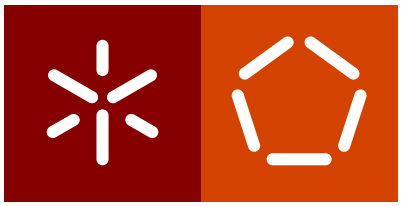
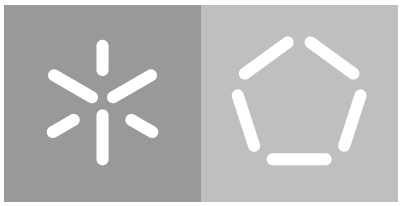


Universidade do Minho
Escola de Engenharia
Departamento de Informática

Tiago Manuel Sampaio Branco

Exploring Adversarial Attacks and Defenses

February 2024



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Tiago Manuel Sampaio Branco

Exploring Adversarial Attacks and Defenses

Master dissertation
Master Degree in Master´s Informatics Engineering

Dissertation supervised by
António José Borba Ramires Fernandes

February 2024

COPYRIGHT AND TERMS OF USE FOR THIRD PARTY WORK

ACKNOWLEDGEMENTS

I would like to express my gratitude to my primary supervisor, António Ramires, who guided me throughout this project. I would also like to thank my friends and family who supported me and offered deep insight into the study.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity.

I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

ABSTRACT

Deep Learning classifiers are capable of an outstanding performance. Yet, they are vulnerable to adversarial attacks, i.e. it is possible to craft a slightly modified version of a correctly classified image that, although its contents are still clearly recognisable to a human being, the classifier outputs an incorrect classification.

In this thesis we evaluate the effectiveness of adversarial attacks, namely their transferability to other models, and some proposed defenses. Transferability occurs when an adversarial sample is crafted with a model, and it succeeds in achieving a misclassification in another model.

To make this study as comprehensive as possible, we explore several attack methods, namely: Fast Gradient Sign Method (FGSM), Deepfool, Jacobian Saliency Map Attack (JSMA), Carlini, Projected Gradient Descent (PGD) and Few Pixels.

To evaluate the impact of the model's architecture in the transferability rate we use several common architectures: VGG16, three ResNet with different depths, and a small Convolution Neural Network. Two common datasets were used for evaluation: CIFAR-10 and German Traffic Sign Recognition Benchmark (GTSRB).

Different attack methods use different approaches and parameters to craft adversarial samples. Hence, it is not trivial to control the degree of perturbation. To be able to achieve the same level of perturbation with every method we resorted to an image comparison metric: Structural Similarity Index Measure (SSIM).

For each method we performed a search within its parameter space to find the parameters that on average attain a specific level of perturbation. To evaluate the impact of the level of perturbation on transferability rates, we evaluate two different values for the SSIM metric.

Our results show that while it is possible to craft an adversarial sample in a particular model, the transferability rates vary considerably from method to method.

Regarding defensive methods we explored Adversarial Training and Defensive Distillation. The results show that the ability to prevent an adversarial attack, or robustness, varies significantly depending on the conditions that the attack is performed and on the defensive methods used. Furthermore, there is a trade-off between robustness and accuracy, with defensive models having lower accuracy than non-defended models.

Keywords: Adversarial Attacks, FGSM, DeepFool, JSMA, PGD, Carlini, Adversarial Training, Defensive Distillation, SSIM, CIFAR-10, GTSRB

RESUMO

Modelos classificativos de Deep Learning são capazes de performances extraordinárias, superando mesmo a capacidade humana, no entanto são vulneráveis a ataques adversariais, por exemplo é possível criar uma versão modificada de uma imagem bem classificada, que apesar do ser facilmente reconhecível por um ser humano, o modelo classifica incorrectamente.

Nesta tese avaliamos a transferabilidade dos ataques adversariais, e algumas propostas de defesa. A transferabilidade occorre quando um adversarial gerado por um modelo é capaz de fazer outro modelo o classificar incorrectamente.

Para fazer este estudo o mais compreensivo possível, explorámos vários tipos de métodos de ataques, nomeadamente: FGSM, Deepfool, JSMA, Carlini, PGD and Few Pixels.

Para avaliar o impacto da arquitectura do modelo na taxa de transferabilidade, usamos várias arquitecturas conhecidas: VGG16, 3 ResNet com diferentes números de camadas, e uma rede pequena de 3 camadas convolucionais. Usamos 2 datasets conhecidos para esta avaliação: CIFAR-10 e GTSRB.

Diferentes métodos de ataques usam abordagens e parâmetros diferentes para gerar exemplos adversariais. Por isso, não é trivial controlar o grau de perturbação gerada pelos exemplos adversariais. Para sermos capazes de alcançar o mesmo nível de perturbação com todos os métodos recorremos a uma métrica de comparação de imagem: SSIM.

Para cada método fizemos uma procura no domínio dos parâmetros para encontrar os valores que em média geram um nível específico de perturbação. Para avaliar o impacto do nível de perturbação na taxa de transferabilidade, avaliámos 2 valores diferentes para a métrica SSIM.

Os nossos resultados mostram que enquanto é possível gerar um exemplo adversarial num modelo particular, a taxa de transferabilidade varia consideravelmente de método para método.

Em relação às defesas exploramos Adversarial Training e Defensive Distillation. Os resultados mostram que a capacidade de prevenir um ataque adversarial varia significativamente dependendo das condições em que o ataque é feito e nos próprios métodos.

Palavras-Chave: Ataques Adversariais, FGSM, DeepFool, JSMA, PGD, Carlini, Treino Adversarial, Defesa via Destilação, SSIM, CIFAR-10, GTSRB

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CNN Convolutional Neural Network. 1, 4, 26, 27

DE Differential Evolution. 19, 20, 21
DNN Deep Neural Network. 4

FGSM Fast Gradient Sign Method. iv, v, viii, ix, x, xi, xii, xiii, xiv, xvii, xviii, 6, 7, 8, 11, 18, 22, 28, 31, 34, 35, 36, 38, 40, 41, 54, 64, 65, 66, 68, 69, 70, 71, 72, 80, 84, 86, 87, 89, 91, 94, 95, 96, 97, 101, 106, 113, 114, 115, 126, 127, 128, 129, 130, 131

GTSRB German Traffic Sign Recognition Benchmark. iv, v, viii, xi, xii, xviii, xix, xx, xxi, 2, 18, 25, 26, 27, 64, 65, 72, 81, 84, 86, 87, 89, 90, 91, 92, 97, 98, 99, 100, 101, 103, 104, 105, 113

JSMA Jacobian Saliency Map Attack. iv, v, viii, ix, x, xiii, xvii, 11, 15, 18, 29, 31, 42, 43, 44, 45, 54, 56, 58, 63, 65, 117, 118

PGD Projected Gradient Descent. iv, v, viii, ix, x, xi, xiii, xv, xvi, xvii, xix, xx, 18, 19, 29, 31, 48, 49, 50, 51, 53, 54, 58, 61, 63, 64, 65, 75, 76, 77, 79, 80, 82, 86, 87, 89, 90, 91, 94, 96, 99, 101, 104, 106, 107, 120, 121, 122, 123, 134, 135, 136, 137, 138, 139

SSIM Structural Similarity Index Measure. iv, v, viii, ix, x, xi, xii, xiii, xiv, xv, xvi, xvii, xviii, xix, 2, 28, 29, 30, 31, 35, 36, 35, 37, 36, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 50, 51, 52, 53, 54, 55, 58, 61, 64, 65, 66, 68, 69, 70, 71, 72, 73, 74, 73, 74, 75, 76, 77, 78, 79, 80, 82, 83, 85, 86, 87, 90, 93, 100, 101, 104, 107, 108, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139

1

INTRODUCTION

It seems like everywhere one looks sees the words deep learning and neural networks. The ubiquity of these terms in the technological world is a reality. In the last few years, deep learning has led to very good performance on a broad variety of problems, such as visual recognition, speech recognition, and natural language processing.

On the other hand, such a success will naturally provoke research on how to fool these systems. These are called adversarial attacks, consisting, in summary, in finding small perturbations to images such that these adulterated images end up being misclassified.

This can cause major problems in some domains, for instance autonomous driving. Consider, for instance, that a Stop sign is adulterated in such way that a deep learning system sees it as a Right-of-way at an intersection.

These perturbations can be made so small that the original image and the adversarial attack are indistinguishable to the human eye Goodfellow et al. (2015). Even visible perturbations may have the appearance of noise, making it harder for humans to detect adversarial images. Several research studies have shown that Convolutional Neural Network (CNN)s can be led to misclassify these adversarial images with a variable degree of success Moosavi-Dezfooli et al. (2016)Su et al. (2019), although a human would have still correctly classify them with high confidence.

An adversarial attack can occur in a broad range of scenarios Zhang et al. (2020). The easiest being when the attacker has full knowledge of the object system it is attacking. In such a scenario, the attacker has access to the model architecture, its weights, and the datasets that were used to train the model. Such a scenario is called a white box attack.

On the other extreme, the attacker has no knowledge about the system. This implies that the attacker must select a model, which is potentially different from the object model, and train it with a potentially different dataset. This is a black box attack.

Between these two extremes there is a wide range of scenarios where the attacker has partial knowledge about the system, for instance, the dataset used to train the object model can be known.

Attacks can also aim at not only causing a misclassification, but lead the model to output a classification in a particular class. This is a targeted attack. These are harder to achieve than non-targeted attacks, but nonetheless possible.

Whenever the attacker model is different from the target model, the degree of success of an attack can be defined as the transferability rate, i.e., the percentage of adversarial samples that are able to fool the target system.

An action causes an equal reaction is true not only in physics, hence, researchers are also working on defenses on these attacks. The goal of a defensive method is to promote robustness to adversarial attacks, preventing misclassifications.

Different methods to generate adversarial samples have been devised in the last few years Ilyas et al. (2019) Papernot et al. (2016) with different proposals and with different degrees of success, contrastingly defenses against theses type of attacks have been developing concurrently, although at a slower pace.

1.1 MOTIVATION AND OBJECTIVES

The main motivation is to study the transferability of adversarial attacks, and robustness of the defenses.

Comparing these methods is not a trivial task as there is not a common setting of parameters. Different levels of perturbation will lead to different success on the attacks and defenses. To be able to compare these methods, a methodology is required, that allows us to compare theses techniques against each other on similar grounds. For example, attacks that change almost all the pixels of an image cannot be directly compared to an attack that only changes a small selection of pixels. To do so we must use some metric that allow us to create comparable attacks using different methods, i.e., to unify all different attacks under a common metric. In this work we will use SSIM Wang et al. (2004) to be able to evaluate the perceptual level of degradation introduced when crafting an adversarial sample.

Attacks will be evaluated considering transferability using gray box attacks, i.e., attacks are always crafted in a model and tested on a different model. The level of grayness is based on the architecture of the model, i.e., several architectures will be used to test transferability.

Attacks and defenses will be evaluated on two well known datasets: CIFAR-10 and GTSRB. While the former is common in these evaluations, the later has not been used in the available research on these issues. This allows our work to be compared with others based on the CIFAR-10 results, while at the same time introduce a new case study which is particularly relevant to autonomous driving.

The same approach will be followed when evaluating defensive methods.

1.2 DOCUMENT STRUCTURE

Besides the present chapter this thesis is organized as follows:

- Chapter 2 presents an overview of the methods in the areas relevant for this work, namely adversarial attack methods (Section 2.1), and defenses (Section 2.2);
- In Chapter 3 an evaluation is performed on the attack methods described in Chapter 2 focused on transferability;
- Chapter 4 presents an evaluation on defensive methods;
- Chapter 5 presents the conclusions and future work.

2

STATE OF THE ART

This chapter assumes the reader has a general knowledge of how neural networks work, more concretely convolutional neural networks. For a good introduction see Stanford University online course (Li et al.), Goodfellow, Bengio, and Courville's book Goodfellow et al. (2016) and Keras's creator Francois Chollet's book Chollet (2017).

This thesis addresses the weaknesses of neural networks and how effectively they can be deceived with adversary attacks. In Section 2.1 several methods of generating adversary attacks are described.

As a counter response some researches have explored methods to make CNNs more robust to the above mentioned attacks. These methods are reviewed in Section 2.2.

2.1 ADVERSARIALS ATTACKS

Considering a Deep Neural Network (DNN) model as a function F that receives an input X and outputs Y, this can be written as: $F(X) = Y$. The aim of an adversarial sample is to fool the model by applying a small perturbation δ to a legitimate sample X, such as $X^* = X + \delta$, that makes the model misclassify as $F(X^*) = Y^* \neq Y$. When this occurs, X^* is said to be an adversarial sample. When the minimum perturbation is the goal, this can be described as in Equation 1.

$$\arg\min_{\delta} \|(\delta)\| \ \ s.t.\ F(X+\delta) = Y^* \tag{1}$$

where Y^* can be any output as long as it is different from Y, and $\|\cdot\|$ is the appropriate norm to compare points in the input domain.

Crafting an adversarial sample is searching for a small perturbation of the input that will change the output to Y^*. Searching for this perturbations can be done using optimization techniques, simple heuristics, or brute force. The goal is not only to get the model to misclassify a given sample but to do it with high confidence (Goodfellow et al. (2015)).

Attacks can be classified regarding the goal of the attack as targeted, requiring the new output to belong to a particular class, or non-targeted, where any output is welcome as long as it is not of the original class. This will be discussed in Section 2.1.1.

Adversarial attacks can be further classified regarding the knowledge the attacker has of the original network and training procedure. Under this classification attacks can be classified as white (full knowledge), grey (some knowledge) or black (no knowledge). This will be discussed in Section 2.1.2.

Following these definitions, some of the most relevant methods to perform adversarial attacks are presented in Sections 2.1.3 to 2.1.8.

2.1.1 *Targeted vs. non-targeted attacks*

As Equation 1 states, an attack consists in adding a perturbation to a sample so that its classification is no longer correct.

The new target classification can be a specific class or just any class that is different from the correct class. In the first case this is called a targeted attack, as the attack has the specific goal of producing a particular classification. On the other hand, when it is considered sufficient to get the model to provide an incorrect classification the attack is labelled as non-targeted (Goodfellow et al. (2015)).

A targeted attack aims at minimizing the loss relative to the new desired output class, whereas a non-targeted attack is based on maximizing the loss for the respective class.

2.1.2 *From White to Black Box Attacks*

In a white box attack the attacker has full knowledge of the model under attack. The network architecture is fully known, as well as its trained weights. Furthermore, the data set used to train the network is also known.

On the other side of the spectrum there is no knowledge available to the attacker regarding the model, and by consequence the trained weights, as well as the training set. The attacker has access to the trained model but only to get the classification label, as an oracle (Papernot et al. (2017)) . In some situations the attacker may even be limited to the number of times permitted to query the model.

Being able to query the model is essential in this case. With a sufficient number of queries the attacker can create a training dataset to train a new model. This model can then be used to produce adversarial samples, which can be tested on the original model.

Between these two extremes there is a wide range of situations, for instance the architecture may be known but not its weights. In this case having access to the training set could potentially make it easier to create a successful attack as the attacker could train an identical model from scratch.

Having knowledge of the data set used to train the model, but not the architecture, allows the attacker to create a model that provides the same classifications, replicating more closely the behaviour of the original model.

The methods to create adversarial images presented in this chapter mostly are gradient based. These methods require full knowledge of the model and its weights. However, these images will latter be used to attack other networks with both different weights and different architectures. Hence, the evaluation will be based not only on the ability of a method to create an adversarial image for a specific model, but also to on the transferability of the adversarial trait of the generated images when tested on non white box attacks.

