TU WIEN Informatics

Comparison and Evaluation of the attacks and defenses against Adversarial attacks

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Logic and Computation

eingereicht von

Aleksandar Jankovic
Matrikelnummer 01636308

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Privatdoz. Mag.rer.soc.oec. Dipl.-Ing. Dr.techn. Edgar Weippl
Mitwirkung: Univ.Lektor Mag.rer.soc.oec. Dipl.-Ing. Rudolf Mayer

Wien, 13. Oktober 2021

Aleksandar Jankovic	Edgar Weippl

TU WIEN Informatics

Comparison and Evaluation of the attacks and defenses against Adversarial attacks

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Logic and Computation

by

Aleksandar Jankovic
Registration Number 01636308

to the Faculty of Informatics

at the TU Wien

Advisor: Privatdoz. Mag.rer.soc.oec. Dipl.-Ing. Dr.techn. Edgar Weippl
Assistance: Univ.Lektor Mag.rer.soc.oec. Dipl.-Ing. Rudolf Mayer

Vienna, 13th October, 2021

Aleksandar Jankovic	Edgar Weippl

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Aleksandar Jankovic

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 13. Oktober 2021

Aleksandar Jankovic

Acknowledgements

This master thesis represents the crown of my educational career and comes as a result of hundreds of hours of hard work. I am very grateful for finding myself in this position. I want to thank my advisors Edgar Weippl and Rudolf Mayer, who made this work possible, for their guidance, support, and commitment.

I especially want to thank my colleague and fiancee, Andrea Milakovic, for her endless support. Words cannot express how much it meant to me her encouragement and thoughtfulness throughout this journey.

I am also very grateful to my friends and family for their support. In particular, I want to thank my parents for always being there for me. Without their support and guidance, I would not be here where I am today.

Kurzfassung

In den letzten Jahren hat Deep Learning den Bereich des maschinellen Lernens dominiert und konventionelle Techniken in Bereichen wie Sprach-, Bild- und Texterkennung übertroffen. Diese Domänen haben eine sehr große praktische Bedeutung, weshalb Deep Learning auch viel Aufmerksamkeit erhält. Dies hat zu Anwendungen von Deep Learning Techniken bei sicherheitskritischen Aufgaben geführt. Neural Networks sind jedoch anfällig für "adversarial" Beispiele, gut ausgearbeitete kleine Anpassung der Eingabe. Daher ist die Frage der Widerstandsfähigkeit und Sicherheit von Deep Learning Modellen zu einem wichtigen Thema geworden.

In dieser Arbeit werden mehrere state-of-the-art white-box- und black-box Angriffe, wie Carlini und Wagner L^2 und L^∞, HopSkipJump und Universal Perturbations, gegen state-of-the-art Convolutional Neural Networks (CNN) in der Bilderkennungsdomäne unter verschiedenen Zieleinstellungen verglichen und bewertet. Verschiedene Abwehrtechniken, wie zum Beispiel "adversarial" Training und Pre-processing Abwehr, werden gegen solche Angriffe verwendet, um die Verbesserung der Widerstandsfähigkeit der CNNs zu bewerten. Darüber hinaus wird eine Kombination dieser Abwehrmechanismen getestet, in der Hoffnung, neue Abwehrmechanismen mit einer erhöhten Widerstandsfähigkeit zu bekommen.

Die Experimente zeigen, dass der Einsatz von Abwehrmechanismen notwendig ist, um CNNs eine höhere Widerstandsfähigkeit zu verleihen, insbesondere in der ungezielten Zieleinstellung. Die Auswertung der Ergebnisse zeigt, dass "adversarial" Training im Vergleich zu Pre-Processing Abwehr ähnliche Widerstandsfähigkeit bietet, jedoch mit dem Preis einer geringeren Genauigkeit des Modells auf den Originaldaten. Die Pre-Processing Techniken waren dagegen sehr effektiv gegen "adversarial" Angriffe, die "adversarial" Beispiele erzeugten, die den Originalbildern, gemessen in der Metrik L^2, sehr nahe kommen.

Abstract

Over the past few years, deep learning has been dominating the field of machine learning, outperforming conventional techniques in domains such as speech, image, and text recognition. These domains have very big practical significance, which is also why deep learning is receiving a lot of attention. This has led to applications of deep learning techniques in safety-critical tasks. However, neural networks are vulnerable to adversarial examples, well-crafted small perturbations of the input. Therefore, the question of robustness and security of deep learning models has become a major concern, indirectly also affecting safety.

In this thesis, several state-of-the-art white-box and black-box adversarial attacks, like Carlini and Wagner L^2 and L^∞, HopSkipJump and Universal Perturbations, are compared and evaluated against state-of-the-art Convolutional neural networks (CNN) under different target settings in the image recognition domain. Additionally, defense techniques against such attacks, like adversarial training and pre-processing defenses, are used to evaluate the improvement of the robustness of the CNNs. Furthermore, a combination of these defenses is tested with the hope to potentially obtain new defenses that have an increased level of robustness.

The experiments show that the use of defense mechanisms is necessary to provide CNNs with a higher level of robustness, especially in the untargeted setting. The evaluation of results indicates that adversarial training provides a similar level of robustness compared to pre-processing techniques, but with the cost of lower accuracy of the model on the original data. The pre-processing techniques were very effective against adversarial attacks that generated adversarial images, which are very close to the original images measured in the L^2 metric.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Over the past few years, deep learning has been dominating the field of machine learning, outperforming conventional techniques in domains such as speech, image, and text recognition. For example, in the image recognition domain, they can recognize objects with very high accuracy. Other domains where deep neural networks also excel include natural language processing and playing games. These domains have very big practical significance, which is also why deep learning is receiving a lot of attention. This has led to applications of deep learning techniques in safety-critical tasks. For example, in healthcare, convolutional neural networks are used to classify skin cancer based on photographic images, referable diabetic retinopathy based on optical coherence tomography (OCT) images of the retina, and pneumonia based on chest X-ray images. Another example, in autonomous vehicles, convolutional neural networks are used to recognize road signs.

As promising results in this field kept piling up, and machine learning systems are consequently employed in an increasing number of systems involved in potentially autonomous decision making, naturally, the question of security of machine learning models is raised. Imagine if the model fails to recognize skin cancer. That would potentially lead to fatal patient outcomes. If the model fails to recognize the STOP sign and the vehicle does not stop, it would lead to a dangerous situation that could result in a car accident. Therefore, the question of robustness and security of deep learning models has become a major concern, indirectly also affecting safety.

In recent years, researchers have demonstrated several ways to successfully attack machine learning models, but also, to some extent, to defend against these attacks.

1.2 Threat model

We discuss in this section, how to model threats against the mentioned attacks. We define the attacker's goal, knowledge, and capability of manipulating the input data, to subsequently define an optimization problem corresponding to the optimal attack strategy, which we mainly base on the discussion by Biggio and Roli (2018) [BR18]. The solution to this problem provides a way to manipulate input data to achieve the attacker's goal.

Attacker's Goal

The attacker's goal is defined in the following terms:

- *Security violation.* The attacker may aim to cause: an integrity violation, i.e. to evade detection without compromising normal system operation; an availability violation, i.e. to compromise the normal system functionalities available to legitimate users; or a privacy violation, to obtain private information about the system, its users or data by reverse-engineering the learning algorithm. Integrity, availability, and confidentiality are also known as the CIA triad and represent the fundamental principles of information security [Per08].
- *Attack specificity.* The attack can be either targeted or untargeted. Targeted attacks aim to cause the model to misclassify a specific set of samples (to target a given system user or protected service), while with the untargeted attacks the attacker aims to cause misclassification of any sample (to target any system user or protected service).
- *Error specificity.* It can be either specific if the attacker aims to have a sample misclassified as a specific class; or generic, if the attacker aims to have a sample misclassified as any of the classes different from the true class.

Attackers Knowledge

Additionally, different attack scenarios can be described based on the attacker's knowledge of the targeted system. This includes different levels of knowledge such as the training data, the feature set, the learning algorithm along with the objective function and possibly even its trained hyper-parameters.

- *White-box setting* - attacker is assumed to know everything about the targeted system.
- *Gray-box setting* - attacker is assumed to know something about the targeted system.
- *Black-box setting* - attacker is assumed to know nothing about the targeted system.

The white-box setting allows one to perform a worst-case evaluation of the security of the learning algorithm and is the main focus of this thesis.

Attacker's Capability

This characteristic depends on the influence that the attacker has on the input data and application-specific data manipulation constraints.

- *Attack influence.* It can be causative if the attacker can manipulate both training and test data, or exploratory if the attacker can only manipulate test data. These scenarios are more commonly known as poisoning and evasion attacks.
- *Data manipulation constraints.* Another aspect related to the attacker's capability depends on the presence of application-specific constraints on data manipulation. For example, to evade malware detection, malicious code has to be modified, but without compromising its intrusive functionality.

Summary

Table 1.1 shows a categorization of these attacks based on the attacker's goals (CIA triad) and capabilities to manipulate test and/or training data.

	Integrity	**Availability**	**Confidentiality**
Test data	Evasion	-	Model extraction / stealing and model inversion
Training data	Poisoning (to allow subsequent intrusions) - e.g. backdoors or neural network Trojans	Poisoning to maximize classification error	-

Table 1.1: Attacks against machine learning models based on the discussed threat model, taken from Biggio and Roli (2018) [BR18]

Evasion attacks, a.k.a adversarial attacks are the type of attacks that this thesis will be focused on. These are attacks that manipulate the test data, aimed to maximize the test error. They use imperceptible, adversarially chosen perturbations to the test data. In many cases, the changes done are so subtle that a human observer does not even notice them, but the classier still make a mistake. They can be successfully performed even if the adversary has no access to the underlying model, i.e. in a black-box setting, which increases security concerns even more and most existing machine learning classifiers are highly vulnerable to them.

To give a practical example of an adversarial attack, if we were to feed a state-of-the-art image recognition model a picture of an animal, say a panda, the model should recognize the panda. But, one can add a specifically tailored noise to this image, which would result in the model recognizing the panda as a shoe instead. The noise aims to be minimal so that to the human eye, the original and modified image look the same.

To this date, different types of adversarial attacks and defenses have been proposed. However, only a few defense mechanisms can be used to defend against more than one attack type.

1.3 Aim of the Work

This thesis focuses primarily on image recognition applications. The first goal is to develop recommendations for which defense mechanisms work best for which types of adversarial attacks under given settings and datasets. These recommendations will be created based on the results obtained through empirical tests and evaluations of adversarial attacks and defenses. The experiments will be conducted on different image datasets under various settings. The robustness of neural networks against adversarial attacks will be measured with different metrics.

Another goal is to improve detection of and defenses against adversarial attacks. The detection of adversarial attacks means that the model can detect adversarial examples as adversarial and refuse to classify them, whereas the defense means that the model can classify the adversarial examples correctly, despite the adversarial perturbation. Therefore, a combination of existing defenses will be tested with the hope to potentially obtain new defenses that have an increased level of robustness. Such attempts have proven to be fruitful in the past, since some proposed defense mechanisms are a fusion of two or more defense mechanisms, like Fine-pruning defense against backdooring attacks proposed by Liu et al. (2018) [LDG18].

This thesis aims to answer the following research questions:

RQ1. Against the state-of-the-art adversarial attacks, which defense mechanisms have the highest level of robustness, measured using different robustness evaluation metrics?

The practical significance of the neural networks implies the importance of the robustness of neural networks against these attacks. We aim to investigate the current state-of-the-art in the field of adversarial defenses. And in addition to answering the proposed question, we aim to analyze the possible defense strategies that could potentially offer higher levels of robustness against these attacks.

RQ2. How do different neural network architectures impact the robustness of the defense mechanisms?

Adversarial examples that are adversarial on one model are often also adversarial on another model, even if the two have different architectures or were trained on

different training sets, so long as both models were trained to perform the same task. However, it is not clear to which extent does this holds and whether the defense mechanisms then provide the same (or at least similar) levels of robustness as when the adversarial examples are generated with the same (defended) model. To showcase the importance of this topic, consider that an attacker may train their substitute model, craft adversarial examples against the substitute, and transfer them to a victim model, with very little information about the victim. For this reason, it is important to better understand the transferability behavior of different neural networks regarding adversarial attacks. We aim to answer this question by evaluating defense mechanisms with several state-of-the-art neural network architectures.

RQ3. How do different characteristics of datasets impact the robustness of the defense mechanisms?

Different image datasets have different characteristics, such as the size of an image, level of details, resolution, quality, etc. These differences have a big impact on what architectures and accuracy results do the models achieve on these datasets. For example, a simpler dataset means that a less complex model is needed to obtain high accuracy scores. We look to investigate these differences with adversarial attacks and defenses by performing the same experiments on different datasets.

RQ4. How well do defense mechanisms work against universal perturbations?

Different from other attacks, universal perturbations represent modifications that are applied to each image, i.e. they are not custom for each image, but the same for all attacked samples. It is a different approach to generating adversarial images that should be considered when evaluating adversarial attacks. In our experiments, we aim to evaluate different state-of-the-art defense mechanisms against universal perturbations.

RQ5. To which extent can we improve the defense against adversarial attacks by adapting or combining different defense mechanisms?

Since there is no clear best state-of-the-art defense that defends against all or most of the adversarial attacks, any improvement in that direction would be considered good. By its definition, most of the defense mechanisms are compatible with other defense techniques. This allows us to, for example, stack them together with the hope of achieving better results. We aim to investigate and analyze such possibilities.

1.4 Structure of the work

The rest of this thesis is organized as follows:

Chapter 2 Background presents an overview of machine learning techniques that are needed to follow the rest of the thesis. The chapter is mainly focused on Convolutional Neural Networks.

Chapter 3 State-of-the-art presents the existing state-of-the-art adversarial attacks and defense mechanisms in the image recognition domain. Additionally, an adversarial robustness metric called CLEVER is discussed.

Chapter 4 Experimental Setup presents the experimental setup of attacking and defending the pre-trained models. The datasets used in the experiments are also described.

Chapter 5 Experimental results chapter presents the experimental results, failed attempts, and also discusses other observations from the experiments.

Chapter 6 Summary and Conclusion chapter summarizes and concludes the work of this thesis, to form guidelines for best defenses against different adversarial attacks. This includes the analysis of the modifications and combinations performed on the defense mechanisms.

CHAPTER 2

Background

This chapter briefly explains the background needed to follow the work of this thesis. First, we explain the basics of machine learning, followed by the introduction to neural networks and their core features. We close this chapter with a discussion on convolutional neural networks (CNNs).

2.1 Machine Learning

One of the first definitions of machine learning is attributed to the work of Samuel [Sam59] as "a field of study that gives computers the ability to learn without being explicitly programmed". A more precise and formal definition of machine learning was given by Tom Mitchell in his book Machine Learning (1997) [Mit97] as "a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". For this ability "to learn from experience", machine learning is seen as a part of artificial intelligence. Over the past two decades, machine learning has become one of the most popular technologies. It is used almost everywhere, from automating mundane tasks to offering intelligent insights. Some of the main tasks that it tackles are classification, image and speech recognition, medical diagnosis, financial trading, and so on. In this section, we focus on aspects of machine learning that are important for the scope of this thesis, including a brief introduction to the image classification problem, so that the reader can follow the upcoming chapters.

Image classification is a supervised learning approach, where the goal is to learn a function $f : \mathbb{R}^n \mapsto \{1, ..., k\}$ that maps an n-dimensional input vector to one or more of k classes. There is another variant of the function f, where f outputs a probability distribution over all k classes, thus stating how likely each class is. The learning process is called training, whereas the trained function is referred to as a trained model. Models that are used for classification tasks are also called classifiers. The best state-of-the-art algorithms

for most image classification tasks are convolutional neural networks, a special type of neural network that we discuss later in this chapter.

One of the main properties of supervised learning approaches is that a set $D = \{(x_i, y_i) : i \in [N]\}$ of N input-output pairs is given. The output elements y_i are called the ground truth (or, sometimes, the gold standard). They carry the information of which is the correct class for the corresponding input x_i. The set D is called the dataset.

Naturally, we are interested in knowing the effectiveness of the learned function f, i.e. knowing how good does the function f classifies the input images. Different measures describe this, and they vary depending on the domain of the problem, but the most commonly used measure is the *accuracy* of the model, which is the ratio of the correctly classified images over the total number of classified images. For example, if the classifier correctly classifies 758 out of 1 000 images, then its accuracy is 75.8 %.

To simulate how the model performs and behaves in production we measure its performance on a set of input data that the model did not see during the learning (training) process. This set of data is called the *test set* and is often obtained by splitting the dataset D into two parts before the training process starts. The other part is then used for training and is called the *training set*.

Furthermore, the training set is often split into two disjoint subsets, the training set, and the validation set. The validation set is used e.g. to optimize hyperparameters, which are parameters that control the behavior of the learning process, or for deciding on an early stopping criterion. We start the training by fitting models with different hyperparameters on the training subset. Next, we evaluate the candidate models on the validation set, and then we select the one with the best performance. We call this process hyperparameter tuning. Hence, the main purpose of the validation set is to verify whether an increase of accuracy on the training subset also yields an increase of accuracy on a set of the model unknown images.

The optimal scenario that we are aiming to achieve is that a trained classifier has high accuracy during training, and then also on the test set. However, a scenario that can occur is that a classifier has high accuracy on the training and validation sets, but low on the test set. In this case, we say the classifier overfits.

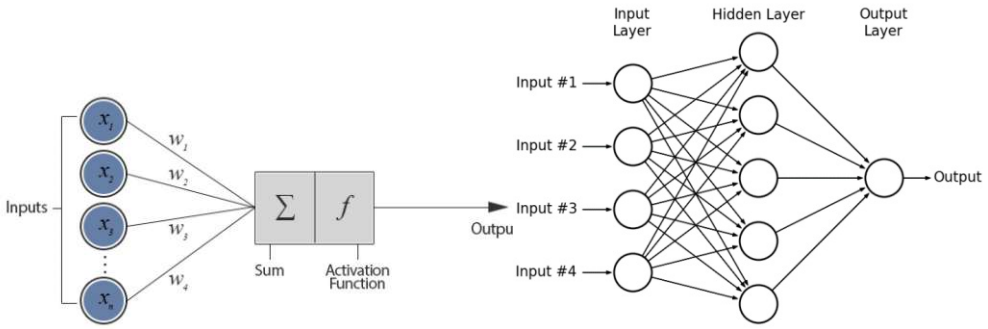
Deep learning and Convolutional Neural Networks are the most popular field of machine learning in the domain of image classification tasks, and we thus cover these in detail in the following. In Section 2.2 we give an introduction to neural networks, followed by introductions to Gradient Descent in Section 2.3 and Backpropagation in Section 2.4. The Gradient Descent is used for the optimization of cost functions, whereas the Backpropagation is used to compute a derivation of a composition of functions in backward direction within neural networks. Lastly, we discuss Convolutional Neural Networks (CNNs) and some of the most important architectures of CNNs in Section 2.5.

2.2 Feedforward Neural Networks

Feedforward neural networks [ZMH+94] are the simplest type of artificial neural network [Dre90]. A good understanding of how these networks work helps one understand and follow more complicated architectures such as Convolutional [LGTB97] and Recurrent Neural Networks [MC01].

The goal of a feed-forward neural network is to approximate some function f. Say that this function f maps an input x to a value y. A feedforward neural network defines a mapping $y = f(x;\theta)$ and learns the value of the parameters σ that then hopefully results in the best function approximation. This approximation function is a composition of different functions. The first function in the composition is called the input layer, whereas the last function is called the output layer. The functions in-between are called hidden layers. If there are no hidden layers, then a feedforward neural network is known as a single-layer perceptron [Ros60], otherwise, it is known as a multi-layer perceptron. The number of hidden layers is also referred to as the depth of a neural network. An example of the architecture of a multi-layer perceptron and one of its neurons can be seen in Figure 2.1.

A feed-forward network is a directed graph, whose vertices are called neurons, and whose edges are called weights. Neurons compute an output by applying the activation function of the sum of inputs multiplied with weights, i.e. $output = f(\sum_{i=1}^{n} x_i w_i)$. Edges (weights) define the flow of information, which in the case of the feedforward neural networks is, as the name suggests, is in forward direction only.



(a) A Neuron with an Activation Function [1] (b) Multi-layer Perceptron Architecture [2]

Figure 2.1: An example of a Multi-layer Perceptron and a single neuron within this architecture

When using these networks in a classification task, a so-called softmax layer is used as the output layer, which outputs a vector of probabilities that a given input belongs

[1] Figure taken from https://www.researchgate.net/figure/A-hypothetical-example-of-Multilayer-Perceptron-Network_fig4_303875065

[2] Figure taken from https://www.cc.gatech.edu/~san37/post/dlhc-fnn/

to a specific class. The sum of all these probabilities equals 1 and the predicted class corresponds to the index with the highest probability.

2.3 Gradient Descent

The learning process mentioned previously means that we are trying to find the best weights. To express how wrong the model is on a given input for some weights, we use loss functions. Naturally, we want that our model makes as few mistakes as possible, which leads us to the optimization problem of minimizing the loss function.

For classification tasks, the most common loss function used is cross-entropy, which is used to quantify the difference in probability distributions p and $q \in \mathbb{R}^n$ and is defined as:

$$H(p, q) = -\sum_{i=1}^{n} p_i \times \log q_i \tag{2.1}$$

If we denote p to be the ground-truth label and q to be the predicted softmax probabilities, then H measures how (dis)-similar are the true and predicted probabilities on a single sample. On the entire dataset, cross-entropy is then defined as:

$$L(\theta) = \frac{1}{|D|} \sum i = 1|D|H(p_i, q_i) \tag{2.2}$$

where p_i and q_i are the ground-truth label and predicted softmax probabilities for the sample i. Other commonly used loss functions include Mean Square Error, Mean Absolute Error, etc.

In general, optimization methods are divided into two groups, constrained and unconstrained. Unconstrained methods are further divided into closed-form and iterative form methods. To give an example of closed-form methods, consider that the steepest descent converges when every element of the gradient is zero, or very close to zero. Hence, we can in simple cases easily solve the equation $\nabla_x f(x) = 0$ for x. If the loss function has many variables, a closed-form method could become very complicated and extremely expensive to calculate. This is where iterative methods work better.

Gradient Descent [Lem12] is such an iterative optimization method. Recall from calculus that for a function $f : \mathbb{R}^N \mapsto \mathbb{R}$ the vector that contains all partial derivatives $\nabla_x f(x)$ of the function f w.r.t x is called the gradient. Gradient descent is the technique of moving x in small steps with the opposite sign of the derivative. In other words, the positive gradient points uphill, and the negative downhill. Hence by moving in the direction of the negative gradient we get lower and lower values of our cost function.

Formally, if we are at point x, the next point x' is calculated as:

$$x' = x - \alpha \nabla_x f(x) \tag{2.3}$$

where α represents the learning rate, i.e. the size of the step. The choice of learning great can have a big impact on the overall results. Larger learning rates have a higher chance

of missing the global minimum, as the learning curve will show large oscillations of the cost function values. In contrast, a smaller learning rate leads to a slow convergence – if the learning rate is too low, the learning process may even get stuck with high-cost value.

The pseudo-code for this method is outlined in Algorithm 2.1. Note that the gradient is calculated on the whole dataset.

Algorithm 2.1: Gradient descent

Input: x, f, ϵ

do

  $x' = x - \epsilon \nabla_x f(x)$

while $x' = \vec{0}$;

The only requirement of Gradient Descent is that f must be differentiable. It is a simple and efficient optimization method, but it also has limitations. For a good generalization of a problem, we should have a large training set, which comes with an increased computational cost. As the training set grows in size, the time it takes to make a single gradient step becomes longer and the algorithm eventually becomes infeasible.

To overcome this challenge, instead of evaluating the algorithm on a whole dataset, we take a random subset, called mini-batch, of the training samples of some cardinality C, at each step of the algorithm. An algorithm obtained this way is called the Stochastic Gradient Descent (SGD) [Bot10]. It is called stochastic because samples are selected randomly for each iteration. The cardinality of the mini-batches is, for efficiency reasons, usually taken to be a power of 2 (64, 128, 256, or 512). It is important to note that the mini-batches have to be sampled randomly. The learning rate is a crucial parameter for SGD. It is recommended to decrease the learning rate during the execution. That means that our ϵ now depends on the iteration and is denoted as ϵ_k.

2.4 Backpropagation

Feedforward neural networks process the input x in the forward direction and produce an output y. The loss function f is then applied on y. By using backpropagation [RDGC95], the flow of data goes back through the layers, to compute the gradient. To achieve this, the so-called chain rule is used:

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} \tag{2.4}$$

This rule gives the way to calculate derivatives of a composition of functions. If you recall, an edge from one neuron to another is a simple function. Hence, moving forward from one layer to another means that we are building a composition of compositions of functions. This is where we need the chain rule. It allows us to go back one step at a time calculating the partial derivatives and in the end the entire gradient.

2.5 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [LGTB97] are a type of feedforward neural network that is primarily used in the field of pattern recognition within images. The name convolutional is derived from the mathematical operation convolution, which these networks aim to imitate. The reason for their supreme application in the image recognition tasks lies in the fact that one of the largest limitations of traditional forms of neural networks is that they struggle with spatial structures, partially, due to the computational complexities involved.

CNNs are made of three types of layers: convolutional layers, pooling layers, and fully connected layers. An example of a CNN architecture is shown in Figure 2.2.

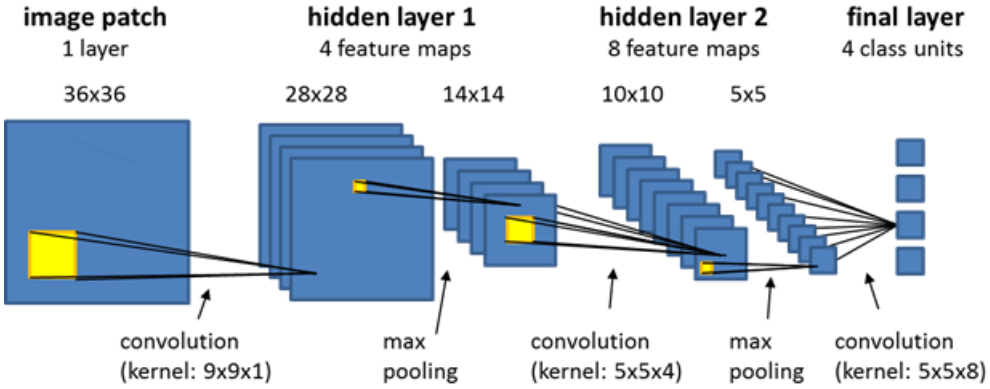


Figure 2.2: A graphic example of CNN layers [3]

The convolutional layer is the most important layer of CNNs. It uses learnable kernels that have smaller width and height, but the same depth as the input. The layer convolves each filter across the height and width of the output to create an activation map, a mapping that corresponds to the activation of different parts of the image. As we move through the input, the layer calculates the scalar products for each value in the kernel. By doing this, the layer obtains kernels that are triggered by specific features. This means that every kernel has its corresponding activation map. This layer reduces significantly the number of weights, while still also successfully learning the features of the input.

