Increasing Transferability by Imposing Linearity and Perturbation in Intermediate Layer with Diverse Input Patterns

by

SHAH MEET ASHVINKUMAR
202011047

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY
in
INFORMATION AND COMMUNICATION TECHNOLOGY
to

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

June, 2022

Declaration

I hereby declare that

i) the thesis comprises of my original work towards the degree of Master of Technology in Information and Communication Technology at Dhirubhai Ambani Institute of Information and Communication Technology and has not been submitted elsewhere for a degree,

ii) due acknowledgment has been made in the text to all the reference material used.

Meet Shah

Shah Meet Ashvinkumar

Certificate

This is to certify that the thesis work entitled "Increasing Transferability by Imposing Linearity and Perturbation in Intermediate Layer with Diverse Input Patterns" has been carried out by Shah Meet Ashvinkumar (202011047) for the degree of Master of Technology in Information and Communication Technology at *Dhirubhai Ambani Institute of Information and Communication Technology* under my/our supervision.

Shruti

Prof. Shruti Bhilare
Thesis Supervisor

[illegible]

Prof. Srimanta Mandal
Thesis Co-Supervisor

Acknowledgments

"Wisdom is not a product of schooling but of the lifelong attempt to acquire it."

-Albert Einstein

Life is a never-ending learning experience. Resuming formal education by pursuing MTech after Industrial experience was a challenging endeavor. This journey was made possible by experienced lecturers, an engaging curriculum, helpful classmates, and the wonderful DAIICT campus. My MTech Thesis is also an important part of this learning process. This thesis presents not only my work but also the motivation and support of the people around me. Hence, I want to take this opportunity to convey my heartfelt gratitude to everyone who helped in this thesis work.

First and foremost, I would like to express my deepest gratitude and reverence to my supervisor Prof. Shruti Bhilare, who gave me the chance to explore this subject and motivated me to try new things throughout the research. Her consistent support and guidance inspired me, and her dedication will continue to inspire me in the future. I'm also grateful to Prof. Srimanta Mandal, my co-supervisor, for pointing me in the right direction and correcting me at every turn along the way. In addition, I would like to thank Prof. Avik Hati for offering valuable assistance on several aspects of the thesis.

I would like to thank Shivangi Gajjar and Krunal Mehta for sharing their experience and knowledge and helping me kick-start my work in this domain till the completion of my thesis. I'm indebted to all my friends who have provided inspiration and moral support throughout this journey. A very special thanks to my parents and sister for their love, care, enthusiasm, and unwavering support throughout my life. The things I have achieved in life, I owe it to them.

Contents

Abstract

Despite high prediction accuracy, deep neural networks are vulnerable to adversarial attacks, introduced by perturbations that humans may not even perceive. Hence, adversarial examples can mislead the trained networks. As a consequence, the security of such systems can get compromised. The process of generating adversarial examples can assist us in investigating the robustness of different models. Many developed adversarial attacks often fail under challenging black-box settings. It is required to improve the success rate of misleading a network by adversarial examples crafted to trick another model. This phenomenon is known as transferability. In contrast to the existing methods, we propose to increase the rate of transferability by inducing linearity in a few intermediate layers of architecture. The proposed design does not disturb the original architecture much. The intermediate layers play significant roles in generating feature maps suitable for a task. Hence, by analyzing the feature maps of architecture, a particular layer can be perturbed more to improve the transferability. The performance is further enhanced by considering diverse input patterns. Experimental results demonstrate the success in increasing the transferability of our proposition.

Keywords: ***Deep Neural Network, Perturbations, Adverasarial Examples, Transferability***

List of Principal Symbols and Acronyms

AT	Adversarial Training
BN	Batch Normalization
C&W	Carlini and Wagner
DI-FGSM	Diverse Inputs Iterative - Fast Gradient Sign Method
DNN	Deep Neural Network
DT	Decision Tree
FGSM	Fast Gradient Sign Method
I-FGSM	Iterative - Fast Gradient Sign Method
ILA	Intermediate Level Attack
JSMA	Jacobian-based Saliency Maps Attack
KNN	K Nearest Neighbor
L-BFGS	Limited-memory Broyden–Fletcher–Goldfarb–Shanno
LinBP	Linearized Backpropagation
LinS	Linear Substitution
LR	Linear Regression
MI-FGSM	Momentum Iterative - Fast Gradient Sign Method
PGD	Projected Gradient Descent
ReLU	Rectified Linear Unit
SVM	Support Vector Machine
TI-FGSM	Translation Invariant - Fast Gradient Sign Method

VGG	Visual Geometry Group
WRN	Wide Residual Network

List of Tables

List of Figures

CHAPTER 1
Introduction

Now-a-days deep learning based methods lead in several applications such as automated driving, biometric recognition, machine translation, etc., due to their learning efficiency [1, 2]. Vast corpora of training examples are used to create models with high prediction accuracy on unseen samples. Hence, deep neural networks (DNNs) [3] can be employed in many scenarios, including security-sensitive applications. As the usage of these networks increased rapidly in a real-world application, the threat to these networks also gets evolved in the form of *adversarial samples*.

The implications for the existence of adversarial examples in the real world can not be underestimated. Figure 1.1 demonstrates an example of an adversarial sample. In the figure, the original image is classified correctly by the model as "Fish" with 0.9 probability. After adding perturbations, the model misclassifies that adversarial image as "Cat" with 0.9 probability. The perturbation is generated in a way such that the model gets deceived with high confidence score while perceptually remaining same. Moreover, the recent works [4, 5, 6, 7] show the success of adversarial attacks in misleading the DNNs. It implies that the basic building blocks of DNNs and training methods generate some blind spots in the model leading to incorrect predictions.

Adversarial attacks can be classified into two types based on the attacker's adversarial knowledge: white-box attack and black-box attack. In a white-box attack, the adversary has complete information about the target model, including its architecture, gradient information, and predictions. Adversary utilizes this knowledge to generate attacks specific to the target model. On the other hand, in a black-box attack [8], the adversary does not have knowledge of the target model, which is the most likely scenario in real-world applications, and its success mainly relies on the transferability [9] of the adversarial samples. Transferability is one of the intriguing properties of adversarial examples that have been highlighted [10, 11], suggesting that adversarial examples crafted to misclassify a specific model