2.1.3 *Fast Gradient Sign Method and some variations*

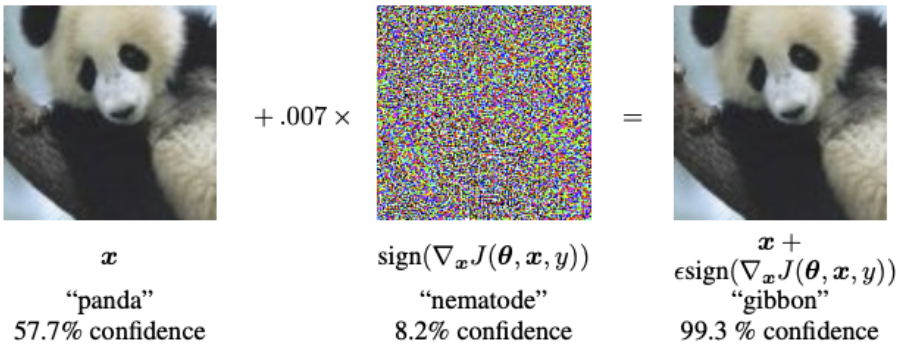


Figure 1.: FGSM example applied to GoogLeNet network on ImageNet dataset, where $\epsilon = 0.007$. Source: Goodfellow et al. (2015)

FGSM is an adversarial attack that uses the model gradients presented in Goodfellow et al. (2015). As the name implies, this is a time efficient method to generate adversarial samples. The approach of this attack is to add a perturbation to an input image that maximizes the loss function of the model on the original class, thus reducing the model prediction confidence on the original class, and eventually misclassify the sample.

This perturbation is found by performing a single iteration of the gradient ascent based on the derivative of the loss function with respect to the input image. This derivative gives a matrix of the same size as the input image with values that details how much each pixel of the image contributes to the loss value. FGSM is summarized in equation form on Equation 2.

$$X_* = X + \epsilon \cdot sign(\nabla_X J(X, y_{true})) \tag{2}$$

where X is the input image, ∇_X denotes the gradient of the loss function J with respect to the input image X with correct label y_{true}, *sign* is the sign operator which returns 1 for positive values and -1 for negative values while 0 remains unchanged, and ϵ is a small value that denotes the variation parameter which controls the perturbation's amplitude. An example of FGSM adversarial attack can be seen in Figure 1 where the network was fooled by misclassifying a panda image as a gibbon with high confidence. In this image we can see that for small values of ϵ the perturbation is hardly perceptible to the human eye. Note that this attack changes almost all image pixels, with only the ones whose gradient is zero not being updated.

Norms

The basic version of this attack is defined as L_∞ norm with the use of *sign* function but it can also be modified to apply other norms such as L_1 and L_2. As suggested in Papernot et al. (2017), to implement the L_1 norm we substitute the expression in Equation 2 for the expression seen in Equation 3, where $num\_occurr\_max\_value$ is the total number of occurrences of max gradient absolute value.

$$\begin{aligned} abs\_grad &= |\nabla_X J(X, y_{true})| \\ max\_abs\_grad &= \max(abs\_grad) \\ X_* &= X + \epsilon \cdot sign(\nabla_X J(X, y_{true})) * max\_abs\_grad / num\_occurr\_max\_value. \end{aligned} \tag{3}$$

Similarly, to implement the L_2 norm we modify the final expression for the one seen in Equation 4.

$$\begin{aligned} square\_sum\_grad &= \sum (\nabla_X J(X, y_{true}))^2 \\ X_* &= X + \epsilon \cdot \frac{\nabla_X J(X, y_{true})}{\sqrt{square\_sum\_grad}}. \end{aligned} \tag{4}$$

All variations on FGSM attack further described in this section can also be modified to implement these norms by using the appropriate equations.

Basic Iterative FGSM

This is a straightforward extension of the basic FGSM which applies it iteratively using a small step size (Kurakin et al. (2017a)). This method can be described in equation form as seen in Equation 5.

$$X_0 = X, \qquad X_{N+1} = Clip_{X,\epsilon}\{X_N + \alpha \cdot sign(\nabla_X J(X_N, y_{true}))\} \tag{5}$$

where X is the original image, X_i is the adversarial sample at iteration i, ϵ is the max perturbation allowed, and α the amount of perturbation per pixel in each iteration. Function

$Clip_{X,\epsilon}\{X_N\}$ performs per-pixel clipping of the image X_N, to limit pixel alterations to $[x_i - \epsilon, x_i + \epsilon]$.

The authors of Kurakin et al. (2017b) used $\alpha = 1$ considering the range $[0, 255]$, and defined the number of iterations to be $min(\epsilon + 4, 1.25\epsilon)$. They chose this function heuristically since it proved sufficient, yet restricted enough to keep the computational cost manageable.

Target Class FGSM

FGSM as presented before is a non-targeted attack. This version of the FGSM aims at make a model misclassify the adversarial samples as some specific class. This is achieved by, instead of increasing the loss for the original class, decreasing the loss for a particular target class. To increase the probability of some target class, $p(y_{target}|\mathbf{X})$, we perform the gradient descent, as shown in Equation 6

$$X_* = X - \epsilon \cdot sign(\nabla_X J(X, y_{target})) \tag{6}$$

A variation of this method, as suggested in Kurakin et al. (2017a), is the *least likely class* method, where it is chosen the least likely class from the model prediction as the target class, $y_{LL} = \arg\min_y \{p(y|\mathbf{X})\}$. , thus we get Equation 7.

$$X_* = X - \epsilon \cdot sign(\nabla_X J(X, y_{LL})) \tag{7}$$

The targeted version of FGSM can also be defined as an iterative approach as seen in Equation 8.

$$X_0 = X, \qquad X_{N+1} = Clip_{X,\epsilon}\{X_N - \alpha \cdot sign(\nabla_X J(X_N, y^*))\} \tag{8}$$

where y^* can be either y_{target} or y_{LL}.

2.1.4 *Deepfool*

Deepfool (Moosavi-Dezfooli et al. (2016)) is a non-targeted method that aims at minimizing the amount of perturbation required to craft an adversarial image by using an iterative linearization approach. In Figure 2 we can see a comparison between FGSM and Deepfool adversarials and their corresponding perturbation, Deepfool leads to a smaller perturbation Moosavi-Dezfooli et al. (2016), found to be estimated as 5 times smaller than FGSM.

The main idea of this attack is to take steps in the direction of the closest decision boundary until it crosses to the other side thus making the network misclassify the image as another class. To achieve this the authors assume that the neural networks classes are sep-

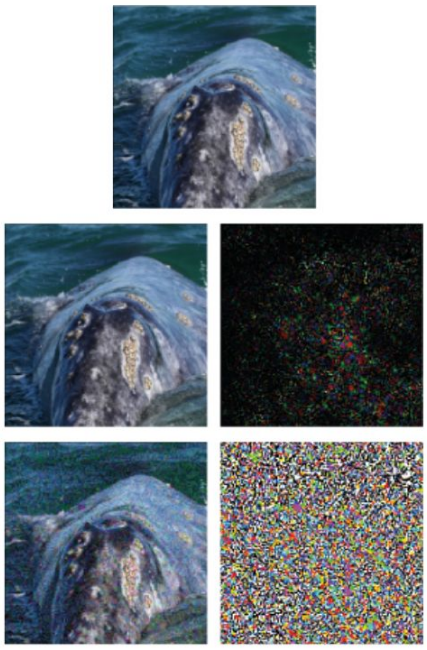


Figure 2.: Example of adversarial perturbations. First row: the original image X that is classified as "whale". Second row: the image $X + r$ classified as "turtle" and the corresponding perturbation r computed by Deepfool. On third row: the image classified as "turtle" and corresponding perturbations computed using FGSM. Note that these perturbations are amplified for visualization purposes. Source:Moosavi-Dezfooli et al. (2016)

arated by hyper-planes. This is a linear approximation and the iterative behaviour of the algorithm allows to reach the other side of the decision boundary. In Figure 3 we can see this linear approximation between the classes boundaries and the real ones.

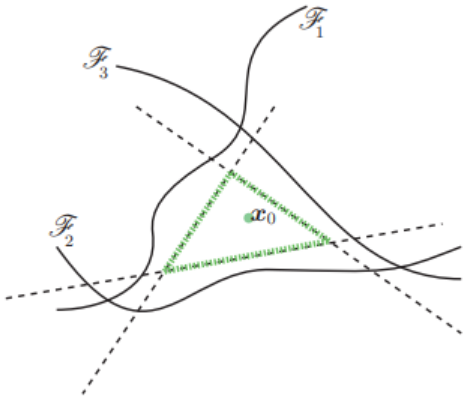


Figure 3.: Decision boundaries linear approximation and real boundaries of a model. Source:Moosavi-Dezfooli et al. (2016)

To calculate the distance between the image the closest decision boundary we perform an orthogonal projection. In Figure 4 we can see this projection on a binary classifier, where $\mathcal{F} = x : w^T x + b = 0$ is the hyperplane, x_0 is the data point, and $\Delta(x_0; f)$ is the robustness of function f at point x_0 which is the smallest amount of perturbation needed to arrive to the decision boundary. To change the classification of data point x_0 we must add a small step to make the data point cross the boundary. This distance is calculated using the orthogonal projection of x_0 onto $\mathcal{F}$ as seen in Equation 9, where w corresponds to the gradient of function f with respect to the data point x_0.

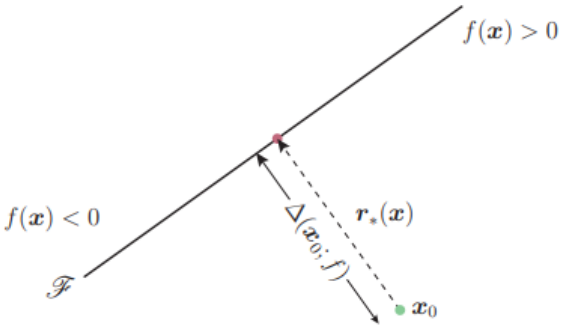


Figure 4.: Distance between a sample and the boundary on a linear classifier. Source:Moosavi-Dezfooli et al. (2016)

$$\begin{aligned} r_*(x_0) &:= \arg\min \|r\|_2 \\ \text{subject to } & sign(f(x_0 + r)) \neq sign(f(x_0)) \\ &= -\frac{f(x_0)}{\|w\|_2^2} w. \end{aligned} \tag{9}$$

At each iteration f, the binary classifier, is linearized around the current point x_i and the minimal perturbation of linearized classifier is computed as seen in Equation 10.

$$\arg\min_{r_i} \|r_i\|_2 \text{ subject to } f(x_i) + \nabla f(x_i)^T r_i = 0. \tag{10}$$

Where perturbation r_i at iteration i is calculated using Equation 9.

This can be generalized for the multiclass non linear classifiers, where we first need to calculate the closest decision boundary, and then apply the perturbation on that direction

that corresponds to the minimum perturbation $r_*(x_0)$ which is the vector that projects x_0 on the hyperplane indexed by $\hat{l}(x_0)$.

$$\hat{l}(x_0) = \arg\min_{k \neq \hat{k}(x_0)} \frac{|f_k(x_0) - f_{\hat{k}(x_0)}(x_0)|}{\|\nabla f_k(x_0) - \nabla f_{\hat{k}(x_0)}(x_0)\|_2}. \tag{11}$$

To calculate the closest decision boundary we use Equation 11, where $\hat{k}$ corresponds to the class with highest score, and to calculate the perturbation we use Equation 12, where the minimum perturbation $r_*(x_0)$ is the vector that projects x_0 on the hyperplane indexed by $\hat{l}(x_0)$ found in Equation 11.

$$\hat{r_*}(x_0) = \frac{|f_l(x_0) - f_{\hat{k}(x_0)}(x_0)|}{\|\nabla f_l(x_0) - \nabla f_{\hat{k}(x_0)}(x_0)\|_2^2} \cdot (\nabla f_l(x_0) - \nabla f_{\hat{k}(x_0)}(x_0)) \tag{12}$$

The pseudo code of this algorithm can be seen in Algorithm 1. Note that this algorithm operates in a greedy way and is not guaranteed to converge to the optimal perturbation.

Algorithm 1: Deepfool: multi-class case

Input: Image x, classifier f.

Output: Perturbation $\hat{r}$.

Initialize $x_0 \leftarrow x, i \leftarrow 0$.
while $\hat{k}(x_i) = \hat{k}(x_0)$ **do**
  for $k \neq \hat{k}(x_0)$ **do**
    $w'_k \leftarrow \nabla f_k(x_i) - \nabla f_{\hat{k}(x_0)}(x_i)$
    $f'_k \leftarrow f_k(x_i) - f_{\hat{k}(x_0)}(x_i)$
  $\hat{l} \leftarrow \arg\min_{k \neq \hat{k}(x_0)} \frac{|f'_k|}{\|w'_k\|_2}$
  $r_i \leftarrow \frac{|f'_{\hat{l}}|}{\|w'_{\hat{l}}\|_2^2} w'_{\hat{l}}$
  $x_{i+1} \leftarrow x_i + r_i$
  $i \leftarrow i + 1,$
return $\hat{r} = \sum_i r_i$.

2.1.5 *Jacobian Saliency Map Attack*

JSMA is a class of algorithms to craft adversarial samples. Below is described the original JSMA algorithm based on Papernot et al. (2015), also known as Targeted JSMA, and the non-targeted attack variant introduced in Wiyatno and Xu (2018), where it removes the requirement to specify a target class as required in the original JSMA.

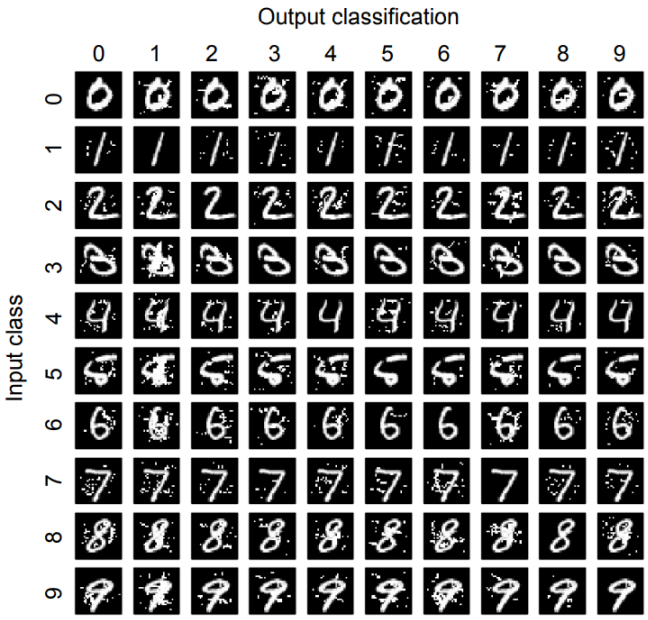


Figure 5.: Targeted adversarial samples using JSMA for different digits from MNIST dataset. Source:Papernot et al. (2015)

The main difference regarding the result of this method when compared to FGSM, is that in a single iteration FGSM can potentially affect all pixels, whereas JSMA will only affect up to two pixels.

Although only a small number of pixels may get disturbed, these perturbations end up as being clearly visible as will be shown, for instance, in 3.4.4.

Targeted JSMA

This method starts by finding the contribution of each pixel of the input image to each class. A selection of the pixels that positively influences the output for the given class is then gathered, with some perturbation being applied to those selected pixels. Some examples of this targeted attack can be seen in Figure 5.

To calculate the contribution of each pixel from the input image to each class of the network output we can calculate the Jacobian matrix to obtain the partial derivatives.

The Jacobian matrix give us the information about how each input pixel, or feature, influences the output, where large positive values mean that increasing the respective pixels will yield a large increase in the output by the model. Conversely, components with large negative values correspond to pixels that yield large decreases in the output by the model when their value is increased.

In the second stage of the process, we must determine the best pixel candidates to be perturbed to guide the model into misclassifying an input image in a chosen target class.