The activation function that is most commonly used in CNNs is the ReLU (rectified linear unit), defined as:

$$ReLU(x) = max(0, x) \tag{2.5}$$

Figure 2.3 shows an illustration of the ReLU function. Other common activation functions include Sigmoid, Hyperbolic Tangent (Tanh), Leaky ReLU, etc.

[3] Figure taken from https://aditimukerjee.medium.com/step-by-step-vgg16-implementation-in-keras-for-beginners-for-image-classification-problem-cbeec9c0d7a3

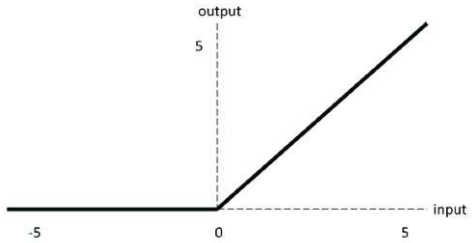


Figure 2.3: ReLU Activation function

The main goal of a pooling layer is to further reduce the dimensions of the input and the number of parameters. In the case of a max-pooling layer, the layer operates over each activation map and scales its dimensionality using the max function. In other words, it takes only the maximum output in the scanned neighborhood.

Finally, the fully connected layer consists of multiple standard feed-forward, fully connected neural network layers used to predict the class probabilities based on the features extracted by previous convolutional and pooling layers.

Even though the CNNs are made of only three types of layers, there is no clear recipe for designing a CNN architecture. Over the years, several well-working CNN architectures have been proposed for different tasks, scoring state-of-the-art results. In this thesis, we use three of these well-known CNN architectures, namely ResNet, Inception, and MobileNet, which we discuss in the following sections. They have been chosen because they are state-of-the-art architectures that vary in complexity, performance, and accuracy.

2.5.1 Deep Residual Network Architecture

With Deep Residual Network (ResNet), proposed by He et al. (2015) [HZRS16], we can train models with up to hundreds of layers, while maintaining efficiency. The proposal of ResNet architecture has improved the results in tasks such as image classification, object detection, or face recognition. It scored the first position at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2015 competition.

The ResNet architecture successfully tackled one of the biggest problems when adding more and more layers is the problem of the vanishing gradient. As the gradient is backpropagated to previous layers, the big number of multiplication operations can make the gradient infinitely small, thus saturating the performance of the network.

The main base element of this architecture is the residual block. As can be seen in Figure 2.4, the residual block takes an input and if it does not have any additional weight layers, it simply outputs the identity, meaning the input itself. If there are some additional weight layers the block tries to learn some delta, a "residual" from the input X.

[4]Figure taken from https://neurohive.io/en/popular-networks/resnet/

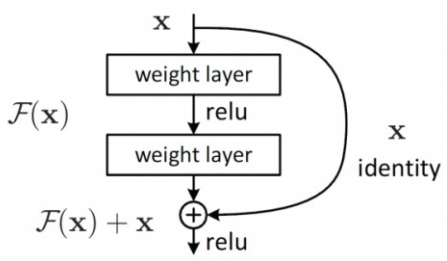


Figure 2.4: A residual block in ResNet architecture [4]

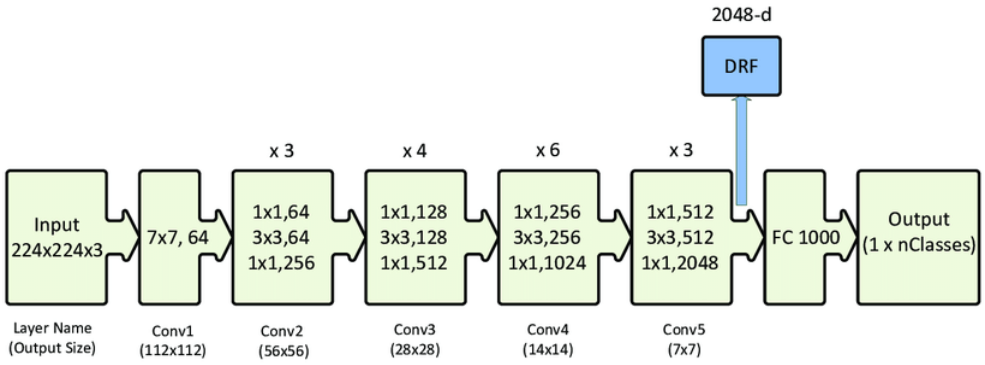


Figure 2.5: The architecture of ResNet-50, taken from [MOB+20]

We have many variants of ResNet architecture, ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-110, etc. The trailing number indicates the number of neural network layers, the concept of the architecture is the same between these variants. Figure 2.5 shows the entire architecture of the ResNet-50.

2.5.2 Inception Neural Network Architecture

The Inception architecture by Szegedy et al. (2014) [SLJ+15] is complex, heavily engineered, and has been evolving constantly. So far, several versions of this architecture are proposed, InceptionV1 (a.k.a GoogleNet) [SLJ+15], InceptionV2 [SVI+16], InceptionV3 [SVI+16], InceptionV4 and Inception-ResNet [SIVA17].

The core idea behind this architecture and also the reason why the network is called Inception is the following. If we observe an image of a dog, then the area occupied by the dog is different in each image. In some images, the dog occupies a big portion of the image, but in other a rather small portion of the image. Because of the huge variation in the size and position of the relevant information, choosing the right kernel size for the convolution layers becomes hard.

[5]Figure taken from https://www.programmersought.com/article/90163515785/

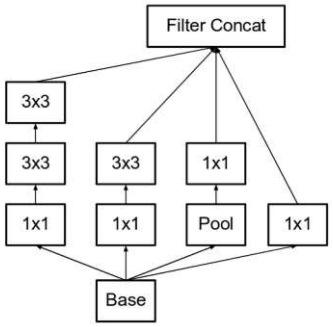


Figure 2.6: An example of an inception block [5]

A larger kernel is preferred for images where the dog occupies a bigger portion of the image, and likewise, a smaller kernel is preferred for images where the dog occupies a smaller portion. But what if we add filters with multiple sizes that operate on the same level? Figure 2.6 shows an example of an inception block from the first version. It performs convolution on an input with 3 different sizes of filters (1×1, 3×3 and 5×5). Max pooling is also performed, the outputs are concatenated along the depth dimension and sent to the next inception module.

As stated before, deep neural networks are computationally expensive. To make it cheaper, the authors reduce the number of input channels by adding an extra 1×1 convolution before the 3×3 and 5×5 convolutions. A 1×1 convolution simply maps an input pixel with all its channels to an output pixel. To see how do 1 convolutions reduce the input, consider the following example. Given an input of size $28 \times 28 \times 192$, if we only apply 5×5 convolutions with same 32 filters we get an output of size $28 \times 28 \times 32$. The computational cost of computing this output is $28 \cdot 28 \cdot 32 \cdot 5 \cdot 5 \cdot 192$, which totals to approximately 120M.

On the other hand, if we first apply 1×1 with 16 same filters and then apply 5×5 convolutions with the same 32 filters, we again get an output of size $28 \times 28 \times 32$. This time, the computational cost is $(28 \cdot 28 \cdot 16 \cdot 192 \cdot 1 \cdot 1) + (28 \cdot 28 \cdot 32 \cdot 5 \cdot 5 \cdot 16)$, which total to approximately 12.4M. Thus, even though adding an extra operation may seem counter-intuitive, 1×1 convolutions are far cheaper than 5×5 convolutions, and the reduced number of input channels contributes to cheaper computation.

Figure 2.7 shows the architecture of the first version of Inception networks, InceptionV1, which was popularly known as GoogLeNet.

Lastly, we discuss some of the tweaks made between the second and third version. In InceptionV2 [SVI+16], the 5×5 convolution has been split into two 3×3 convolutions to improve computational speed. Authors have shown empirically that a 5×5 convolution

[6] Figure taken from https://towardsdatascience.com/10-invaluable-tips-tricks-for-building-successful-neural-networks-566aca17a0f1

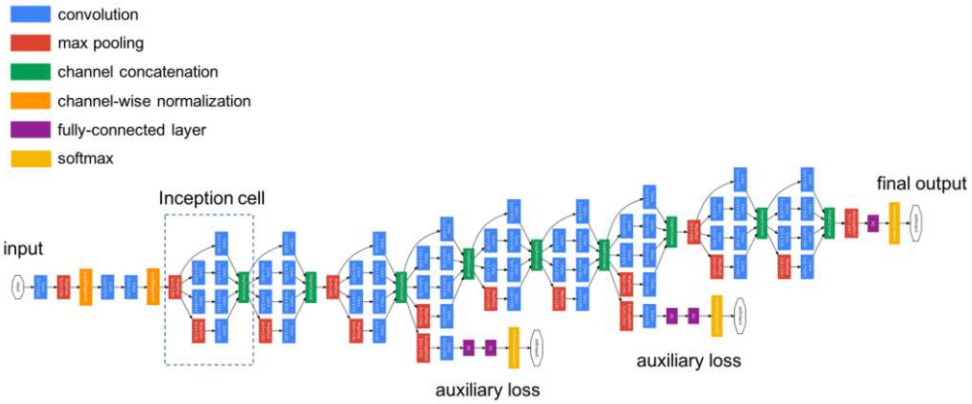


Figure 2.7: The architecture of InceptionV1 (also known as GoogleNet) [6]

is 2.78 times more expensive (parameter-wise) than a 3×3 convolution. This means that stacking two 3×3 convolutions together indeed leads to a boost in performance.

Convolutions of size $N \times N$ have been split into a combination of $1 \times N$ and $N \times 1$ convolutions. An $N \times N$ convolution is equivalent to first performing a $1 \times N$ convolution, and then performing a $N \times 1$ convolution on its output. But the authors have measured empirically that a combination of, for example, a 1×3 and a 3×1 convolution is 33% cheaper than a single 3×3 convolution. These two alterations to the inception block allowed for 7×7 convolutions to be included in the architecture without significantly impacting the performance in the InceptionV3.

2.5.3 MobileNet Neural Network Architecture

Howard et al. (2017) [HZC+17] introduced a neural network architecture optimized for mobile devices. They wanted to create a model that could deliver high accuracy, but also that could keep the number of parameters and the number of mathematical operations as low as possible. Achieving this was necessary to bring neural networks to low-performance devices such as smartphones.

The MobileNet architecture uses depth-wise separable convolutions, which are made from two operations, depth-wise convolution and point-wise convolution. The architecture thus significantly reduces the number of parameters when compared to networks with regular convolutions and with the same depth. This idea originated from the idea that a filter's depth and spatial dimension can be separated.

To explain the depth-wise separable convolutions and the improvements that they bring, we will use a simple example. Say, we have an input image of size $12 \times 12 \times 3$. If we perform convolution using a $5 \times 5 \times 3$ kernel, we will get an output of the size $8 \times 8 \times 1$.

[7] Figure taken from https://www.pluralsight.com/guides/transfer-learning-in-deep-learning-using-tensorflow-2.0

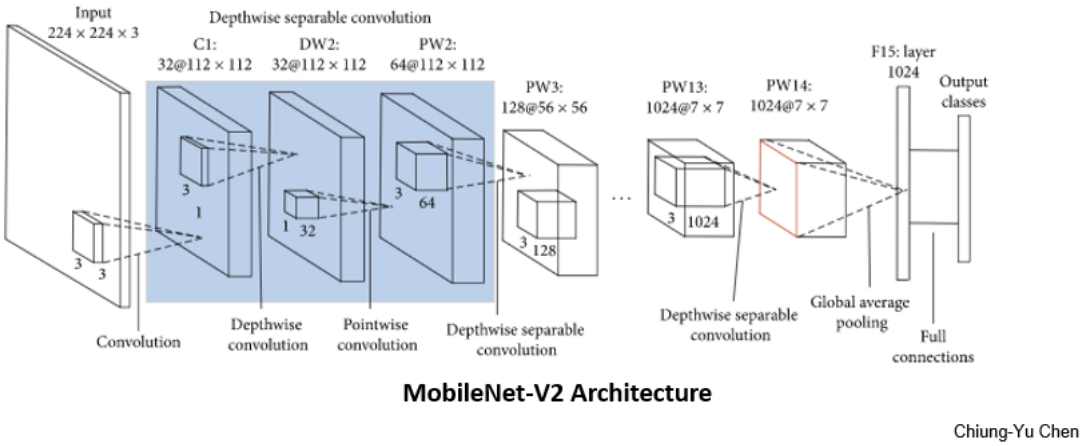


Figure 2.8: The architecture of MobileNetV2 [7]

Additionally, during the convolution operations, we specify that we need N number of channels in output. This means that the same operation is repeated N times with different kernels. Suppose $N = 10$. The total computational cost in this example equals to $8 \cdot 8 \cdot 5 \cdot 5 \cdot 3 \cdot 10 = 48\,000$.

Now, consider first the depth-wise convolution. We have three channels for input and we have three $5 \times 5 \times 1$ kernels. A $5 \times 5 \times 1$ kernel iterates over the corresponding channel of the input image to produce an $8 \times 8 \times 1$ output. Next, we stack all three of these outputs together and obtain an output of the size $8 \times 8 \times 3$.

Next, consider the point-wise convolution. A $1 \times 1 \times 3$ kernel is used on the $8 \times 8 \times 3$ output from the previous step. We repeat this with 10 different $1 \times 1 \times 3$ kernels to produce 10 feature maps and stack these features maps together. This creates an output of the size $8 \times 8 \times 10$, the same as in the standard case. The total computational cost with depth-wise separable convolutions equals to $8 \cdot 8 \cdot 5 \cdot 5 \cdot 3 \cdot 8 \cdot 8 \cdot 10 \cdot 3 = 4\,800 + 1\,920 = 6\,720$, which is a computational improvement of more than 7 times.

The entire MobileNet architecture consists of 28 convolutional layers and 1 fully connected layer, followed by a softmax layer, as can be seen in Figure 2.8.

CHAPTER 3

State-of-the-Art

In this chapter, we will discuss the adversarial attacks and defenses that this thesis relies on. These attacks and defenses are currently state-of-the-art in this field. In addition to this, we are also going to discuss an adversarial robustness metric called CLEVER, which is used to measure and compare the robustness of models.

3.1 Adversarial Attacks

Szegedy et al. [SZS+14] first noticed the existence of adversarial examples in the image classification domain. They showed that despite their high accuracies, current state-of-the-art neural networks are surprisingly vulnerable to adversarial attacks. These attacks add small perturbations to images, unrecognizable to the human eye. Such small transformations can cause a neural network to change its prediction. What is even worse, the attacked models usually report high confidence for the adversarial images.

The degree to which attackers can find adversarial examples impacts the applications in which neural networks can be used. For instance, in self-driving cars, adversarial examples could allow an attacker to cause the car to take unwanted actions, such as ignoring certain traffic signs. A successful attack like that could lead to a tragic outcome, as it would impact safety. This is just one example, but it clearly shows the dangerous potential of adversarial attacks. This has inspired research on how to harden neural networks against these kinds of attacks. However, since the discovery of Szegedy et al, researchers have also proposed many new ways to successfully craft adversarial examples.

This thesis primarily assumes a white-box setting, which means that the adversary has complete access to a neural network, including the architecture and the parameters. The reason for this assumption is twofold. On the one hand, white box is the harder setting to defend against – if CNNs can be protected when the attacker knows all the information about its target, then one could conclude that CNNs can be protected in the same way

and to the same degree even when the attacker knows less. On the other hand, Papernot et al. [PMG16] have shown that it is possible to train a substitute model given black-box access to a target model, and by attacking the substitute model, we can then transfer these attacks back to the target model. Thus, it is possible to transform, to some extent, an originally black-box setting into a white(r) box setting.

Based on their importance, performance, and diversity, we have selected the following adversarial attacks, which we compare and evaluate in our experiments:

- Fast Gradient Sign Method (FGSM) [GSS15]

  Published in 2014, this attack has become arguably the most popular adversarial attack, because it is simple and is capable of crafting close adversarial examples fast. It is optimized for the L_∞ norm, which means that authors used this norm as a distance metric to quantify the similarity between original and adversarial samples. Since its publication, many changes and adaptations have been proposed that improve the attack's success rate to overcome newly proposed defense techniques. This attack is discussed in more detail in Section 3.1.1.

- Carlini and Wagner L_2 and L_∞ attacks (CW) [CW17]

  Published in 2016, these attacks are still one of the strongest attacks known. Apart from proposing the attacks, Carlini and Wagner have made another contribution. Initially, Papernot et al [PMW+16] proposed a defense technique called distillation and have believed that they have solved the problem of adversarial attacks since the distillation technique worked on all of that moment known adversarial attacks. However, Carlini and Wagner have refuted that statement, using their L_p attacks as counter-examples. These attacks are discussed in more detail in Section 3.1.2.

- Universal Perturbations [MFFF17]

  While the FGSM and CW attacks make transformations specific to one single image to fool the model on that image, a universal adversarial perturbation is able to fool a network on many images with the same perturbation, also with high confidence. These image-agnostic perturbations also remain quasi-imperceptible. This attack is discussed in more detail in Section 3.1.3.

- HopSkipJump [CJW20]

  Since we are primarily interested in the white-box setting, this is the only black-box setting attack that we consider. It is based on a novel estimate of the gradient direction using binary information at the decision boundary. The proposed attack includes both untargeted and targeted attacks optimized for L_2 and L_∞ metrics respectively. The authors have empirically shown that the HopSkipJump attack is also strong against several state-of-the-art defense techniques. This attack is discussed in more detail in Section 3.1.4.

- Shadow Attack [GSG20]

  Shadow attack is a generalization of the well-known Projected gradient descent (PGD) [MMS+18] attack, a descendent of the aforementioned FGSM attack. It exploits the labeling function of a classifier. The proposed method applies large perturbations that place images far from a class boundary while maintaining the imperceptibility property of adversarial examples. This attack is discussed in more detail in Section 3.1.1.

3.1.1 Fast Gradient Sign Method (FGSM)

Intuitively, for each pixel, the Fast Gradient Sign method [GSS15] uses the gradient of the loss function to determine in which direction the pixel's intensity should be changed (whether it should be increased or decreased) to minimize the loss function. Then, it shifts all pixels simultaneously. More formally, given an image x the Fast Gradient Sign method sets:

$$x' = x - \epsilon \cdot sign(\nabla loss_{F,t}(x)) \tag{3.1}$$

where ϵ is chosen to be sufficiently small to be undetectable, F is the full neural network including the softmax layer, and t is the target label.

It is important to note that the fast gradient sign attack was designed to be fast, rather than optimal.

Basic Iterative Method (BIM)

Sometimes referred to as the Iterative Gradient Sign method, Basic Iterative Method (BIM) [KGB17] is a simple improvement of the FGSM, where a single step of size ϵ in the direction of the gradient sign is replaced with multiple smaller steps α. Additionally, the result is clipped by the same ϵ.

To be more precise, we set the initial sample to:

$$x'_0 = x \tag{3.2}$$

and then in each iteration, the next samples is defined as:

$$x'_i = clip(x'_{i-1} - \alpha \cdot sign(\nabla loss_{F,t}(x'_{i-1}))) \tag{3.3}$$

Projected Gradient Descent (PGD)

The Projected Gradient Descent attack [MMS+18] is an iterative method in which, after each iteration, the perturbation is projected on an l_p-ball of specified radius. That is done in addition to clipping the values of the adversarial sample so that the samples lie in the permitted data range.

The attack is formulated as a constraint optimization problem to find a perturbation that maximizes the loss function used to train the CNN model such that the crafted

perturbation is inside the L_p ball of the original sample:

$$\max_{\delta} loss(x + \delta) \tag{3.4}$$

such that

$$\|\delta\|_p \leq \epsilon. \tag{3.5}$$

As it can be seen in Equation (3.5), the L_p ball represents all attacked samples whose L_p distance to the given sample is smaller or equal to ϵ. The algorithm can be summarized with the following steps:

1. Start from a random perturbation in the L_p ball around the given sample;
2. Take a gradient step in the direction of greatest loss;
3. Project perturbation back into L_p ball if necessary:
4. Repeat steps 2 and 3 until convergence.

More formally, the next sample is defined as:

$$x'_i = \prod_{x+S} (x'_{i-1} + \alpha \cdot sign(\nabla_x loss_{F,t}(x))) \tag{3.6}$$

This attack is a more sophisticated version of the BIM and was proposed for adversarial training purposes, which we discuss later in this chapter in Section 3.2.2.

Shadow Attack

Instead of solving the constrained optimization problem that the PGD attack solves, the Shadow attack [GSG20] solves the following problem featuring a range of penalties:

$$\max_{\delta} loss(x + \delta) - \lambda_c C(\delta) - \lambda_{tv} TV(\delta) - \lambda_s Dissim(\delta) \tag{3.7}$$

where λ_c, λ_{tv}, λ_s are scalar penalty weights.

Penalty $TV(\delta)$ forces the perturbation δ to have small total variation (TV), and so appear more smooth and natural.

Penalty $C(\delta)$ limits the perturbation δ globally by constraining the change in the mean of each color channel c. This constraint is needed since the total variation is invariant to constant/scalar additions to each color channel, and it is desirable to suppress extreme changes in the color balances of images.

Penalty $Dissim(\delta)$ promotes perturbations δ that assume similar values in each color channel. In the case of an RGB image of shape $3 \times W \times H$, if $Dissim(\delta)$ is small, then the perturbations to red, green, and blue channels are similar, i.e., $\delta_{R,w,h} \approx \delta_{G,w,h} \approx \delta_{B,w,h}, \forall (w, h) \in W \times H$. This amounts to making the pixels darker/lighter, without changing the color balance of the image. Two effective ways of enforcing such similarity between RGB channels are proposed:

- 1-channel attack that strictly enforces $\delta_{R,i} \approx \delta_{G,i} \approx \delta_{B,i}, \forall i$ by using just one array to simultaneously represent each color channel $\delta_{W \times H}$. On the forward pass, we duplicate δ to make a 3-channel image. In this case, $Dissim(\delta) = 0$, and the perturbation is greyscale.
- 3-channel attack that uses a 3-channel perturbation $\delta_{3 \times W \times H}$, along with the dissimilarity metric $Dissim(\delta) = \|\delta_R - \delta_B\|_p + \|\delta_R - \delta_G\|_p + \|\delta_B - \delta_G\|_p$.

Altogether, the three penalties minimize the perception of perturbations by forcing them to be a) small, b) smooth, and c) without dramatic color changes. At the same time, these penalties allow perturbations that are very large in L_p-norm.

3.1.2 Carlini and Wagner (CW)

Carlini and Wagner [CW17] proposed three different attacks, one for each of the following L_p norms, L_0, L_2 and L_∞. These attacks are still among the strongest attacks. For this thesis, we use L_2 and L_∞ attacks, as they are stronger than the L^0 attack.

Carlini and Wagner formally define the problem of finding an adversarial transformation for a sample x as follows:

$$\begin{aligned} &\text{minimize } D(x, x+\delta) \\ &\text{s.t. } C(x+\delta) = t \\ &x + \delta \in [0,1]^n \end{aligned} \tag{3.8}$$

where x is fixed, D represents the distance metric, either L_0, L_2 or L_∞, and the goal is to find δ that minimizes $D(x, x+\delta)$. That is, we want to find some small change δ that we can make to an image x, such that it changes the classification of that image, but such that the result is still a valid image, i.e. the pixel values are in [0, 1].

Since the above formulation is difficult for existing algorithms to solve directly, as the constraint $C(x+\delta) = t$ is highly non-linear, the problem is reformulated as an appropriate optimization instance that can be solved by optimization algorithms. An objective function f such that $C(x+\delta) = t$ if and only if $f(x+\delta) \leq 0$ is defined. A total of seven possible choices for f are empirically tested and the following definition was found to be the most effective:

$$f(x') = (\max_{i \neq t}(Z(x')_i) - Z(x')_t)^+ \tag{3.9}$$

At this point, the formulation of the problem becomes:

$$\begin{aligned} &\text{minimize } \|\delta\|_p + c \cdot f(x+\delta) \\ &x + \delta \in [0,1]^n \end{aligned} \tag{3.10}$$

Another empirical result is that the best way to choose c is to use the smallest value of c for which the resulting solution x^* has $f(x^*) \leq 0$. This causes gradient descent to minimize both of the terms simultaneously instead of picking only one to optimize over first.

L_2 attack

Given x, they chose a target class t (such that $t \neq C^*(x)$) and then they searched for w that solves

$$\text{minimize } \left\| \frac{1}{2}(\tanh w + 1) - x \right\|_2^2 + c \cdot f(\frac{1}{2}(\tanh w + 1)) \tag{3.11}$$

with f defined as

$$f(x') = \max(\max Z(x')_i : i \neq t - Z(x')_t, -\kappa). \tag{3.12}$$

This f is based on the best objective function found earlier, modified slightly so that one can control the confidence with which the misclassification occurs by adjusting κ.

L_∞ attack

The L_∞ distance metric is not fully differentiable and standard gradient descent does not perform well for it. The $\|\delta\|_\infty$ term only penalizes the largest (in absolute value) entry in δ and has no impact on any of the other. As such, gradient descent very quickly becomes stuck oscillating between two sub-optimal solutions.

Carlini and Wagner resolved this issue by using an iterative attack. They replaced the L_2 term in the objective function with a penalty for any terms that exceed τ (initially 1, decreasing in each iteration). This prevents oscillation, as this loss term penalizes all large values simultaneously. Specifically, in each iteration they solve

$$\text{minimize } c \cdot f(x + \delta) + \sum_1 [(\delta_i - \tau)^+] \tag{3.13}$$

After each iteration, if $\delta_i < \tau, \forall i$, they would reduce τ and repeat; otherwise, they would terminate the search.

3.1.3 Universal Perturbations

Let μ denote a distribution of images in $\mathbb{R}^d$, and F define a classification function that outputs for each image $x \in \mathbb{R}^d$ an estimated label $F(x)$. The main focus of this attack is to find perturbation $\delta \in \mathbb{R}^d$ that fool the classifier F on most samples from μ. This perturbation is called universal because it is a fixed perturbation that does not depend on a particular sample and that causes label change for most images in the dataset [MFFF17].