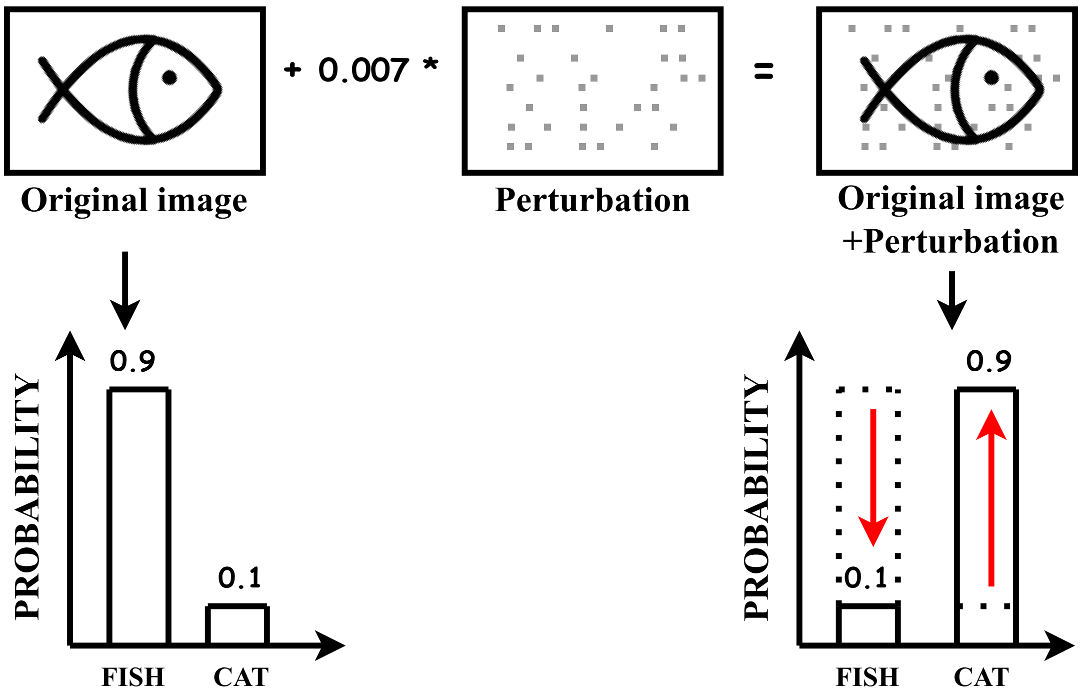


Figure 1.1: Adversarial sample misleading the model

can be utilized to misclassify another model. The adversaries can exploit this property in order to fool an unknown model using a known one. Hence, it is imperative to study the extent of transferability of the adversarial attacks on the state-of-the-art DNN models.

1.1 Motivation

The primary reason for focusing more on the transferability property is to make state-of-the-art attacks more generalized and stronger as, in order to build a more robust defense, it is essential to analyze the extent of the threat posed by more transferable variants of stronger attacks. As shown in Table 1.1, among different machine learning techniques such as Support Vector Machine, Linear Regression, Decision Tree, K Nearest Neighbor, and Deep Neural Networks, attacks on DNNs have the lowest transferability, which makes adversarial attacks on DNNs ideally suitable for real-world applications. To understand the reliability of these DNNs in case someone managed to improve the transferability of these attacks drastically, it is crucial to study the attack to investigate the threat posed by their more transferable variants.

Table 1.1: Cross - transferability in machine learning techniques

SVM	LR	DT	KNN	DNN
100.0	91.43	89.29	41.65	38.27

1.2 Problem Statement

To enhance the transferability of adversarial attacks in the black-box setting and to analyze the robustness of state-of-the-art deep neural network architecture against these attacks.

1.3 Contribution

The key contributions of the proposed work are as follows:

- We propose to induce linearity in a few intermediate layers of the architecture. During forward propagation, there is no change in the training process. However, only a few non-linear activation functions are skipped in a few intermediate layers during backpropagation.
- This method has a similarity with LinBP [12]. An important difference is a position and the number of activation layers for linearization. We investigate to find out that linearizing only 2 layers in the middle and 2 layers just before the fully-connected layer during backpropagation is sufficient to improve transferability. Hence, the proposed design does not affect the architecture much.
- We consider the importance of intermediate feature representations of the adversarial example. Increasing the perturbation in a particular layer can increase the transferability [13].
- We consider diverse input patterns [14] by applying random transformations to the input image. It helps address the over-fitting issue of the network by augmenting the training data with label-preserving transformations.

1.4 Document Outline

The rest of the work is divided into four sections. Chapter 2 discusses the previous works on adversarial attacks and techniques to improve the transferability of those attacks. Chapter 3 discusses the proposed method. Chapter 4 and 5 showcases the improvement in our results as compared to the state-of-the-art methods. Chapter 6 concludes the work.

CHAPTER 2
Literature Review

By demonstrating the vulnerabilities of multiple state-of-the-art deep neural networks, Szegedy et al. [10] made significant progress in adversarial machine learning. DNNs trained with high accuracy also have certain blind spots, where they can be vulnerable to adversarial attacks. The adversary exploits these areas to generate adversarial samples. Even by changing the data distribution of the input slightly, a machine learning model can easily be fooled [4]. Major attack approaches developed to minimize the perturbation and increase the generalizability of adversarial samples and defense methods to mitigate those attacks developed till now are as follows.

2.1 Adversarial Attack

2.1.1 Limited-memory-BFGS Attack

Szegedy et al. [10] defined adversarial example as inputs that look very similar to their real counterparts according to a distance metric but one that causes a classifier to misclassify it. They generated adversarial examples using box-constrained L-BFGS. Given an image x_0, their method finds a different image x that is similar to x_0 under L_2 distance, yet is labeled differently by the classifier. They model the problem as a constrained minimization problem:

$$\begin{aligned} \text{minimize} \quad & \|x - x_0\|_2^2 \\ \text{such that} \quad & f(x) = y \\ & x \in [0,1]^n \end{aligned} \tag{2.1}$$

Figure 2.1 shows images that were classified correctly on the left, perturbation added to them in the middle, and images that were misclassified after adding perturbation on the right.

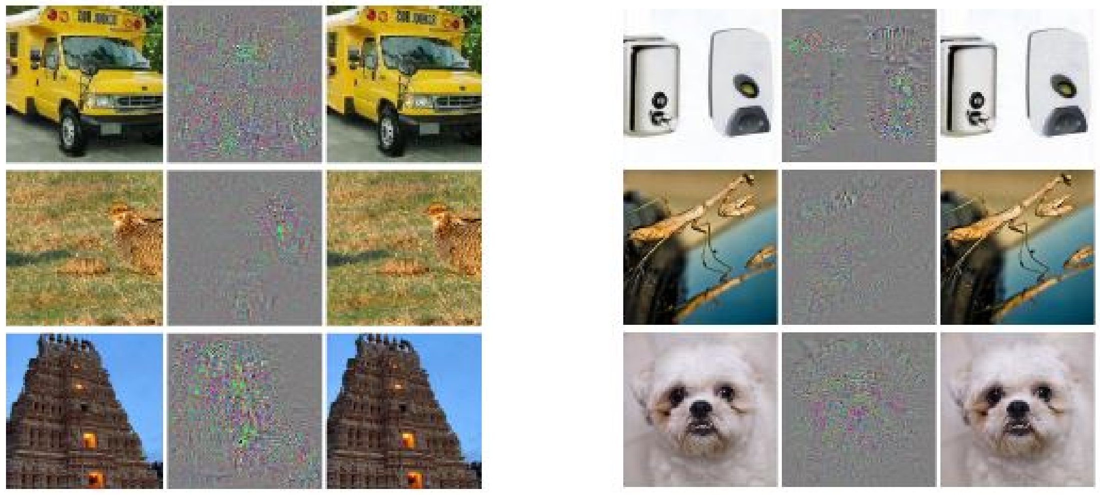


Figure 2.1: col1: original image, col2: perturbation, col3: adversarial image
Adversarial samples generated from L-BFGS [10]

This problem can be challenging to solve, however, so they solved the following problem, which aims to find a perturbation that:

$$\text{minimize } c \cdot \|x - x_0\|_2^2 + \text{loss}(f(x), y) \text{ , such that } x \in [0,1]^n \tag{2.2}$$

The above formulation aims to make the classifier f misclassify x as class y. The loss function used here is the cross-entropy loss, and line search is used to find the minimum constant c where $c > 0$ until an adversary is found.

2.1.2 Fast Gradient Sign Method

FGSM [4] is an example of a white-box attack because the attacker needs to know the model's architecture and parameters to perform backpropagation. FGSM searches for the direction in which the loss function increases fastest for a target machine learning model. Once the gradient is computed, one can push the input towards the adversarial gradient by a small amount.