To achieve this, a saliency map is computed, where each pixel is given a score based on how much impact that pixel will have on the target class and in all other classes. This pixel score is defined as the product of the gradient of the target class and the sum of the gradients of all other classes. We exclude pixels whose derivative of target class is negative or whose sum of the gradients of all other classes is positive. This ensures that we exclude pixels which negative impact the target class and those who have an overall positive contribution to the all other classes.

$$S^{+}(X,t)[i] = \begin{cases} 0 & \text{if } \frac{\partial F_t(X)}{\partial X_i} < 0 \text{ or } \sum_{j \neq t} \frac{\partial F_j(X)}{\partial X_i} > 0 \\ \left(\frac{\partial F_t(X)}{\partial X_i}\right) \left| \sum_{j \neq t} \frac{\partial F_j(X)}{\partial X_i} \right| & \text{otherwise} \end{cases} \tag{13}$$

In this saliency map, as presented in Equation 13, i is an input feature, and $F(X)$ refers to the last hidden layer logits output. The condition on the first line sets the score to zero for input features that have a negative target derivative, or an overall positive derivative on other classes. The derivative $\frac{\partial F_t(X)}{\partial X_i}$ must be positive in order for $F_t(X)$ to increase when the feature X_i increases and analogous, the sum of all the derivatives of the other classes, $\sum_{j \neq t} \frac{\partial F_j(X)}{\partial X_i}$, must be negative, or stay constant, so that if we increase the value of the input X_i it will not influence positively the output in those classes. Also, that is why the absolute value is taken in the second line, because the value should be negative. Thus, with the second line we can warrant that the pixels with highest score are the ones that increase the target class, or decrease all other classes significantly or both. This saliency map considers increasing the value of the input features in order to achieve misclassification.

The counter part to this saliency map is one that considers decreasing the input features value instead to attain the misclassification. It's follows the same construction principle, the only difference being in the inequalities that are flipped and the location of the absolute value in the second line as seen in Equation 14.

$$S^{-}(X,t)[i] = \begin{cases} 0 & \text{if } \frac{\partial F_t(X)}{\partial X_i} > 0 \text{ or } \sum_{j \neq t} \frac{\partial F_j(X)}{\partial X_i} < 0 \\ \left|\frac{\partial F_t(X)}{\partial X_i}\right| \left(\sum_{j \neq t} \frac{\partial F_j(X)}{\partial X_i}\right) & \text{otherwise} \end{cases} \tag{14}$$

The saliency maps corresponding to increase and decrease of pixel values are expressed as S^{+} and S^{-} respectively.

In the last step in the algorithm, after finding the input feature with the highest score in the positive saliency map, we perturbed the pixels to realize the adversary sample. In each iteration of the algorithm this step is repeated until the misclassification is achieved.

In Papernot et al. (2015) the authors proposed a more robust heuristic that instead of searching for the best candidate pixel it searches the best pair candidate. This approach is preferable because selecting two pixels instead of one is less strict and very few pixels

would meet the criteria when using the initial heuristic. This new search heuristic can be seen in Equation 15.

$$\arg\max_{p1,p2} \left(\sum_{i=p1,p2} \frac{\partial F_t(X)}{\partial X_i} \right) \times \left| \sum_{i=p1,p2} \sum_{j \neq t} \frac{\partial F_j(X)}{\partial X_i} \right| \tag{15}$$

For example, if pixel $p1$ has a target derivative of 5 but a sum of other classes derivatives equal to 0.1, while $p2$ as a target derivative equal to -0.5 and a sum of the other classes derivatives -6. Alone neither pixel won't match the search criteria in Equation 13, but combined the pair matches the saliency criteria defined in Equation 15. Using pairs improves the chance of obeying the selection criteria more often, although the method becomes computationally more expensive.

The full attack algorithm can be seen in Algorithm 2, showing 3 different stop conditions:

1. the adversarial sample is classified by the network as being the target class t
2. the max number of iterations has been reached
3. the feature domain Γ is empty, Γ refers to the number of input features valid for perturbation, all the pixels that do not have the maximum value in the color range (ex: 1 or 255) if we are using the incremental saliency map S^+, or the pixels that do not have the minimum value in the range(0) if we are using the decremental saliency map S^-.

The number of maximum iterations max_iter is defined by $max\_iter = \left[\frac{\#\ input\ features \cdot \gamma}{2} \right]$, where γ is the percentage of the total number of input features allowed to be perturbed.

Besides γ, the algorithm has another parameter, θ, that defines the amount of perturbation applied to each pixel.

Algorithm 2: JSMA full algorithm

Input: input $\mathbf{X}$, $\mathbf{Y}^*$ is the target classifier output, $\mathbf{F}$ is the classifier, $\boldsymbol{\gamma}$ is the percentage of the total number of input features allowed to be perturbed, $\boldsymbol{\theta}$ is the amount of perturbation applied to pixels, $\mathbf{n}$ the number of features of $\mathbf{X}$

Output: Adversarial sample $\mathbf{X}^*$

$X^* \leftarrow X$

$\Gamma = \{1...|X|\}$ `// search domain is all pixels`

$\text{max\_iter} = \frac{n \cdot \gamma}{2 \cdot 100}$

$s = \arg\max_j F(X^*)_j$ `// source class`

$t = \arg\max_j Y^*_j$ `// target class`

while $s \neq t$ *&* $iter < max\_iter$ *&* $\Gamma \neq \emptyset$ **do**

  Compute derivative $\nabla F(X^*)$

  $p_1, p_2 = \text{Saliency\_Map}(\nabla F(X^*), \Gamma, Y^*)$

  Modify p_1 and p_2 in X^* by θ

  Remove p_1 from Γ if $p_1 == 0$ for S^- or $p_1 == 1$ for S^+

  Remove p_2 from Γ if $p_2 == 0$ for S^- or $p_2 == 1$ for S^+

  $s = \arg\max_j F(X^*)_j$

  iter $++$

return X^*

Non-Targeted JSMA (NT-JSMA)

In Wiyatno and Xu (2018) proposed a variant of JSMA where there is no need to specify a target class to perform the attack. This algorithm instead of increasing the probability of some target class, it decreases the model's prediction confidence of the true class label. This is done by swapping the saliency maps employed in JSMA, so if we are increasing the value of the pixels we use the S^- map, and if we are decreasing the values we use the S^+ instead.

This variant has a more relaxed success criterion in comparison with JSMA because here we only need to ensure that the sample is misclassified, regardless of the final class.

2.1.6 *Carlini Method*

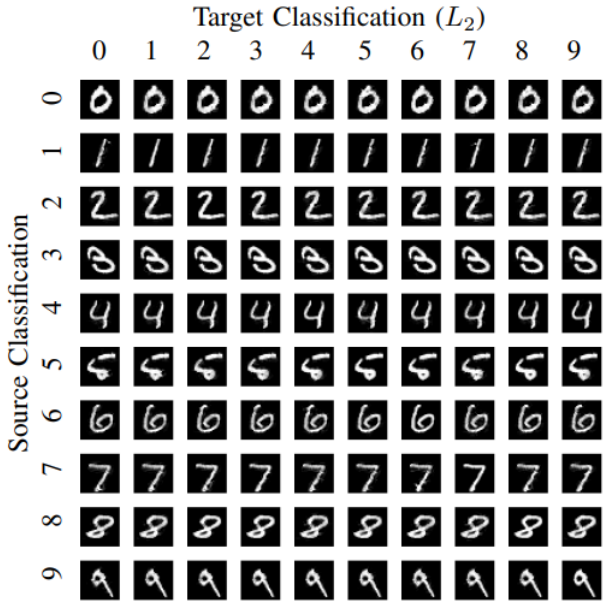


Figure 6.: Carlini L_2 adversary applied to the MNIST dataset performing a targeted attack for every source/target pair. Source: Carlini and Wagner (2017)

In Carlini and Wagner (2017) the authors propose a new attack methodology. The main idea of this attack is to implement all the constraints required to craft an adversarial sample as an objective function that can be solved by using a gradient descent optimizer such as Adam.

To understand how this is done first we need to formulate the constraints necessary to create an adversarial attack. This can be seen in Equation 16, where X is the input image, δ is the perturbation, D a distance function, C the classifier model, n are the dimensions of the input image, in case of a RGB image $n = 3$ and t is the target class.

$$\begin{aligned} &\text{minimize: } D(X, X+\delta) \\ &\text{such that: } C(X+\delta) = t \qquad \text{- constraint 1} \\ &X+\delta \in [0,1]^n \qquad \text{- constraint 2} \end{aligned} \tag{16}$$

In the first line we want to ensure that the adversarial sample it's close to original sample by using some distance metric, typically the distance metrics are specified in terms of L_p norms such as L_0, L_2 or L_∞. The second line shows a constraint that ensures that the adversarial sample is indeed misclassified by the classifier, this constraint is highly non-linear Carlini and Wagner (2017). And finally in the last line we have the constraint 2 where we want to ensure that the crafted adversarial is a valid image.

To solve this optimization problem the authors suggest to replace $C(X+\delta) = t$ for a function f, where this objective function f must satisfy $C(X+\delta) = t$ if and only if the output of this objective function on the perturbed image $X+\delta$ is less than or equal to zero, i.e. $f(X+\delta) \leq 0$. The authors evaluated different candidates for the function f in original paper and found that best suited to be Equation 17.

$$f(X) = \max(\max_{i \neq t} Z(X)_i - Z(X)_t, -k) \tag{17}$$

The term $\max_{i \neq t} Z(X)_i$ represents what is the highest logits value among non-targeted classes, $Z(X)_t$ is the logits of the target class. Thus $\max_{i \neq t} Z(X)_i - Z(X)_t$ is the difference between 'what the classifier thinks the current image probably is' and 'what we want to misclassify as'. The purpose of the parameter k is to control the strength of the adversarial sample, adjust the trade-off between the size of the perturbation and the success of the attack. Larger values for k may allow for more significant perturbations but could increase the risk of detection, on the other hand smaller values may result in smaller perturbations but may also reduce the likelihood of successful adversarial attacks. The optimal value for "k" may depend on the particular dataset, model architecture, or other factors.

Constraint 2 from Equation 16 with lower and upper bounds set to 0 and 1 respectively, cannot be optimized using gradient descent because it is not a differentiable function. To solve this it is necessary to convert this constraint to a differentiable function with the same codomain. To achieve this purpose the authors suggest the use of the *tanh* function. The range of *tanh* is $[-1, 1]$, hence we must add $+1$ and multiply by $\frac{1}{2}$ to force to be within the codomain required, see Equation 18. Note that W is the image converted in tanh space, this is done by converting the image to the range $[-1, 1]$ and then performing the tanh inverse function: arctanh.

$$\begin{aligned} \text{constraint: } X + \delta &= \frac{1}{2}(\tanh(W) + 1) \\ \delta &= \frac{1}{2}(\tanh(W) + 1) - X, \end{aligned} \tag{18}$$

Now putting everything in an equation we get Equation 19, where $\|\|^2$ is the L_2 norm and c is a constant that steers the trade off between the left and right terms of the equation (image similarity vs misclassification confidence), c must be greater than zero, having a small value for c results in the attack rarely succeeding and having a large value of c results in attacking being less effective (large value of L_2 distance) but always succeeding. The value of c can be found empirically by performing binary search, we start with a very small

value for c perform the attack and check if we attained misclassification if not we double the value of c and repeat until we get misclassification.

$$\begin{aligned} &\text{minimize: } \|\frac{1}{2}(\tanh(W)+1-X)\|_2^2 + c \cdot f(\frac{1}{2}(\tanh(W)+1) \\ &\text{where, } f(X) = \max(\max\{Z(X')_i : i \neq t\} - Z(X')_t, -k) \end{aligned} \tag{19}$$

Some examples of this attack can be seen in Figure 6.

In Carlini and Wagner (2017) it is stated that L_0 and L_2 norms of this attack produced lower perturbations (from 2 to 10 times smaller) than other attacks such Deepfool and JSMA. When comparing Figure 6 with the results for JSMA (Figure 5) it is clear that Carlini produces adversarial attacks that are less perceptible to the human eye.

2.1.7 *Projected Gradient Descent (PGD)*

Figure 7.: PGD L_∞ adversary applied to the GTSRB dataset performing a non-targeted attack, with maxim distortion allowed of $\epsilon = 0.03$, using a ResNet-50.

PGD (Madry et al. (2017)) is an iterative attack that aims to create a perturbation that maximizes the loss of a model while keeping the total perturbation magnitude bounded by a threshold ϵ. It's essentially a FGSM (Section 2.1.3) performed in each iteration, where the total perturbation is projected to the L_p ball between $\pm\epsilon$. Then we add the projected perturbation to the original unperturbed image and clip to valid values. This method

ensures that the perturbation sits between a defined range despite the number of steps and step sizes used. Some adversarial samples are shown in Figure 7

The pseudo code for a non-targeted L_∞ attack (considering images within $[0, 1]$ range) can be seen in Algorithm 3.

Algorithm 3: Non-targeted PGD: L_∞ norm

Input: Image x, classifier f, ϵ, epsilon per iteration α, number of iterations n, class of $x = c$

Output: Adversarial sample adv_x

adv_x=x
adv_x =clip(adv_x,0.,1.)
for $i < n$ **do**
    // FGSM
    $grad = \nabla f(adv_x)_c$
    $grad = sign(grad)$
    $adv_x = adv_x + grad * \alpha$
    adv_x =clip(adv_x, 0., 1.)

    // perturbation projection
    $eta = adv_x - x$
    eta =clip(adv_x, $-\epsilon$, $+\epsilon$)
    $adv_x = x + eta$
    adv_x =clip(adv_x, 0., 1.)
return adv_x

In the targeted version we calculate the gradients with respect to the target class and instead of adding the gradients to the image to increase the loss, we subtract to decrease the loss of our target class.

This attack can also be applied using other norms such as L_1 and L_2. For this purpose we must change the expressions used to calculate the gradients as described in Section 2.1.3.

2.1.8 *One Pixel Attack*

The One Pixel Attack (Su et al. (2019)) is an iterative method that perturbs a single pixel (or a small set of pixels) per iteration, in Figure 8 we can see some adversarial samples from CIFAR-10. Although the paper is entitled One Pixel Attack, it's possible to perturb more than one pixel. Namely, the authors also tested with 3 and 5 pixels perturbations. As

Figure 8.: Few pixels attacks with one pixel perturbation that fooled three types of models trained on CIFAR-10 dataset. Source: Su et al. (2019)

opposed to the other methods presented so far, it is not gradient based. Instead the method relies in a Differential Evolution (DE) algorithm to determine which pixels to alter.

This method only requires the probability labels, the inner information of the network such as the model structure or the gradients aren't needed. This makes this attack almost a black-box attack on the model itself. As this attack doesn't use the gradients it can attack more types of models, including non-differentiable ones.

Let f be a neural network image classifier, $X = (x_1, ..., x_n)$ the vector representing an input image of n-dimensions, and t the correctly classified class. The probability that X belongs to class t is $f_t(X)$. The vector $e(X) = (e_1, ..., e_n)$ is an additive adversarial perturbation applied to X, the target class being adv. The limitation of the maximum modification of the original input d is measured by the length of vector $e(X)$ (zero norm). In the case of targeted attacks the goal is to find a solution $e(X)^*$ as seen in Equation 20.

$$\begin{aligned} \underset{e(X)^*}{\text{maximize}} \quad & f_{adv}(X + e(X)) \\ \text{subject to} \quad & \|e(X)\|_0 \leq d \end{aligned} \tag{20}$$

This attack involve finding two values: which pixels need to be perturbed and the corresponding strength of the modification.

DE is a method that optimizes a solution by iteratively trying to improve a candidate solution with regard to a given measure of quality. According to the authors, this algorithm is expected to efficiently find higher quality solutions than gradient-based solutions or other kinds of evolutionary algorithms because the method of selecting the candidate population keeps great diversity among the search space thus have higher probability of finding global

minima (Su et al. (2019)). This work uses a simplified version of the DE algorithm, that does not includes the crossover step, only mutations.