More formally, the attack is searching for δ such that $F(x + \delta) \neq F(x)$ for "most" $x \sim \mu$. Furthermore, this kind of perturbation is desired to be small in terms of the L_p norm with $p \in [1, \infty)$). The goal is therefore to find δ that satisfies the following two constraints:

$$\begin{aligned} \|\delta\|_p &\leq \xi, \\ P_{x \sim \mu}(F(x + v) \neq F(x)) &\geq 1 - \gamma. \end{aligned} \tag{3.14}$$

The parameter ξ controls the magnitude of the perturbation vector δ, and γ quantifies the desired fooling rate for all images sampled from the distribution μ.

The experiments indeed showed the existence of such small universal perturbations. Furthermore, universal perturbations generalize well across different classification models and result in universal perturbations that are both image- and network-agnostic. The authors believe that an explanation for the existence of such transferable perturbations is probably due to the geometrical correlations between different regions of the decision boundary.

3.1.4 HopSkipJump

HopSkipJump [CJW20] is a query algorithm and a decision-based generator of adversarial examples which utilizes distance-vector as a hyperparameter. It is based on an estimate of the gradient direction using binary information at the decision boundary. Proposed are both untargeted and targeted versions of the attack, which are optimized for l_2 and l_∞ similarity metrics respectively.

The algorithm works as follows. For an untargeted attack, it is initialized with a sample in the target class, while for a targeted attack, the initialization is with a misclassified sample blended with uniform noise. Each iteration of the algorithm has three major steps.

- Push the iterate from the last iteration towards the decision boundary by using the so-called binary search algorithm,
- Estimate the gradient direction,
- Update step size along the gradient direction and decrease it via geometric progression[1] until perturbation becomes successful.

The next iteration starts with projecting the perturbed sample back to the boundary again. Figure 3.1 shows an intuitive visualization of the three steps in l_2 metric. For a detailed technical explanation of how the algorithm works, please refer to the original paper [CJW20].

Experiments show that the HopSkipJump attack requires significantly fewer model queries than several state-of-the-art decision-based adversarial attacks, like Boundary Attack [BRB18], Limited Attack [IEAL18] and Opt Attack [CLC+19]. It also achieved competitive performance in attacking several widely-used defense mechanisms, such as defensive distillation [PMW+16] and adversarial training Section 3.2.2. Through the experimental analysis, the authors also suggest that the HopSkipJump attack can be used as a simple and efficient first step for researchers to evaluate new defense mechanisms.

[1] i.e. a sequence where each subsequent term after the first is found by multiplying the previous term by a fixed number

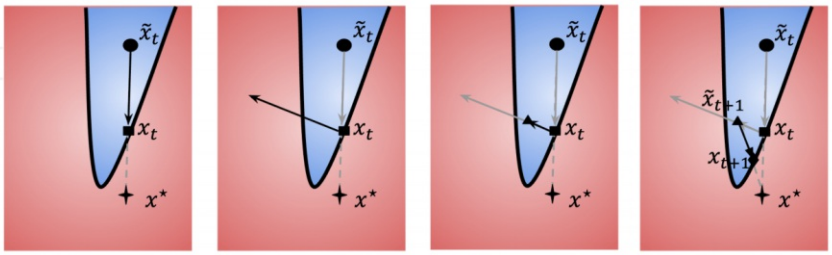


Figure 3.1: Intuitive illustration of HopSkipJump attack, taken from [CJW20]. (a) Perform a binary search to find the boundary. (b) Estimate the gradient at the boundary point. (c) Geometric progression. (d) Perform another binary search

3.2 Adversarial Defenses

Adversarial examples demonstrate that most of the state-of-the-art neural networks are vulnerable and can be fooled. Since this discovery, the research community has been working on practical defenses against adversarial examples. However, adversarial examples are hard to defend against, due to the following reasons:

- A theoretical solution to the process of generating adversarial examples is difficult to construct. This is due to adversarial sample crafting being a non-linear and non-convex complex optimization problem. If we do not have the proper theoretical background to describe the solution to these optimization problems, then it is equally hard to derive any theoretical conclusions that a given defense indeed improves robustness against adversarial examples.
- If a defense mechanism makes a considerable modification to the model, then this may affect the ability of the model to correctly predict legitimate, unmodified inputs. The increase of robustness could thus lead to a general drop in the effectiveness of the model.

Furthermore, most of the current defense strategies are not fit against all types of adversarial attacks, as one method may mitigate one kind of attack, but be still vulnerable to some other types of adversarial attack.

The current defense mechanisms against adversarial attacks can be split into four distinct categories [NST+18]:

1. Preprocessor – a type of defense that does not modify the model, but instead the input (image) that the model is supposed to classify. The modification is done, as the name suggests before the image is passed to the model. Many different preprocessing defense mechanisms were proposed in recent years, and we will discuss selected approaches in Section 3.2.1

2. Trainer – this defense technique is used to re-train the model in a specific way so that it is more robust to adversarial samples. The input images and the outputs are not modified, but adversarial examples crafted by the defender are added to the training set. An example of this defense is the so-called Madry's Protocol defense [MMS+18], which performs adversarial training with a PGD attack as the adversary. These defenses are described in Section 3.2.2.

3. Transformer - this defense technique is somewhat similar to the trainer technique, as the model is also re-trained. However, transformers can also introduce an additional transformation to the network architecture. An example of this defense is the so-called Defensive Distillation technique [PMW+16]. At the moment of publication, the authors believed that they have solved the problem of adversarial attacks, mainly because it was believed that the reason adversarial examples exist is due to "blind spots" (as Szegedy et al. (2013) [SZS+13] call them) in highly non-linear neural networks. However, the publication of the Carlini and Wagner [CW17] attacks showed that this is incorrect. Namely, these attacks have a 100 % success rate on both distilled and undistilled neural networks.

4. Detector - these defense techniques primarily attempt to detect the adversarial images. In other words, they try to find the subtle adversarial perturbations added to inputs. This technique is utilized more in other types of attacks on neural networks, such as Poisoning attacks. Nonetheless, there are several successful detectors for adversarial examples. For example, Detector based on the Fast Generalized Subset Scan [SSR+18], which adapts the so-called subset scanning methods from the anomalous pattern detection domain to the task of detecting adversarial examples for neural networks. The anomalous pattern detection task arises in many domains: customs monitoring, where we attempt to discover patterns of illicit container shipments; disease surveillance, where we must detect emerging outbreaks of disease in the very early stages; network intrusion detection, where we attempt to identify patterns of suspicious network activities; and various others.

In this thesis, we focus primarily on the first two categories, preprocessors and adversarial training for several reasons. We are more interested in defenses and their combinations that enable the classifier to correctly classify images despite the adversarial noise. Furthermore, pre-processors and adversarial training are currently state-of-the-art, whereas for the best transformer defense (Defensive Distillation [PMW+16]) Carlini and Wagner [CW17] have proposed attacks that are not affected by this defense at all. In the following sections, we discuss some of the best-known defenses from these two categories.

3.2.1 Preprocessors

Spatial Smoothing

The Spatial Smoothing [XEQ18] defense is a part of the approach called Feature Squeezing proposed by Xu et al. (2017) [XEQ18]. The core motivation behind feature squeezing is

that the feature input spaces in image recognition tasks are often very large, and hence enable a lot of freedom for an adversary to craft adversarial examples. Therefore, the goal is to shrink the adversary's freedom by "squeezing" out unnecessary input features. After the original input is preprocessed, both the original and preprocessed inputs are given to the model to classify. If the predictions are significantly different, then the input is regarded as an adversarial example. Xu et al. (2017) focus primarily on two simple types of Feature Squeezing: reducing the color depth of images and using smoothing (local and non-local) to reduce the variation among pixels. Namely because of its superiority, we use the latter type in this thesis, also known as Spatial Smoothing, which is a group of techniques widely used in image processing for reducing image noise, sometimes regarded as blur

Local smoothing methods make use of the nearby pixels to smooth each pixel. There exist different techniques for weighing the neighboring pixels, but this defense uses the so-called median smoothing, also known as median blur or median filter. The median filter runs a sliding window over each pixel of the image and replaces the center pixel with the median value of the neighboring pixels within that window. The number of pixels in the image remains unchanged. The feature squeezing process thus means making neighboring pixels more similar to each other.

The size of the window can be defined from 1 pixel to up to the image size, whereas the shape is square. If the size of the window setting is configured to 1 pixel, then the image remains unchanged, whereas if it is configured to the image size, the image is changed to one color.

Median smoothing is particularly effective at removing sparsely occurring black and white pixels in an image (known as "salt-and-pepper noise"), whilst preserving edges of objects well. Authors reported that empirical results show that it performs especially well against the adversarial attacks based on the L_0 norm.

In contrast, non-local smoothing smooths over similar pixels in a much larger area instead of just nearby pixels.

For this reason, the Spatial Smoothing defense can be easily combined with other defenses, because it does not modify the model, only the input image before it is given to the classifier. The authors have found that this defense works very well when coupled with adversarial training.

JPEG Compression

The inspiration for this defense approach lies in the fact that most datasets that are used for training the neural networks for image classification have been JPG compressed. When a JPG image is transformed into an adversarial example, likely, that image is no longer in the JPG space. This leads to the idea behind that JPEG re-compression [DGR16] could potentially revert the adversarial perturbation, such that the image is no longer adversarial and the classifier outputs the original prediction.

By performing JPG re-compression to reverse adversarial perturbations, the authors have hypothesized that the following could happen:

- Adversarial perturbations could be very sensitive and reverted by most image processing steps
- Adversarial perturbations might be "orthogonal" to the JPG subspace, in which case we would expect the modifications to be removed by JPEG compression.

The former hypothesis was shown to be false – standard image processing does not revert the tailored adversarial perturbations. However, the latter hypothesis was not contradicted. Namely, for small perturbations, JPEG compression was able to successfully revert the adversarial perturbation. On the other hand, if the adversarial perturbations are larger, JPEG compression does not reverse the adversarial perturbation. This result is not promising because these larger perturbations are still barely visible to an untrained human eye.

Even though this defense alone does not provide strong protection to the model, it is fairly simple and can be easily combined with other defenses. For this, we speculate that JPEG Compression combined with other defenses could increase the robustness of the model.

Total Variance Minimization

Guo et al. (2018) [GRCvdM18] proposed a new defense technique against adversarial attacks that is a Compressed Sensing approach [2] that combines pixel dropout with total variation minimization (Rudin et al., 1992 [ROF92]). The technique works by randomly selecting a small set of pixels and reconstructing an image that is consistent with the selected pixels. Because the adversarial perturbations are usually small and localized, the reconstructed image is not adversarial anymore.

More formally, they first select a random set of pixels by sampling a Bernoulli random variable $X(i, j, k)$ for each pixel location (i, j, k). A pixel is maintained when $X(i, j, k) = 1$. Next, they use total variation minimization to constructs an image z that is similar to the (perturbed) input image x for the selected set of pixels, whilst also being "simple" in terms of total variation by solving:

$$\min_z ||(1 - X) \odot (z - x)||_2 + \lambda_{\text{TV}} \cdot \text{TV}_p(z) \tag{3.15}$$

where $\odot$ denotes element-wise multiplication, and $\text{TVp}(z)$ represents the L_p-total variation of z:

$$\text{TV}_p(Z) = \sum_{k=1}^{K} \left[\sum_{i=2}^{N} ||z(i, :, k) - z(i - 1, :, k)||_p + \sum_{j=2}^{N} ||z(:, j, k) - z(:, j - 1, k)||_p \right]. \tag{3.16}$$

[2] Compressed Sensing is a signal processing technique for efficiently acquiring and reconstructing a signal [DDEK12]

The total variation (TV) measures the amount of fine-scale variation in the image z, which leads to the removal of adversarial perturbations by Total Variance Minimization.

Wang et al. (2016) [WGZ+16] proposed a strategy for adversary-resistant deep neural networks, stating that a strong input-transformation defense should be non-differentiable and randomized. Total variation minimization fulfills both of the properties:

- It is difficult to differentiate because it involves solving a complex minimization of a function that is inherently random.
- It randomly selects the pixels that it uses to reconstruct the image.

The randomness of the defenses is important because it makes it difficult to attack the model - it implies that the adversary has to find a perturbation that changes the prediction for the entire dataset, which is harder than attacking a single image as shown in [MFFF17].

Lastly, the authors note that Total Variation Minimization has an advantage over adversarial-training approaches, because adversarially trained networks are differentiable, implying that they can be successfully attacked easier. An additional disadvantage of adversarial training stated by the authors is that it is based on one specific adversarial attack, whereas the Total Variance Minimization generalizes well across different adversarial attacks.

3.2.2 Adversarial Training

Adversarial training defense works by generating adversarial examples on the fly during training and including the newly crafted adversarial example into the training set. However, the correct choice of the adversary is not clear. The research of this defense technique started with the choice of the FGSM attack, mainly because of its speed of crafting adversarial robustness and the fact that many state-of-the-art adversarial attacks are extensions and generalizations of the FGSM.

One of the most significant results for this defense technique comes from the work of Madry et al. (2017) [MMS+18], who have shown empirically that FGSM adversaries do not increase robustness for large ϵ. This is due to the that when training the network using adversarial examples generated with the FGSM, it can be observed that the network easily overfits these adversarial examples. They believed that this behavior stems from the fact that the adversary produces a very restricted set of adversarial examples that the network can overfit on. Moreover, these networks have poor performance on natural examples and do not exhibit any kind of robustness against PGD adversaries. On the other hand, for the case of smaller ϵ the loss is often linear enough in the L_∞-ball around natural examples so that the FGSM finds adversarial examples close to those found by PGD. Thus being a good choice of an adversarial attack to train against.

Instead of the FGSM, Madry et al. (2017) proposed to use their PGD attack as the re-training adversary. This became the default adversarial training defense and has also remained very robust since the publications. However, this approach comes at a non-trivial additional computational cost, often increasing training time by an order of magnitude over standard training because the adversarial images are generated in each training step and for all samples in the batch.

Due to this high computational overhead, scaling to more complex state-of-the-art neural networks was difficult. Training such a neural network is already a computational challenge on its own. To overcome this issue, the last couple of years have seen many improvements in the performance of the algorithms that craft adversarial examples. Shafahi et al., 2019 [SNG+19] proposed an algorithm that eliminates the overhead cost of generating adversarial examples by recycling the gradient information computed when updating model parameters. Their so-called "free" adversarial training algorithm achieves comparable robustness to Madry et al.. However, even with such improvements, the training of neural networks is significantly slowed down when adversarial training defense is utilized, depending on the dataset even by the factor of 10.

Wong et al., 2020 [WRK20] showed empirically that adversarial training with FGSM with random initialization is as robust as with PGD. On top of that, adversarial training with FGSM with random initialization combined with techniques for efficient training achieves a training runtime that is significantly faster compared to the overhead-free adversarial training from Shafahi et al., 2019. Specifically, they were able to train an ImageNet model in 12 hours, whereas the same training takes 50 hours with free adversarial training. Thus, adversarial become computationally feasible.

A key difference between these two methods is in one of the key properties of free adversarial training, namely that the perturbation from the previous iteration is used as the initial starting point for the next iteration. However, Wong et al. argued that there is no reason to believe that an adversarial perturbation from the previous mini-batch should be carried over into the next mini-batch. Furthermore, they claimed that the benefit that the free adversarial training has achieved comes from simply starting the mini-batches from a non-zero initial perturbation. With this small modification, Wong et al. achieved a defense as robust as the adversarial training with PGD. Finally, by utilizing the top-performing training methods, such as AIACC-Training [3] from the DAWNBench competition (Coleman et al., 2017 [CNK+17]), they achieve the above mentioned computational improvements.

Lastly, Wong et al. explain what is the reason behind the results that claimed that the FGSM adversarial training overfits the network for larger ϵ. This phenomenon has been the reason why FGSM has been avoided for years in adversarial training. Wong et al. observed empirically that the model would indeed very rapidly appear to overfit to the FGSM adversarial examples. What was previously a reasonably robust model will quickly transform into a non-robust model which suffers 0% accuracy with respect to

[3] https://www.alibabacloud.com/help/doc-detail/162795.htm

PGD attack. They refer to this as "catastrophic overfitting". One of the reasons for this failure lies in the lack of diversity in adversarial examples generated by FGSM. Using a zero initialization or using the random initialization scheme will result in adversarial examples whose features have been perturbed by $-\epsilon, 0, \epsilon$, and so the network learns a decision boundary that is robust only at these perturbation values. On the other hand, the PGD attack can easily overcome such a defense.

3.3 Adversarial Robustness Metrics

3.3.1 CLEVER

CLEVER [WZC+18], which is short for Cross Lipschitz Extreme Value for nEtwork Robustness, is a robustness metric that is attack-agnostic and computationally feasible for large neural networks. The CLEVER score is well supported by the theoretical analysis on formal robustness guarantees and the use of extreme value theory which we briefly present here.

Lemma (Lipschitz continuity and its relationship with gradient norm [PŽ06]). Let $S \subset \mathbb{R}^d$ be a convex bounded closed set and let $h(x) : S \mapsto \mathbb{R}$ be a continuously differentiable function on an open set containing S. Then, $h(x)$ is a Lipschitz function with Lipschitz constant L_q if the following inequality holds for any $x, y \in S$:

$$|h(x) - h(y)| \leq L_q ||x - y||_p, \tag{3.17}$$

where $L_q = \max\{||\nabla h(x)||_q : x \in S\}$, $\nabla h(x) = (\frac{\partial h(x)}{\partial x_1}), ..., \frac{\partial h(x)}{\partial x_d})^T$ is the gradient of $h(x)$ and $\frac{1}{p} + \frac{1}{q} = 1$, $1 \leq p, q \leq \infty$.

Theorem (Formal guarantee on lower bound β_L for untargeted attack). Let $x_0 \in \mathbb{R}^d$ and $f : \mathbb{R}^d \mapsto \mathbb{R}^K$ be a multi-class classifier with continuously differentiable components f_i, and let $c = \arg\max_{1 \leq i \leq K} f_i(x_0)$ be the class which f predicts for x_0. For all $\delta \in \mathbb{R}^d$ with

$$||\delta||_p \leq \min_{j \neq c} \frac{f_c(x_0) - f_j(x_0))}{L_q^j} \tag{3.18}$$

$\arg\max_{1 \leq i \leq K} f_i(x_0 + \delta) = c$ holds with $\frac{1}{p} + \frac{1}{q} = 1$, $1 \leq p, q \leq \infty$ and L_q^j is the Lipschitz constant for the function $f_c(x) - f_j(x)$ in l_p norm. In other words:

$\beta_L = \min_{j \neq c} \frac{f_c(x_0) - f_j(x_0))}{L_q^j}$ **is a lower bound of minimum distortion**.

The intuition behind this Theorem is shown in **??** with a one-dimensional example.

The value $g(x_0 + \delta$ can be bounded by $g(x_0)$, δ and the Lipschitz constant L_q. When $g(x_0) + \delta$ decreases to 0, an adversarial example is found.

Corollary (Formal guarantee on β_L for untargeted attack). Let L_{q,x_0}^j be local Lipschitz constant of function $f_c(x) - f_j(x)$ at x_0 over some fixed ball

$B_p(x_0, R) := \{x \in \mathbb{R} | ||x - x_0||_p \leq R\}$ and let $\delta \in B_p(0, R)$. By the previous Theorem, we obtain the bound in (Hein and Andriushchenko, 2017 [**?**]):

$$||\delta||_p \leq \min\{\min_{j \neq c} \frac{f_c(x_0) - f_j(x_0)}{L^j_{q,x_0}}, R\} \tag{3.19}$$

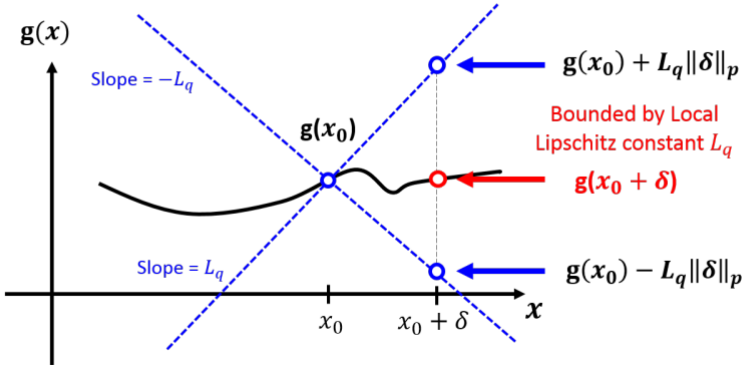


Figure 3.2: The intuition behind the formal guarantee on lower bound β_L for the untargeted attack, taken from [WZC+18].

An important use case of the above-stated Theorem and Corollary is the bound for targeted attacks:

Corollary (Formal guarantee on β_L for targeted attack). Assume the same notation as in the previous Theorem and Corollary. For a specified target class j, we have

$$||\delta||_p \leq \min\{\frac{f_c(x_0) - f_j(x_0)}{L^j_{q,x_0}}, R\} \tag{3.20}$$

Based on this theoretical conclusion, Weng et al. [WZC+18] proposed an algorithm to compute with the aid of extreme value theory a robustness score of a neural network, where CLEVER can be viewed as an efficient estimator of the lower bound β_L and is the first attack-agnostic score that applies to any neural network classifiers. Through experiments, the authors have shown that the CLEVER score corresponds to the practical robustness indication of several state-of-the-art neural networks, even when a defense mechanism is deployed.

One approach to compute L^j_{q,x_0} is through sampling a set of points $x^{(i)}$ in a ball $B_p(x_0, R)$ around x_0 and taking the maximum value of $||\nabla g(x)^{(i)}||_q$. However, a significant amount of samples might be needed to obtain a good estimate of $\max ||\nabla g(x)||_q$ and it is unknown how good the estimate is compared to the true maximum. Fortunately, Extreme Value Theory ensures that the maximum value of random variables can only follow one of

the three extreme value distributions, which is useful to estimate $\max ||\nabla g(x)||_q$ with only a tractable number of samples. The details of this algorithm are out of the scope of this thesis. For the detailed analysis and explanations, please refer to the original publications.

CHAPTER 4

Experimental setup

In this chapter, the experimental setup is described. In Section 4.1 we discuss the datasets (ImageNet, CIFAR-10) and the neural network architectures (MobileNetV2, InceptionV3, ResNet-50 and PreActResNet18) that are used in the experiments. Next, the process of generating adversarial examples with different attacks is presented in Section 4.3. Finally, in Section 4.4 we describe defense strategies and how we evaluate the defenses against the generated adversarial examples.

4.1 Datasets and Neural Network Architectures

Table 4.1: Benchmark datasets used with adversarial attacks and defenses

Dataset	References
ImageNet	[CH20], [MFFF17], [GSS15], [CW17], [XEQ18], [GRCvdM18], [DGR16], [WRK20]
CIFAR-10	[CH20], [MMS+18], [GSS15] [CW17], [MFFF17], [XEQ18], [GSG20], [WRK20]
CIFAR-100	[CH20]
MNIST	[CH20], [MMS+18], [GSS15], [CW17], [XEQ18], [WRK20]

Table 4.1 shows an overview of the datasets of the related work on adversarial attacks and defenses and which datasets they use. Based on this overview, we have selected ImageNet and CIFAR-10 datasets, as two datasets that are more complex and are most commonly used.

CIFAR-10 [KH09] dataset consists of 6 000 examples for each of its 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck). Each image has dimensions of 32×32 pixels. The total of 60 000 images are split into 50 000 training and 10 000 test images. CIFAR-10 is a dataset that is commonly used in research as a benchmark (please refer to Table 4.1 for the overview). The CNN models that we use for this dataset are the PreActResNet18 (from [WRK20]) and adversarially trained version of it that we refer to as AdvTrainedPreActResNet18. The adversarial training that we use is the procedure called "Faster is better than free", discussed in Section 3.2.2. To adversary used in the adversarial training is the Projected Gradient Descent which we discussed in Section 3.1.1. Since we, among others, want also to evaluate the current state-of-the-art adversarial training technique, it also makes sense to use the same model which was used in the paper. Hence the reason why this model has been selected.

For our experiments, we selected randomly a test set consisting of 1000 images from the CIFAR-10 test data, such that all 10 classes are evenly distributed. This test set is used for generating adversarial images. Table 4.2 shows the state-of-the-art accuracy and the accuracy of these two models on our selected test set.

Model	Accuracy on our test set	State-of-the-art accuracy
PreActResNet18	83.0 %	85.18 %
AdvTrained- PreActResNet18	95.1 %	96.03 %

Table 4.2: Model accuracy on the CIFAR-10 dataset

ImageNet [DDS+09] is an image dataset organized according to the WordNet hierarchy. Each meaningful concept in WordNet, possibly described by multiple words or word phrases, is called a "synonym set" or "synset". There are more than 100 000 synsets in WordNet; the majority of them are nouns (80,000+). In ImageNet, they aim to provide on average 1 000 images to illustrate each synset. Images of each concept are quality-controlled and human-annotated, with a total of more than 14 million images. The dataset contains more than 20 000 categories.[1] For our experiments, we selected randomly a test set consisting of 1000 images from the validation images from the ImageNet Large Scale Visual Recognition Challenge 2012. This test set is used for generating adversarial images.

[1]The entire description taken from https://image-net.org/about.php

For ImageNet, we use three state-of-the-art pre-trained CNN models, MobileNetV2, ResNet-50, and InceptionV3. These selected models vary in complexity, performance, and accuracy on ImageNet.

Model	Accuracy on our test set		State-of-the-art accuracy		
	Top-1	Top-5	Top-1	Top-5	No. of params.
InceptionV3	75.8 %	92 %	78.95 %	94.4 %	24M
MobileNetV2	63.2 %	84 %	74.7 %	90.81 %	3.4M
ResNet-50	62.8 %	85.7 %	83.2 %	96.5 %	23M

Table 4.3: Model accuracy on ImageNet dataset

As such, they have been selected to evaluate if and how do different CNN models affect adversarial attacks and defenses. The pre-trained weights are loaded from the Keras and Pytorch frameworks. Table 4.3 shows the state-of-the-art accuracy, the accuracy of the selected models on our selected test set, and the number of parameters that they have.

4.2 Adversarial Robustness Toolbox (ART)

The Adversarial Robustness Toolbox (ART) [NST+18] is a Python library for Machine Learning Security, developed by IBM. ART provides tools that enable developers and researchers to defend and evaluate Machine Learning models and applications against the adversarial threats of Evasion, Poisoning, Extraction, and Inference. We use ART implementations of the selected adversarial attacks for generating adversarial images. We also use ART implementations of the selected defensive techniques to defend our classifiers from adversarial attacks. Lastly, we use the ART implementation of the CLEVER robustness metric to evaluate the robustness of our classifiers.

4.3 Generating Adversarial Examples

Due to their diversity, strength, and importance, we use the following adversarial attacks: Auto-PGD [CH20], Carlini and Wagner L^2 and L^∞ [CW17], FGSM [GSS15], HopSkipJump [CJW20], Shadow attack [GSG20] and Universal Perturbations [MFFF17]. These attacks are widely used in the research as benchmarks for other adversarial attacks and defenses, to see we refer the reader to the publications of the selected attacks and defenses. They are also among the strongest state-of-the-art adversarial attacks and their implementations are available in the ART.