It is single-step algorithm to generate an adversarial sample x_{adv}, given an original input image x with the model's output y as shown below,

$$x_{adv} = x + \epsilon \cdot \text{sign}\left(\nabla_x loss(f(x), y)\right) \tag{2.3}$$

Here, ϵ is a hyperparameter that controls the magnitude of the perturbation, and $loss(.)$ denotes the model f's loss function. Figure 2.2 shows an adversarial image generated using FGSM. FGSM differs from L-BFGS by 2 key points: 1. It is opti-

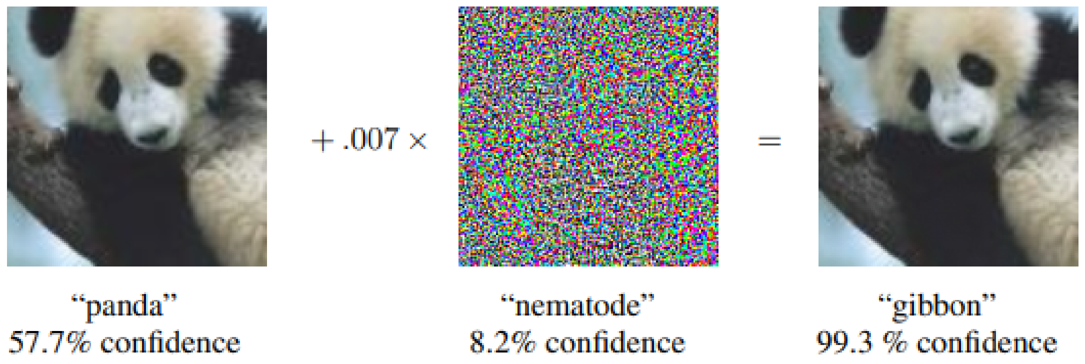


Figure 2.2: Adversarial sample generated using FGSM [4]

mized for the L_1 distance metric 2. It is designed primarily to be fast instead of producing very close adversarial examples.

FGSM has several variants primarily focused on the same technique but with some modifications. Iterative FGSM (I-FGSM) [15] is an FGSM version that uses the FGSM's single-step rule to update the original input with perturbations computed iteratively. In order to avoid poor local maxima, Momentum Iterative FGSM (MI-FGSM) [5] collects gradients of the loss function at each iteration. As a result, stable optimization will be obtained. Diverse Inputs FGSM (DI-FGSM) [14] augments input data as transformed images to avoid the problem of MI-FGSM overfitting. The gradients of the untranslated pictures are convolved with a preset kernel in Translation Invariant FGSM (TI-FGSM) [16], which optimizes the perturbations.

2.1.3 Carlini and Wagner Attack

The adversarial attack proposed by Carlini and Wagner is by far one of the strongest attacks [17]. They formulate targeted adversarial attacks as an optimization problem, take advantage of the internal configurations of a targeted DNN for attack guidance, and use the L_2 norm (i.e., Euclidean distance) to quantify the difference between the adversarial and the original examples.

Given an image x_0, let x denote the adversarial example of x_0 with a targeted class label y toward misclassification. The C&W attack finds the adversarial example x by solving the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \|x - x_0\|_2^2 + c \cdot f(x) \\ \text{subject to} \quad & x \in [0,1]^n \end{aligned} \tag{2.4}$$

where $\|\mathbf{v}\|_2 = \sqrt{\sum_{i=1}^{n} v_i^2}$ denotes the Euclidean norm (L_2 norm) of a vector $\mathbf{v} = [v_1, \ldots, v_n]^T$, and $c > 0$ is a regularization parameter. The first term $\|x - x_0\|_2^2$ in equation is the regularization used to enforce the similarity between the adversarial example x and the image x_0 in terms of the Euclidean distance, since $x - x_0$ is the adversarial image perturbation of x relative to x_0. The second term $c \cdot f(x)$ is the objective function that reflects the level of unsuccessful adversarial attacks, and y is the target class. Carlini and Wagner compared several candidates for objective function $f(x)$ and suggested the following form for effective targeted attacks [6]:

$$f(x) = \max\left\{\max_{i \neq y}[Z(x)]_i - [Z(x)]_y, -\kappa\right\} \tag{2.5}$$

where $Z(x) \in \mathbb{R}^K$ is the logit layer representation (logits) in the DNN for x such that $[Z(x)]_y$ represents the predicted probability that x belongs to class y, and $\kappa \geq 0$ is a tuning parameter for attack transferability. The parameter κ is used to control the strength of adversarial examples: the higher κ, the more powerful the adversarial example's classification. By increasing κ, we can construct high-confidence adversarial examples. Carlini and Wagner set $\kappa = 0$ for attacking a targeted DNN and suggested large κ when performing transfer attacks. The rationale behind the use of the loss function can be explained by the softmax classification rule based on the logit layer representation; the output (confidence score) of a DNN $F(x)$ is determined by the softmax function:

$$[F(x)]_k = \frac{\exp([Z(x)]_k)}{\sum_{i=1}^{K} \exp([Z(x)]_i)}, \forall k \in \{1, \ldots, K\} \tag{2.6}$$

here, $\max_{i \neq y}[Z(x)]_i -$ $[Z(x)]_y \leq 0$ implies that the adversarial example x attains the highest confidence score for class y and hence the targeted attack is successful. On the other hand, $\max_{i \neq y}[Z(x)]_i - [Z(x)]_y > 0$ implies that the targeted attack using x is unsuccessful. The role of κ ensures a constant gap between $[Z(x)]_y$ and $\max_{i \neq y}[Z(x)]_i$, which explains why setting large κ is effective in transfer attacks. The box constraint $x \in [0, 1]^n$ indicates that the generated adversarial example has to be in valid image space. This constraint is removed by replacing x with $\frac{1+\tanh \mathrm{w}}{2}$, where w is an optimizer. With this change-of-variable method, the optimization problem converted as following with optimal w where, $f(x)$ will be the function choosen by carlini and wagner and discussed above

$$\min_{w} \left\|\frac{1}{2}(\tanh(w) + 1) - x_0\right\|_2^2 + c \cdot f\left(\frac{1}{2}(\tanh(w) + 1)\right) \tag{2.7}$$

Furthermore, Carlini and Wagner also showed that their attack could successfully bypass 10 different detection methods designed for detecting adversarial examples [18]. Figure 2.3 demonstrates the adversarial samples generated from L_2 attack, which is the strongest among them. These attacks were proposed to evaluate the robustness of DNNs by generating adversarial samples with high confidence.

