-truth class.

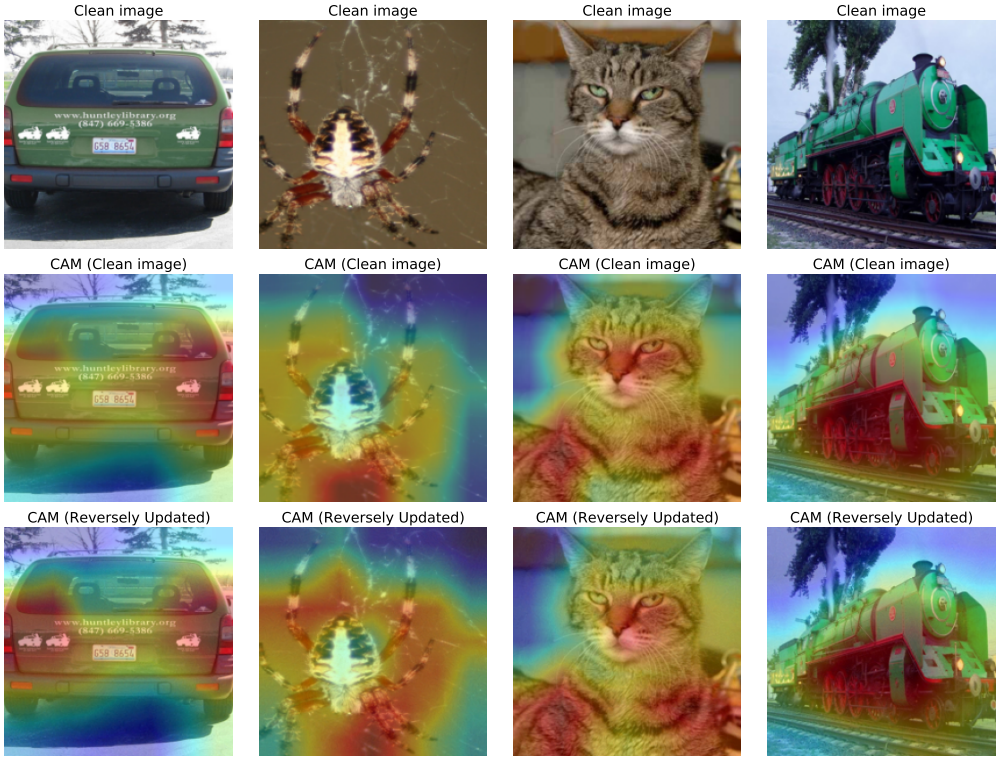


Figure B.1: CAM visualization of the images after reverse adversarial update.

To test the effect of reversely updated examples on targeted models, we also feed them into the remaining undefended models and observe the change in accuracy. The result is reported in Table B.2, which shows that reverse adversarial update not only benefits the source model, but also many of the other models with similar architecture. It shows that reverse adversarial update can be beneficial to many of the target models under proper choices of hyper-parameters such as the source model.

Table B.2: Top-1 accuracy of the models under the example with reverse adversarial update from different source models. The source model "-" indicates clean example without any perturbation.

Source model	Target model				
	ResNet50	Inception v3	WRN	VGG19	PNASNet
-	75.08%	76.44%	77.58%	70.54%	71.34%
ResNet50	92.20%	79.48%	82.22%	74.06%	72.84%
Inception V3	78.58%	94.60%	80.70%	73.62%	72.88%
VGG19	78.46%	79.56%	80.06%	89.98%	71.76%

Source model	Target model				
	DenseNet	ResNeXt	MobileNet	SENet	Average
-	75.86%	78.30%	70.48%	75.36%	74.55%
ResNet50	80.70%	82.70%	73.70%	80.70%	79.84%
Inception V3	80.54%	81.54%	73.04%	78.80%	79.37%
VGG19	80.58%	81.02%	74.38%	79.00%	79.42%