We calculate the success of an adversarial attack by its success rate, i.e. the ratio of the successfully attacked images over the total number of classified images. An image is considered successfully attacked if, given the target setting, the classifier predicts the targeted class for the attacked image. In the case of the untargeted setting, an image is

considered successfully attacked if the classifier classifies the image as any other class than the original. We evaluate attacks only on the portion of the test set that was correctly predicted by the classifier. One of the main benefits of this is that we can evaluate the success of an attack more precisely.

We also calculate the (dis)-similarities of adversarial images and original images by measuring their L^2 and L^∞ pixel-wise distances.

4.4 Defending against Adversarial Attacks

For defending against adversarial attacks we use adversarial training, called "Fast is better than free" [WRK20], and three pre-processing defensive techniques: Spatial Smoothing [XEQ18], JPEG Compression [DGR16], and Total Variance Minimization [GRCvdM18]. These defense techniques are commonly used in the research as benchmarks for how strong the adversarial attacks are, as well as for evaluating other adversarial defenses. Additionally, they are easy to combine together and have implementations available in the ART.

Same as with the adversarial attacks, we evaluate defenses only on the portion of the test set that was correctly predicted by the classifier. We calculate the success rates of the attacks on the classifier that additionally uses these selected defenses. We are interested to determine if and how much do the defenses reduce the success rates of the attacks. Additionally, we combine defenses and evaluate if this improves the robustness of the classifier. We exhaust all different combinations.

4.5 Hardware and Software setup

We run our experiments on several machines equipped with the GeForce RTX 2080 Ti and the Tesla V100-PCIE-32GB GPUs.

For our experiments, we use the following software:

1. Python 3.7
2. ART 1.8.0
3. TensorFlow 2.3.1
4. Keras 2.4.3
5. PyTorch 1.8.1

CHAPTER 5

Experimental results

This chapter presents the results from the experiments described in Chapter 4.

We structure it into two parts:

1. results obtained from the experiments on the CIFAR-10 dataset, where we compare two models, PreActResnet18 and an adversarially re-trained version of this model called AdvTrainedPreActResNet18.

2. results obtained from the experiments on the ImageNet dataset, where we compare three models, MobileNetV2, InceptionV3, and ResNet-50.

The first part of this chapter focuses on the experimental results obtained by attacking the models with different adversarial attacks, without applying any pre-processing defensive technique beforehand. Here, we focus on the following measures:

1. Success rates of the attacks against the models

2. (Dis)-similarities of the adversarial images and the original images, calculated and presented by L^2 and L^∞ metrics.

3. CLEVER scores

4. Runtimes for generating adversarial images and evaluating them

In the second part, we discuss the same measures, but with pre-processing defense techniques used, both separately and combined. The discussion is first grouped by target setting and by dataset.

5.1 Adversarial attacks without pre-processing defense techniques

CIFAR-10 dataset

For this dataset, we present and compare results for PreActResNet18 and AdvTrained-PreActResNet18 models, which use adversarial training as a defense mechanism.

While PreActResNet performs slightly better on the original (clean) data, it is affected more by adversarial images. As shown in Figures 5.1 to 5.3, the success rates of our selected adversarial attacks are generally higher on the PreActResNet model under all three settings. This, however, does not hold for Auto-PGD and Carlini and Wagner L^∞ attacks in the targeted settings. In both targeted settings, these two attacks have approximately 80 % success rate against AdvTrainedPreActResNet18, compared to on average approximately 45 % success rate against PreActResNet18. It is hard to speculate as to why this is the case since the original paper [WRK20] does not present results against these two attacks. Compared to the original paper, we can only conclude that we obtained very similar results for the FGSM attack in the *untargeted* setting. In their work, they report that the AdvTrainedPreActResNet18 model has a lower accuracy on adversarial images than on original images by about 30 %.

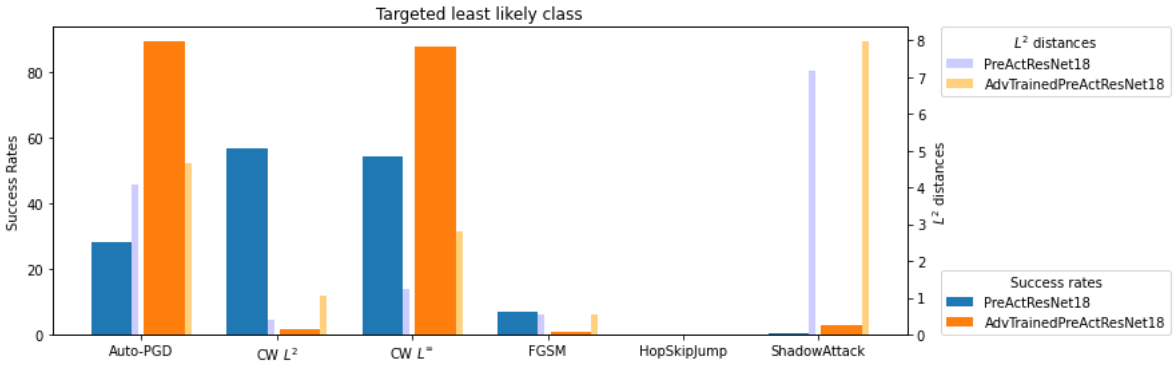


Figure 5.1: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models targeting the least likely class

In our experiments, adversarial training has been very effective in both targeted settings against every selected attack, except the mentioned Auto-PGD and Carlini and Wagner L^∞. But while these two attacks have a high success rate against adversarial training, they also have had to modify original images more than the other attacks to make them adversarial. The only exception to this is the ShadowAttack, which has the highest L^2 distances, but a very low success rate in both targeted settings. For the case of Auto-PGD, the mean L^2 distance of the adversarial images and the original images is 4.66 for the *targeted least likely class* and 4.74 for the *targeted next class* settings. For PreActResNet, it is 4.09 for the *targeted least likely class* and 4.08 for the *targeted next*

class settings. The larger the modification, the less stealthy the attack is. If the attack fools the classifier, but the introduced adversarial modification is very substantial and easy to notice by a human, then we can argue that in this instance the attack is rather unsuccessful.

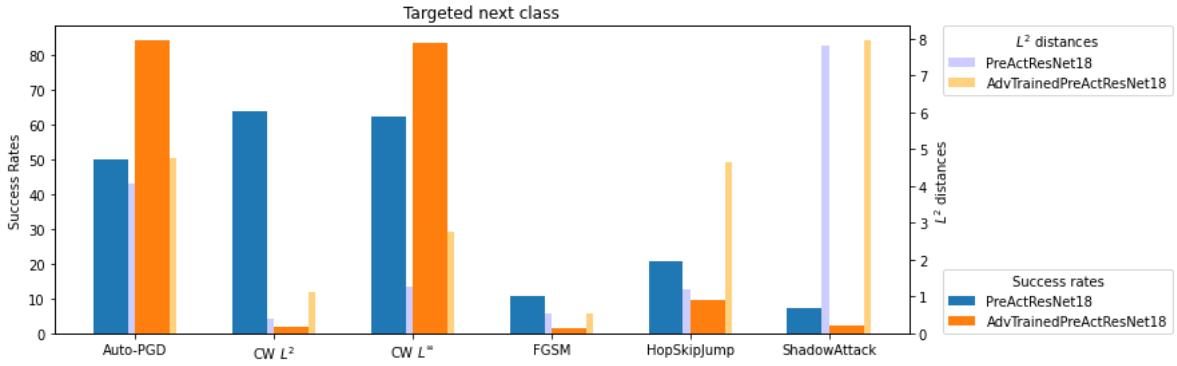


Figure 5.2: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models targeting the next class

It is important to notice that CW L^2 attack has very low success rates in both targeted settings against AdvTrainedPreActResNet18, whereas it has about 60 % success rate against the original model. However, while it was successful against most of the attacks (except Auto-PGD, CW L^∞) in the targeted settings, HopSkipJump has managed to penetrate the adversarial training defense in the *untargeted* settings, scoring % and 90.12 success rates, but with a high L^2 distance. We also have to highlight that other attacks as well have scored higher success rates than in the targeted settings. For example, ShadowAttack and Universal Perturbations, with 23.6 % and 26.5 % success rate. Such behavior was expected since, for an *untargeted* setting, the attacker aims for any misclassification, contrary to the targeted setting, where it aims at a specific one.

Looking at L^2 (dis)-similarities between adversarial images and the original images, there are several points to notices. Against the original model, CW L^2 and L^∞ introduced only small changes to original images, but achieve almost perfect success rates in all three target settings. Auto-PGD modified the images much more than CW L^∞ attack (approximately 4.5 compared to approximately 1.5), still with a similar success rate in all three target settings. On the other hand, the Shadow attack, despite causing the highest modifications (on average around 8) of the original images (compared to other attacks), achieved low success rates in both targeted settings, even on the original model.

In general, across all attacks and all three settings, attacks had to modify images much more for them to be adversarial against adversarially re-trained PreActResNet18 than against the original model. Some examples of that:

1. Carlini and Wagner L^∞ - in the *targeted least likely class* 1.2 L^2 for PreActResNet18,

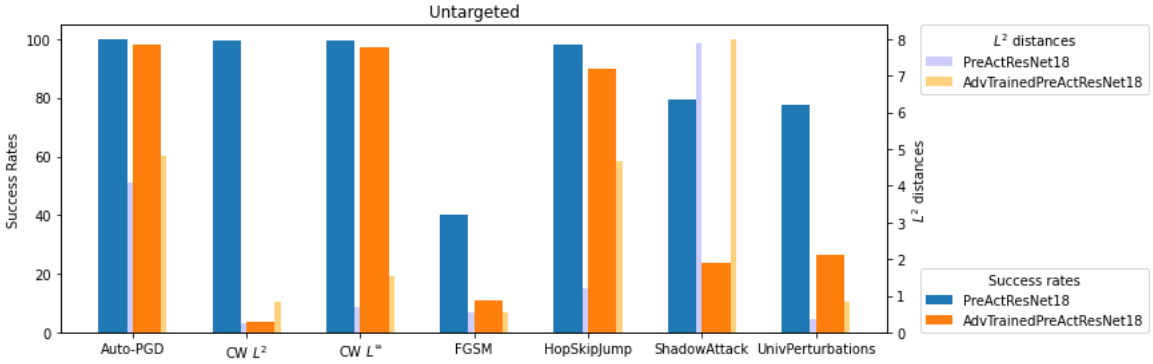


Figure 5.3: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models with no specific target

2.8 for AdvTrainedPreActResNet18

2. HopSkipJump - in the *untargeted* setting 1.2 L^2 for PreActResNet18, 4.6 for AdvTrainedPreActResNet18

As for runtimes, there is not much to report for the CIFAR-10 dataset, since all attacks generated adversarial images at the rate of approximately 1 second per image, except in the case of the CW L^2 attack, which took approximately 15 seconds per image. But even at the rate of 15 seconds per image, we could not notice any significant runtime issues. This was, however, an indicator that we could potentially have computational struggles on the ImageNet dataset with CW attacks.

ImageNet dataset

For this dataset, we evaluate InceptionV3, MobileNetV2 and ResNet-50 models. For their different architectures and baseline accuracy, it is very interesting to see how do different adversarial attacks and pre-processing defenses affect them.

Figures 5.4 to 5.6 show the success rates of the different adversarial attacks and also the L^2 distances of the adversarial images and the original images under all three target settings. The first observation is that under all three settings, the selected adversarial attacks scored (slightly) lower success rates against InceptionV3 than against MobileNetV2 and ResNet-50. We speculate that this is due to the InceptionV3 being a more complex neural network architecture and having more parameters than the other two. While this behavior is less apparent in the *untargeted* and the *targeted next class* setting, it is quite obvious in the *targeted least likely class* with Carlini and Wagner attacks. With relatively similar L^2 distances between adversarial and original images, both Carlini and Wagner L^2 and L^∞ attacks have scored 80+ % success rates against MobileNetV2 and

ResNet-50, whereas InceptionV3 remained fairly robust with very low success rates of 7.27 % and 22.2 % respectively.

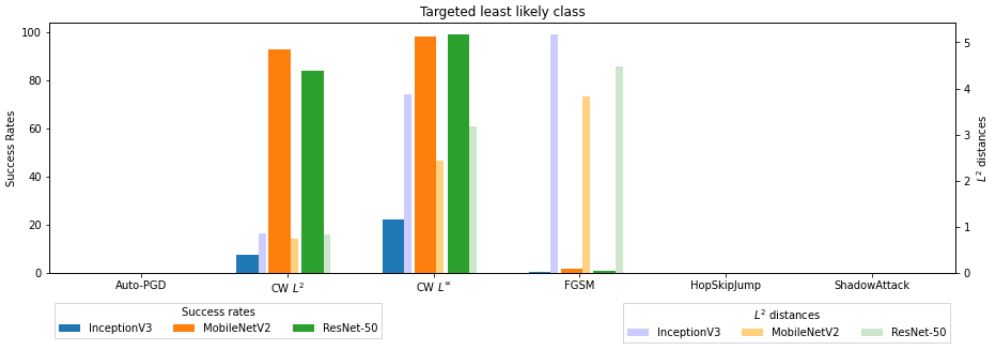


Figure 5.4: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2, and ResNet-50 models targeting the least likely class

Interestingly, the other attacks in the *targeted least likely class* setting have been unsuccessful and did not produce any success rates. This observation highlights several points, one of them being the testament to how strong the Carlini and Wagner attacks are. Another point is that it is hard to successfully attack a specific class in the probability distribution such that the modification to the images is "almost" unnoticeable. This trend holds also for the *targeted next class* setting. Albeit, the success rates in this targeted setting are slightly better than in the first targeted setting. We speculate that the reason for this could be the fact that ImageNet has 1000 classes and some of the classes are closer than the others, like for example pan and wok pan. Hence, it would make sense that attacks with such targets are more successful than when attacks with targets that are more distant, like for example least likely class.

Figure 5.5: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2, and ResNet-50 models targeting the next class

However, the results are quite different, when it comes to the *untargeted* setting. The general conclusion is that all of the attacks have scored high success rates against all of the models In particular, Auto PGD, Carlini and Wagner attacks, HopSkipJump, and Universal Perturbations have scored 80+ % success rates against all three models. A very interesting result to highlight is the Shadow attack on InceptionV3, which has had to generate adversarial images that are more distant to original images than the ones generated against the other two models, but still scored a much lower success rate (40, 3 % compared to 88.6 and 80.8 %).

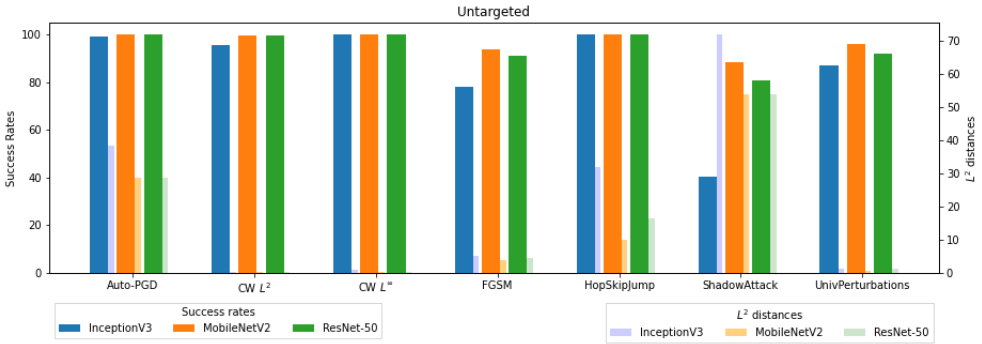


Figure 5.6: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2, and ResNet-50 models with no specific target

The L^2 distances between adversarial and original images are low for the Carlini and Wagner attacks, FGSM and Universal Perturbations, which means that these attacks have generated adversarial images that are close to the original images. These distances are higher for the other three attacks, Auto-PGD, HopSkipJump, and especially for Shadow attacks. We conclude that in this experiment the Shadow attack is the weakest because it scored the lowest success rates while generating the adversarial images with the highest L^2 distance.

Lastly, we comment on the runtimes when generating adversarial images. While it is hard to compare our results directly, because we generated adversarial images with different attacks on different GPUs, there are some interesting trends to be highlighted. Namely, all attacks except for Carlini and Wagner L^2 and L^∞ and Universal Perturbations have been fast with a runtime of approximately 1 seconds per sample. Among them, we highlight only the HopSkipJump that had a slightly higher runtime of 3 seconds per sample. For Carlini and Wagner L^2 and L^∞ and Universal Perturbations, the runtimes are 234, 19, and 134 seconds respectively.

5.2 Adversarial attacks with pre-processing defense techniques

5.2.1 targeted least likely class setting

CIFAR-10 dataset

The first observation (and perhaps also the most important) is that all defense scenarios have reduced the success rates of all attacks except Auto PGD on the AdvTrainedPreActResNet18 model. The reduced success rates are very similar to the success rates against the adversarially re-trained model without pre-processing defenses, except in the case of Auto-PGD and CW L^∞, which successfully broke the adversarial training defense, as discussed in the previous section. For example, In other words, the effect of pre-processing defense techniques was about the same (with an exception of Auto-PGD and CW L^∞ attacks) as adversarial training alone. But, AdvTrainedPreActResNet18 paired with pre-processing defenses has overall reduced success rates to single digits against most of the attacks.

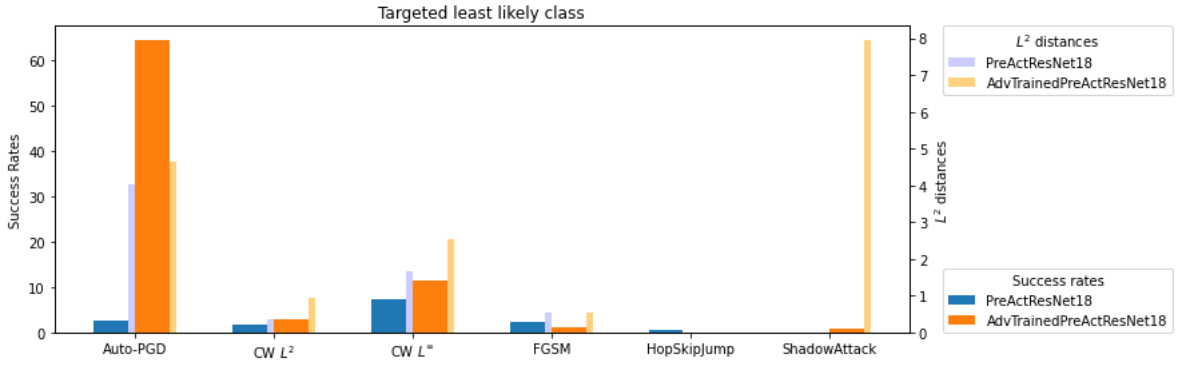


Figure 5.7: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models targeting the least likely class, with Spatial Smoothing as the pre-processing defense

It is important to highlight that while the pre-processing defenses while adversarial training could not defend against Auto-PGD and CW L^∞ attacks in the baseline, the pre-processing defenses have had much better success in that regard. For example, Auto-PGD and CW L^∞ have scored only about 2 % success rate against PreActResNet18 with Total Variance Minimization defense. But perhaps even more interesting is that CW L^∞ has scored much lower success rates against AdvTrainedPreActResNet18 with the pre-processing defenses (for example, from 87.7 % in the baseline to 3.7 % with all three pre-processing defenses combined), while only the Total Variance Minimization defense managed to reduce the success rate of the Auto-PGD attack, from 89.4 % to 15.5 %.

Comparing the baseline results without pre-processing defenses from Figure 5.1 for the target least likely class, we notice that the pre-processing defenses and their combinations

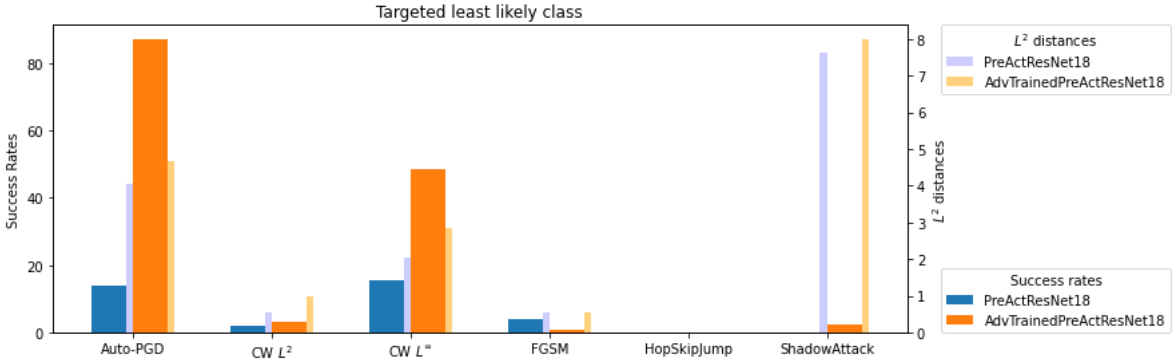


Figure 5.8: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models targeting the least likely class, with JPEG Compression as the pre-processing defense

have overall reduced the success rates for several attacks on the original model (for example CW L^2 from about 60 % to less than 5 % on average and CW L^∞ from about 60 % to approximately 10 % on average).

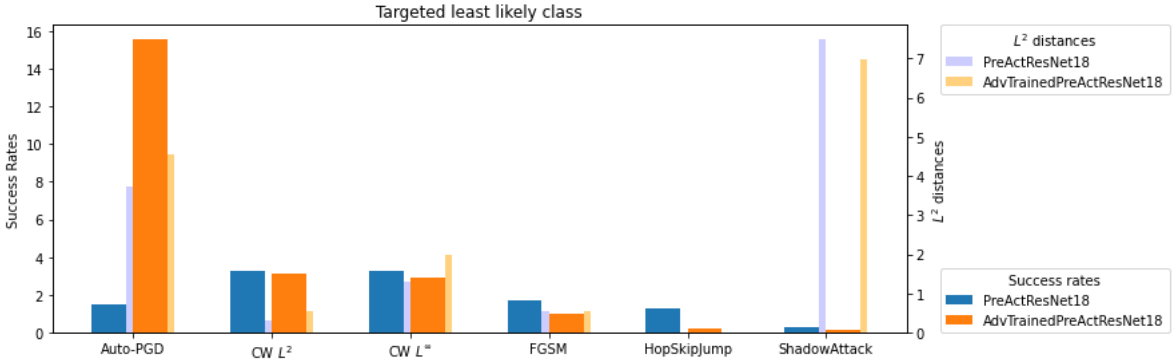


Figure 5.9: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models targeting the least likely class, with Total Variance Minimization as the pre-processing defense

For the complete overview of the success rates of our selected adversarial attacks against the pre-processing defenses in this targeted setting, please refer to Figures 5.7 to 5.13.

We are now turning our attention to L^2 distances of the adversarial images pre-processed with different combinations of the pre-processing defenses to the original images (see Figures 5.10 to 5.13). While one could have expected that these images would be closer (more similar) to the original images in the L^2 metric than the baseline non-pre-processed adversarial images, that was not the case. What is even more interesting, when two or more pre-processing defenses are combined, the L^2 metric norms for the images are higher than when pre-processing with only one defense. But, this does not translate

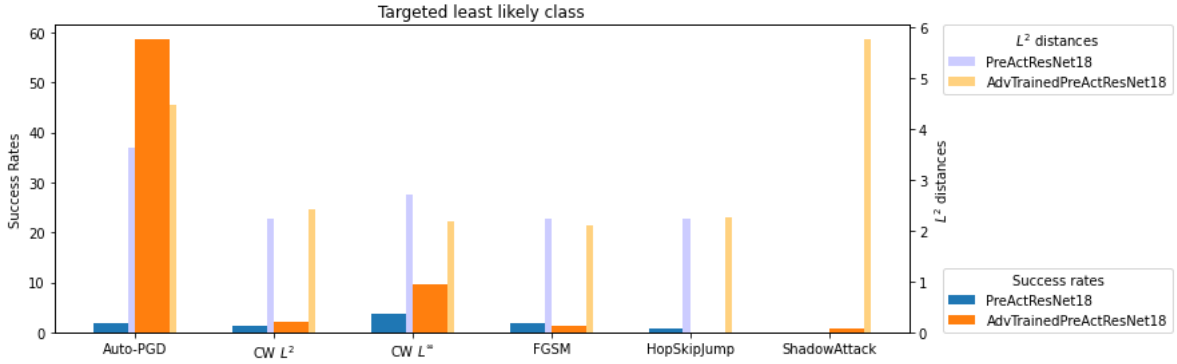


Figure 5.10: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models targeting the least likely class, with Spatial Smoothing and JPEG Compression as the pre-processing defenses

to higher success rates of the attacks. The combinations in this target setting do not improve success rates significantly either.

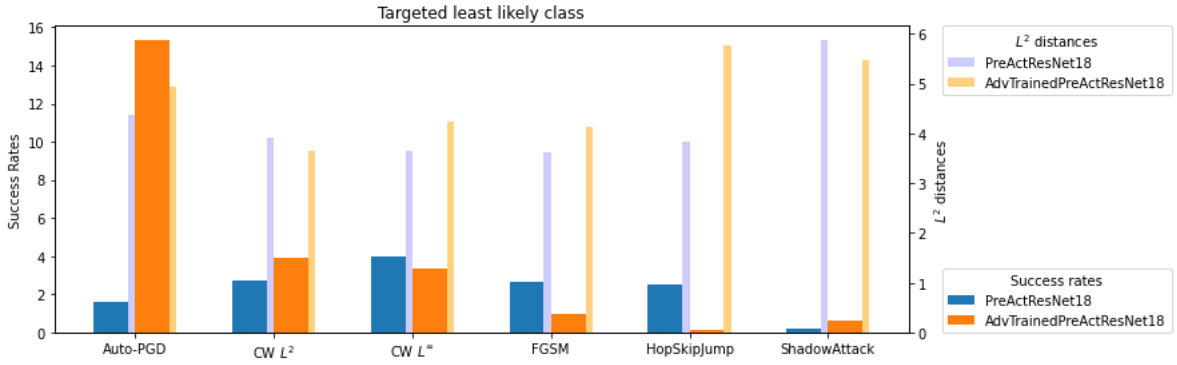


Figure 5.11: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models targeting the least likely class, with Spatial Smoothing and Total Variance Minimization as the pre-processing defenses

Overall, it is hard to conclude which pre-processing defenses were the best in this targeted setting. However, using Total Variance Minimization only scored the lowest success rates, while also keeping the adversarial images and original images very similar for most of the attacks.