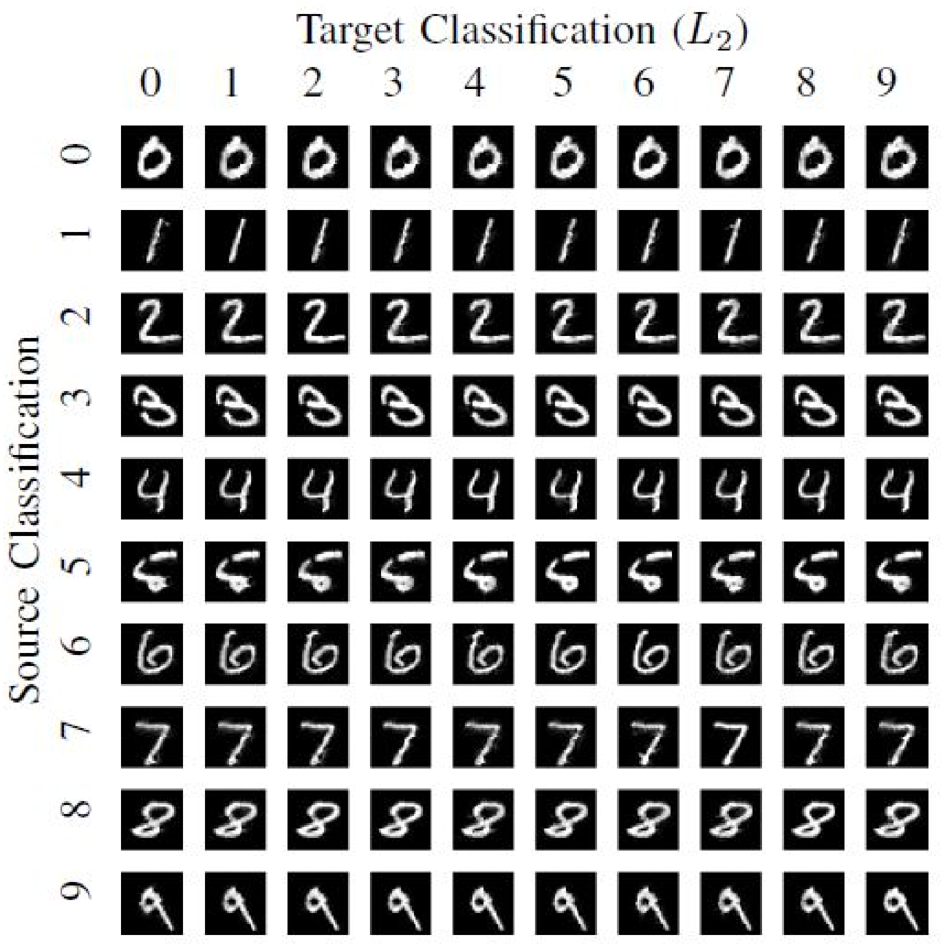


Figure 2.3: Source/Target pair generated from C&W [6] L_2 attack

2.2 Transfer-based Attacks

As the foundation of many black-box attacks, the transferability of adversarial examples has drawn attention since Goodfellow's work [4], where it is attributed to the linear nature of modern DNNs. Transfer-based methods generate adversarial examples on a source model and expect them to fool the target model. A somewhat non-trivial finding is that single-step attacks are more transferable than their multi-step counterparts, which are undoubtedly more effective in the white-box

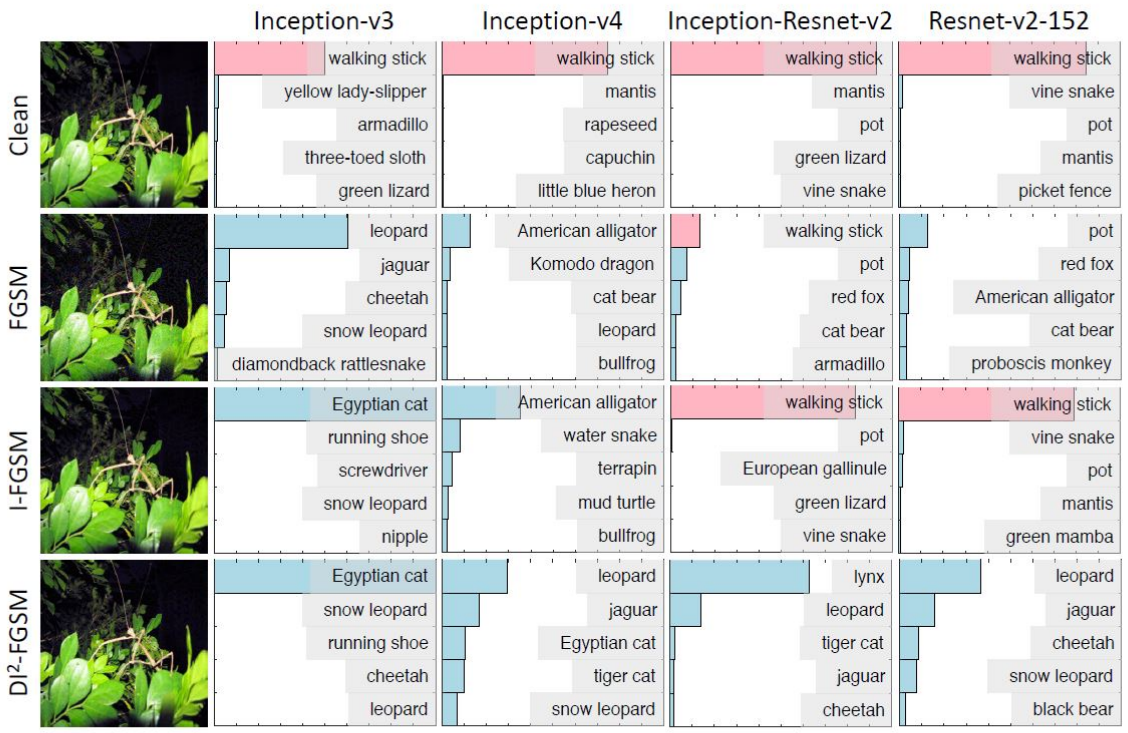


Figure 2.4: The comparison of success rates using three different attacks [14]

context. Specific techniques like variance tuning, skip gradients method, and intermediate level attack [13, 19, 20] have been developed to boost the transferability of adversarial samples and to easily integrate them with current state-of-the-art attacks.

2.2.1 Diverse Input Iterative-FGSM

The diverse inputs method, introduced by Xie et al. [14] applies random transformations to the inputs and feeds the modified samples into the classifier for gradient computation. At the start of each new iteration, they select input for current iteration from the process with probability of p. Here, the stochastic transformation function $T\left(X_n^{adv}; p\right)$ is:

$$T\left(X_n^{adv}; p\right) = \begin{cases} T\left(X_n^{adv}\right) & \text{with probability } p \\ X_n^{adv} & \text{with probability } (1-p) \end{cases} \tag{2.8}$$

They have also integrated momentum term [5] with iterative algorithm [15] in this approach to stabilize the update directions and avoid poor local maxima during the iterations, resulting in more transferable adversarial examples. Figure 2.4 shows the comparison of success rates between 3 attacks FGSM, I-FGSM, and DI^2-

FGSM, where adversarial examples are generated from Inception-v3 transferred to target networks. In the top-5 confidence distribution plots of Figure 2.4, the ground-truth "walking stick" is colored pink, while the other classes are colored blue.

2.2.2 Intermediate Level Attack

By modifying a pre-specified layer of the source model, Huang et al. [13] have proposed an intermediate level attack (ILA) for improved black-box transferability. ILA refines existing adversarial samples, leveraging state-of-the-art adversarial attacks. It fine-tunes samples at specified layer l for higher norm and attempts to maintain the output difference's direction, keeping the original adversarial structure intact. They must balance staying close to the original direction and increasing the norm by maximizing the projection onto the original adversarial perturbation. They examined that the intermediate feature maps, perceptibility is no longer intrinsically tied to the norm in an intermediate feature map and able to increase the norm of the perturbation in that feature space significantly with no change in perceptibility in the image space.

2.2.3 Linearized Backpropagation

Guo et al. [12] have used the linearity hypothesis: "The cause of adversarial examples and their surprising transferability is the linear nature of modern DNNs" [4] and have suggested eliminating non-linear layers from architecture, giving idea for more linear substitution (LinS) model. Since such a direct linearization generates a retrogressive impact on the network's adversarial success rate, they have tried to obtain a reasonable trade-off between transferability and adversarial success rate. They have decoupled the influence of ReLU removal in forward and backward computing for constructing adversarial examples, and they analyze how the specific "linearization" consistently affects both passes. After observing the effects of linearity on both passes, Guo et al. [12] focused on the linearized backpropagation (LinBP) method as it makes little sense to couple a "linearized" forward with a non-linear backpropagation. LinBP technique can supplement existing state-of-the-art attacks for superior transferability. However, it does not consider the importance of intermediate layers. Figure 2.5 compares the flow of Normal methods with the LinS and LinBP methods.