One candidate solution contains a fixed number of perturbations and each perturbation is a tuple holding five elements: x-y coordinates and RGB value of the perturbation. One perturbation modifies one pixel. In Su et al. (2019) 400 candidates were used as the initial number of candidate solutions (initial population) and for each iteration another 400 solutions (children) were generated using the DE formula in Equation 21.

$$\begin{gathered} x_i(g+1) = x_{r1}(g) + \alpha \times (x_{r2}(g) - x_{r3}(g)), \\ r1 \neq r2 \neq r3, \end{gathered} \tag{21}$$

Where g is the current index of generation, x_i is an element of the candidate solution, x_{r1}, x_{r2}, x_{r3} are random candidate solutions picked from the population, the authors set α to be 0.5.

Once generated each child $x_i(g+1)$ is compared to the $x_i(g)$ and the winner is kept on the population. The fitness function is the probabilistic label of the target class for target attack or the label of true class for the non-targeted attack.

The maximum number of iterations is set to 100 and the early stop criterion will be triggered when the probability label of target class exceeds 90% for target attacks, in case of the non-targeted attack when the true class is lower than 5%.

The initial population is initialized by using uniform distributions $U(1, 32)$ for CIFAR-10 images, for generating x-y coordinate (CIFAR-10 image size is 32X32), and Gaussian distributions $N(\mu = 128, \sigma = 127)$ for the RGB values.

2.2 Defense methods against adversarial attacks

Due to the success of adversarial attacks, soon researchers started developing methods to create more robust models, i.e. models that can receive adversarial images and still correctly classify them. In this section two widely different approaches are explored: Adversarial Training (Section 2.2.1) and Defensive Distillation (Section 2.2.2).

There are other defenses methods, an overview can be found at Chakraborty et al. (2018).

2.2.1 *Adversarial Training*

The main idea behind this approach, initially proposed in Szegedy et al. (2014), is to add adversarial examples to the training set, continually generating new adversarial examples at every step of training. The goal is to train the model to assign the same label to the adversarial example as to the original example and thus increasing the robustness of the model against future adversarial samples.

This method can be applied using the modified loss function defined in Equation 22 to train the model. This loss function allows independent control of the number of adversarial samples used and the relative weight of the adversarial samples in each batch.

$$Loss = \frac{1}{(m-k)+\lambda k}\left(\sum_{i \in Clean} L(X^N|y^N) + \lambda \sum_{i \in Adv} L(X^i_{adv}|y^i)\right) \tag{22}$$

where $L(X|y)$ is a loss on a single example X with true class y, m is the total number of training examples in the training batch including the adversarial samples, k is the number of adversarial examples in the minibatch and λ is a parameter which controls the relative weight of adversarial examples in the loss.

The training algorithm according to Kurakin et al. (2017a) can be seen in Algorithm 4:

Algorithm 4: Adversarial training algorithm

Input: network **N**, training set **X**, size of the training minibatch is **m**, number of adversarial images in the minibatch is **k**

1 Randomly initialize network N
2 **while** *training not converged* **do**
3 Read minibatch $B = \{X^1, ..., X^m\}$ from training set X
4 Generate k adversarial examples $\{X^1_{adv}, ..., X^k_{adv}\}$ from corresponding training set samples $\{X^1, ..., X^k\}$ using current state of the network N
5 Make new minibatch $B' = \{X^1_{adv}, ..., X^k_{adv}, X^{k+1}, ..., X^m\}$
6 Do one training step of network N using minibatch B'

In Kurakin et al. (2017a) it is recommended for the examples to be grouped into batches containing both normal and adversarial examples before taking each training step. Note that the method used to generate adversarial samples used by the authors was the FGSM Iterative least-likely class method. They found that using a fixed value for ϵ in the FGSM for the training made the networks robust only to that specific value of ϵ. They recommend to choose the value of ϵ randomly for each training example instead. Using ϵ drawn from a truncated normal distribution defined in interval $[0, 16]$ with underlying normal distribution $N(\mu = 0, \sigma = 8)$ the authors achieved the best results for imageNet dataset using Inception v3 model.

2.2.2 *Defensive Distillation*

Distillation Hinton et al. (2015) was initially proposed as an approach to reduce a large model (the teacher) down to a smaller distilled model. At a high level, distillation works by first training the teacher model on the training set in a standard manner. Then we use the

teacher to label each instance in the training set with soft labels(the output vector from the teacher network). For example the hard label for an image of a hand-written digit 7 will say it is classified as a seven, the soft labels may say it has 80% chance of being a seven and 20% of being a one. Afterwards, we train the distilled model on the soft labels from the teacher rather than the hard labels from the original training set. Distillation can potentially increase accuracy for the distilled model on the test set.

Defensive distillation Papernot et al. (2016) uses distillation in order to increase the robustness of a neural network, but with two significant differences. First, both the teacher model and the distilled model are identical in size, hence, defensive distillation doesn't result in smaller models. Second, and more importantly, defensive distillation uses a large distillation temperature to force the distilled model to become more confident in it's predictions. Recall that, the $softmax$ function is the last layer of a neural network. Defensive distillation modifies the $softmax$ function to also include a temperature constant T as seen in Equation 23.

$$softmax(x, T)_i = \frac{e^{\frac{x_i}{T}}}{\sum_j e^{\frac{x_j}{T}}} \tag{23}$$

Note that $softmax(x, T) = softmax(x/T, 1)$. Intuitively, increasing the temperature causes a "softer" maximum, and decreasing it causes a "harder" maximum. As the limit of the temperature goes to 0, $softmax$ approaches max, as the limit goes to infinity, $softmax(x)$ approaches a uniform distribution. Defensive distillation proceeds in four steps:

1. Train a network, the teacher network, by setting the temperature of the $softmax$ to T during the training phase. In Papernot et al. (2016) results are present with T varying from 1 to 100, the highest robustness being achieved with $T = 100$.

2. Compute soft labels by applying the teacher network to each instance in the training set, again evaluating the $softmax$ at temperature T.

3. Train the distilled network (a network with the same shape as the teacher network) on the soft labels using $softmax$ at temperature T.

4. Finally, when running the distilled network at test time (to classify new inputs), use temperature 1.

In the example mentioned above, a hard label for an image of a hand-written digit 7 will say it is classified as a seven, the soft labels may say it has 80% chance of being a seven and 20% of being a one. Training a network with this explicit relative information about classes prevents models from fitting too tightly to the data, and contributes to a better generalization around training points.

Using high temperature systematically reduces the model sensitivity to small variations of it's inputs when defensive distillation is performed at training time. At test time, the temperature is set to $T = 1$ in order to make predictions on unseen inputs. The intuition is that this doesn't affect the model's sensitivity as weights learned during training will not be changed by this change in temperature, and decreasing the temperature only makes the class probability vector more discrete, without changing the relative ordering of classes. In a way, the smaller sensitivity imposed by using a high temperature is encoded in the weights during training and is thus still observed at test time.

3

ATTACK EVALUATION

This chapter presents an evaluation on the transferability of an attack generated by a particular model when tested on a set of different models.

By transferability it is meant if an adversarial sample crafted on a model is also misclassified on other models. When considering targeted attacks we only consider an attack as successful if the models classify the samples as the target class.

For each attack method, a model is used to generate a set of adversarial samples that will be used to evaluate the transferability to the other models.

The chapter starts by describing the experimental setup, including the training procedures and attacks performed. The datasets used for these tests are introduced in Section 3.1, followed by the architectures and model training procedure in Section 3.2. The procedure to build the adversarial datasets is presented in Section 3.3.

Section 3.4 and Section 3.5 present the results considering different attack methods for each dataset.

3.1 DATASETS

For this work two datasets were considered: CIFAR-10 and GTSRB. The first is a widely used non-trivial dataset, while the second is relevant in real world applications such as autonomous driving. They differ significantly in their intra-class and inter-class variability, with GTSRB having a smaller variability in both items.

3.1.1 *CIFAR-10*

CIFAR-10 (Krizhevsky (2009)) is a labeled subset of the Tiny Images dataset. The Tiny Images dataset had 80 million 32×32 images and was originally created in 2006 in MIT, however, it was withdrawn in June 2020 (Torralba et al. (2020)). This dataset was chosen due to the fact that it is widely known and reported.

The CIFAR-10 dataset consists in 60000 32×32 colour images in 10 different classes, with 6000 images per class. These were labelled by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The test set has 10000 images. From the remaining 50000 images a split for training/validation was performed at 80%/20%.

Images in this dataset consists of natural shapes such as animals and some more geometrical ones like vehicles. Some samples can be seen in Figure 9.

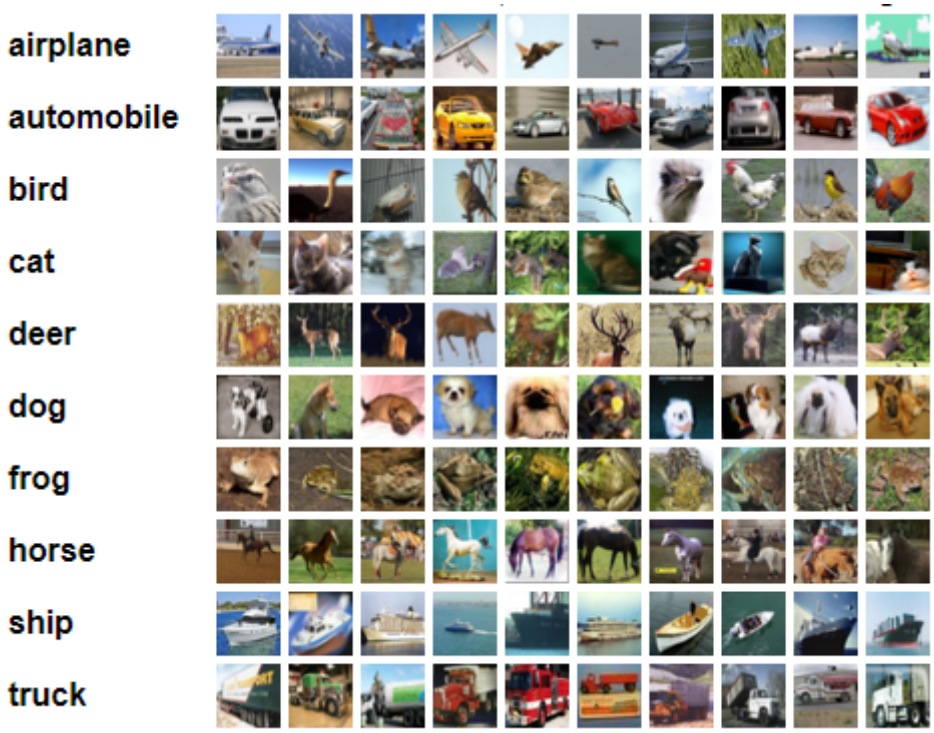


Figure 9.: CIFAR-10 samples from https://www.cs.toronto.edu/ kriz/cifar.html

3.1.2 *German Traffic Sign Recognition Benchmark*

GTSRB (GTSRB (2010)) is provided by the Real-Time Computer Vision group from the Institut für Neuroinformatik, Ruhr-Universität Bochum. It was initially made available for a competition at IJCNN 2011. This dataset was selected due to its relevance for autonomous driving.

This is an image classification dataset based in traffic sign images, originally divided in a training set with 39209 images, and a test set with 12630 images, totalling 51839 samples divided by 43 classes. As opposed to CIFAR-10 this is not a balanced set, with the number of images per class ranging from 210 to 2250 in the training set. Each image is in RGB with dimension ranging from 25×25 to 232×266 pixels. This dataset is composed of the sequence of frames taken from 1 second videos at 30 fps. This implies that the validation set

must be built using full sequences, instead of randomly picking images from the original training set. The validation set was built with approximately 20% of the sequences for each class.

Some samples can be seen in Figure 10.

Figure 10.: GTSRB samples from http://benchmark.ini.rub.de

3.2 MODELS AND TRAINING PROCEDURE

This project requires several architectures to be used to evaluate the transferability of adversarial attacks under different conditions. Well known architectures were selected, such as VGG16, ResNet (with three variations), together with a shallower conventional CNN architecture that we called ConvNet. For each model, 5 training runs were performed on each dataset.

All models use pre-processing to feed the network subtracting from the image the training set mean for each channel. Training was performed with data augmentation with Keras ImageDataGenerator. Specific details on the training procedure for each dataset are provided in the respective sections.

3.2.1 *VGG16*

VGG16 is a deep conventional CNN with 13 convolutional layers and 3 fully connected layers. This architecture won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) competition in 2014. Our implementation of this architecture is based on Geifman (2018).

3.2.2 *ResNets*

ResNet architecture was made famous when it won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) competition in 2015. This architecture introduces the concept of short cut connections allowing the definition of models with a large number of layers without the vanishing gradients problem.

All ResNets used in this work are based on a narrow version described in He et al. (2015), following the formula $6n + 2$ for the number of layers. We chose three architectures varying the number of layers: ResNet-20, 20 layers with $n = 3$, ResNet-50, 50 layers with $n = 8$, and ResNet-110, 110 layers with $n = 18$. Our implementation is based on Chollet.

3.2.3 *ConvNet*

This is a shallower architecture built for this project that consists in only 3 convolutional layers followed by 2 dense layers. a diagram of this model and their parameters can be seen in Figure 11.

3.3 BUILDING THE ADVERSARIAL SAMPLE SET

An adversarial sample is an image that has been perturbed so that it causes a misclassification on a particular model. The level of perturbation present in an adversarial sample deeply influences the outcome, and some metric is required to evaluate this perturbation.

Furthermore, different methods have different parameters, therefore, some measure of image perturbation is required in order to fairly compare the methods.

We have adopted SSIM Wang et al. (2004), a well known and studied metric used to measure image degradation. The implementation used is provided by scikit-image library, in which the SSIM index for a pair of images is a value between -1.0 and 1.0, where 1.0 is only attainable with identical images. A value of -1.0 indicates that there are no structural similarities between the two images. To generate adversarial samples we considered two indices of perturbation: 0.95 and 0.80. The first index is achievable for pairs of images where there is almost no perceptual difference between them. The second index already implies some perceptual degradation, yet the items in the perturbed image are still easily recognisable. In this work, the SSIM value is computed between the original image and the perturbed, adversarial sample.

The goal is therefore to create two sets of adversarial samples, each having an average SSIM value close to those specified above. This is done for all the methods discussed in Chapter 2, namely:

- FGSM (Section 2.1.3)

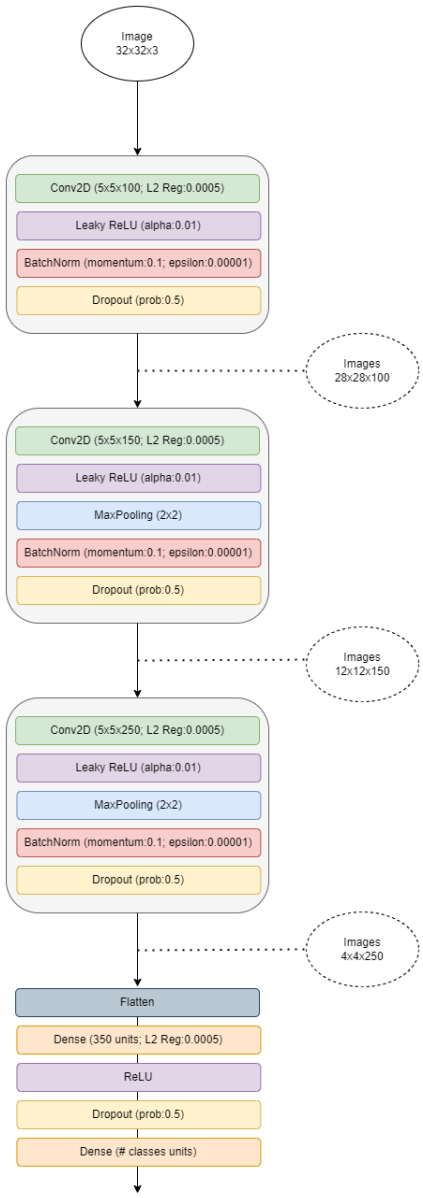


Figure 11.: ConvNet architecture diagram

- Deepfool (Section 2.1.4)
- JSMA (Section 2.1.5)
- Carlini (Section 2.1.6)
- PGD (Section 2.1.7)
- One pixel/ Few Pixels (Section 2.1.8)

The adversarial sample sets are defined by the pair (SSIM, method). For each pair we had to perform a parameter search for the particular method to achieve a dataset with the specified average SSIM value.