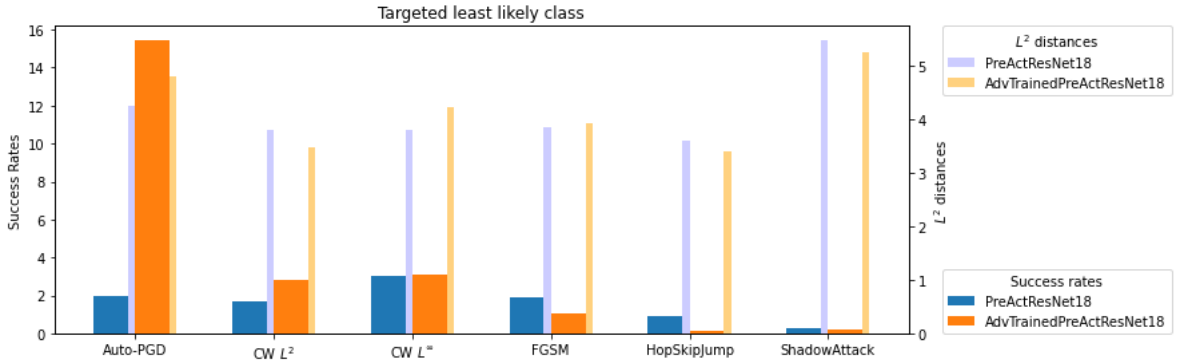


Figure 5.12: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models targeting the least likely class, with JPEG Compression and Total Variance Minimization as the pre-processing defenses

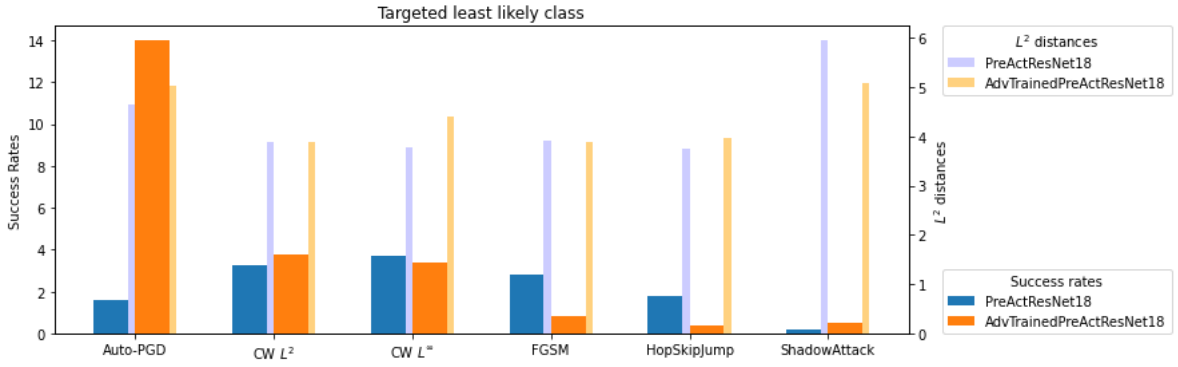


Figure 5.13: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models targeting the least likely class, with Spatial Smoothing, JPEG Compression, and Total Variance Minimization as the pre-processing defenses

ImageNet dataset

While the discussion and the experimental results for the *targeted least likely class* setting have been very interesting on the CIFAR-10 dataset, here we have fewer highlights. This is mainly because the adversarial attacks were unsuccessful in the baseline as well. Auto PGD, FGSM, HopSkipJump, and Shadow attacks remained unsuccessful, with almost no success rate.

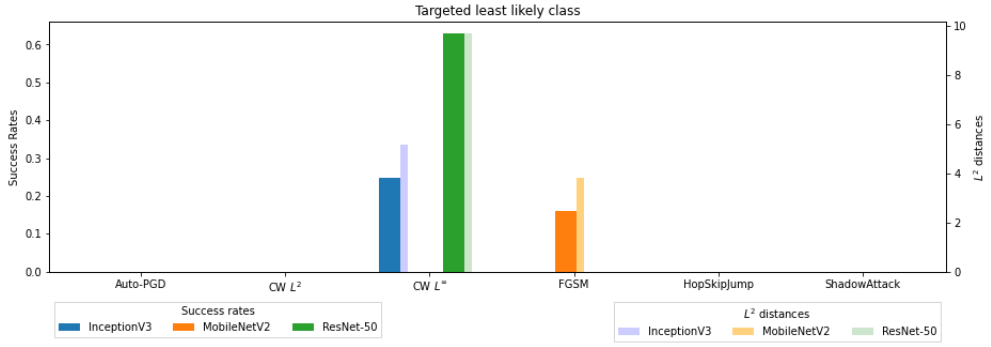


Figure 5.14: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2, and ResNet-50 models targeting the least likely class, with Spatial Smoothing as the pre-processing defense

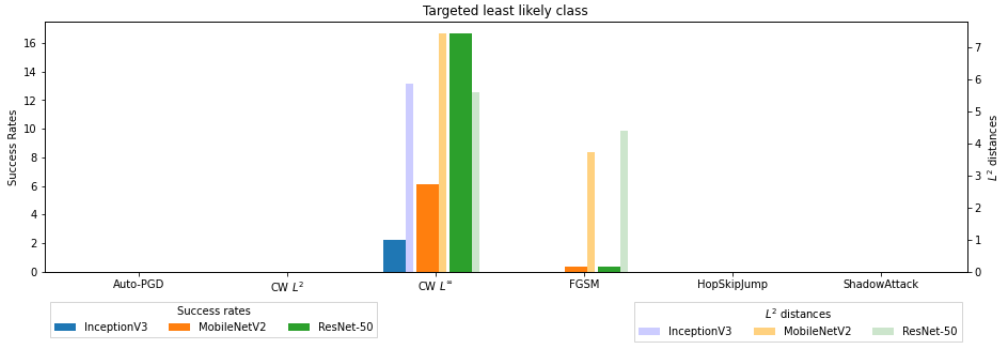


Figure 5.15: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2, and ResNet-50 models targeting the least likely class, with JPEG Compression as the pre-processing defense

Hence the only interesting case is the Carlini and Wagner attacks, where we report that pre-processing defenses have managed to defend the models very well. If you recall from Figure 5.4, these two attacks have had very high success rates against MobileNetV2 and ResNet-50, whereas now with pre-processing defenses they have almost no success rates. Only in the case of the JPEG Compression (see Figure 5.15) has Carlini and Wagner L^∞ had some success, scoring a success rate of 16.6 % on ResNet-50, 6.2 % on MobileNetV2 and 2.22 % on InceptionV3. This result is still a good improvement from the baseline

case. Carlini and Wagner L^2 did not record any success.

Since the discussion above is very much applicable to all defenses and their combinations in this targeted setting, we present here only the results for Spatial Smoothing (Figure 5.14) and JPEG Compression (Figure 5.15) pre-processing defenses individually, and present the others in Appendix Chapter A in Figures A.8 to A.12.

Overall, we cannot conclude which of the defenses has had the best success in this setting, because all of them were equally successful. We can only conclude that the JPEG Compression has the worst results because it was unable to reduce the success rate of Carlini and Wagner L^∞ attack as others did.

5.2.2 Targeted next class setting

CIFAR-10 dataset

The trends and observations for the experiments when targeting the next class are very similar as in the previous case when targeting the least likely class, only with slight differences. For example, with the Total Variance Minimization defense (see Figure A.3 on the PreActResNet18 model, the ShadowAttack has scored a success rate of 37.7 % (compared to 2.2 % on the AdvPreActResNet18). This result becomes even more interesting when we consider that in the non-defended baseline, ShadowAttack scores a 7.4 % success rate against PreActResNet18. This occurrence is however unique in the targeted settings and we cannot speculate what is the reason for it.

Because the discussion and the conclusions are very similar as in the case of the *targeted least likely class* and for better readability of this chapter, we show the results for this targeted setting in Appendix A and refer to Figures A.1 to A.7 for detailed results that are structured and presented in the same way as in the previous target setting.

ImageNet dataset

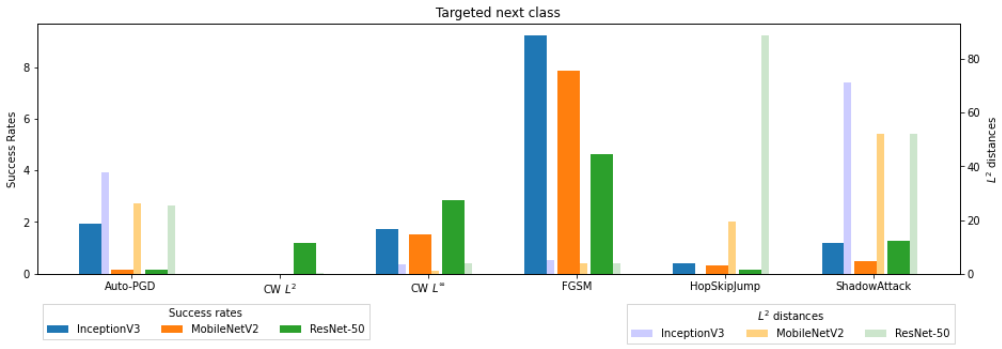


Figure 5.16: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2, and ResNet-50 models targeting the next class, with Spatial Smoothing as the pre-processing defense

The trends and observations for the experiments when targeting the next class are again very similar as in the case when targeting the least likely class, only with slight differences. The success of the pre-processing defenses has been only slightly worse than in the targeted least likely class setting. We speculate that this directly correlates with the properties of the ImageNet dataset. Namely, ImageNet has 1000 classes and some of the classes are closer than the others, like for example pan and wok pan. Hence, it would make sense that attacks with such targets are more successful than when attacks with targets that are more distant, like for example least likely class.

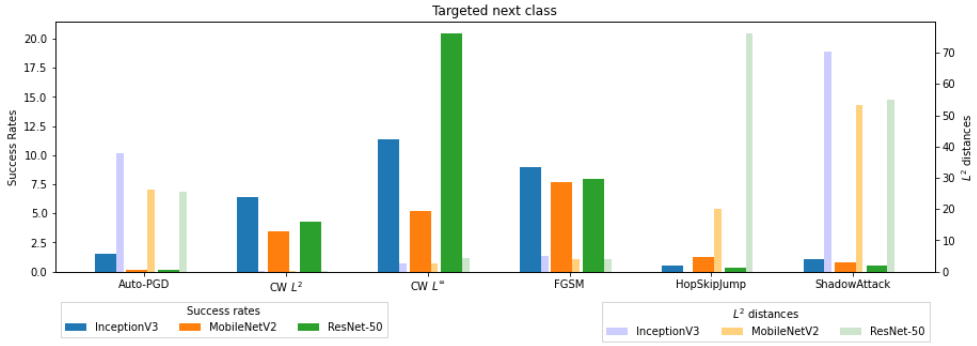


Figure 5.17: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2, and ResNet-50 models targeting the next class, with JPEG Compression as the pre-processing defense

As we saw in the baseline, three attacks had the most success in this target setting, namely Carlini and Wagner L^2 and L^∞, and FGSM. Similar to in the *targeted least likely class* setting, all three defenses and their combinations have managed to reduce these success rates significantly (to single digits). The JPEG Compression defense had again less success against Carlini and Wagner L^∞ attack.

Figure 5.18: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2, and ResNet-50 models targeting the next class, with Total Variance Minimization as the pre-processing defense

As can be seen in Figures 5.16 to 5.18, the recorded success rates for this target setting indicate that, in contrast to the not defended baseline case, all attacks are now equally successful (or rather unsuccessful). We highlight the FGSM attack for being stronger than Auto-PGD (its successor) in almost all scenarios in this targeted setting.

Between the three models, InceptionV3 turned out to be less robust with pre-processing defenses than the other two models, which is also opposite to the baseline case. Albeit, this does not hold in every scenario, especially against Carlini and Wagner attacks and with the Total Variance Minimization pre-processing defense (and also combinations that include this defense).

For better readability and because they do not show any improvement we present the results for the different combinations of the pre-processing defenses in Appendix Chapter A in Figures A.13 to A.16. It is again hard to conclude which defense scored the best results in this targeted setting, with success rates being > 0 overall. By a slight margin, the Total Variance Minimization pre-processing defense and the combinations which include this defense have overall lower success rates, but probably the most important observation is that all of the pre-processing defenses have been successful in the *targeted next class* setting.

5.2.3 Untargeted setting

CIFAR-10 dataset

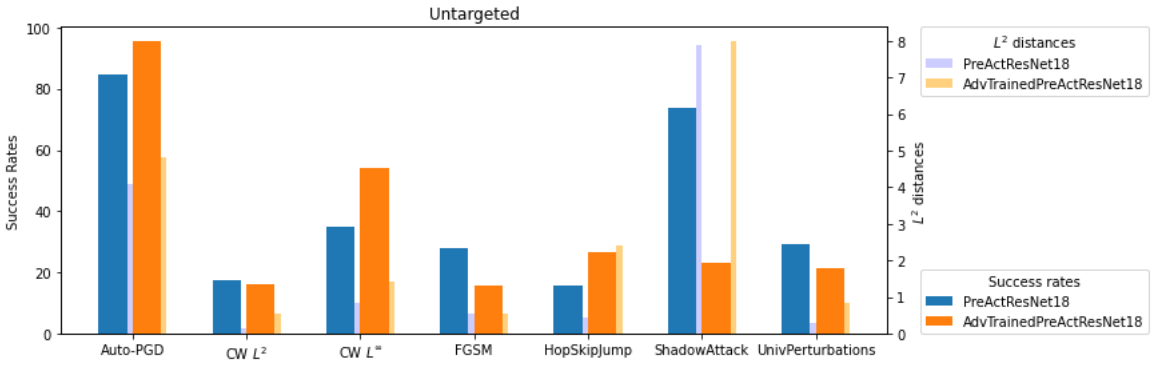


Figure 5.19: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models with no specific target and Spatial Smoothing as the pre-processing defense

Since the attacker in the *untargeted* setting aims for any misclassification, it makes sense to expect higher success rates overall than in the targeted setting, similarly as we saw higher success rates in the *untargeted* setting in the baseline discussion. This has been confirmed in our experiments, which we discuss here.

Different from the targeted setting, where adversarial training combined with Total Variance Minimization provided very good protection, here this combination fails in at

least 20 % of adversarial images, see Figure 5.21. For example, Auto PGD which had a success rate (against the combination of these defenses??) of 4, 6 % in the *targeted next class* setting now has 74.2 % in the *untargeted* setting.

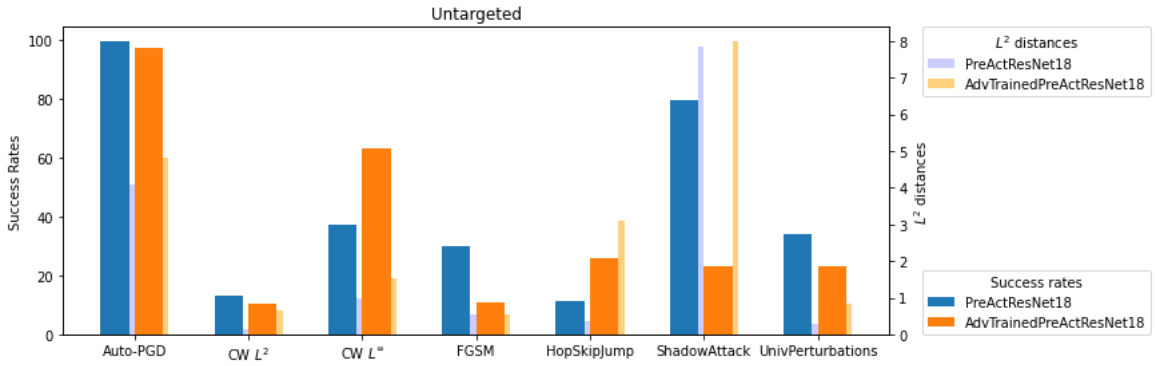


Figure 5.20: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models with no specific target and JPEG Compression as the pre-processing defense

However, if we consider the baseline results (from Figure 5.3) we notice overall a great reduction in success rate on the PreActResNet18 model. These results are similar to the results obtained in the baseline with the adversarially trained model, albeit slightly worse. But interestingly, the adversarially trained model did not introduce a further significant reduction in success rates. AdvTrainedPreActResNet18 paired with pre-processing defenses only provided better defenses by on average approximately $10 - 20$ %. These results are not at all optimal. For example, we have, 30.8 % success rate of FGSM with AdvTrainedPreActResNet18 and Total Variance Minimization (Figure 5.21) or 25 % success rate of Shadow attack with AdvTrainedPreActResNet18 and Spatial Smoothing and JPEG Compression combined (Figure 5.22).

ShadowAttack remained very strong against all defense combinations and both models. The highest measured success rate with the original model is 79.3 %, whereas with the adversarially trained model it is 23.1 %, both with JPEG Compression (Figure 5.20). Only noteworthy improvement of the defense was recorded with the combination of Spatial Smoothing and JPEG Compression, which managed to reduce the success rate of the ShadowAttack on the AdvTrainedPreActResNet18 model to 3.6 %.

All three defenses individually offer a very similar level of protection and the trends between the attacks translate from defense to defense. While Total Variance Minimization is again the only defense that can reduce the success rates of the Auto-PGD (albeit to 60.9 % on the PreActResNet18 and 79 % on the AdvTrainedPreActResNet18, which is still a high success rate), the JPEG Compression has kept CW L^2 attack at a low success rate against both PreActResNet18 and AdvTrainedPreActResNet18, 13.3 % and 10.3 % respectively. it provides the best protection against HopSkipJump and also combined

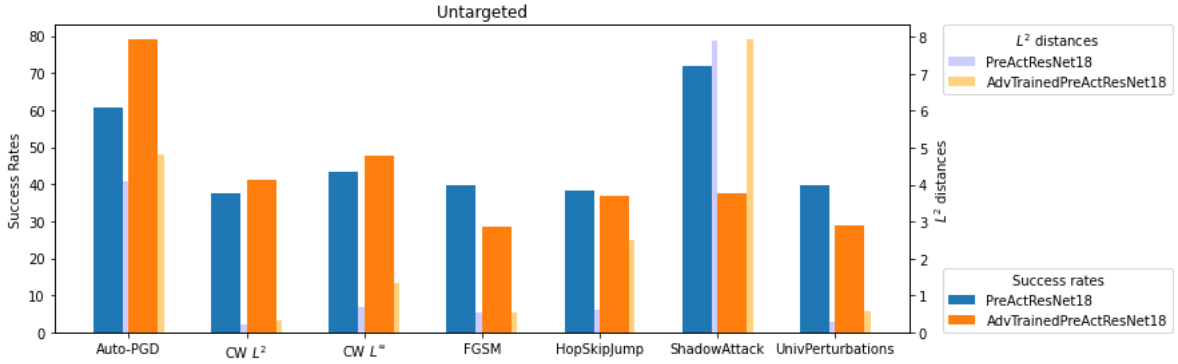


Figure 5.21: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models with no specific target and Total Variance Minimization as the pre-processing defense

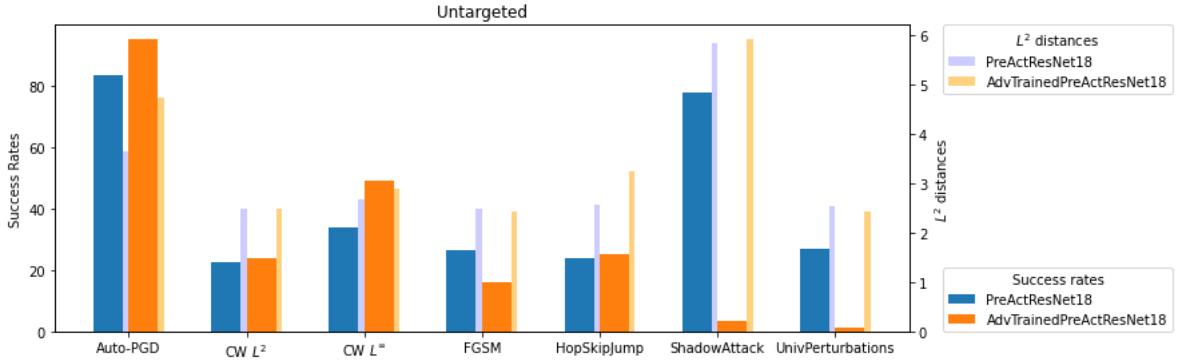


Figure 5.22: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models with no specific target and Spatial Smoothing and JPEG Compression as the pre-processing defenses

with Spatial Smoothing it provides the best protection against Universal Perturbations. For this and more details, please refer to Figures 5.19 to 5.21.

When we look at the combination of these defenses, no significant improvement can be noticed, apart from that L^2 distances are higher than in the individual cases. The most notable is the combination of Spatial Smoothing and JPEG Compression, where they combined best together, keeping CW attacks, HopSkipJump and Universal Perturbations to their lower success rates of about 20 % (compared to other results in the *untargeted* setting). The drawback of this combination is that Auto PGD against both models and ShadowAttack against PreActResNet18 remained very strong against it. To be more precise, Auto PGD scored almost perfect 95 % success rate against AdvTrained-PreActResNet18, and 83.4 % against dPreActResNet18, whereas ShadowAttack scored

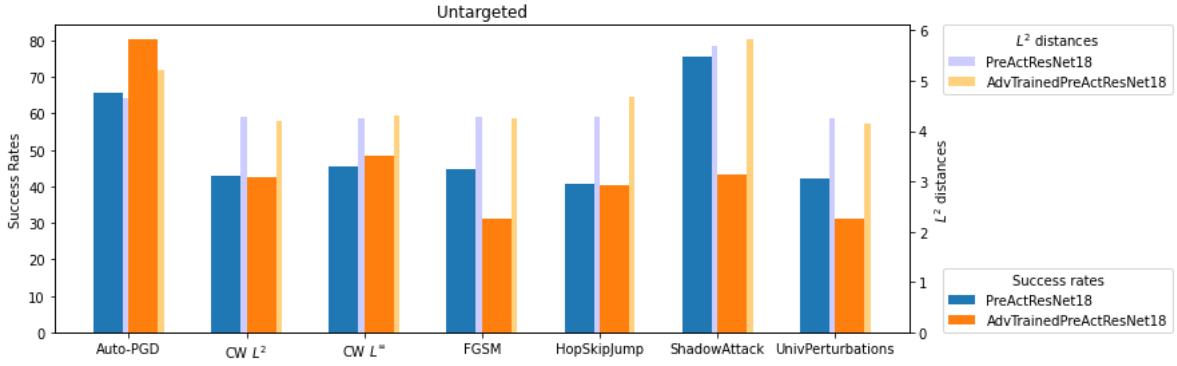


Figure 5.23: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models with no specific target and Spatial Smoothing and Total Variance Minimization as the pre-processing defenses

77.8 % success rate against PreActResNet18. For this and more details, please refer to Figures 5.22 to 5.25.

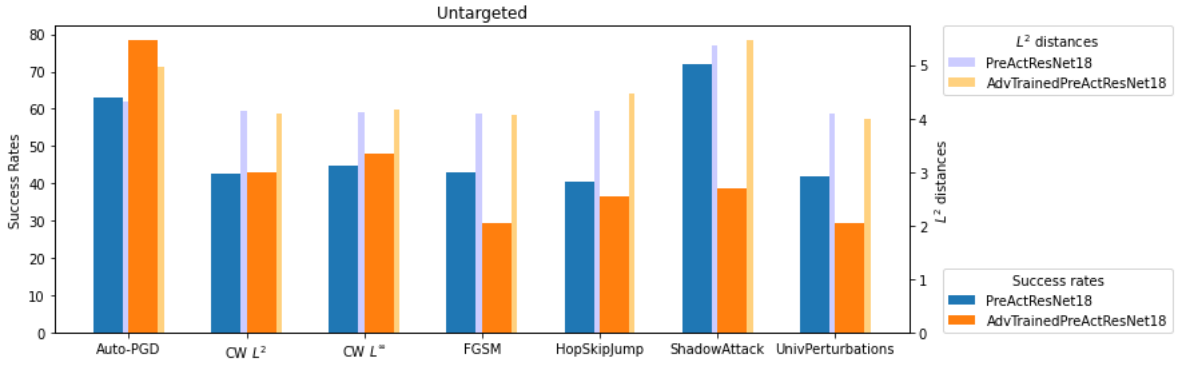


Figure 5.24: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models with no specific target and JPEG Compression and Total Variance Minimization as the pre-processing defenses

The trends regarding the L^2 distance of the adversarial images and original images are the same as in the previous two targeted settings.

We now give an example of one of the images from the CIFAR-10 dataset (see Figure 5.26). In this example, we have the original image of the *truck* correctly predicted with PreActResNet18. Next, we have the adversarial version of this image, generated with the Carlini and Wagner L^2 in the *untargeted* setting, which PreActResNet18 recognized as an *airplane*. But then we show that the Spatial Smoothing defense defends (pre-processes)

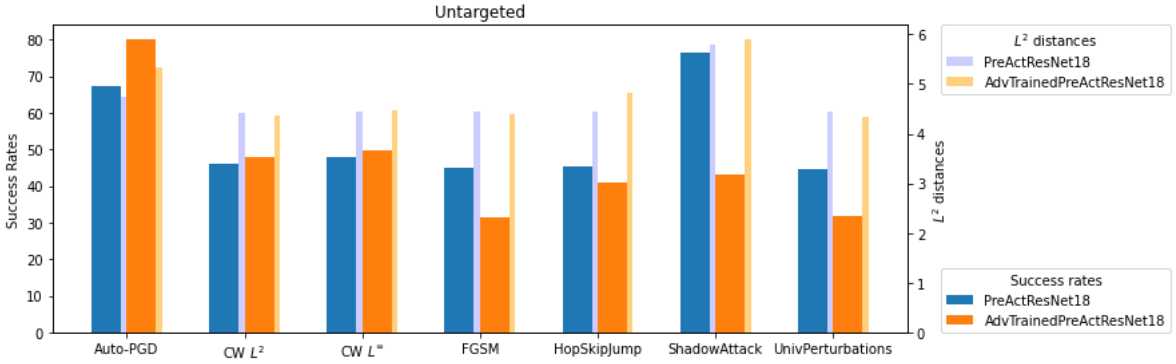


Figure 5.25: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models with no specific target and Spatial Smoothing, JPEG Compression and Total Variance Minimization as the pre-processing defenses

this image successfully, meaning that the model was able to predict the image correctly as a *truck*.

The last thing that we want to highlight in this example is the L^2 distances. Namely, the L^2 difference between the adversarial image and the original is 14.68, whereas the L^2 difference between the defended image and the original is 14.49, which shows us that (in the L^2 metric) the defended image is not much closer to the original image than the adversarial image, but the model has predicted the defended image correctly.