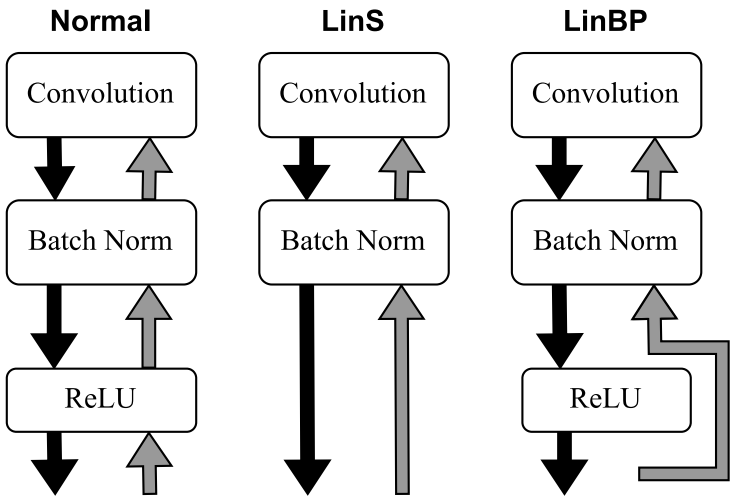


Figure 2.5: The comparision of Flow in LinS and LinBP with normal method

2.3 Adversarial Defense

2.3.1 Adversarial Training

Conversely, many methods have been proposed recently to defend against adversarial examples. Adversarial training (AT) is by far one of the best defense techniques that defend the DNNs from adversarial attacks. Kurakin et al. proposed [21] to inject adversarial examples with their correct labels into the training data so that the model learns how to handle them and increases the network robustness. Tram´er et al. [22] pointed out that such adversarially trained models still remain vulnerable to adversarial examples and proposed ensemble adversarial training, which augments training data with perturbations transferred from other models, in order to improve the network robustness further. Although these are effective, they are computationally expensive and will not guarantee total security against stronger attacks.

2.3.2 Network Distillation

Papernot et al. [23] proposed a defensive distillation method to defend against attacks. In distillation training, one model is trained to predict the output probabilities of another model trained on an earlier baseline standard to emphasize accuracy. As shown in Figure 2.6, This trains the second model to behave like the first model, and the soft labels convey additional hidden knowledge learned by the first model. As indicated by the suggestion that the defense should defend

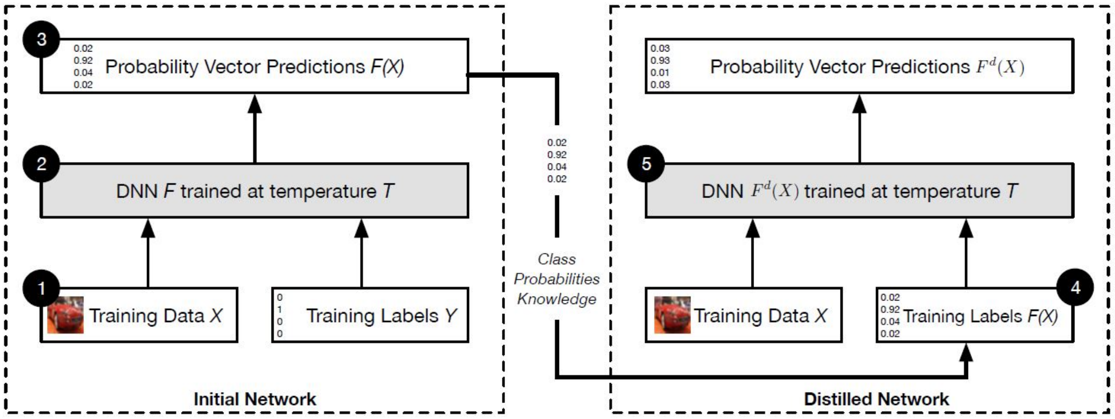


Figure 2.6: An overview of defensive distillation mechanism [23]

against the strong attack [18], several others [24, 25, 26] tried to develop detection mechanisms where we can distinguish between original and adversarial samples. Although these methods achieve quiet success in FGSM [4] and JSMA [27] attacks, none proved to be capable enough to detect adversarial samples generated by C&W attack.

CHAPTER 3
Proposed Approach

Figure 3.1 shows the end-to-end process for our method, where at each iteration, we are generating diverse input patterns from an adversarial sample and calculating the perturbation while removing two ReLU layers from the middle and last parts each. The pre-specified layer at which we are increasing perturbation is highlighted as intermediate layer. After the process completes, generated adversarial sample will have higher transferability.

3.1 Linearization

Our method is developed based on the concept of linearization. However, linearization may affect the performance of a network. Hence, we design the learning in such a manner that the forward propagation does not get affected at all. Only a few non-linear activation layers are skipped during backpropagation. If the source architecture can be represented by a function $h : \mathbb{R}^N \rightarrow \mathbb{R}^K$, which labels one of the K classes to the N dimensional data $\mathbf{x}$. The architecture is parameterized by a sequence of parameters $\theta_1, \theta_2, \ldots, \theta_L$. Then the forward propagation of the architecture can be modeled mathematically as

$$h(\mathbf{x}) = \theta_L^T \mathcal{A}\left(\theta_{L-1}^T, \ldots, \mathcal{A}(\theta_1^T \mathbf{x})\right). \tag{3.1}$$

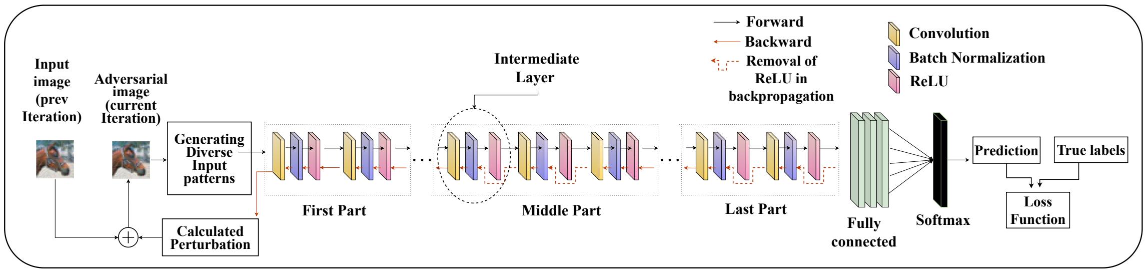


Figure 3.1: Illustration of proposed method.

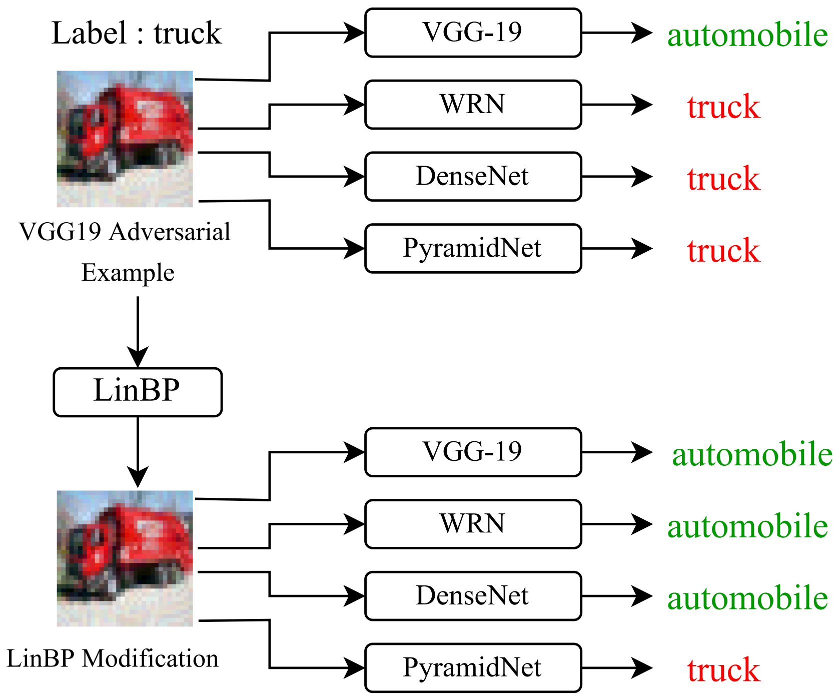


Figure 3.2: An example from CIFAR-10[28] dataset perturbed using LinBP into adversarial example for VGG-19