A trained ResNet-50 was used to craft all sets of adversarial samples. As mentioned before, five models were trained for each architecture. We selected the ResNet-50 with median accuracy to be the adversarial sample generator. These datasets will then be used to evaluate the transferability ratio to other models. The selection of ResNet-50 allows to test in similar architectures both shallower (ResNet-20) and deeper (ResNet-110), as well as significantly different architectures (VGG16 and ConvNet).

When attacking the other ResNet-50 models, we are doing an almost white box attack: only the weights differ from the generator model. Considering the other trained models, the attack is closer to a black box attack since only the training set is known.

To build the adversarial sample sets, N images from the respective test set are selected such that all trained models correctly classify these images. The number of selected images and the selection procedure varies by dataset and is detailed in Section 3.4 and Section 3.5.

In this project, all models expect a preprocessed input by subtracting the training dataset mean from the image pixels. This shifts the input values to be centered around zero, meaning that the model input values are in the range $[-1, 1]$. When crafting an adversarial samples these are the max boundaries allowed. When the misclassification is attained, the train mean is added back to the adversarial sample, and the values clipped to the range $[0, 1]$ to generate a valid RGB image.

This remapping from different domains can cause an adversarial sample to lose its adversarial property, i.e, although the adversarial sample with values in $[-1, 1]$ causes a misclassification, the RGB image obtained after summing the training set mean and clipping can still be correctly classified, or in the case of a targeted attack, it may not classify the image in the desired class.

As mentioned before, for each adversarial sample configuration (SSIM, method), N adversarial samples per class are attempted. N is an upper bound on the final number of adversarial samples in each set. A set may have fewer samples due to two reasons: the method to generate the adversarial sample may fail, i.e. the method fails to converge; or the preprocessing issue mentioned above caused the sample to lose its adversarial effect.

In practice, to generate the adversarial sample set we are carrying white box attack, and keeping the samples that result in a successful attack.

The main goals of this work is to assert the transferability of the studied adversarial attacks methods between models with different architectures, or shared architecture but different weight states.

3.4 CIFAR-10: ADVERSARIAL ATTACKS

This section details all the experiments performed on models trained with CIFAR-10. It starts by describing the specific procedure details for this dataset in Section 3.4.1. Section 3.4.2 to Section 3.4.7 detail the initial parameter search for each of the attacks involved, considering both SSIM values, where the attack is performed in a single model for each architecture. Finally, Section 3.4.8 presents the aggregate test results, using all models for each architecture.

3.4.1 *Procedure details and attack parameters*

Tests on the CIFAR-10 dataset use all the trained models on this dataset: VGG16, ConvNet, ResNet-20, ResNet-50, and ResNet-110. The architectures of these models is described in Section 3.2.

Table 1 shows the average accuracy obtained in the test set for each architecture.

	VGG16	**ConvNet**	**ResNet-20**	**ResNet-50**	**ResNet-110**
CIFAR-10	92.80 ± 0.17	86.89 ± 0.33	90.71 ± 0.09	91.52 ± 0.13	91.39 ± 0.25
# train params	14991946	2725360	273066	760266	1734666

Table 1.: Average accuracy and standard deviation on test set for all models trained on CIFAR-10 dataset and their corresponding number of trainable parameters.

Dynamic data augmentation was performed with settings as presented in Table 2.

	ImageDataGenerator
width shift range	0.1
height shift range	0.1
fill mode	constant
horizontal flip	True

Table 2.: CIFAR-10 data augmentation

All the attack methods described from Section 2.1.3 to Section 2.1.8 are used with this dataset, namely: FGSM, Deepfool, JSMA, Carlini, PGD, and One pixel/Few Pixels. The implementation for FGSM, Deepfool, JSMA (non-targeted), Carlini, and PGD are from the Adversarial-Robustness-Toolkit (ART) library by IBM Nicolae et al. (2018). For the non-targeted version of JSMA we used an implementation based on Youlixx (2020), and for Few Pixels attack we used the implementation of Song (2019) based on Su et al. (2019).

To build a dataset for each pair (SSIM, method), first it is required to find the parameters for each method that generate an average level of distortion close to the selected SSIM indices. This implies exploring the parameter space of each attack method.

As mentioned before, the adversarial samples are created from a ResNet-50 model. Initially, 500 samples correctly classified in all trained models are randomly selected from CIFAR-10 test set. Note that, as mentioned in Section 3.3, this number is an upper bound on the cardinality of the datasets generated.

Regarding the targeted version, on CIFAR-10 the attack uses as the target class the true class $+1(\mod 10)$, for example if the true class is 8 which contains horses the new target will be class 9, containing ships, if the true class is the last, in this case class 9 which contains truck, the new target will be 0, the airplane class.

Starting from Section 3.4.2 to Section 3.4.7 the parameter exploration for each method is presented. In addition, these sections also present an evaluation for the trained models for each architecture.

As mentioned before, to evaluate the best performing norm, initially we use a single model of each architecture (the chosen model was the one with median test accuracy across all 5 instances of the same architecture, in the case of the ResNet-50 we picked the second model with best test accuracy out of 4) and calculate the average transferability in these models on the adversarial samples. The norm that attains the highest average between the models will be chosen to report the final results. Section 3.4.8 presents a more comprehensive test with all models.

The remaining of this subsection presents the training details for each architecture.

VGG16

The VGG16 was trained using the same protocol used in Geifman (2018), the parameters can be seen in Table 3. This training protocol has two callbacks functions:

- ModelCheckpoint - saves current epoch model if the validation accuracy is better than the best saved model so far;
- LearningRateScheduler - changes the learning rate along the epochs.

The learning rate schedule in derived from the formula in Equation 24, where $\left\lfloor \frac{\cdot}{\cdot} \right\rfloor$ denotes the integer division. The graph of this schedule can be seen in Figure 12.

$$\text{new\_lr} = \text{learning\_rate} \cdot \left(0.5^{\left\lfloor \frac{\text{epoch}}{\text{lr\_drop}} \right\rfloor}\right) \tag{24}$$

ResNet based models

All the ResNet base models follow the same training protocol, which was based on Chollet. The training parameters can be seen in Table 4. This training protocol has the same callbacks functions as VGG-16 (Table 3.4.1) with a different equation for the LearningRateScheduler, as seen in Equation 25. A graph of this schedule can be seen in Figure 13.

	VGG16 train params
epochs	250
batch size	128
optimizer	SGD
learning rate	0.1
momentum	0.9
learning rate decay	0.000001
learning rate drop	20
nesterov	True

Table 3.: VGG16 model training parameters on CIFAR-10

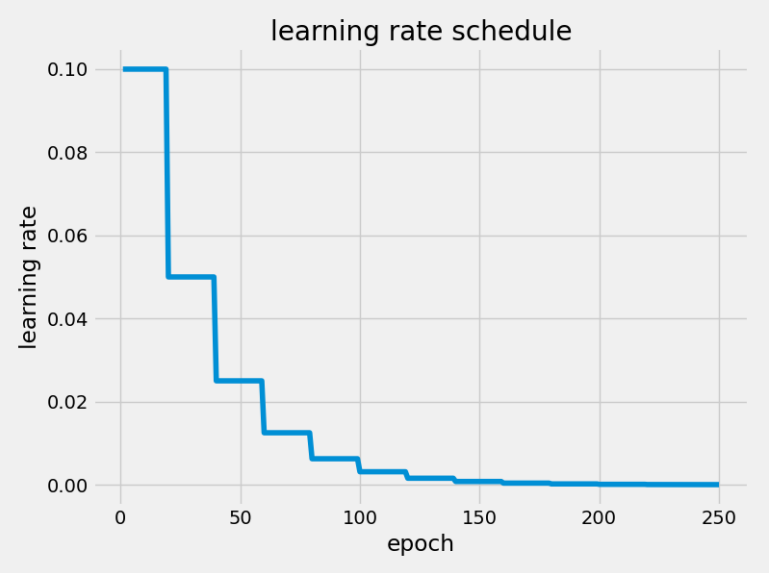


Figure 12.: VGG16 learning rate schedule for CIFAR-10

	ResNet train params
epochs	200
batch size	32
optimizer	Adam
learning rate	0.001

Table 4.: ResNet models training parameters on CIFAR-10

$$\text{new\_lr}(\text{epoch}) = \begin{cases} 0.0005 \cdot 0.001 & \text{if epoch} > 180 \\ 0.001 \cdot 0.001 & \text{if } 160 < \text{epoch} < 180 \\ 0.01 \cdot 0.001 & \text{if } 120 < \text{epoch} < 160 \\ 0.1 \cdot 0.001 & \text{if } 80 < \text{epoch} < 120 \\ 0.001 & \text{if epoch} < 80 \end{cases} \tag{25}$$

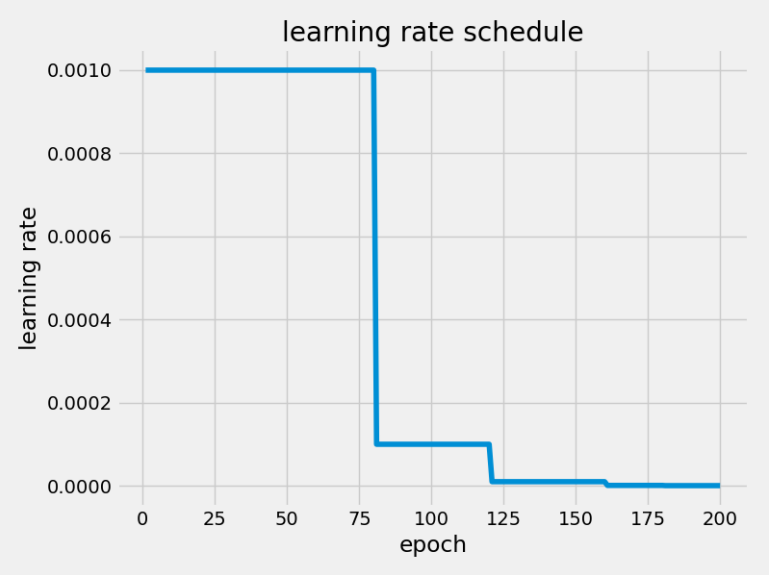


Figure 13.: ResNet learning rate schedule for CIFAR-10

ConvNet

The ConvNet model training parameters can be seen in Table 4, this training protocol has two callbacks: ModelCheckpoint, similar to Table 3.4.1, and ReduceLROnPlateau. The parameters for this latter callback can be seen in Table 5.

	ConvNet train params
epochs	200
batch size	128
optimizer	Adam
learning rate	0.001

Table 5.: ConvNet model training parameters on CIFAR-10

	ConvNet ReduceLROnPlateau
monitor	val loss
factor	0.2
patience	5
min lr	0.0001

Table 6.: ConvNet ReduceLROnPlateau callback parameters

3.4.2 *FGSM*

FGSM has a single parameter ϵ and supports both targeted and non-targeted attacks. Both versions were tested in three variants based on the norms: L_∞, L_1, and L_2.

For each norm, a different range of ϵ values had to be explored in order to find the values that create adversarial samples with the desired SSIM index when compared to the original image.

Non-targeted L_∞ norm

With the L_∞ norm, the ϵ explored range was $[0.01, 0.085]$, in 0.005 increments. Examples of an adversarial sample for some ϵ considered can be seen in Figure 14 .

Figure 15 shows the success attack rate on each model per ϵ, their corresponding average SSIM, and total percentage of successful adversarial samples on model generator after converting to valid images.

After some tuning the SSIM index 0.95 was found at $\epsilon = 0.025$ attaining an average success attack rate of 39.39%. For SSIM index 0.80 the value found was $\epsilon = 0.062$ attaining an average success attack rate of 73.11%.

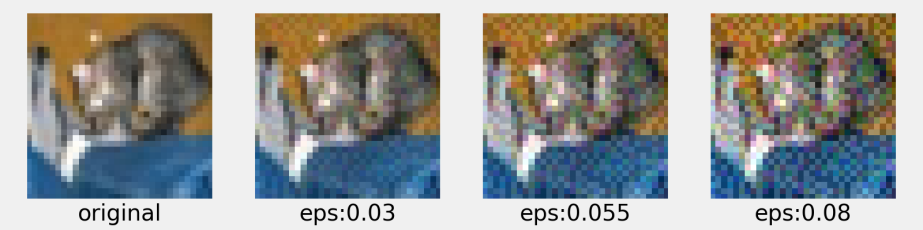


Figure 14.: Non-targeted FGSM L_∞ norm adversarial sample per ϵ

Non-targeted L_1 norm

With the L_1 norm, the range considered for ϵ was $[20, 170]$, in 10.0 increments. Examples of an adversarial sample for some ϵ values within this range can be seen in Figure 16.

Figure 17 shows the success attack rate on each model per ϵ, their corresponding average SSIM, and total percentage of successful adversarial samples on model generator after converting to valid images.

For SSIM index 0.95 the epsilon found was $\epsilon = 40$ attaining an average success attack rate of 39.35%. For SSIM index 0.80 the epsilon found was $\epsilon = 125$ attaining an average success attack rate of 66.01%.

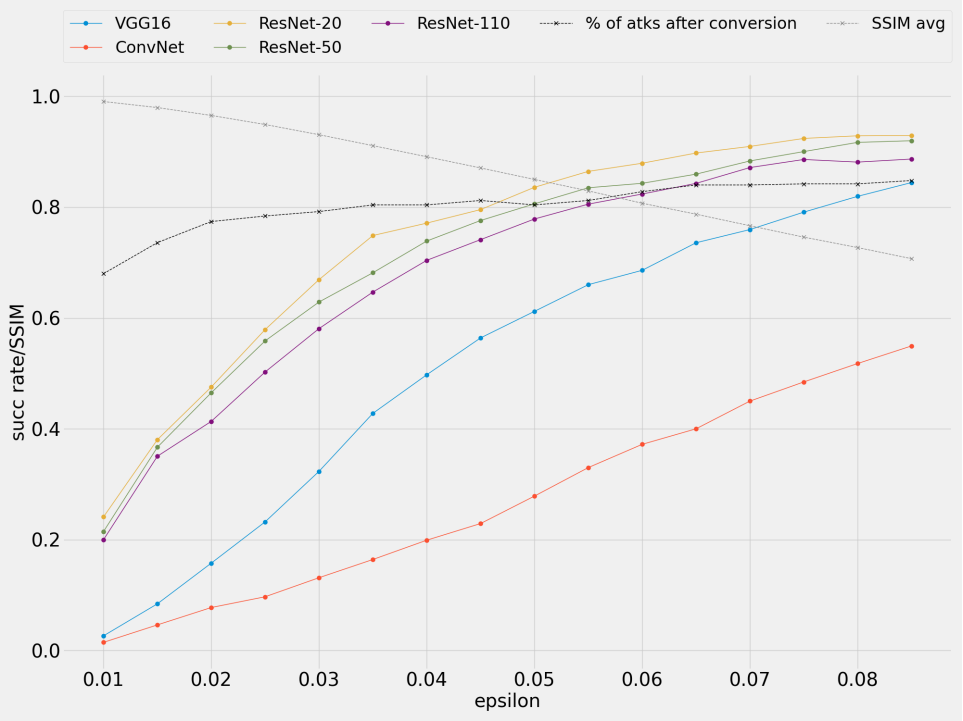


Figure 15.: Success attack rate of non-targeted FGSM L_∞ norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

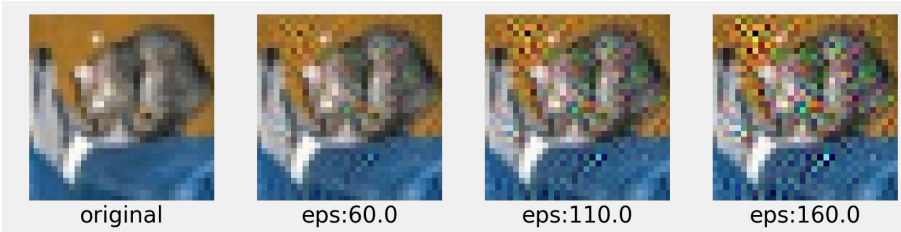


Figure 16.: Non-targeted FGSM L_1 norm adversarial sample per epsilon

Non-targeted L_2 norm

When using the L_2 norm, the ϵ explored range was in the interval $[1, 8.5]$, in 0.5 increments. Examples of an adversarial sample for ϵ values within this range can be seen in Figure 18 .