(a) Original Image, predicted as *truck* (b) Adversarial image, predicted as *airplane* (c) Defended Image, predicted as *truck*

Figure 5.26: A comparison of the original image (a), the same image attacked with Carlini and Wagner L^2 attack (b) in the *untargeted setting* and the attacked image defended with the Spatial Smoothing defense (b). The figure shows that the attack was successful, but also that the defense successfully defended against the attack. The L^2 difference between the adversarial image and the original is 14.68, whereas the L^2 difference between the defended image and the original is 14.49.

ImageNet dataset

As expected, the *untargeted* setting on the ImageNet dataset was the most difficult to defend against, because the attacker is satisfied with any misclassification. And since in the baseline (see Figure 5.6) all of the attacks were very successful, this setting gives us a good playground to evaluate the effect of the pre-processing defenses with different adversarial attacks.

The spatial Smoothing had the most success against Carlini and Wagner attacks, HopSkipJump and Universal Perturbations, see Figure 5.27 The adversarial images generated with these attacks have also lower L^2 distances compared to others, which indicates a correlation between the success of a pre-processing defense and a small distance of the adversarial and original images. In other words, the less modification the attack introduces to the original images, the easier it is for a pre-processing defense to correct the classification. On the other hand, Spatial Smoothing had less success against FGSM, and against Auto PGD and Shadow attack it made almost no difference. The adversarial images generated with these two latter attacks have the highest L^2 distance, which only supports our observation. Interestingly, the FGSM attack remained strong, scoring approximately a $65 - 70$ % success rate, but did not beat the Auto-PGD as seen in the previous target settings.

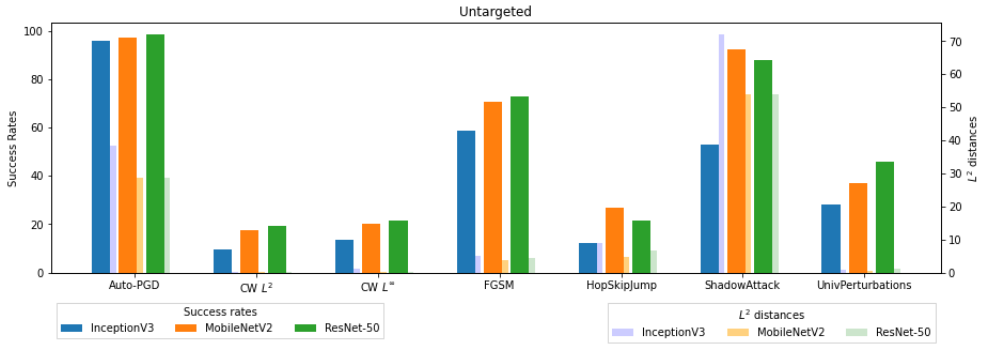


Figure 5.27: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2, and ResNet-50 models with no specific target and Spatial Smoothing as the pre-processing defense

The JPEG Compression was not as good as Spatial Smoothing but did manage to reduce the success rates on average by approximately 50 %, see Figure 5.28. While this seems like quite a good result, keep in mind that this means that every other adversarial image was misclassified, which is not an optimal defense. The trends for which attacks were stronger and which were weaker against this defense are similar as in the case of Spatial Smoothing.

Unlike the Spatial Smoothing, The Total Variance Minimization was able to reduce the success rate of the Auto PGD attack (see Figure 5.29), but was, on the other hand, less successful than the Spatial Smoothing against the HopSkipJump and the Universal

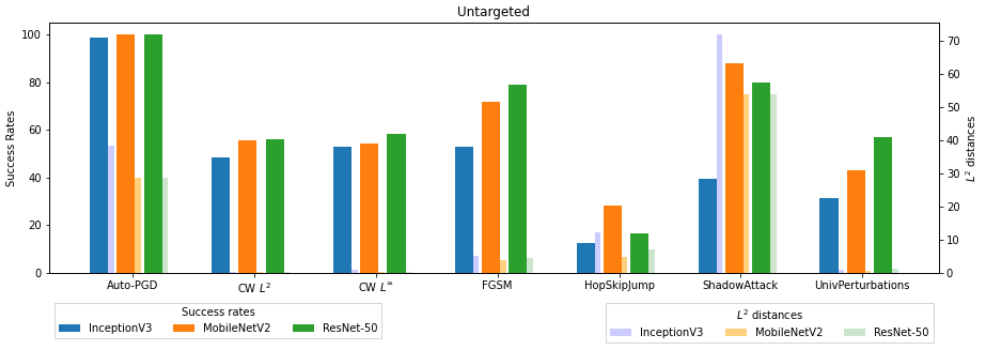


Figure 5.28: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2, and ResNet-50 models with no specific target and JPEG Compression as the pre-processing defense

Perturbations. It also provided the same level of protection for Carlini and Wagner attacks as the Spatial Smoothing.

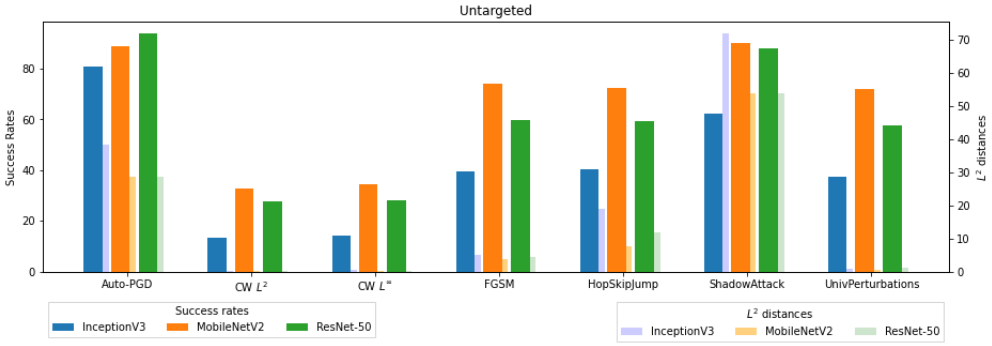


Figure 5.29: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2, and ResNet-50 models with no specific target and Total Variance Minimization as the pre-processing defense

But interestingly, the combination of Spatial Smoothing and Total Variance Minimization, as can be seen in Figure 5.31, has scored worse results than the defenses individually. In addition, it has increased the L^2 distances significantly. This is most obvious in the case of the Carlini and Wagner attacks, which in other scenarios had very low L^2 distances. This example indicated that combining defenses requires careful tailoring and that the outcome is highly situational.

Other combinations of defenses have also notably increased the L^2 distances between the adversarial images and the original images, but have managed to keep the improvements from the individual cases or even slightly improve them. A good example is the combination of Spatial Smoothing and JPEG Compression (see Figure 5.30).

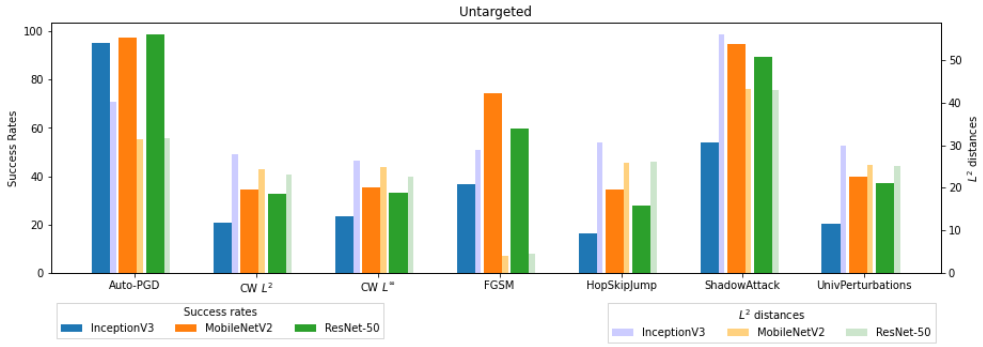


Figure 5.30: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2, and ResNet-50 models with no specific target and Spatial Smoothing and JPEG Compression as the pre-processing defense

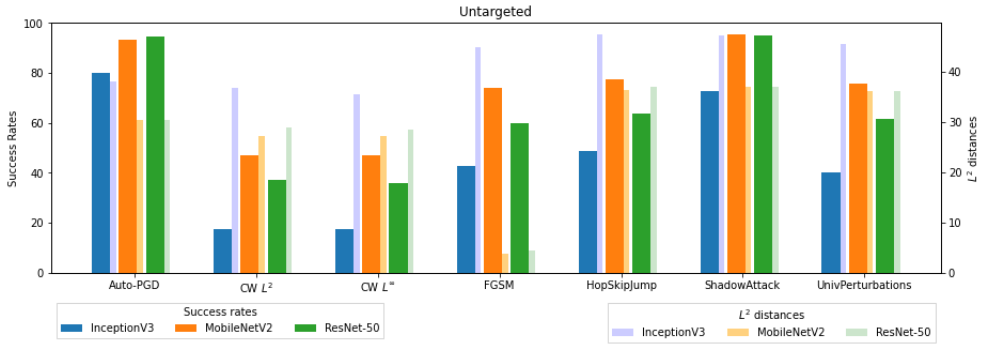


Figure 5.31: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2, and ResNet-50 models with no specific target and Spatial Smoothing and Total Variance Minimization as the pre-processing defense

Turning our attention to adversarial attacks, we have to stress that the pre-processing defenses have made little to no impact on the adversarial images generated by the Shadow attack. This is interesting because this attack was the weakest one in the baseline experiment. Now, in the *untargeted* setting, it is the strongest attack alongside Auto PGD.

To conclude, we have seen that the pre-processing defenses indeed reduce the success rates of most of the attacks significantly. However, in the *untargeted* setting, none of the classifiers managed to defended successfully against the attacks, where with successfully we mean that the success rate was on average a single-digit number. The lowest success rates were recorded with the Spatial Smoothing defense against Carlini and Wagner attacks. We also see from our experiments that the attacks that generate more distant adversarial images are stronger against these defenses. We conclude that Spatial Smoothing and the combination of Spatial Smoothing and JPEG Compression were overall the best defenses

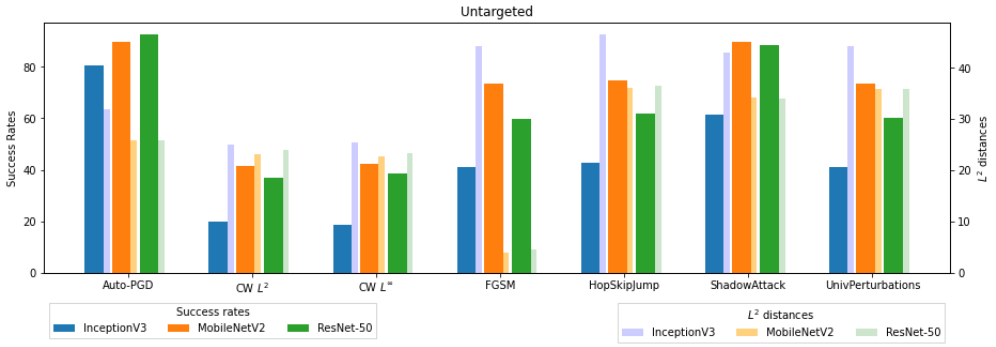


Figure 5.32: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2, and ResNet-50 models with no specific target and JPEG Compression and Total Variance Minimization as the pre-processing defense

in this setting.

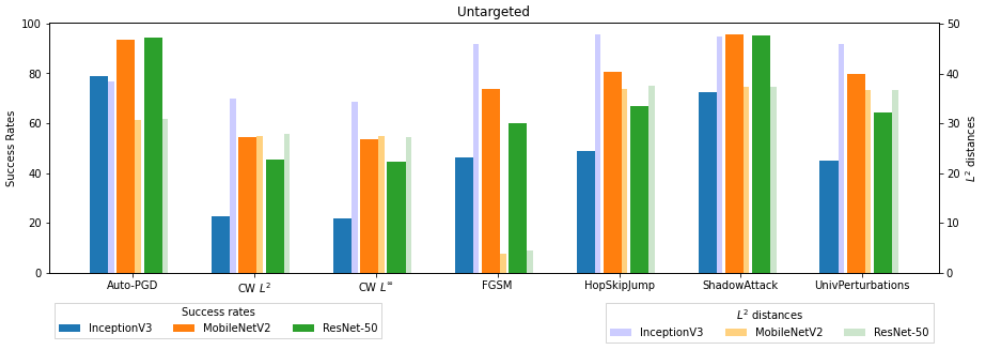


Figure 5.33: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2, and ResNet-50 models with no specific target and Spatial Smoothing, JPEG Compression and Total Variance Minimization as the pre-processing defense

We give an example of one of the images from the ImageNet dataset (see Figure 5.34). In this example, we have the original image of the *cabbage butterfly* correctly predicted with MobileNetV2 with 92.6 % confidence. Next, we have the adversarial version of this image, generated with the HopSkipJump in the *untargeted* setting, which MobileNetV2 recognized as a *plastic bag* with 10.7 % confidence. The attack was successful. But then show that the Spatial Smoothing defense defends (pre-processes) this image successfully, meaning that the model was able to predict the image correctly as a *cabbage butterfly*, with the 94.9 % confidence.

The last thing that we want to highlight in this example is the difference in L^2 distances. Namely, the L^2 difference between the adversarial image and the original is 21.02, whereas

the L^2 difference between the defended image and the original is 13.71, which shows us that the defense has managed to bring the adversarial image closer to the original image, which helps the model to be able to make the correct decision again.

(a) Original Image, predicted as *cabbage butterfly* with 92.6% confidence

(b) Adversarial Image, predicted as *plastic bag* with 10.7 % confidence

(c) Defended Image, predicted as *cabbage buttetfly* with 94.9 % confidence

Figure 5.34: A comparison of the original image (a), the same image attacked with HopSkipJump attack (b) in the *untargeted* setting and the attacked image defended with the Spatial Smoothing defense (b). The figure shows that the attack was successful, but also that the defense successfully defended against the attack. The L^2 difference between the adversarial image and the original is 21.02, whereas the L^2 difference between the defended image and the original is 13.71. The classifier is MobileNetV2.

Table 5.1: CLEVER scores of the PreActResNet18 and AdvTrainedPreActResNet18 on the adversarial images generated by Carlini and Wagner L^2 adversarial attack under different target settings and with different pre-processing defenses

	CLEVER norm L^2			CLEVER norm L^∞		
	least likely class	next class	no target class	least likely class	next class	no target class
PreActResNet18						
No defenses	0.11729	0.12792	0.11187	0.00252	0.00286	0.00295
SS	0.02599	0.02626	0.0236	0.00071	0.00072	0.00072
JPEG	0.04642	0.06952	0.04384	0.00149	0.00186	0.0012
TVM	0.00399	0.01235	0.01937	0.0000	0.00013	0
SS + JPEG	0.00558	0.00574	0.00609	0.0002	0.00018	0.00023
SS + TVM	0.00593	0.0219	0.00337	0.00027	0.00101	0.00013
SS + JPEG + TVM	0.0118	0.01104	0.00387	0.00043	0.00043	0.00015
JPEG + TVM	0.05579	0.04603	0.03179	0.00136	0.001	0.00135
AdvTrainedPreActResNet18						
No defenses	0.15272	0.05773	0.00132	0.00338	0.00304	5e-05
SS	0.02314	0.00647	0.01898	0.00072	0.00017	0.00058
JPEG	0.04138	0.02351	0.0007	0.00306	0.0008	3e-05
TVM	0.01287	0.01305	0.01225	0.00012	0.00027	0.00042
SS + JPEG	0.00552	0.02028	0.0082	0.00021	0.0008	0.00032
SS + TVM	0.0267	0.01433	0.00719	0.00079	0.00078	0.00033
SS + JPEG + TVM	0.00338	0.00204	0.02905	9e-05	8e-05	0.00116
JPEG + TVM	0.05983	0.0345	0.02796	0.002	0.00178	0.00097

5.3 CLEVER scores

In this subsection, we discuss CLEVER scores recorded in our experiments with and without pre-processing defenses, where a higher CLEVER score indicates the higher robustness of the classifier.

CIFAR-10 dataset

Our main goal when examining the obtained CLEVER scores is to evaluate if they correlate positively with the success rates through different settings. CLEVER calculates the robustness score and if it is reliable in most cases, then it would be a practical measurement for evaluating attacks and defenses. Table 5.1shows the CLEVER scores grouped by the Carlini and Wagner L^2 adversarial attack. From this table, there are several interesting observations to be made.

If we look at the baseline case (no defenses), we can see that in the least likely class

targeted setting, the AdvTrainedPreActResNet18 scored a higher CLEVER score w.r.t L^2 norm than the PreActResNet18 (0.15272 compared to 0.11729, which is in line with success rates of CW L^2 attack in the baseline, see Figure 5.4. But consider also that 0.11729 is one of the highest scores in Table 5.1 for the L^2 norm. This indicates that the classifier, in this case, is more robust than the others. However, when we look at the success rates for the baseline, we see that CW L^2 attack has a high success rate (of about 60 %) against PreActResNet18. The same conclusion holds for the other two settings (*targeted least likely* and *untargeted*) as well.

With pre-processing defenses, CW L^2 attacks have very similar success rates in all three target settings, which is why one would expect that their CLEVER scores are also similar as well. This indeed is the case for several cases, like for example 0.01235 and 0.01305 with Total Variance Minimization in the *targeted next class* setting or 0.05579 and 0.05983 with JPEG Compression combined with Total Variance Minimization in the *targeted least likely class* setting. However, this also does not hold for several cases, like for example 0.04384 and 0.0007 with JPEG Compression in the *untargeted* setting.

This means that while some CLEVER scores are in line with the success rates and observations made in the previous sections, others are not, and so using the CLEVER score as the only indicator of robustness would not be very useful for our experiments on CIFAR-10. When we compare the CLEVER scores for the CW L^2 attack with CLEVER scores for other attacks we notice that similar conclusions that we just discussed also hold in these cases as well. For better readability, the complete results are presented in Appendix A in Tables A.1 to A.6 and discuss here all important highlights.

Lastly, one would expect that with pre-processing defenses, the models would be more robust and that the CLEVER scores would reflect that. However, our experimental results do not allow us to conclude that. We would argue that the results are slightly worse with pre-processing defenses, indicating a lower level of robustness.

ImageNet dataset

Similar to CIFAR-10, we present CLEVER scores for ImageNet grouped by a single adversarial attack in **??**. This way, we can see the changes in the CLEVER score with different pre-processing defenses under different target settings. We notice that the CLEVER scores from different adversarial attacks lead to similar trends and conclusions and thus we only present here the CLEVER scores obtained with the FGSM adversarial attack. The complete results are presented in Appendix Chapter A in Tables A.7 to A.9 and A.11 to A.13.

From the Table A.10 we can notice some trends that correlate positively and some that correlate negatively with the actual success rates of the attacks. For example, we notice that in cases with individual pre-processing defenses we have in most cases an increase in the CLEVER score. However, this does not hold for the combinations of defenses.

An interesting observation is that the combination of Spatial Smoothing and JPEG

Table 5.2: CLEVER scores of the InceptionV3, MobileNetV2 and ResNet-50 on the adversarial images generated by FGSM adversarial attack under different target settings and with different pre-processing defenses

	CLEVER norm L^2			CLEVER norm L^∞		
	least likely class	next class	no target class	least likely class	next class	no target class
InceptionV3						
No defenses	0.16298	0.23112	2.0	0.00036	0.00791	0.00668
SS	0.24742	0.00494	2.0	0.00112	0.00136	0.00366
JPEG	0.68033	0.78041	2.0	0.002	0.00229	0.00629
TVM	0.03564	0.02002	0.03752	0.0001	0.00007	0.0001
SS + JPEG	0.79638	0.80107	1.86768	0.00144	0.00229	0.00235
SS + TVM	0.02222	0.01779	0.02942	0.0001	0.00014	8.00E-05
SS + JPEG + TVM	0.03715	0.04149	0.03496	0.00012	0.00011	0.0001
JPEG + TVM	0.0289	0.02801	0.02677	0.0001	0.00012	7e-05
MobileNetV2						
No defenses	0.18426	0.14277	1.01925	0.00061	0.00062	0.00331
SS	0.25983	0.26468	2.0	0.00118	0.00112	0.00272
JPEG	0.47149	0.58493	1.24635	0.0027	0.00263	0.00483
TVM	0.03391	0.00014	0.02647	0.0001	0.0001	0.0001
SS + JPEG	0.64274	0.33064	0.90888	0.00117	0.00128	0.00174
SS + TVM	0.04123	0.03478	0.03833	0.00014	0.0001	0.0002
SS + JPEG + TVM	0.05212	0.03565	9e-05	9e-05	0.00017	
JPEG + TVM	0.02359	0.01683	0.02974	7e-05	7e-05	0.00012
ResNet-50						
No defenses	2.0	0.65432	2.0	0.011	0.00336	0.04222
SS	2.0	0.38601	2.0	0.00449	0.00184	0.00917
JPEG	2.0	1.72062	2.0	0.00606	0.00336	0.02721
TVM	0.02947	0.02442	0.02637	0.00014	0.00015	0.00012
SS + JPEG	2.0	1.01856	2.0	0.00734	0.0032	0.01041
SS + TVM	0.03284	0.01449	0.02144	0.00013	6e-05	7e-05
SS + JPEG + TVM	0.01557	0.01153	0.00678	9e-05	3e-05	0.00011
JPEG + TVM	0.01015	0.00939	0.01425	8e-05	6e-05	7e-05

Compression has higher CLEVER scores, which somewhat correlates with the discussions above, where this combination usually performed very well (compared to others).

It is also noteworthy that the *untargeted* setting scores higher CLEVER scores, which is not expected because the success rates have been far higher in that setting than in the other two targeted settings. Similar conclusions and trends that we made in the case with the CIFAR-10 dataset, can be also made here. This means again that while some CLEVER scores are in line with the success rates and observations made in the previous sections, others are not, and so using the CLEVER score as the only indicator of robustness would not be very useful for our experiments on ImageNet as well.

CHAPTER 6

Conclusion and Future Work

This thesis compares and evaluates several different adversarial attacks and defenses on ImageNet and CIFAR-10 datasets under three different target settings. Additionally, adversarial defenses have been combined and evaluated with the hope to potentially obtain new defenses that have an increased level of robustness.

Let us look at our research questions and give answers to them.

RQ1. Against the state-of-the-art adversarial attacks, which defense mechanisms have the highest level of robustness, measured using different robustness evaluation metrics?

In the targeted setting, we have observed that models on both datasets (ImageNet and CIFAR-10) are much more robust against adversarial attacks, i.e. it is more difficult to force a specific class. The only attacks that managed to target specific classes with higher success rates were Carlini and Wagner attacks on both datasets and Auto PGD on the CIFAR-10. In the *untargeted* setting, PreActResNet18, MobileNetV2, InceptionV3 and ResNet-50 were weak against most of the attacks. While AdvTrainedPreActResNet18 was vulnerable against Auto PGD, CW L^∞, and HopSkipJump, it was stronger against other attacks. Comparing the results of the pre-processing defenses in the targeted setting, we notice a common trend where the defenses have almost entirely removed the success rates of the attacks. With an exception being Auto PGD against AdvTrainedPreActResNet18. It is also interesting to note that while Auto PGD was the strongest attack on the CIFAR-10 dataset in this setting, it was not successful on ImageNet at all.

When comparing the two targeted settings, we have to stress that the *targeted least likely* and *targeted next class* settings were not that similar on ImageNet as they were on the CIFAR-10 dataset.

When it comes to the *untargeted* settings the results match, as all models are very weak against all of the attacks. This does not hold for the AdvTrainedPreActResNet-

18, but as this model is adversarially re-trained, it hence already has a defense mechanism applied. We have noticed good improvements (i.e. reductions of success rates) in the *untargeted* setting with different pre-processing defenses on both datasets, with very similar trends.

Most notably, Auto PGD and Shadow attacks were very strong against all of the pre-processing defenses. Against those, the defenses have barely reduced their success rates. We argue here that pre-processing defenses provide better protection against attacks that generate adversarial images which are closer to the original images. An example of such an attack is Carlini and Wagner L^2 attack, which has been affected by the pre-processing defenses the most throughout our experiments.

Lastly, we highlight that adversarial training combined very nicely with the pre-processing defenses. In the experiments on the CIFAR-10 dataset, the different attacks on the AdvTrainedResNet-18 have scored lower success rates in most of the scenarios. However, adversarial training comes with the cost of lower accuracy of the model on natural images. In our case, we had a drop of about 12 % (from 95.1 to 83 %), which is significant and poses the question of whether this defense should be used.

RQ2. How do different neural network architectures impact the robustness of the defense mechanisms?

First, We noticed that attacks have lower success rates against InceptionV3 in the baseline, without defenses. This trend continued in the *untargeted* setting when we introduced pre-processing defenses as well. Even though it was close, we conclude that in the *untargeted* setting the attacks had slightly higher success rates against MobileNetV2 compared to the other two models. Lastly, the trend from the baseline that the attacks have lower success rates against InceptionV3 was not true for the targeted case. In future work, one could investigate these relations in more detail, possibly with more state-of-the-art neural network architectures.

RQ3. How do different characteristics of datasets impact the robustness of the defense mechanisms?

On ImageNet, the success of the pre-processing defenses has been slightly worse than in the *targeted least likely* class setting than in the *targeted next class* setting. We speculate that this directly correlates with the properties of the ImageNet dataset. Namely, ImageNet has 1000 classes and some of the classes are closer than the others, like for example pan and wok pan. Hence, it would make sense that attacks with such targets are more successful than when attacks with targets that are more distant, like for example least likely class. On CIFAR-10 we had more similar results between these two targeted settings.

RQ4. How well do defense mechanisms work against universal perturbations?

Overall in our experiments, the Universal Perturbations attacks have scored on average about $55 - 60$ % success rate on ImageNet and about 40 % success rate

on CIFAR-10, which is a reduction from about 90 % on ImageNet and 50 % on CIFAR-10 in the baseline without defenses. Still, we have to consider that the 50 % success rate means that every other image is misclassified. This indicates that the defense mechanisms do not work very well against this attack. We point out the adversarial training, Spatial Smoothing, and JPEG Compression combination of the defenses which reduced the success rate of this attack to 1.2 % on the CIFAR-10 dataset.

RQ5. To which extent can we improve the defense against adversarial attacks by adapting or combining different defense mechanisms?

We have seen that adversarial training and pre-processing defenses combine easily together. We have also seen that in the general case they either maintain the success rates of the attack from the cases when the defenses are used individually or they slightly improve the results. Some of the combinations, like adversarial training, Spatial Smoothing, and JPEG Compression, have had more success than others. We conclude that while combining defenses is easy, finding the optimal combination is challenging and also specific to the given case. In our experiments, we did not have any combination that worked well in all (or at least most) of the cases. In future work, one could dive deeper into the analyses of how to engineer/tailor the best defense combination for which case.