Here $\mathcal{A}$ is non-linear ReLU unit. For simplicity of expression, consider skipping last two ReLU functions during backpropagation,

$$\hat{\nabla}_x C(h(\mathbf{x}), k) = \frac{dC(h(\mathbf{x}), k)}{d\mathbf{y}_m} \theta_L \theta_{L-1} \frac{d\mathbf{y}_n}{d\mathbf{x}} \tag{3.2}$$

$C(h(\mathbf{x}), k)$ is the cost function between the predicted class and true class. $\hat{\nabla}$ represents linearize gradient of the cost function. $\mathbf{y}_n := f(\mathbf{x})$, where $f(\cdot)$ represents the entire network except the last two convolution layers.

$$\mathbf{y}_m = \theta_L^T \mathcal{A}(\theta_{L-1}^T \mathbf{y}_n) \tag{3.3}$$

Figure 3.2 depicts the LinBP adaptation of an adversarial example for VGG19. LinBP alters the adversarial example to make it more transferable. It is worth noting that for this sample, while the initial VGG19 adversarial example fooled VGG-19 [29], it did not fool the WRN [30], DenseNet [31] and PyramidNet [32]. The LinBP variant of this adversarial sample, on the other hand, is more transferrable and can trick more networks.

3.2 Intermediate Layers

Analyzing the representation of adversarial samples in the intermediate layers can illuminate different aspects of transferability [13]. Based on the analysis, one layer can be pre-specified and modified such that the norm of the output at this layer increases while deviating minimally from the original output direction. Even though the perturbation norm increases, intermediate-level attack (ILA) achieves visually similar adversarial examples. This is because the intermediate layer typically induces less recognizable distortion in the resulting output. Thus, the intermediate layer l is fine-tuned to have a higher perturbation in a direction close to the previous adversarial example. It will help us to achieve higher transferability without sabotaging the adversarial structure. It may be noted that the ILA can be carried out without prior knowledge of the target model. Thus, we can easily integrate the ILA attack with our method on the source model to get superior transferability on the target models.

3.3 Diverse Input Patterns

M-DI^2-FGSM attack overcomes overfitting, resulting in more transferable samples. M-DI^2-FGSM attack uses the data augmentation technique termed as diverse input pattern [14]. It transforms images using different transformation functions such as random resizing and random padding at each iteration with probability p, maximizing the loss function with respect to the original inputs (Eq. (3.5)). At the start of each new iteration, they select the input for current iteration from the process shown in Figure 3.3 with probability of p. Moreover, the momentum factor in the attack stabilizes the update direction and avoids local maxima, thus alleviating overfitting and resulting in even more transferable adversarial samples.

Updation step for I-FGSM attack is,

$$\begin{aligned} X_0^{\text{adv}} &= X \\ X_{n+1}^{\text{adv}} &= \text{Clip}_X^{\epsilon}\left\{X_n^{adv} + \alpha \cdot \text{sign}\left(\nabla_X L\left(X_n^{adv}, y^{\text{true}}; \theta\right)\right)\right\} \end{aligned} \tag{3.4}$$

where, X_{n+1}^{adv} is the adversarial sample, n is the iteration number, α is the step size, and Clip_X^{ϵ} indicates the resulting image is clipped within the ϵ-ball of the original image X. The updating step for M-DI^2-FGSM is similar to I-FGSM with small change, where g_n is the accumulated gradient at iteration n and μ is the decay

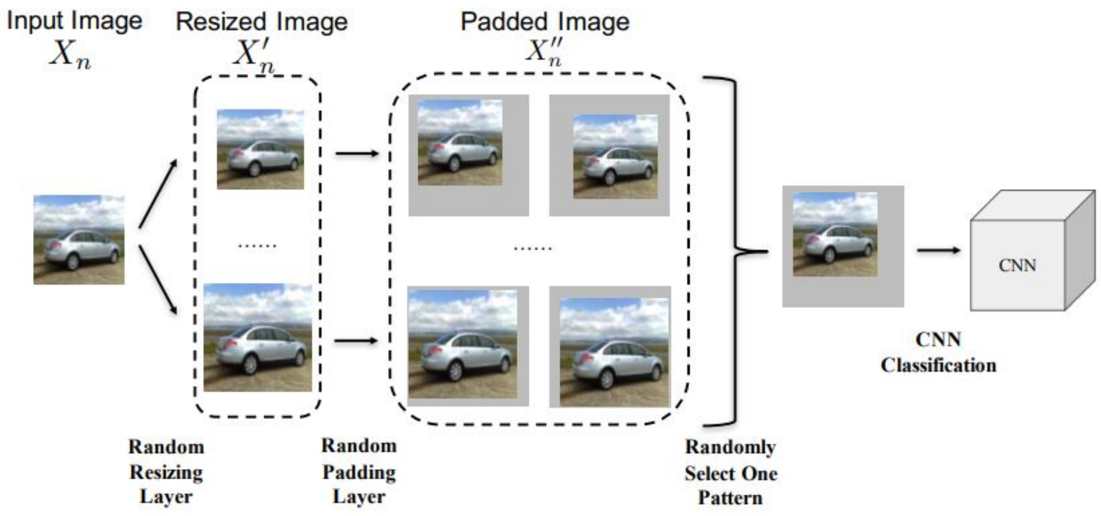


Figure 3.3: The process of using transformation function at each iteration to select input for current iteration [33]

factor of the momentum term.

$$\begin{aligned} g_{n+1} &= \mu \cdot g_n + \frac{\nabla_X L\left(T\left(X_n^{adv}; p\right), y^{\text{true}}; \theta\right)}{\left\|\nabla_X L\left(T\left(X_n^{adv}; p\right), y^{\text{true}}; \theta\right)\right\|_1} \\ X_{n+1}^{\text{adv}} &= \text{Clip}_X^{\epsilon}\left\{X_n^{adv} + \alpha \cdot \text{sign}\left(g_{n+1}\right)\right\} \end{aligned} \tag{3.5}$$

CHAPTER 4
Experimental Setup

4.1 Dataset

For training and evaluation of the proposed method, experiments are performed on the CIFAR-10 dataset [28]. CIFAR-10 imageset is a well-known and established image dataset used for object detection and computer vision applications. This dataset has 60,000 RGB color images, each of size 32x32x3 of a total 10 mutually exclusive classes. These are a set of 60,000 small sized images, much smaller than typical photograph meant for research work into computer vision domain. Out of these, 50,000 corresponds to training images, and the rest are test images. Labels of the class and its associated values are listed below in Table 4.1.

4.2 Source Architecture

We use VGG-19 [29] with batch normalization as the source model for crafting adversarial samples and transferred them to other victim networks. Table 4.2 displays the architecture of VGG-19 with batch normalization.