Figure 19 shows the success attack rate on each model per ϵ, their corresponding average SSIM and total percentage of successful adversarial samples on model generator after converting to valid images.

After some tuning, for SSIM index 0.95 we found $\epsilon = 1.2$ attaining an average success attack rate of 42.05%. For SSIM index 0.80 the value found was $\epsilon = 3.7$ attaining an average success attack rate of 68.09%.

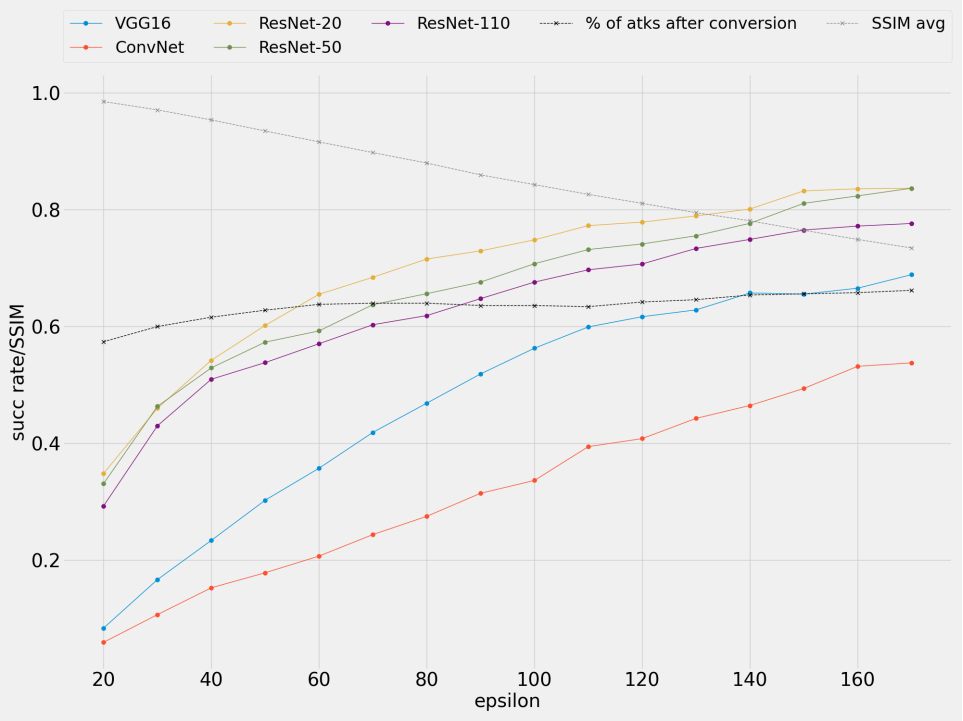


Figure 17.: Success attack rate of non-targeted FGSM L_1 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

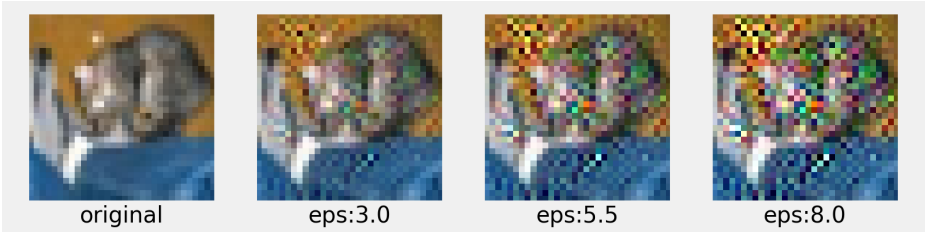


Figure 18.: Non-targeted FGSM L_2 norm adversarial sample per epsilon

Summary for non-targeted attacks

Several trends are noticeable when testing FGSM in all three norms:

- As expected, as ϵ increases the SSIM between the original image and its adversarial counterpart decreases;
- The percentage of successful adversarial attacks on the generator model, although increasing as ϵ increases, does not seem to suffer much influence with the varying ϵ, implying that attack method is capable of computing adversarial attacks even when ϵ is small;

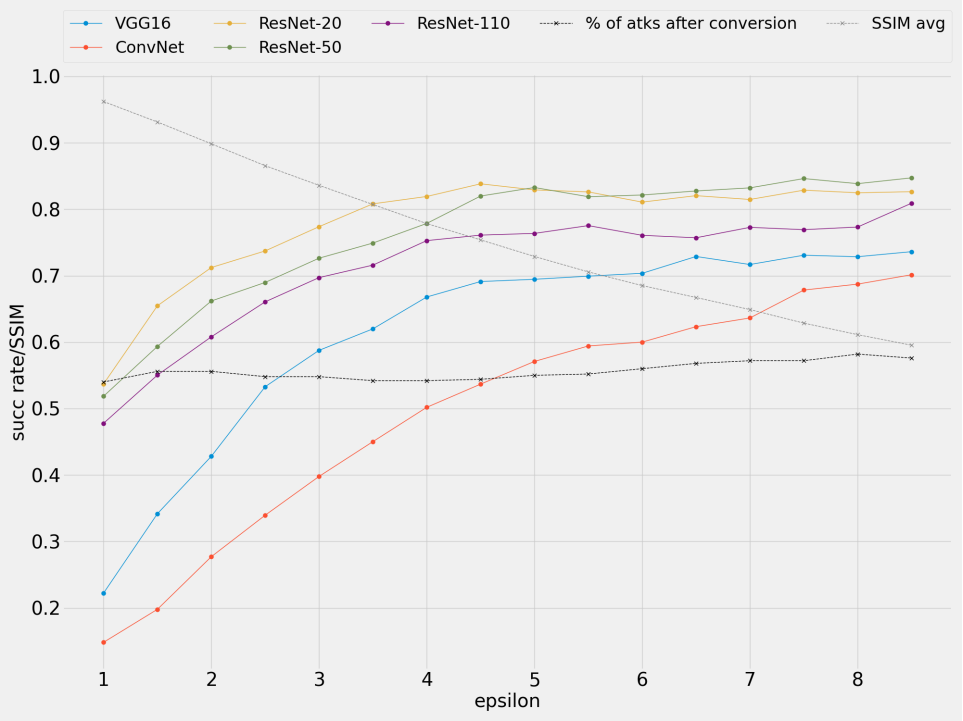


Figure 19.: Success attack rate of non-targeted FGSM L_2 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

- The family of ResNet models shows a higher transferability rates than the other models. This may be due to the architecture similarity;
- The ConvNet model is the most resilient to the adversarial attacks, showing the lowest transferability rate.

Table 7 reports the results of success attack rate per SSIM and epsilon for each norm. The norm that achieved the highest average transferability result for $SSIM = 0.80$ was L_∞ and for $SSIM = 0.95$ was L_2. These norms will be used in the attacks reported in Section 3.4.8.

FGSM	L_∞		L_1		L_2	
SSIM ($\pm$0.005)	0.95	0.80	0.95	0.80	0.95	0.80
epsilon	0.025	**0.062**	40.0	125.0	**1.2**	3.7
avg transf atk	39.39	**73.11**	39.35	66.01	**42.05**	68.09
succ atk on generator	78.40	**83.80**	61.80	65.40	**54.60**	54.60
succ atk aft conv	100.00	**99.76**	99.68	98.78	**100.00**	99.63
total atks aft conv	78.40	**83.60**	61.60	64.60	**54.60**	54.40

Table 7.: FGSM non-targeted average SSIM, epsilons, success attack rate, success attacks after conversion and total successful attacks for each norm

Targeted attack

A similar search for the ϵ values required to obtain SSIM indices was performed for each norm considering targeted attacks. Results are presented in Figure 20.

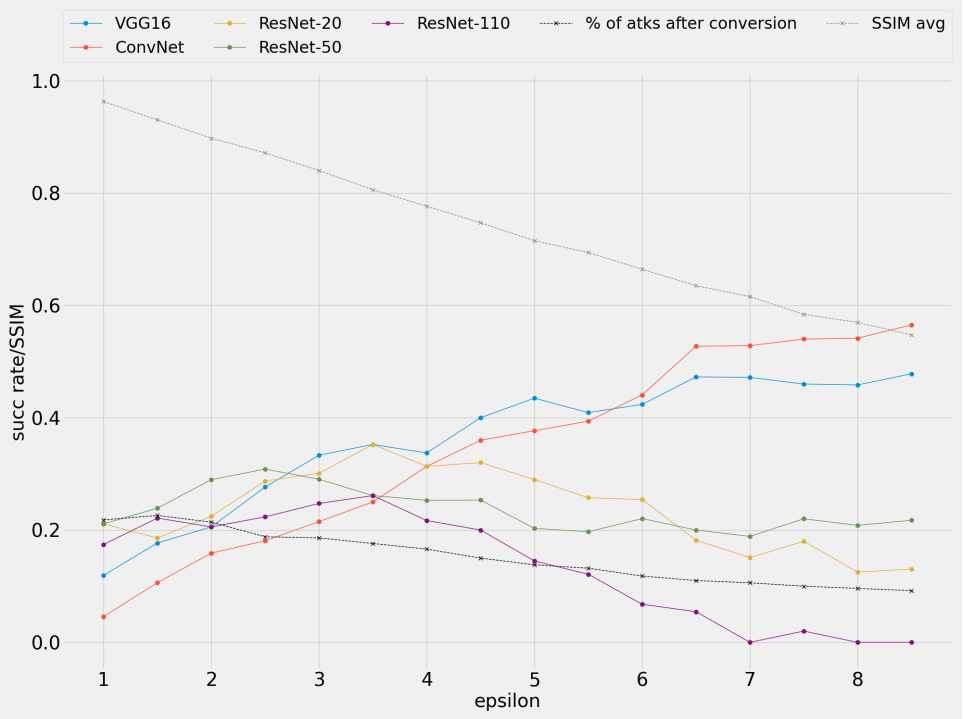


Figure 20.: Success attack rate of targeted FGSM L_2 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

Several trends noticeable for all norms on non-targeted attacks are not evident in the targeted versions, more concretely:

- The percentage of successful adversarial attacks on generator model seems to diminish with the varying ϵ. While a larger degradation tends to lead to more misclassifications, note that a targeted attack is only considered successful if the attacked models produce the required misclassification, i.e. produce a classification on the target class. A large degradation seems to provide a higher uncertainty on which class the attacked models will classify a sample. Hence, it is not a surprise to see architectures having its robustness increase after ϵ goes above certain values. This can be observed in the ResNet family;

- The ConvNet architecture is still the most resistant model for the considered SSIM values. However, its robustness decreases significantly as ϵ increases, ending up being the least robust architecture for larger values of ϵ.;

- A similar behaviour can be seen with VGG16, but in this case the decrease in relative robustness starts earlier.

In Figure 20 we can see this behaviour for targeted FGSM L_2 norm. This behaviour can also be observed in the other norms on the target setting. These graphs are available in Section A.1.1.

Table 8 shows the results of success targeted attack rate per SSIM and epsilon for each norm. As in the non-targeted attacks, the norm that achieved best average transferability for $SSIM = 0.80$ was L_∞ and for $SSIM = 0.95$ was L_2.

FGSM	L_∞		L_1		L_2	
SSIM ($\pm$0.005)	0.95	0.80	0.95	0.80	0.95	0.80
epsilon	0.026	**0.070**	42.0	130.0	**1.2**	3.6
avg transf atk	13.39	**31.11**	16.00	30.11	**16.21**	30.00
succ atk on generator	25.40	**14.60**	23.40	19.00	**23.60**	18.40
succ atk aft conv	100.00	**98.63**	98.29	91.58	**98.31**	93.48
total atks aft conv	25.40	**14.40**	23.00	17.40	**23.20**	17.20

Table 8.: targeted FGSM average, SSIM, epsilons, success attack rate, success attacks after conversion and total successful attacks for each norm

3.4.3 *Deepfool*

This attack, as opposed to FGSM (Section 2.1.3), has no norms. Furthermore, this attack is purely a non-targeted attack. Hence, we only searched the parameters that gave us average SSIM of 0.80 and 0.95. Figure 21 shows an adversarial sample for a range of ϵ values.

Figure 21.: Deepfool image samples per epsilon

Figure 22 shows the success attack rate on each model per ϵ, their corresponding average SSIM and total percentage of successful adversarial samples on model generator after converting to valid images.

For SSIM index 0.95 the epsilon found was $\epsilon = 5$ attaining an average success attack rate of 30.75%. For SSIM index 0.80 the epsilon found was $\epsilon = 18$ attaining an average success attack rate of 67.98%

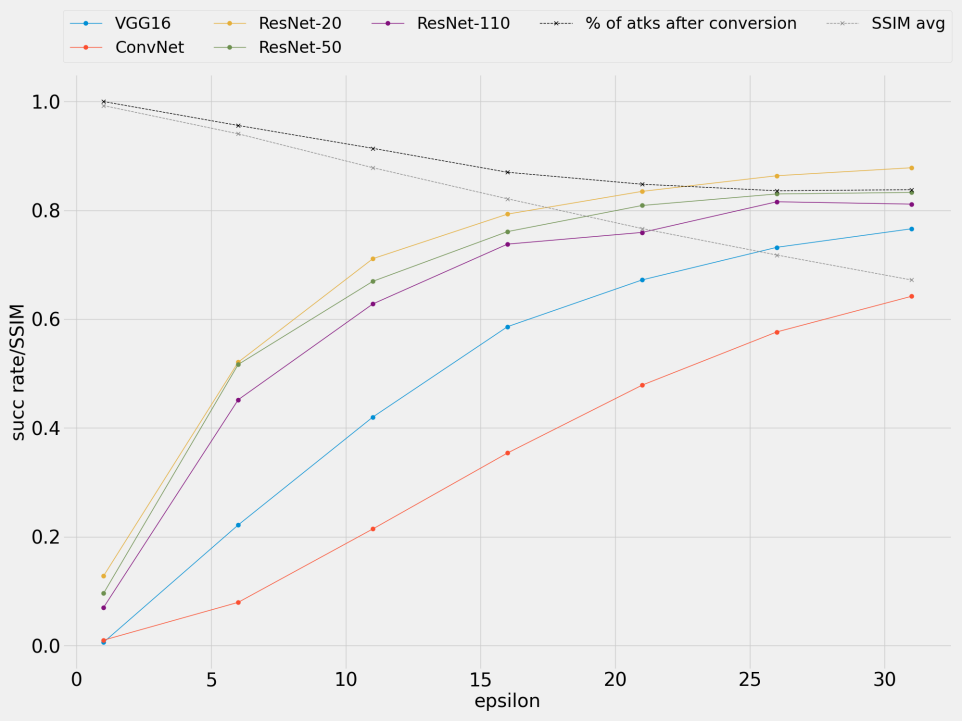


Figure 22.: Deepfool success attack rate on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

An interesting finding, which differs from what was observed with FGSM, is that the percentage of successful adversarial attacks on generator model seems to decrease with the varying ϵ. This behaviour seems to indicate that the algorithm increasingly overshoots the boundaries as ϵ gets larger. Therefore choosing ϵ is important not only due to the amount of degradation it creates in the original image, but also due to its impact on the success rate of adversarial generation.