APPENDIX A

Auxiliary Tables and Figures

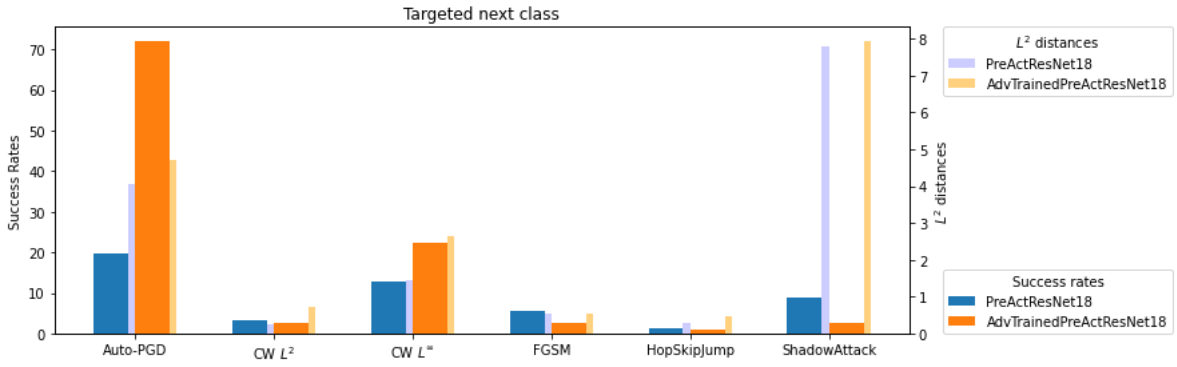


Figure A.1: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models targeting the next class, with Spatial Smoothing as the pre-processing defense

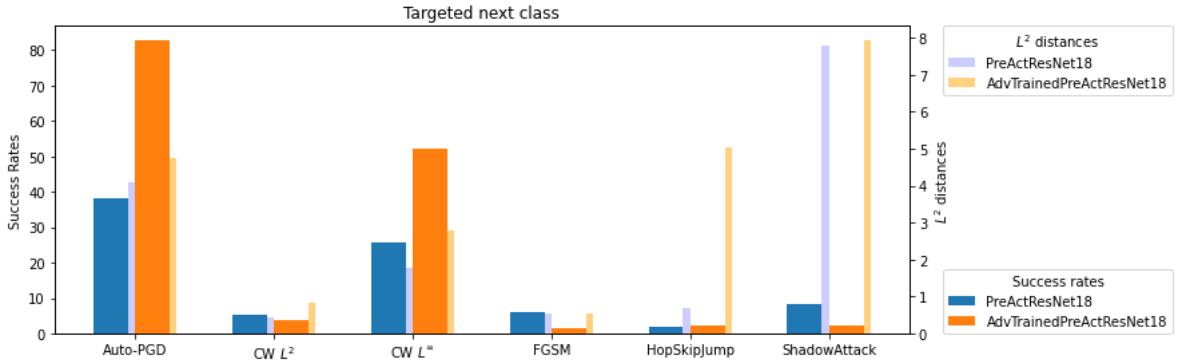


Figure A.2: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models targeting the next class, with JPEG Compression as the pre-processing defense

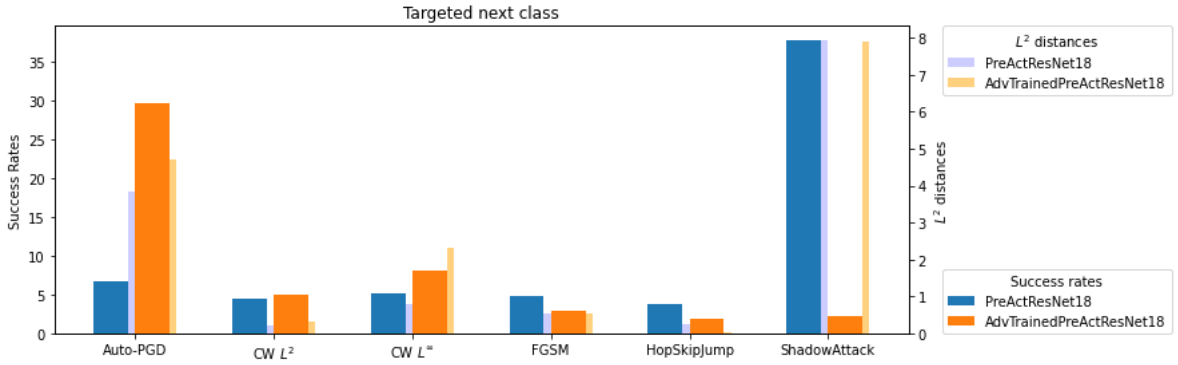


Figure A.3: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models targeting the next class, with Total Variance Minimization as the pre-processing defense

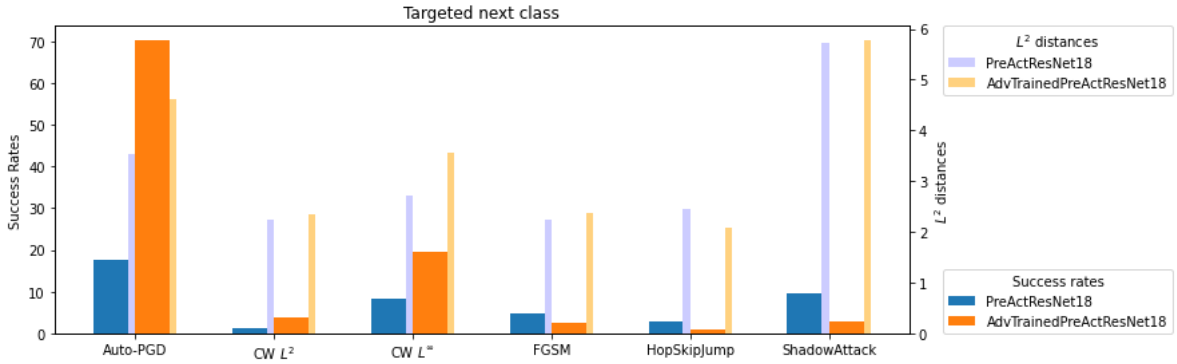


Figure A.4: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models targeting the next class, with Spatial Smoothing and JPEG Compression as the pre-processing defenses

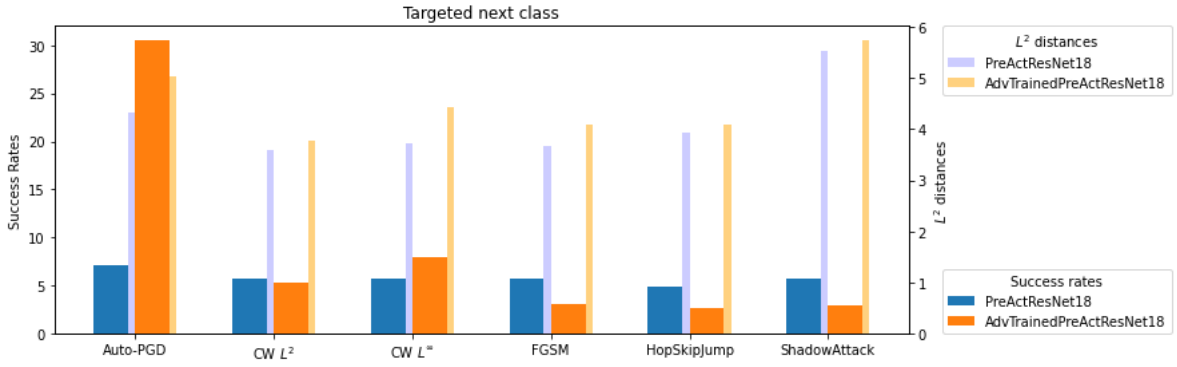


Figure A.5: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models targeting the next class, with Spatial Smoothing and Total Variance Minimization as the pre-processing defenses

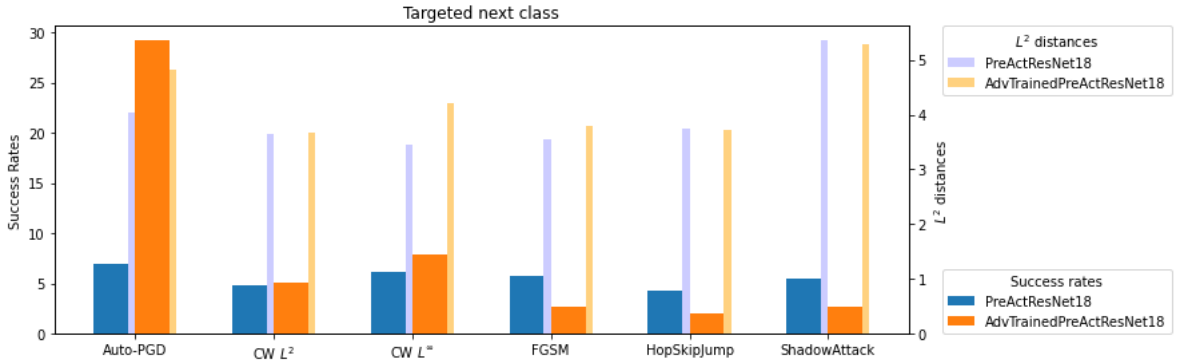


Figure A.6: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models targeting the next class, with JPEG Compression and Total Variance Minimization as the pre-processing defenses

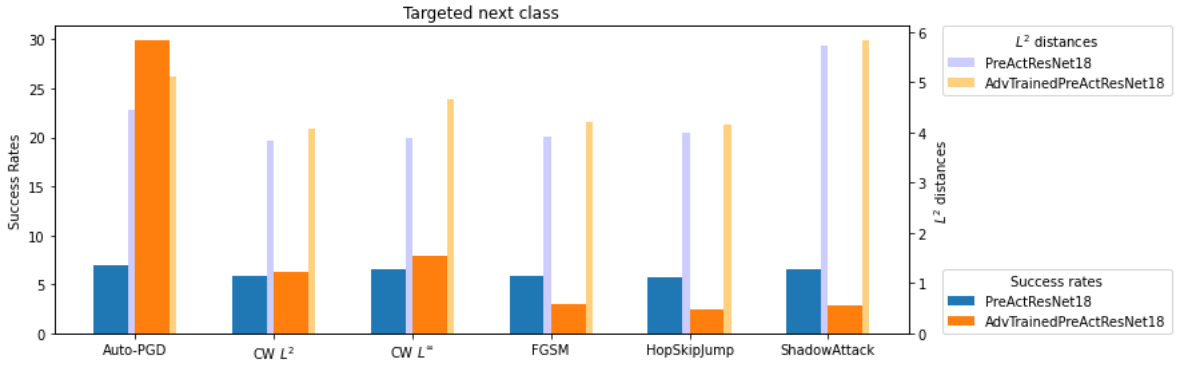


Figure A.7: Success rates of the different adversarial attacks and the L^2 distances to the original images for PreActResNet18 and AdvTrainedPreActResNet18 models targeting the next class, with Spatial Smoothing, JPEG Compression, and Total Variance Minimization as the pre-processing defenses

Table A.1: CLEVER scores of the PreActResNet-18 and AdvTrainedPreActResNet-18 on the adversarial images generated by Auto-PGD adversarial attack under different target settings and with different pre-processing defenses

	CLEVER norm L^2			CLEVER norm L^∞		
	least likely class	next class	no target class	least likely class	next class	no target class
PreActResNet-18						
No defenses	0.9308	0.08752	0.04154	0.02561	0.0322	0.01552
SS	0.25715	0.1233	0.12816	0.00795	0.02307	0.0038
JPEG	0.64702	0.22174	0.07669	0.01998	0.03048	0.01553
TVM	0.00306	0.39472	0.21266	0.01281	0.01225	0.00744
SS + JPEG	0.00108	0.03017	0.03669	0.00527	0.02073	0.00257
SS + TVM	0.34424	0.01748	0.0065	0.00949	0.01457	0.00804
SS + JPEG + TVM	0.00404	0.04759	0.01311	0.00752	0.01347	0.0091
JPEG + TVM	0.49231	0.012	0.01489	0.0168	0.0124	0.00587
AdvTrained PreActResNet-18						
No defenses	0.14538	0.11798	0.02013	0.0037	0.00257	0.02179
SS	0.0079	0.0114	0.05771	0.00024	0.00039	0.00132
JPEG	0.08887	0.15577	0.00468	0.00238	0.00353	0.01799
TVM	0.00029	0.02462	0.02273	-0.00022	0.00132	0.00097
SS + JPEG	0.01771	0.03708	0.01543	0.00077	0.00139	0.00055
SS + TVM	0.0194	0.03339	0.00406	0.00052	0.00134	0.00015
SS + JPEG + TVM	0.01477	0.03611	0.00463	0.00058	0.0012	0.00023
JPEG + TVM	0.01321	0.03585	0.02662	0.00062	0.00169	0.00074

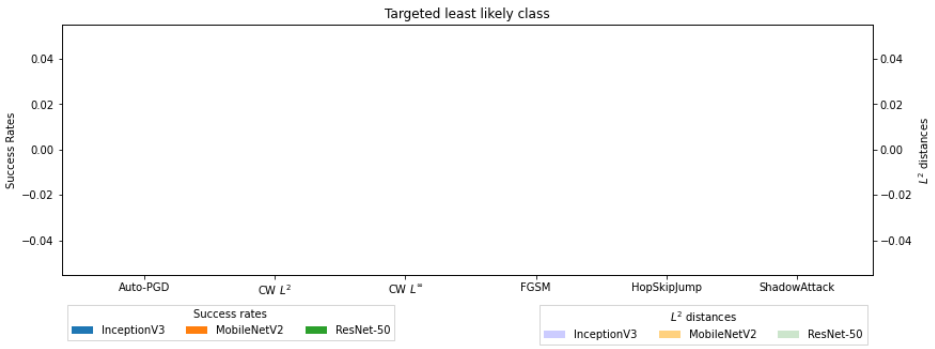


Figure A.8: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2 and ResNet-50 models targeting the least likely class, with Total Variance Minimization as the pre-processing defense

Table A.2: CLEVER scores of the PreActResNet-18 and AdvTrainedPreActResNet-18 on the adversarial images generated by Carlini and Wagner L^∞ adversarial attack under different target settings and with different pre-processing defenses

	CLEVER norm L^2			CLEVER norm L^∞		
	least likely class	next class	no target class	least likely class	next class	no target class
PreActResNet-18						
No defenses	0.02752	0.16636	0.01065	0.00204	0.00328	0.003
SS	0.24904	0.15573	0.00487	0.00694	0.00428	0.00243
JPEG	0.07636	0.00424	0.05161	0.00207	0.00327	0.00149
TVM	0.06215	0.00148	0.26539	0.0075	0.00774	0.0088
SS + JPEG	0.30187	0.18715	0.13131	0.00921	0.00432	0.00406
SS + TVM	0.3129	0.00941	0.51441	0.01112	0.01081	0.0178
SS + JPEG + TVM	0.3369	0.01457	0.50715	0.01196	0.01046	0.01591
SS + TVM	0.0082	0.00509	0.34403	0.00058	0.00782	0.01341
AdvTrained PreActResNet-18						
No defenses	0.16989	0.05629	0.04181	0.00386	0.00196	0.00175
SS	0.02539	0.01131	0.00976	0.00062	0.00034	0.00027
JPEG	0.03794	0.01533	0.02924	0.0021	0.00096	0.00098
TVM	0.02393	- 0.00105	0.00695	0.00029	0	0.00039
SS + JPEG	0.00398	0.02229	0.00939	0.00021	0.00074	0.00037
SS + TVM	0.02311	0.01671	0.00897	0.00078	0.00065	0.00025
SS + JPEG + TVM	0.003	0.00174	0.01437	0.00012	8e-05	0.00063
SS + TVM	0.03702	0.00975	0.02344	0.00162	0.00037	0.00107

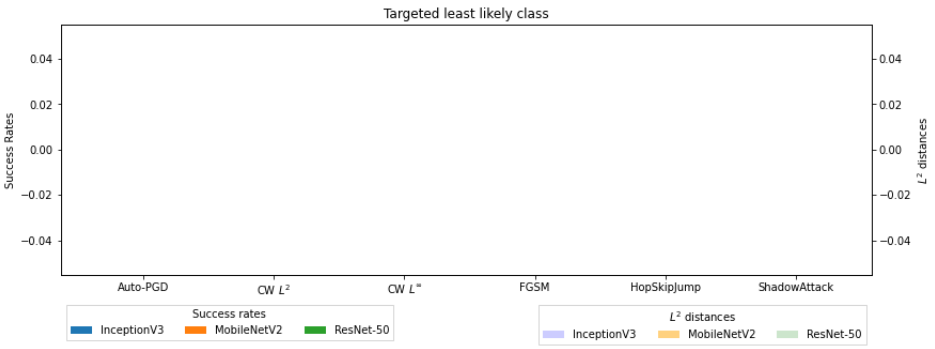


Figure A.9: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2 and ResNet-50 models targeting the least likely class, with Spatial Smoothing and JPEG Compression as the pre-processing defense

Table A.3: CLEVER scores of the PreActResNet-18 and AdvTrainedPreActResNet-18 on the adversarial images generated by FGSM adversarial attack under different target settings and with different pre-processing defenses

	CLEVER norm L^2			CLEVER norm L^∞		
	least likely class	next class	no target class	least likely class	next class	no target class
PreActResNet-18						
No defenses	0.55593	0.61692	0.44223	0.01766	0.01785	0.01305
SS	0.5017	0.01824	0.36401	0.01438	0.01156	0.01021
JPEG	0.60855	0.05411	0.07848	0.02025	0.01817	0.01208
TVM	0.23673	0.2455	0.22036	0.0068	0.00848	0.00799
SS + JPEG	0.53258	0.07995	0.38962	0.01912	0.01379	0.01334
SS + TVM	0.32653	0.01782	0.43475	0.01137	0.01445	0.01517
SS + JPEG + TVM	0.00391	0.02908	0.00731	0.01242	0.01498	0.00581
JPEG + TVM	0.01713	0.00601	0.01345	0.01157	0.01289	0.01194
AdvTrained PreActResNet-18						
No defenses	0.02025	0.03223	0.05117	0.00076	0.00112	0.00151
SS	0.0068	0.00683	0.00651	0.00018	0.00016	0.00024
JPEG	0.03803	0.06079	0.03532	0.0015	0.00242	0.00171
TVM	0.01201	0.00686	0.00167	0.00015	0.00031	0
SS + JPEG	0.01548	0.02878	0.01837	0.00063	0.00091	0.00063
SS + TVM	0.00218	0.01867	0.01809	9e-05	0.00087	0.0006
SS + JPEG + TVM	0.02546	0.00252	0.00486	0.00093	0.00011	0.00018
JPEG + TVM	0.01377	0.00452	0.02025	0.0004	0.00015	0.00077

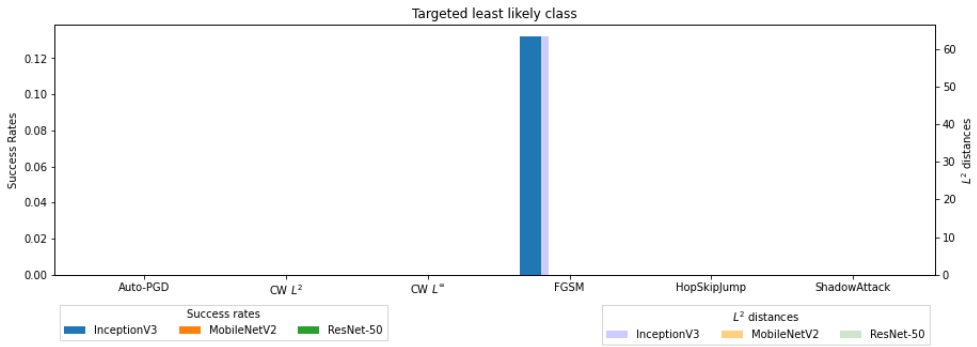


Figure A.10: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2 and ResNet-50 models targeting the least likely class, with Spatial Smoothing and Total Variance Minimization as the pre-processing defense

Table A.4: CLEVER scores of the PreActResNet-18 and AdvTrainedPreActResNet-18 on the adversarial images generated by HopSkipJump adversarial attack under different target settings and with different pre-processing defenses

	CLEVER norm L^2			CLEVER norm L^∞		
	least likely class	next class	no target class	least likely class	next class	no target class
PreActResNet-18						
No defenses	0.55593	0.61692	0.44223	0.01766	0.01785	0.01305
SS	0.5017	0.01824	0.36401	0.01438	0.01156	0.01021
JPEG	0.60855	0.05411	0.07848	0.02025	0.01817	0.01208
TVM	0.23673	0.2455	0.22036	0.0068	0.00848	0.00799
SS + JPEG	0.53258	0.07995	0.38962	0.01912	0.01379	0.01334
SS + TVM	0.32653	0.01782	0.43475	0.01137	0.01445	0.01517
SS + JPEG + TVM	0.00391	0.02908	0.00731	0.01242	0.01498	0.00581
JPEG + TVM	0.01713	0.00601	0.01345	0.01157	0.01289	0.01194
AdvTrained PreActResNet-18						
No defenses	0.02025	0.03223	0.05117	0.00076	0.00112	0.00151
SS	0.0068	0.00683	0.00651	0.00018	0.00016	0.00024
JPEG	0.03803	0.06079	0.03532	0.0015	0.00242	0.00171
TVM	0.01201	0.00686	0.00167	0.00015	0.00031	0
SS + JPEG	0.01548	0.02878	0.01837	0.00063	0.00091	0.00063
SS + TVM	0.00218	0.01867	0.01809	9e-05	0.00087	0.0006
SS + JPEG + TVM	0.02546	0.00252	0.00486	0.00093	0.00011	0.00018
JPEG + TVM	0.01377	0.00452	0.02025	0.0004	0.00015	0.00077

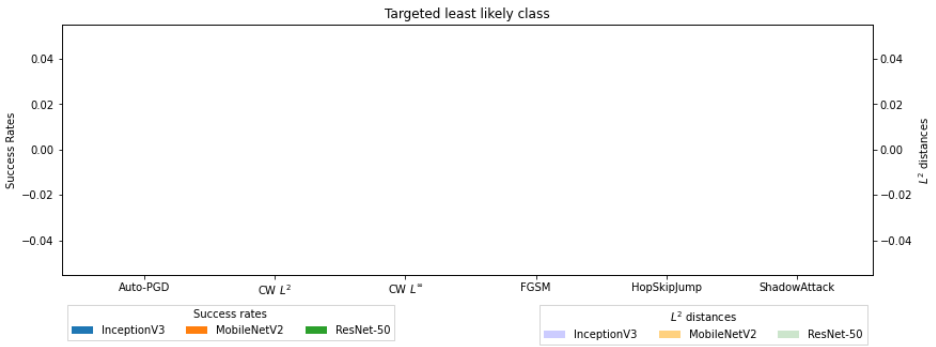


Figure A.11: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2 and ResNet-50 models targeting the least likely class, with JPEG Compression and Total Variance Minimization as the pre-processing defense

Table A.5: CLEVER scores of the PreActResNet-18 and AdvTrainedPreActResNet-18 on the adversarial images generated by ShadowAttack adversarial attack under different target settings and with different pre-processing defenses

	CLEVER norm L^2			CLEVER norm L^∞		
	least likely class	next class	no target class	least likely class	next class	no target class
PreActResNet-18						
No defenses	0.97576	0.02598	0.01818	0.00409	0.02549	0.02273
SS	0.01856	0.00136	0.00625	0.00702	0.01094	0.01339
JPEG	1.10684	0.01587	0.04054	0.03084	0.00878	0.02411
TVM	0.01459	0.06115	0.23896	0.00919	0.0088	0.00751
SS + JPEG	0.00559	0.01256	0.03042	0.01114	0.01501	0.01646
SS + TVM	0.00683	0.39461	0.00958	0.00047	0.01342	0.00726
SS + JPEG + TVM	0.39715	0.40332	0.27376	0.01324	0.01373	0.01097
JPEG + TVM	0.3319	0.3281	0.03402	0.00998	0.01086	0.00776
AdvTrained PreActResNet-18						
No defenses	0.02414	0.05133	0.03456	0.00068	0.00151	0.00088
SS	0.01315	0.01051	0.00414	0.00039	0.00028	8e-05
JPEG	0.03068	0.03399	0.02205	0.00082	0.00094	0.00069
TVM	0.08146	0.0168	0.0481	0.00246	0.0006	0.0012
SS + JPEG	0.00384	0.00169	0.02062	0.00012	5e-05	0.0006
SS + TVM	0.00923	0.04199	0.03576	0.00043	0.00146	0.00139
SS + JPEG + TVM	0.01481	0.03676	0.04863	0.00061	0.00125	0.002
JPEG + TVM	0.04792	0.0065	0.04845	0.00183	0.00025	0.00169

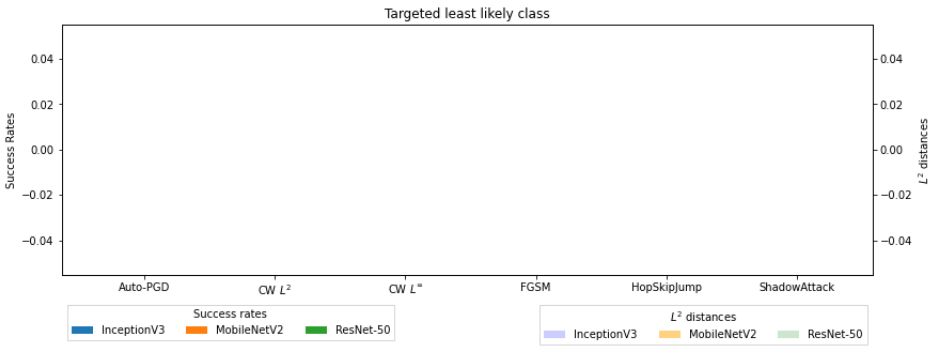


Figure A.12: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2 and ResNet-50 models targeting the least likely class, with Spatial Smoothing, JPEG Compression and Total Variance Minimization as the pre-processing defense

Table A.6: CLEVER scores of the PreActResNet-18 and AdvTrainedPreActResNet-18 on the adversarial images generated by Universal Perturbations adversarial attack with different pre-processing defenses

	CLEVER norm L^2 no target class	CLEVER norm L^∞ no target class
PreActResNet-18		
No defenses	0.59704	0.01764
SS	0.00747	0.01562
JPEG	0.00269	0.01812
TVM	0.27069	0.00866
SS + JPEG	0.62079	0.00103
SS + TVM	0.37318	0.01239
SS + JPEG + TVM	0.39718	0.01291
JPEG + TVM	0.31617	0.01257
AdvTrained PreActResNet-18		
No defenses	0.05259	0.00299
SS	0.00918	0.0002
JPEG	0.03259	0.00141
TVM	0.01556	0.00034
SS + JPEG	0.01653	0.00047
SS + TVM	0.01846	0.00062
SS + JPEG + TVM	0.02389	0.00084
JPEG + TVM	0.02185	0.00094