Table 4.1: CIFAR-10 Dataset

Class number	Class label
0	airplane
1	automobile
2	bird
3	cat
4	deer
5	dog
6	frog
7	horse
8	ship
9	truck

Table 4.2: Model architecture for VGG-19 with batch normalization [29]

Index	Layer Type	Filter
0	Convolution + BN + ReLU	$3 \times 3 \times 64$
1	Convolution + BN + ReLU	$3 \times 3 \times 64$
2	Max Pooling	
3	Convolution + BN + ReLU	$3 \times 3 \times 128$
4	Convolution + BN + ReLU	$3 \times 3 \times 128$
5	Max Pooling	
6	Convolution + BN + ReLU	$3 \times 3 \times 256$
7	Convolution + BN + ReLU	$3 \times 3 \times 256$
8	Convolution + BN + ReLU	$3 \times 3 \times 256$
9	Convolution + BN + ReLU	$3 \times 3 \times 256$
10	Max Pooling	
11	Convolution + BN + ReLU	$3 \times 3 \times 512$
12	Convolution + BN + ReLU	$3 \times 3 \times 512$
13	Convolution + BN + ReLU	$3 \times 3 \times 512$
14	Convolution + BN + ReLU	$3 \times 3 \times 512$
15	Max Pooling	
16	Convolution + BN + ReLU	$3 \times 3 \times 512$
17	Convolution + BN + ReLU	$3 \times 3 \times 512$
18	Convolution + BN + ReLU	$3 \times 3 \times 512$
19	Convolution + BN + ReLU	$3 \times 3 \times 512$
20	Max Pooling	
21	Fully Connected	4096
22	Fully Connected	4096
23	Fully Connected	1000
24	Softmax	10

4.3 Target Networks

Adversarial attacks, namely FGSM, PGD, I-FGSM, and M-DI^2-FGSM, are employed with ℓ_∞ constraint in the untargeted setting. We have considered maximum perturbation (ϵ) to 0.03. In order to evaluate the transferability of the proposed approach, we consider several DNN models trained on CIFAR-10 dataset as victim models namely, ResNeXt [34], WRN [30], and DenseNet [31], GDAS [35] and PyramidNet [32]. All these victim models with their unique architectures were proposed after VGG-19 making the transfer of adversarial examples more challenging. Pre-trained models of the target networks on Torchvision [36] have been used for experimentation.

CHAPTER 5

Experiment and Results

To check how linearization affects the accuracy and transferability of the network, we performed a number of experiments while removing non-linear units (ReLU) from different parts of the architecture. We divide the source architecture into 3 parts as First (F), Middle (M), and Last (L). We removed ReLU layers from the last part of the architecture and increased the number of ReLU layers to be removed to compare the trade-off between accuracy and transferability. All ReLU layers were removed one at a time, starting from the last layer of the architecture up to the first layer. Here (L_{14}) indicates, we removed 14 ReLU layers starting from last part of the architecture in Table 4.2. Table 5.1 shows that transferability does improve while we increase the removal of ReLU layers, but after a certain point, it starts decreasing and hampering accuracy.

To observe the effects of reducing non-linearity, we conduct experiments to remove the ReLU layer from other parts. Table 5.2 depicts different standalone and combinations of parts where F_1 indicates removing 1 ReLU layer from first part, same as M_1 and L_1 indicates middle and last part. As shown in Table 5.2, we found that removing ReLU layers from the middle part gives consistently better results than only the first and last part. Further, we remove ReLU layers from the first-middle (FM), middle-last (ML), and first-last (FL) parts as well. The combinations like F_1L_1 depict removing 1 ReLU layer from the first and 1 from the last part each. It can be observed that the combination of the middle and last part while removing 2 ReLU layers gives the best transferability of all.

Table 5.1: Success rates of transfer-based attacks on CIFAR-10

Attack	VGG-19$^*_{bn}$	ResNeXt	WRN	DenseNet	GDAS	PNet	Average
(L_{14})	99.94	92.11	90.62	89.93	81.24	45.94	79.97
(L_{12})	100	94.96	93.98	93.18	84.8	51.62	83.71
(L_9)	100	95.78	95.38	94.58	83.92	52.45	84.42
(L_6)	100	91.76	91.57	89.86	79.05	43.24	79.10
(L_2)	100	92.23	91.74	90.08	79.31	43.07	79.29

Table 5.2: Success rates of different variations of the attack on CIFAR-10 dataset

Attack	VGG-19$^*_{bn}$	ResNeXt	WRN	DenseNet	GDAS	PNet	Average
F_1	99.91	92.21	91.54	89.98	78.7	43.27	79.14
F_2	99.89	92.15	91.53	89.94	78.52	42.42	78.91
M_1	100	94.88	94.92	93.82	81.68	49.19	82.90
M_2	100	95.18	94.9	94.09	82.5	50.46	83.43
L_1	99.93	92.02	91.59	89.69	78.54	42.58	78.88
L_2	100	92.23	91.74	90.08	79.31	43.07	79.29
F_1L_1	99.93	91.93	91.5	89.78	78.68	43.03	78.98
F_2L_2	99.99	92.04	92.09	90.28	79.47	43.13	79.40
M_1L_1	100	95.16	95.03	94.26	82.88	50.8	83.63
M_2L_2	100	95.32	95.46	94.46	83.1	51.44	83.96
F_1M_1	100	95.12	95	94.19	83.02	50.21	83.51
F_2M_2	100	95.29	95.16	94.19	82.66	50.61	83.58

Table 5.3: Success rates of the variations with removing more ReLU from middle part on CIFAR-10 dataset

Attack	VGG-19$^*_{bn}$	ResNeXt	WRN	DenseNet	GDAS	PNet	Average
M_2L_1	100	95.12	95.11	94.04	82.34	50.95	83.51
M_3L_1	99.99	95.27	95	94.33	83.55	51.6	83.95
M_3L_2	100	94.92	94.96	93.92	82.69	50.5	83.34
$F_1M_1L_1$	100	93.13	92.83	91.65	78.87	46.31	80.55
$F_2M_2L_2$	100	91.41	90.86	89.69	77.07	41.92	78.19
$F_3M_3L_3$	100	88.76	88.21	86.81	72.14	38.34	74.85

To understand the impact of the middle and last part of architecture even more, We experimented while removing more number of ReLU layers from these parts. In Table 5.3, results show that just increasing more number of non-linear units from these parts does not help much in increasing transferability. We also tried to combine all three parts in $F_1M_1L_1$ where we remove the ReLU layer from each part while varying numbers, but in those as well, results for transferability rates do not improve.

In Table 5.4, we present the comparison with the state-of-the-art methods to increase the transferability by removing ReLU layers from the middle and last part only. We compute the transferability using FGSM, PGD, I-FGSM, and M-DI^2-FGSM attacks. It can be seen from Table 5.4 that M-DI^2-FGSM generates more transferable examples than other attacks for all victim models. Moreover, the I-FGSM attack can be combined with ILA [13] and LinBP [12]. It can be observed that all DNN models can be fooled by the adversarial examples generated on the source model with very high confidence for both I-FGSM and M-DI^2-FGSM attacks. It may be noted that the combination of modified LinBP with ILA attack

Table 5.4: Comparison on the success rates of different methods of attacks on CIFAR-10

Attack	VGG-19$^*_{bn}$	ResNeXt	WRN	DenseNet	GDAS	PNet	Average
FGSM [4]	91.11	55.85	49.39	50.45	54.07	18.17	45.59
PGD [39]	92.32	56.6	50.13	51.43	55.52	17.76	46.29
I-FGSM [12]	99.96	64.82	64.32	61.92	49.98	16.6	51.53
ILA + I-FGSM [12]	99.96	88.08	87.66	85.7	72.52	33.74	73.54
LinBP + I-FGSM [12]	100	92.06	91.6	89.86	77.1	41.3	78.38
M-DI^2-FGSM [14]	99.85	75.64	75.76	72.94	63.04	25.93	62.66
Our Result	100	95.32	95.46	94.46	83.1	51.44	83.96

having M-DI^2-FGSM as the baseline attack gives the best transferability. One can observe that the result for PyramidNet [32] is on the lower side. It may be attributed to the fact that it is the best classification model available for CIFAR-10 as it is trained using sophisticated data augmentation [37] and regularization techniques [38].