The robustness relative ordering for the architectures is similar to the results obtained in non-targeted FGSM.

Table 9 presents the information obtained for both SSIM values.

	Deepfool	
SSIM ($\pm$0.005)	0.95	0.80
epsilon	**5.0**	**18.0**
avg transf atk	**30.75**	**67.98**
succ atk on generator	**96.20**	**87.40**
succ atk aft conv	**99.79**	**98.63**
total atks aft conv	**96.00**	**86.20**

Table 9.: Deepfool average SSIM, epsilons, success attack rate, success attacks after conversion and total successful attacks

3.4.4 *JSMA*

This attack does not have norms defined, and has two parameters:

- θ: the amount of perturbation added to the pixel;
- γ: the maximum amount of pixels as number or percentage in the image allowed to be perturbed.

Non-Targeted

For the non-targeted JSMA (NT-JSMA) algorithm we used an implementation based on Youlixx (2020). In Figures 23 and 24 we can see adversarial samples for different thetas and gammas.

Figure 23.: NT-JSMA samples for different gammas and constant theta=0.60

Figure 24.: NT-JSMA samples for different thetas and constant gamma=20%

This attack is very expensive computational-wise so instead of generating 500 samples for each parameter we only used 250.

For the parameter search we varied θ from 0.20 to 1.0 in 0.2 steps and varied γ from 20% to 80%. For this particular dataset and ranges, the γ value did not have any influence on the adversarial generation, i.e., the adversarials are the same regardless of γ.Hence, we fixed $\gamma = 20\%$. This behaviour can be seen in Figure 23.

Figure 25 shows the success attack rate on each model per θ while setting $\gamma = 20\%$, their corresponding SSIM and total percentage of successful adversarial samples on model generator after converting to valid images.

The behaviour observed in Figure 25 shows some relevant features:

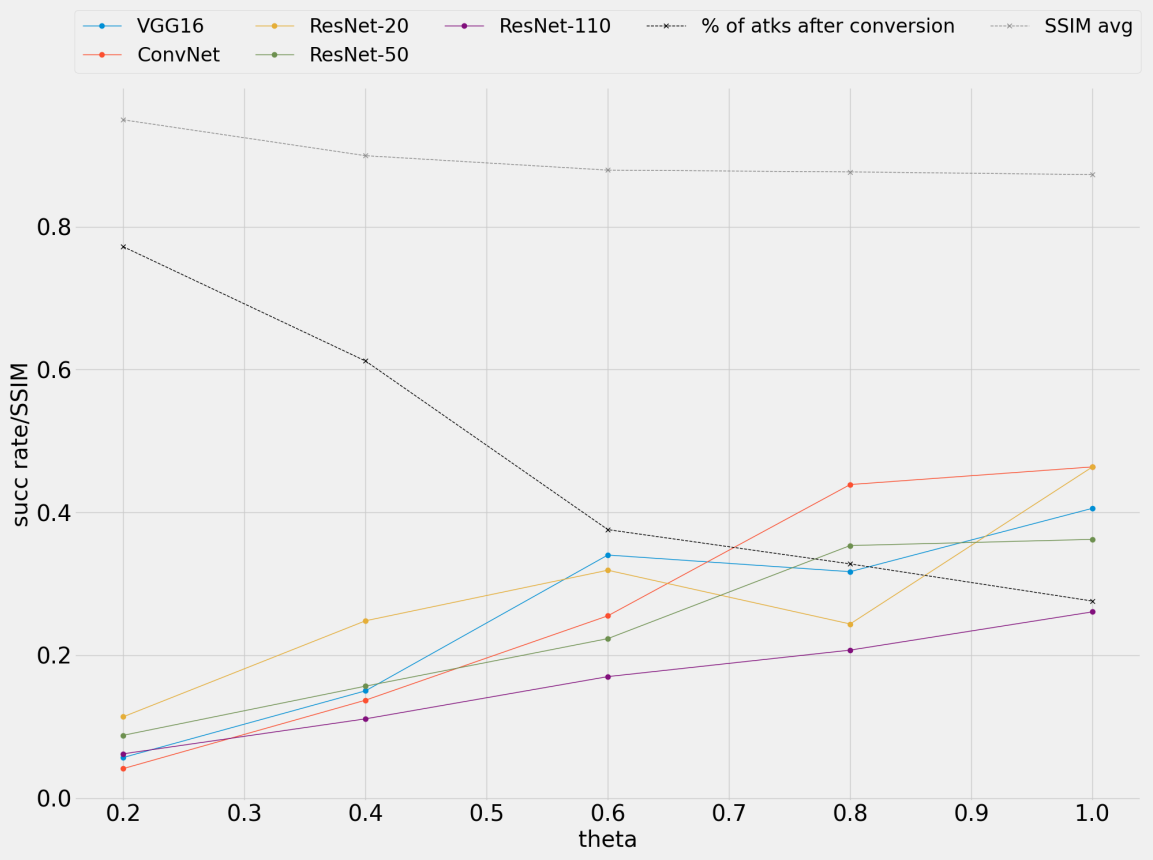


Figure 25.: NT-JSMA success attack rate on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per θ and $\gamma = 20\%$

- Increasing θ appears to have little influence in decreasing the SSIM between the original image and its adversarial counterpart. Together with what was observed for γ this corroborates the authors statement that JSMA crafts adversarials perturbing only a small number of pixels, see Section 2.1.5;

- The percentage of successful adversarial attacks after conversion on the generator decreases significantly as θ increases. As can be seen in Table 10, for the selected SSIM values, all attacks before conversion have 100% success rate. This implies that JSMA tends to create adversarial samples outside the boundaries of valid images, that once converted loose their adversarial status. Note that there is no contradiction with the previous point. As seen in Figure 24 the perturbation is indeed small considering the number of pixels that are affected, however, this perturbation is very significant for the modified pixels;

- as the perturbation increases, the shallower networks have higher transferability rates: ConvNet, ResNet-20, and VGG16 show the highest transferability rate when $\theta = 1.0$. ResNet-110 is the most robust architecture in this test.

As mentioned before, prior to the conversion to a valid range, all attacks had 100% success rate. Hence, we evaluated the transferability on the unconverted adversarial samples. Results are presented in Figure 26. In this graph we can see that the deepest model, ResNet-110, remains the most robust. On the other hand, ConvNet is now the second most robust architecture when $\theta = 1.0$. This shows that converting to a valid image affects different architectures in different scales.

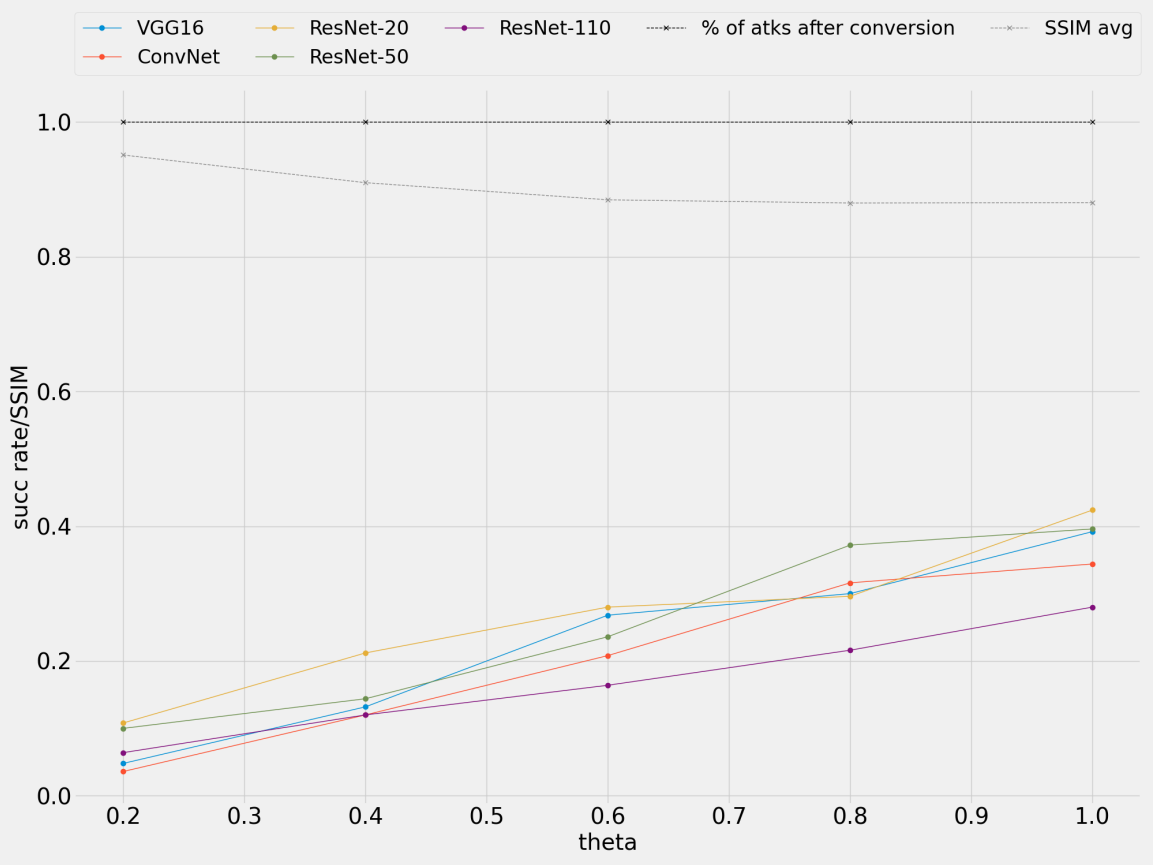


Figure 26.: NT-JSMA success attack rate on each model and their corresponding SSIM and total successful adversarial attacks on model generator without converting to a valid image per θ and $\gamma = 20\%$

For SSIM index 0.95 the parameters found were: $\gamma = 20\%$ and $\theta = 0.20$ with an average transferability success attack rate of 7.25%; we did not achieve a SSIM$= 0.80$ with this algorithm with neither combination of parameters searched because the threshold needed to misclassify all the 250 samples tested were small enough that their average perturbation did not decreased below 0.8729. Although the targeted SSIM was not achieved, we used the

combination $\gamma = 0.2$ and $\theta = 1.0$ as the highest perturbation possible in these conditions. The parameters chosen for the small and large perturbation can be seen in Table 10.

	NT- JSMA	
SSIM(±0.005)	0.95	0.8729
gamma	**0.2**	**0.2**
theta	**0.2**	**1.0**
avg transf atk	**7.25**	**39.13**
succ atk on generator	**100.00**	**100.00**
succ atk aft conv	**77.20**	**27.60**
total atks aft conv	**77.20**	**27.60**

Table 10.: NT-JSMA average SSIM, theta, gamma, success attack rate, success attacks after conversion and total successful attacks

Targeted

Figure 27 shows the success attack rate on each model for each θ while setting $\gamma = 20\%$, their corresponding SSIM and total percentage of successful adversarial samples on model generator after converting to valid images.

As expected there is a significant increase in robustness when considering targeted attacks.

For the targeted attack the parameters chosen can be seen in Table 11.

	Target JSMA	
SSIM(±0.005)	0.95	0.8729
gamma	**0.2**	**0.2**
theta	**0.13**	**0.78**
avg transf atk	**0.98**	**9.00**
succ atk on generator	**100.00**	**100.00**
succ atk aft conv	**73.20**	**24.00**
total atks aft conv	**73.20**	**24.00**

Table 11.: Targeted JSMA average SSIM, theta, gamma, success attack rate, success attacks after conversion and total successful attacks

3.4.5 *Carlini*

Carlini exist in both targeted and non-targeted versions. As these attacks are expensive computation-wise, and both L_∞ and L_0 are derived from the L_2 attack, which was the main attack devised by the authors, we only perform tests using the L_2 norm.

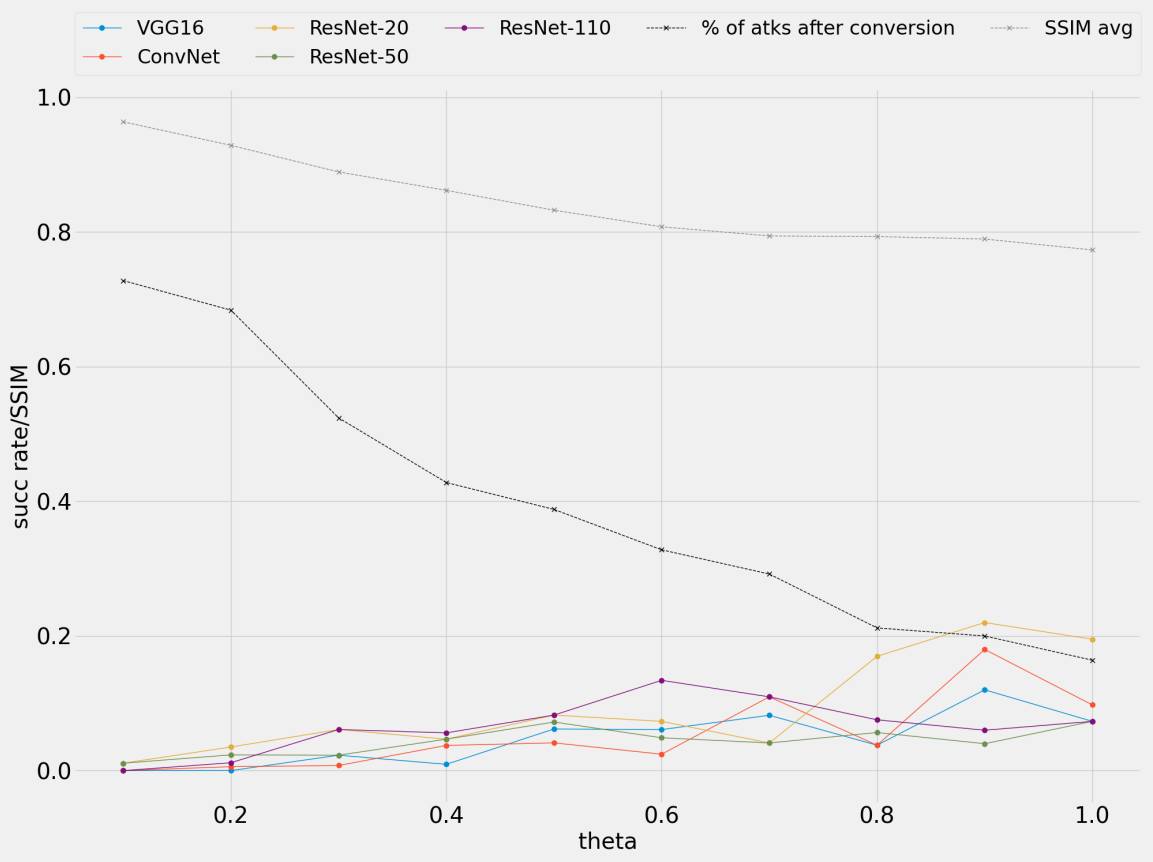


Figure 27.: Targeted JSMA success attack rate on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per θ with $\gamma = 20\%$

In these tests we searched over k, used 1 for the starting value of c, and the other variables used the default values (learning rate $= 0.01$, binary search steps$= 10$, max iterations $= 10$, max halving $= 5$ and max doubling $= 5$).

Non-Targeted

In Figure 28 we can see adversarial samples for different k values.

Figure 28.: Non-targeted Carlini L_2 adversarial sample for some confidence values k

Figure 29 shows the success attack rate on each model per confidence k, their corresponding average SSIM and total percentage of successful adversarial samples on model generator after converting to valid images.