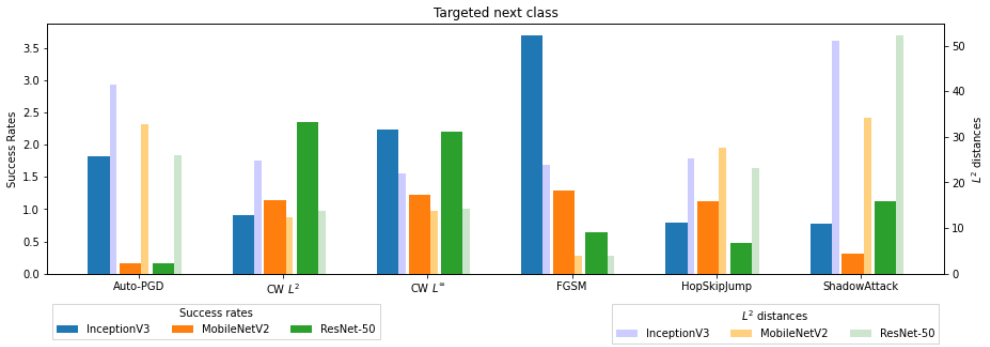


Figure A.13: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2 and ResNet-50 models targeting the next class, with Spatial Smoothing and JPEG Compression as the pre-processing defense

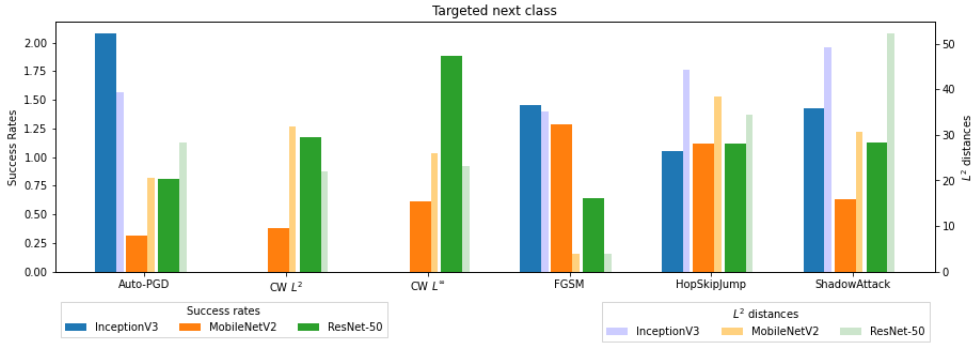


Figure A.14: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2 and ResNet-50 models targeting the next class, with Spatial Smoothing and Total Variance Minimization as the pre-processing defense

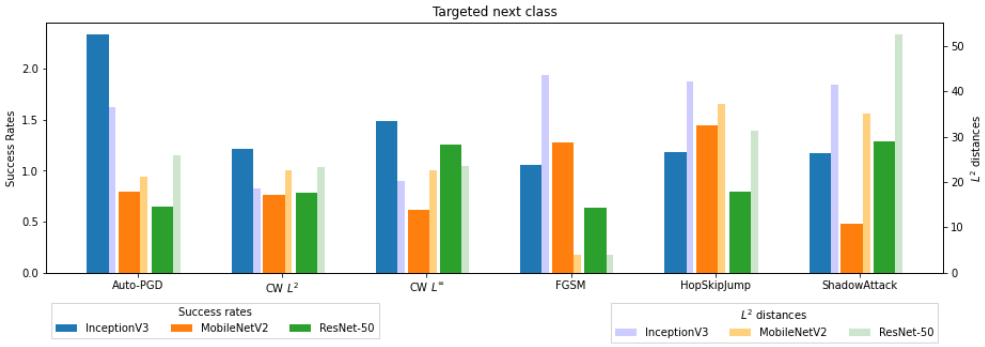


Figure A.15: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2 and ResNet-50 models targeting the next class, with JPEG Compression and Total Variance Minimization as the pre-processing defense

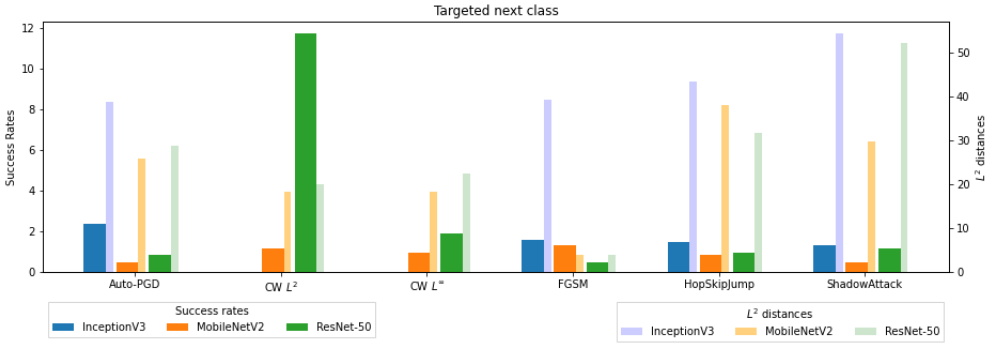


Figure A.16: Success rates of the different adversarial attacks and the L^2 distances to the original images for InceptionV3, MobileNetV2 and ResNet-50 models targeting the next class, with Spatial Smoothing, JPEG Compression and Total Variance Minimization as the pre-processing defense

Table A.7: CLEVER scores of the InceptionV3, MobileNetV2 and ResNet-50 on the adversarial images generated by Auto-PGD adversarial attack under different target settings and with different pre-processing defenses

	CLEVER norm L^2			CLEVER norm L^∞		
	least likely class	next class	no target class	least likely class	next class	no target class
InceptionV3						
No defenses	0.80329	0.8182	0.03852	0.0309	0.00076	0.00014
SS	0.03333	0.0043	0.03158	0.0345	0.00074	0.00005
JPEG	0.00429	0.04037	0.00991	0.0032	0.0015	0.00254
TVM	0.0053	0.0049	0.03275	0.0057	0.00488	0.00299
SS + JPEG	0.00367	0.01527	0.03762	0.0279	0.00253	1.34E03
SS + TVM	0.03129	0.01324	0.03673	0.0268	0.0041	0.00384
SS + JPEG + TVM	0.0054	0.01598	0.00365	0.0369	0.00111	0.00457
JPEG + TVM	0.00533	0.03012	0.03082	0.02483	0.00245	0.00287
MobileNetV2						
No defenses	0.71059	0.7507	0.01548	0.03043	0.03923	0.00639
SS	0.03322	0.02881	0.00572	0.00752	0.00161	0.00277
JPEG	0.03864	0.03418	0.0011	0.03441	0.00648	0.00805
TVM	0.02456	0.02939	0.01727	0.00299	0.00656	0.00279
SS + JPEG	0.00578	0.0213	0.03768	0.0321	0.03116	0.00274
SS + TVM	0.03735	0.03399	0.00431	0.02007	0.0079	0.00701
SS + JPEG + TVM	0.03859	0.03352	0.00732	0.00394	0.01593	0.00385
JPEG + TVM	0.03015	0.01947	0.02097	0.03527	0.02283	0.00558
ResNet-50						
No defenses	0.50749	0.8302	0.02287	0.00086	0.0102	0.0069
SS	0.03752	0.04065	0.01991	0.00672	0.00351	0.00791
JPEG	0.01678	0.01431	0.00928	0.00624	0.00149	0.00151
TVM	0.0407	0.01528	0.00481	0.00097	0.00951	0.00289
SS + JPEG	0.0002	0.01171	0.0074	0.00327	0.00213	0.00147
SS + TVM	0.00114	0.00405	0.0351	0.0083	0.00233	0.00285
SS + JPEG + TVM	0.03789	0.03132	0.03429	0.0043	0.00642	0.00579
JPEG + TVM	0.00811	0.01296	0.01507	0.00814	0.00384	0.0002

Table A.8: CLEVER scores of the InceptionV3, MobileNetV2 and ResNet-50 on the adversarial images generated by Carlini and Wagner L^2 adversarial attack under different target settings and with different pre-processing defenses

	CLEVER norm L^2			CLEVER norm L^∞		
	least likely class	next class	no target class	least likely class	next class	no target class
InceptionV3						
No defenses	0.11259	0.11256	0.11129	0.00031	0.0003	0.0003
SS	0.23928	0.23604	0.20907	0.00047	0.00051	0.00049
JPEG	0.26619	0.24266	0.23917	0.00063	0.00064	0.00055
TVM	0.09552	0.09336	0.09385	0.00025	0.00025	0.00025
SS + JPEG	0.52552	0.43048	0.50905	0.00059	0.0006	0.00045
SS + TVM	0.0516	0.05181	0.05093	0.00013	0.00013	0.00013
SS + JPEG + TVM	0.03214	0.02915	0.02808	0.00013	0.00014	0.00014
JPEG + TVM	0.02124	9e-05	0.01938	7e-05	6e-05	6e-05
MobileNetV2						
No defenses	0.07542	1.43311	0.07616	0.00019	0.00081	0.00018
SS	0.05842	0.12913	0.06278	0.00014	0.00043	0.00019
JPEG	0.06295	0.2868	0.06029	0.00018	0.00064	0.00013
TVM	0.04165	0.02081	0.04194	0.00011	4e-05	0.00013
SS + JPEG	0.03783	0.09011	0.0001	0.00013	0.00027	7e-05
SS + TVM	0.03771	0.01523	5e-05	3e-05	7e-05	3e-05
SS + JPEG + TVM	0.03734	0.00707	0.02697	0.0002	1e-05	0.00018
JPEG + TVM	0.01749	0.01747	0.01386	8e-05	7e-05	8e-05
ResNet-50						
No defenses	0.41826	0.40558	0.34706	0.00054	0.00058	0.00055
SS	0.00014	0.0262	0.02767	0.0002	0.00027	0.00036
JPEG	0.10825	0.0822	0.08155	0.00044	0.00031	0.0004
TVM	0.01405	0.01362	0.01621	0.00012	0.00011	0.00011
SS + JPEG	0.03184	0.03245	0.03273	0.00017	0.00022	0.00016
SS + TVM	0.01348	0.01073	0.00771	6e-05	6e-05	4e-05
SS + JPEG + TVM	0.00645	0.0077	0.00667	6e-05	7e-05	5e-05
JPEG + TVM	0.00855	0.00834	0.01022	5e-05	6e-05	7e-05

Table A.9: CLEVER scores of the InceptionV3, MobileNetV2 and ResNet-50 on the adversarial images generated by Carlini and Wagner L^∞ adversarial attack under different target settings and with different pre-processing defenses

	CLEVER norm L^2			CLEVER norm L^∞		
	least likely class	next class	no target class	least likely class	next class	no target class
InceptionV3						
No defenses	0.11259	0.11256	0.11129	0.00031	0.0003	0.0003
SS	0.23928	0.23604	0.20907	0.00047	0.00051	0.00049
JPEG	0.26619	0.24266	0.23917	0.00063	0.00064	0.00055
TVM	0.09552	0.09336	0.09385	0.00025	0.00025	0.00025
SS + JPEG	0.52552	0.43048	0.50905	0.00059	0.0006	0.00045
SS + TVM	0.0516	0.05181	0.05093	0.00013	0.00013	0.00013
SS + JPEG + TVM	0.03214	0.02915	0.02808	0.00013	0.00014	0.00014
JPEG + TVM	0.02124	9e-05	0.01938	7e-05	6e-05	6e-05
MobileNetV2						
No defenses	0.07542	1.43311	0.07616	0.00019	0.00081	0.00018
SS	0.05842	0.12913	0.06278	0.00014	0.00043	0.00019
JPEG	0.06295	0.2868	0.06029	0.00018	0.00064	0.00013
TVM	0.04165	0.02081	0.04194	0.00011	4e-05	0.00013
SS + JPEG	0.03783	0.09011	0.0001	0.00013	0.00027	7e-05
SS + TVM	0.03771	0.01523	5e-05	3e-05	7e-05	3e-05
SS + JPEG + TVM	0.03734	0.00707	0.02697	0.0002	1e-05	0.00018
JPEG + TVM	0.01749	0.01747	0.01386	8e-05	7e-05	8e-05
ResNet-50						
No defenses	0.41826	0.40558	0.34706	0.00054	0.00058	0.00055
SS	0.00014	0.0262	0.02767	0.0002	0.00027	0.00036
JPEG	0.10825	0.0822	0.08155	0.00044	0.00031	0.0004
TVM	0.01405	0.01362	0.01621	0.00012	0.00011	0.00011
SS + JPEG	0.03184	0.03245	0.03273	0.00017	0.00022	0.00016
SS + TVM	0.01348	0.01073	0.00771	6e-05	6e-05	4e-05
SS + JPEG + TVM	0.00645	0.0077	0.00667	6e-05	7e-05	5e-05
JPEG + TVM	0.00855	0.00834	0.01022	5e-05	6e-05	7e-05

Table A.10: CLEVER scores of the InceptionV3, MobileNetV2 and ResNet-50 on the adversarial images generated by FGSM adversarial attack under different target settings and with different pre-processing defenses

	CLEVER norm L^2			CLEVER norm L^∞		
	least likely class	next class	no target class	least likely class	next class	no target class
InceptionV3						
No defenses	0.16298	0.23112	2.0	0.00036	0.00791	0.00668
SS	0.24742	0.00494	2.0	0.00112	0.00136	0.00366
JPEG	0.68033	0.78041	2.0	0.002	0.00229	0.00629
TVM	0.03564	0.02002	0.03752	0.0001	0.00007	0.0001
SS + JPEG	0.79638	0.80107	1.86768	0.00144	0.00229	0.00235
SS + TVM	0.02222	0.01779	0.02942	0.0001	0.00014	8.00E-05
SS + JPEG + TVM	0.03715	0.04149	0.03496	0.00012	0.00011	0.0001
JPEG + TVM	0.0289	0.02801	0.02677	0.0001	0.00012	7e-05
MobileNetV2						
No defenses	0.18426	0.14277	1.01925	0.00061	0.00062	0.00331
SS	0.25983	0.26468	2.0	0.00118	0.00112	0.00272
JPEG	0.47149	0.58493	1.24635	0.0027	0.00263	0.00483
TVM	0.03391	0.00014	0.02647	0.0001	0.0001	0.0001
SS + JPEG	0.64274	0.33064	0.90888	0.00117	0.00128	0.00174
SS + TVM	0.04123	0.03478	0.03833	0.00014	0.0001	0.0002
SS + JPEG + TVM	0.05212	0.03565	9e-05	9e-05	0.00017	
JPEG + TVM	0.02359	0.01683	0.02974	7e-05	7e-05	0.00012
ResNet-50						
No defenses	2.0	0.65432	2.0	0.011	0.00336	0.04222
SS	2.0	0.38601	2.0	0.00449	0.00184	0.00917
JPEG	2.0	1.72062	2.0	0.00606	0.00336	0.02721
TVM	0.02947	0.02442	0.02637	0.00014	0.00015	0.00012
SS + JPEG	2.0	1.01856	2.0	0.00734	0.0032	0.01041
SS + TVM	0.03284	0.01449	0.02144	0.00013	6e-05	7e-05
SS + JPEG + TVM	0.01557	0.01153	0.00678	9e-05	3e-05	0.00011
JPEG + TVM	0.01015	0.00939	0.01425	8e-05	6e-05	7e-05

Table A.11: CLEVER scores of the InceptionV3, MobileNetV2 and ResNet-50 on the adversarial images generated by HopSkipJump adversarial attack under different target settings and with different pre-processing defenses

	CLEVER norm L^2			CLEVER norm L^∞		
	least likely class	next class	no target class	least likely class	next class	no target class
InceptionV3						
No defenses	0.0125	0.11601	0.05103	0.0009	0.00031	0.00132
SS	0.00216	0.00215	0.00621	0.0010	0.00048	0.00012
JPEG	0.03575	0.25317	0.0655	0.0042	0.00052	0.00532
TVM	0.01033	0.09274	0	0.0002	0.00025	0.00341
SS + JPEG	0.0007	0.44147	0.001	0.0006	0.00063	1.20E-02
SS + TVM	0.01208	0.00052	0.00199	0.0003	0.00013	0.0004
SS + JPEG + TVM	0.03472	0.002	0.0009	0.00017	0.00006	
JPEG + TVM	0.002	0.02088	0.00027	0.0004	0.0001	0.0004
MobileNetV2						
No defenses	1.39674	1.39246	1.37913	0.00085	0.0009	0.00081
SS	0.12904	0.13131	0.13386	0.00045	0.00043	0.00047
JPEG	0.15425	0.3076	0.23951	0.0007	0.00072	0.00075
TVM	0.02055	0.02104	0.01983	4e-05	5e-05	4e-05
SS + JPEG	0.00206	0.08923	0.08058	0.00028	0.00027	0.00029
SS + TVM	0.01576	0.01878	0.01264	8e-05	5e-05	9e-05
SS + JPEG + TVM	0.01575	0.0067	0.0066	3e-05	2e-05	3e-05
JPEG + TVM	0.01458	0.01732	0.01878	6e-05	7e-05	5e-05
ResNet-50						
No defenses	0.97145	0.93654	0.85776	0.0003	0.00035	0.00032
SS	0.13815	0.12662	0.13415	0.00031	0.00027	0.00026
JPEG	0.06248	0.05899	0.06706	0.00054	0.00027	0.00031
TVM	0.00034	0.0001	0.02675	9e-05	0.00011	7e-05
SS + JPEG	0.02896	0.02649	0.02674	0.0001	0.0001	0.0001
SS + TVM	0.00017	0.00017	0.01781	5e-05	4e-05	5e-05
SS + JPEG + TVM	0.0085	0.01906	0.02543	9e-05	2e-05	0.0001
JPEG + TVM	0.01583	0.00715	0.01612	3e-05	4e-05	3e-05

Table A.12: CLEVER scores of the InceptionV3, MobileNetV2 and ResNet-50 on the adversarial images generated by ShadowAttack adversarial attack under different target settings and with different pre-processing defenses

	CLEVER norm L^2			CLEVER norm L^∞		
	least likely class	next class	no target class	least likely class	next class	no target class
InceptionV3						
No defenses	0.203	0.40572	0.02628	0.0047	0.00816	0.00294
SS	0.00868	0.02447	0.0134	0.0060	0.00708	0.00869
JPEG	0.00956	0.02054	0.01725	0.0073	0.00802	0.00636
TVM	0.01345	0.01907	0.01571	0.0080	0.00633	0.00283
SS + JPEG	0.00894	0.01367	0.0306	0.0023	0.00712	2.00E-05
SS + TVM	0.00801	0.0283	0.00395	0.0027	0.00706	0.00324
SS + JPEG + TVM	0.00958	0.031	0.02444	0.0033	0.00372	0.00236
JPEG + TVM	0.03813	0.01466	0.03571	0.00297	0.00516	0.00634
MobileNetV2						
No defenses	0.09708	0.09754	0.08136	0.00157	0.00776	0.00238
SS	0.01136	0.02699	0.00348	0.00928	0.00226	0.00969
JPEG	0.02331	0.00263	0.02664	0.00032	0.00764	0.00626
TVM	0.04076	0.03959	0.00539	0.0067	0.00945	0.00451
SS + JPEG	0.02078	0.00005	0.00277	0.00894	0.0053	0.00253
SS + TVM	0.01222	0.00119	0.01983	0.00956	0.00084	0.00181
SS + JPEG + TVM	0.02728	0.02237	0.00824	0.01008	0.00599	0.00078
JPEG + TVM	0.00076	0.00682	0.02812	0.00825	0.00985	0.00774
ResNet-50						
No defenses	0.73786	0.90548	0.04448	0.00525	0.00639	0.00236
SS	0.01213	0.00398	0.01925	0.00566	0.00617	0.00936
JPEG	0.03686	0.02042	0.03717	0.00262	0.00845	0.00783
TVM	0.00057	0.02638	0.01535	0.00673	0.00902	0.00261
SS + JPEG	0.01347	0.03104	0.00418	0.00091	0.00898	0.00269
SS + TVM	0.00279	0.01219	0.03646	0.00889	0.00525	0.00545
SS + JPEG + TVM	0.0091	0.003	0.00255	0.0089	0.00802	0.00759
JPEG + TVM	0.00961	0.03566	0.00793	0.00653	0.00736	0.00309

Table A.13: CLEVER scores of the InceptionV3, MobileNetV2 and ResNet-50 on the adversarial images generated by Universal Perturbations adversarial attack with different pre-processing defenses

	CLEVER norm L^2 no target class	CLEVER norm L^∞ no target class
InceptionV3		
No defenses	1.2941	0.00473
SS	2.0	0.00333
JPEG	2.0	0.00349
TVM	0.02806	9e-05
SS + JPEG	1.31211	0.00236
SS + TVM	0.02124	4e-05
SS + JPEG + TVM	0.01187	5e-05
JPEG + TVM	0.02636	5e-05
MobileNetV2		
No defenses	0.1142	0.00049
SS	0.07698	0.00035
JPEG	0.17065	0.0005
TVM	0.02082	6e-05
SS + JPEG	0.05791	0.00022
SS + TVM	0.02058	5e-05
SS + JPEG + TVM	0.01268	5e-05
JPEG + TVM	0.03737	0.0001
ResNet-50		
No defenses	0.03724	0.00019
SS	0.06448	0.00021
JPEG	0.30148	0.00607
TVM	0.0248	0.00017
SS + JPEG	0.08048	0.0004
SS + TVM	0.01551	4e-05
SS + JPEG + TVM	0.01545	5e-05
JPEG + TVM	0.01025	3e-05

List of Figures

List of Tables

List of Algorithms

Bibliography

[Bot10] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[BR18] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[BRB18] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[CH20] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020.

[CJW20] Jianbo Chen, Michael I. Jordan, and Martin J. Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 1277–1294. IEEE, 2020.

[CLC+19] Minhao Cheng, Thong Le, Pin-Yu Chen, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[CNK+17] Cody A. Coleman, D. Narayanan, Daniel Kang, Tian Zhao, Jian Zhang, Luigi Nardi, Peter Bailis, K. Olukotun, C. Ré, and M. Zaharia. Dawnbench : An end-to-end deep learning benchmark and competition. 2017.

[CW17] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 39–57. IEEE Computer Society, 2017.

[DDEK12] Mark A Davenport, Marco F Duarte, Yonina C Eldar, and Gitta Kutyniok. Introduction to compressed sensing., 2012.

[DDS+09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[DGR16] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M. Roy. A study of the effect of JPG compression on adversarial images. *CoRR*, abs/1608.00853, 2016.

[Dre90] Stuart E Dreyfus. Artificial neural networks, back propagation, and the kelley-bryson gradient procedure. *Journal of guidance, control, and dynamics*, 13(5):926–928, 1990.

[GRCvdM18] Chuan Guo, Mayank Rana, Moustapha Cissé, and Laurens van der Maaten. Countering adversarial images using input transformations. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[GSG20] Amin Ghiasi, Ali Shafahi, and Tom Goldstein. Breaking certified defenses: Semantic adversarial examples with spoofed robustness certificates. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[HZC+17] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[IEAL18] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2142–2151. PMLR, 2018.

[KGB17] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017.

[KH09] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[LDG18] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In Michael Bailey, Thorsten Holz, Manolis Stamatogiannakis, and Sotiris Ioannidis, editors, *Research in Attacks, Intrusions, and Defenses - 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings*, volume 11050 of *Lecture Notes in Computer Science*, pages 273–294. Springer, 2018.

[Lem12] Claude Lemaréchal. Cauchy and the gradient method. *Doc Math Extra*, 251(254):10, 2012.

[LGTB97] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[MC01] Danilo Mandic and Jonathon Chambers. *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. Wiley, 2001.

[MFFF17] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 86–94. IEEE Computer Society, 2017.

[Mit97] Tom M. Mitchell. *Machine learning, International Edition*. McGraw-Hill Series in Computer Science. McGraw-Hill, 1997.

[MMS+18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[MOB+20] Ammar Mahmood, Ana Giraldo Ospina, Mohammed Bennamoun, Senjian An, Ferdous Sohel, Farid Boussaid, Renae Hovey, Robert B Fisher, and Gary A Kendrick. Automatic hierarchical classification of kelps using deep residual features. *Sensors*, 20(2):447, 2020.

[NST+18] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian Molloy, and Ben Edwards. Adversarial robustness toolbox v1.2.0. *CoRR*, 1807.01069, 2018.

[Per08] Chad Perrin. The cia triad. *Dostopno na: http://www. techrepublic. com/blog/security/the-cia-triad/488*, 2008.

[PMG16] Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *CoRR*, abs/1605.07277, 2016.

[PMW+16] Nicolas Papernot, Patrick D. 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 2016, San Jose, CA, USA, May 22-26, 2016*, pages 582–597. IEEE Computer Society, 2016.

[PŽ06] Remigijus Paulavičius and Julius Žilinskas. Analysis of different norms and corresponding lipschitz constants for global optimization. *Technological and Economic Development of Economy*, 12(4):301–306, 2006.

[RDGC95] David E Rumelhart, Richard Durbin, Richard Golden, and Yves Chauvin. Backpropagation: The basic theory. *Backpropagation: Theory, architectures and applications*, pages 1–34, 1995.

[ROF92] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[Ros60] Frank Rosenblatt. Perceptron simulation experiments. *Proceedings of the IRE*, 48(3):301–309, 1960.

[Sam59] Arthur L. Samuel. Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.*, 3(3):210–229, 1959.

[SIVA17] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 4278–4284. AAAI Press, 2017.

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1–9. IEEE Computer Society, 2015.

[SNG+19] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John P. Dickerson, Christoph Studer, Larry S. Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3353–3364, 2019.

[SSR+18] Skyler Speakman, Srihari Sridharan, Sekou L. Remy, Komminist Weldemariam, and Edward McFowland. Subset scanning over neural network activations. *CoRR*, abs/1810.08676, 2018.

[SVI+16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826. IEEE Computer Society, 2016.

[SZS+13] 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.

[SZS+14] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[WGZ+16] Qinglong Wang, Wenbo Guo, Kaixuan Zhang, Alexander G. Ororbia II, Xinyu Xing, C. Lee Giles, and Xue Liu. Learning adversary-resistant deep neural networks. *CoRR*, abs/1612.01401, 2016.

[WRK20] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[WZC+18] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[XEQ18] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *25th Annual Network*

and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018. The Internet Society, 2018.

[ZMH+94] Andreas Zell, Niels Mache, Ralf Hübner, Günter Mamier, Michael Vogt, Michael Schmalzl, and Kai-Uwe Herrmann. Snns (stuttgart neural network simulator). In *Neural network simulation environments*, pages 165–186. Springer, 1994.