Table 5.5 displays the visual representation of original images from CIFAR-10 dataset and their perturbed images with our method. In First 3 rows, attack successfully able to fool source model as well as all target models whereas in last 3 rows, attack only able to fool source model and not fooling any of the target models.

One can observe in Table 5.6 that the results of our method (combination of modified LinBP, ILA, and Diverse input patterns) are better than the current works. The reason is that the ILA does not consider perturbation, whereas LinBP disregards the importance of intermediate layers. M-DI^2-FGSM considers data-related aspects. Our method considers all three aspects: linearization, focus on intermediate layers, and data augmentation. Thus our method produces the best transferability on all the victim models. The results emphasize the importance of intermediate layers for linearization as well as the perturbation.

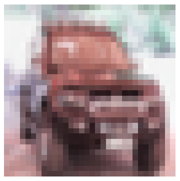
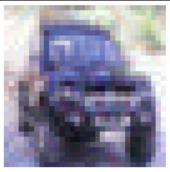
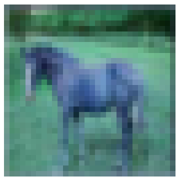
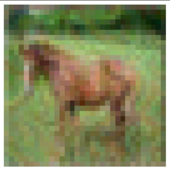
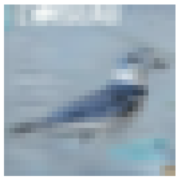
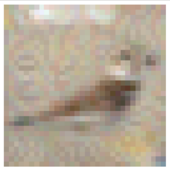
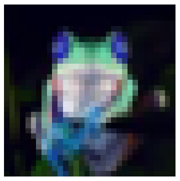
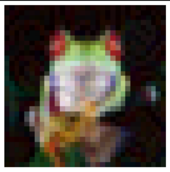
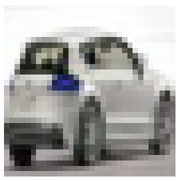
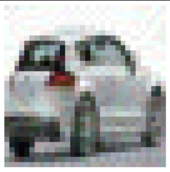
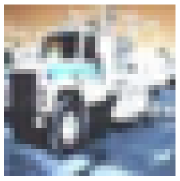
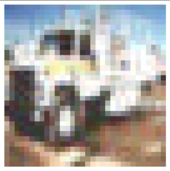
Input Image	Original Class	Perturbed Image	Source Model	Target Models
	Automobile		Truck	Truck
	Horse		Deer	Deer
	Bird		Cat	Cat
	Frog		Dog	Frog
	Automobile		Airplane	Automobile
	Truck		Automobile	Truck

Table 5.5: Visual results of perturbed images from CIFAR-10[28] dataset generated with our method

Table 5.6: Analyzing the effectiveness of different components on CIFAR-10 (I-FGSM as base attack)

Attack	VGG-19^*_{bn}	ResNeXt	WRN	DenseNet	GDAS	PNet	Average
Linearity	100	94.23	93.79	92.75	81.64	48.28	82.14
Linearity + Intermediate Perturb	100	95.14	94.99	94.32	82.8	51.38	83.73
Linearity + Data Augment	100	93.6	93.32	91.73	79.65	43.91	80.44
Intermediate Perturb + Data Augment	99.92	92.17	91.54	90.28	78.44	43.08	79.10
Our Result	100	95.32	95.46	94.46	83.1	51.44	83.96

CHAPTER 6

Conclusions and Future Work

6.1 Conclusions

Higher the transferability, the more vulnerable the state-of-the-art DNN models will be to black-box attacks, thus posing a severe threat to the reliability of the deployed DNN models. Therefore, to assess the threat in this work, we present a method for increasing the transferability of the adversarial examples in the black-box setting. Notably, we modified the architecture of the source model minimally so as to decrease the non-linearity in the model by removing a few intermediate ReLU layers. We have performed a series of experiments using two state-of-the-art adversarial attacks, namely IFGSM and M-DI^2-FGSM. Supplementing these attacks with minor modifications in the model architecture specifically, decreasing non-linearity during backpropagation yields high transferability.

Further, we observed that removing the non-linear layers from the intermediate convolutional blocks results in superior transferability of adversarial examples. With lesser modifications in the architecture of the source model than similar methods, our method achieves the same adversarial success rate in the white-box setting and identical transferability in the black-box setting. We improve the transferability further by using the M-DI^2-FGSM as the attack that uses data augmentation and integrates the ILA attack with our method. The observations partially confirm the linearity hypothesis as reducing non-linearity from architecture improves transferability, thus raising security issues for developing more robust deep learning models.

6.2 Future Work

To make these attacks more powerful, we can extend the enhancement of transferability for targeted variation of the state-of-the-art attacks. Another possible future work is to validate the linearity hypothesis on architecture with different non-linear units other than ReLU.

6.3 Publication

Meet Shah, Shruti Bhilare, Srimanta Mandal and Avik Hati, "Increasing Transferability by Imposing Linearity and Perturbation in Intermediate Layer with Diverse Input Patterns", IEEE International Conference on Signal Processing and Communications (SPCOM), Bangalore, July 2022 (Accepted).

References

[1] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

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

[4] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

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

[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] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[8] Nina Narodytska and Shiva Prasad Kasiviswanathan. Simple black-box adversarial perturbations for deep networks. *arXiv preprint arXiv:1612.06299*, 2016.

[9] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

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

[11] Ekin Dogus Cubuk, Barret Zoph, Samuel Stern Schoenholz, and Quoc V. Le. Intriguing properties of adversarial examples. *arXiv preprint arXiv:1711.02846*, 2017.

[12] Yiwen Guo, Qizhang Li, and Hao Chen. Backpropagating linearly improves transferability of adversarial examples. *CoRR*, abs/2012.03528, 2020.

[13] Qian Huang, Isay Katsman, Horace He, Zeqi Gu, Serge J. Belongie, and Ser-Nam Lim. Enhancing adversarial example transferability with an intermediate level attack. *CoRR*, abs/1907.10823, 2019.

[14] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille. Improving transferability of adversarial examples with input diversity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2730–2739, 2019.

[15] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[16] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4312–4321, 2019.

[17] Gabriel Machado, Eugenio Silva, and Ronaldo Goldschmidt. Adversarial machine learning in image classification: A survey towards the defender's perspective. *CoRR*, abs/2009.03728, 2020.

[18] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14, 11 2017.

[19] Xiaosen Wang and Kun He. Enhancing the transferability of adversarial attacks through variance tuning. *CoRR*, abs/2103.15571, 2021.

[20] Dongxian Wu, Yisen Wang, Shu-Tao Xia, James Bailey, and Xingjun Ma. Skip connections matter: On the transferability of adversarial examples generated with resnets. *CoRR*, abs/2002.05990, 2020.

[21] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *CoRR*, 11 2016.

[22] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *CoRR*, 05 2017.

[23] 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*, pages 582–597, 05 2016.

[24] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *CoRR*, 02 2017.

[25] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations, 2017.

[26] Kang Deng, Anjie Peng, Wanli Dong, and Hui Zeng. Detecting c amp;w adversarial images based on noise addition-then-denoising. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3607–3611, 2021.

[27] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings, 2015.

[28] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. CIFAR-10 (Canadian Institute for Advanced Research). *URL https://www.cs.toronto.edu/ kriz/cifar.html*, 5(4):1, 2010.

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[30] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016.

[31] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[32] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. *CoRR*, abs/1610.02915, 2016.

[33] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *ICLR*, 2017.

[34] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016.

[35] Xuanyi Dong and Yi Yang. Searching for A robust neural architecture in four GPU hours. *CoRR*, abs/1910.04465, 2019.

[36] 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 H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[37] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. AutoAugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[38] Yoshihiro Yamada, Masakazu Iwamura, and Koichi Kise. Shakedrop regularization. *CoRR*, abs/1802.02375, 2018.

[39] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.