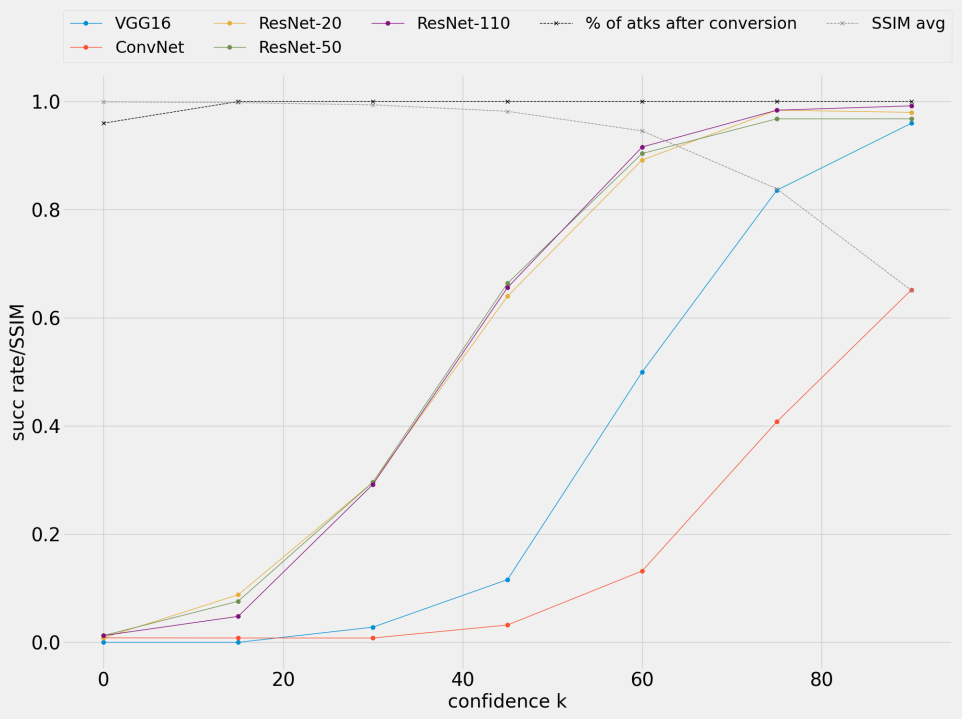


Figure 29.: Non-targeted Carlini L_2 norm success attack rate of on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per k

For SSIM=0.95 the k found was 57 attaining an average transferability attack success rate of 63.80% and for SSIM=0.80 the k is 78 achieving an average transferability attack success rate of 87.00%, see Table 12.

	Carlini L_2	
SSIM($\pm$0.005)	0.95	0.80
k	**57**	**78**
c	**1**	**1**
avg transf atk	**63.80**	**87.00**
succ atk on generator	**100.00**	**100.00**
succ atk aft conv	**100.00**	**100.00**
total atks aft conv	**100.00**	**100.00**

Table 12.: Non-targeted Carlini L_2 norm average SSIM, k, c, success attack rate, success attacks after conversion and total successful attacks

The results in Figure 29 show a similar behaviour for all the ResNet model group. The ConvNet is by a relevant margin the most robust model regarding this attack, with VGG16 having a robustness in between the other models.

Targeted

For the targeted attack we followed the same steps of non-targeted attack, for this experiment the results are: for SSIM= 0.95 the k found was 30 attaining an average transferability targeted attack success rate of 17.80%, and for SSIM= 0.80 the k found was 49 achieving an average transferability attack success rate of 52.21%, see Table 13.

	Target Carlini L_2	
SSIM($\pm$0.005)	0.95	0.8729
k	**30**	**49**
c	**1**	**1**
avg transf atk	**17.80**	**52.21**
succ atk on generator	**100.00**	**95.00**
succ atk aft conv	**100.00**	**100.00**
total atks aft conv	**100.00**	**95.00**

Table 13.: Targeted Carlini L_2 norm average SSIM, k, c, success attack rate, success attacks after conversion and total successful attacks

Figure 91, in Section A.1.3, shows the success attack rate on each model per confidence k. The behaviour is similar to the non-targeted attack, although with a lower overall transferability rate as expected.

3.4.6 *PGD*

PGD has a single parameter ϵ. In this test we allowed this algorithm to run at most 100 iterations, and for each ϵ tested we defined the step size per iteration as $\frac{\epsilon}{100}$. All three norms (L_∞, L_1, L_2) have been tested in both targeted and non-targeted versions.

Non-targeted L∞ norm

In Figure 30 we can see adversarial samples per ϵ.

Figure 31 shows the success attack rate on each model per ϵ, their corresponding average SSIM and total percentage of successful adversarial samples on model generator after converting to valid images.

For SSIM index 0.95 the epsilon found was $\epsilon = 0.08$ attaining an average transferability success attack rate of 71.28%, and for SSIM index 0.80 the epsilon found was $\epsilon = 0.45$ attaining an average transferability success attack rate of 88.60%.

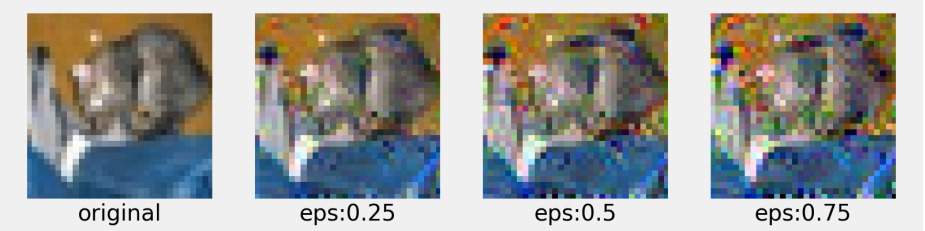


Figure 30.: Non-targeted PGD L_∞ norm adversarial sample per ϵ

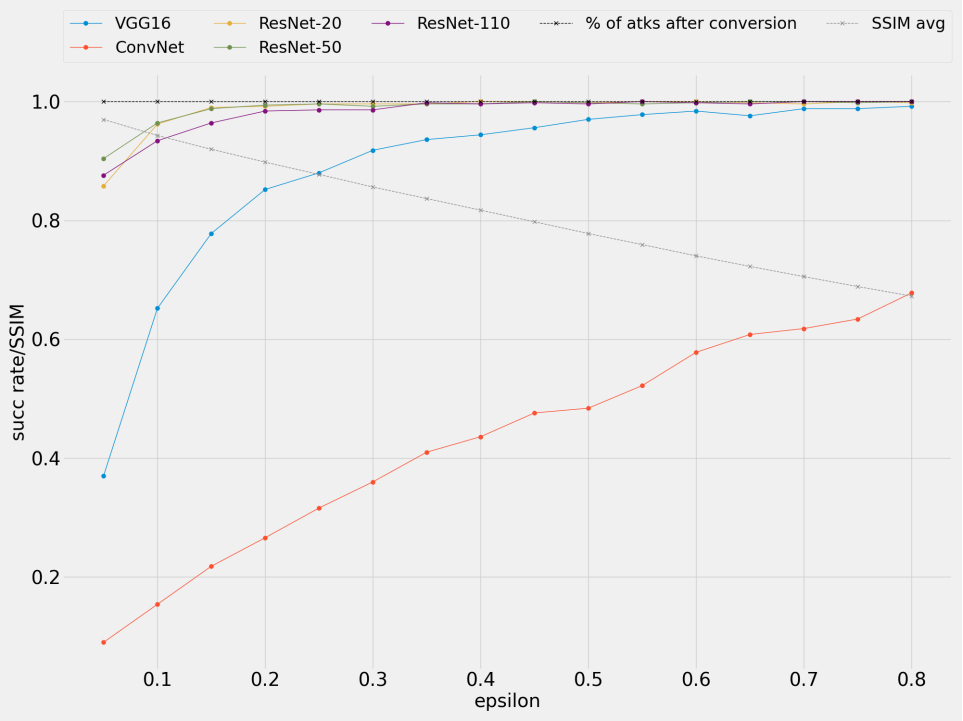


Figure 31.: Success attack rate of non-targeted PGD L_∞ norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

Non-targeted L_1 norm

In Figure 32 we can see adversarial samples per epsilon. Figure 33 shows the success attack rate on each model per ϵ, their corresponding average SSIM and total percentage of successful adversarial samples on model generator after converting to valid images.

For SSIM index 0.95 the epsilon found was $\epsilon = 180$ attaining an average transferability success attack rate of 73.79%, and for SSIM index 0.80 the epsilon found was $\epsilon = 840$ attaining an average transferability success attack rate of 89.71%.

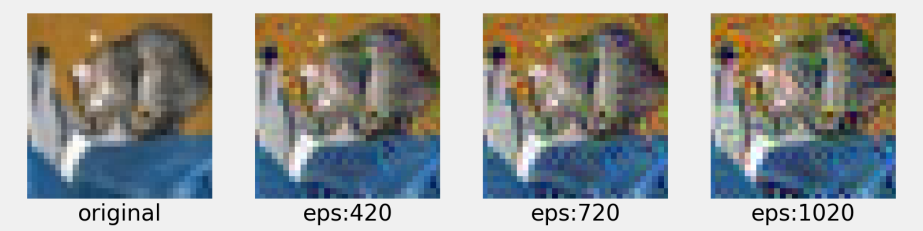


Figure 32.: Non-targeted PGD L_1 norm adversarial sample per epsilon

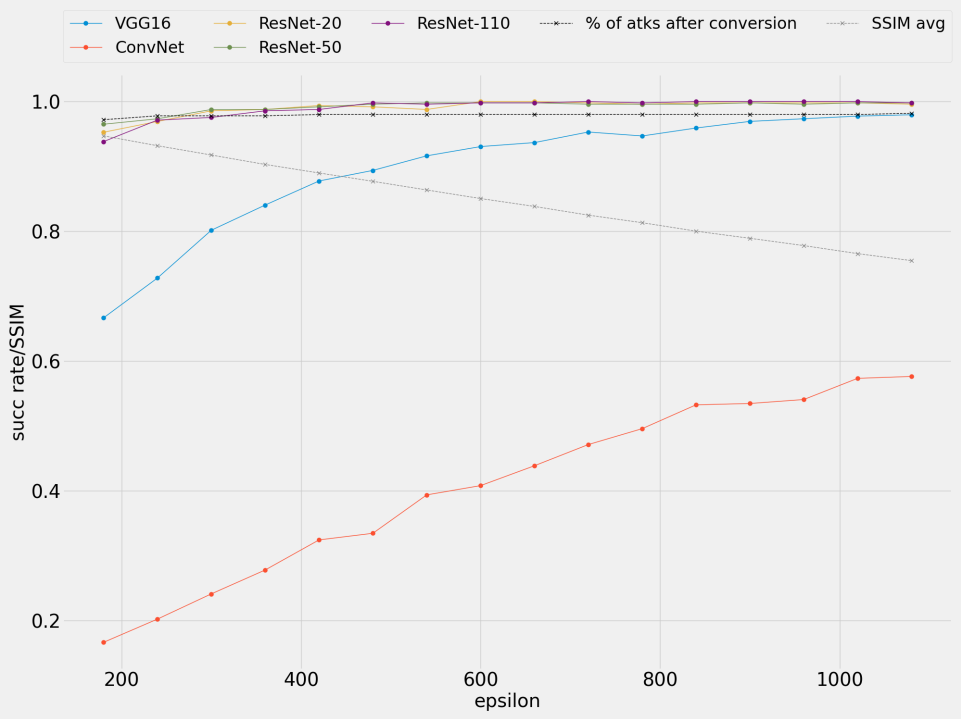


Figure 33.: Success attack rate of non-targeted PGD L_1 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

Non-targeted L_2 norm

In Figure 34 we can see adversarial samples per epsilon. Figure 35 shows the success attack rate on each model per ϵ, their corresponding average SSIM and total percentage of successful adversarial samples on model generator after converting to valid images.

For SSIM index 0.95 the epsilon found was $\epsilon = 5.0$ attaining an average transferability success attack rate of 73.53%, and for SSIM index 0.80 the epsilon found was $\epsilon = 23$ attaining an average transferability success attack rate of 89.30%.

In Table 14 are the results of success attack rate per SSIM for each norm. As we can see the norm that achieved highest average transferability for both SSIM=0.80 and SSIM=0.95 was the L_1 norm, this norm will be used for the full scale attacks.

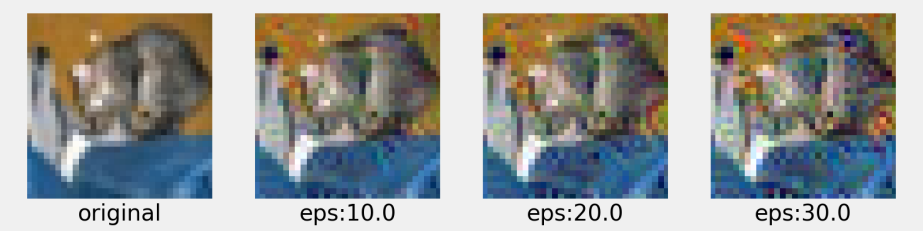


Figure 34.: Non-targeted PGD L_2 norm adversarial sample per epsilon

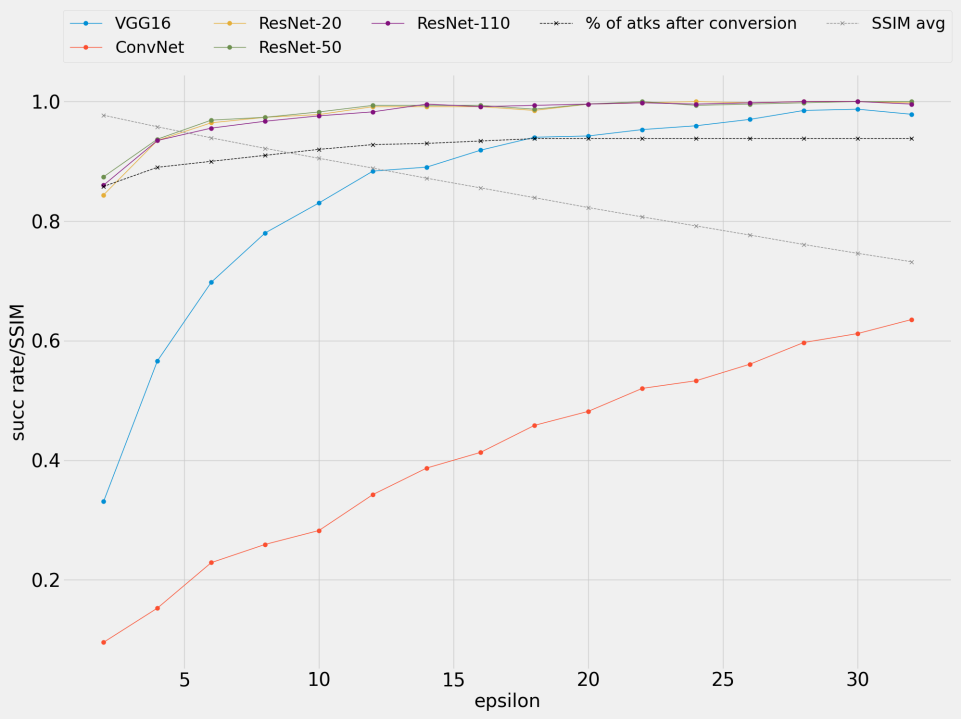


Figure 35.: Success attack rate of non-targeted PGD L_2 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

Summary for non-targeted attacks

All the non-targeted attacks exhibit the same relative behaviour. The ResNet family has the highest transferability rates, while the ConvNet is once again the more robust by a significant margin.

Targeted attack

In Table 15 are the results of successful target attack rate per SSIM and epsilon for each norm. As we can see the norm that archived highest average transferability for SSIM= 0.80 and SSIM= 0.95 was the L_∞ norm, this will be the norm used for the full scale attacks.

PGD	L_∞		L_1		L_2	
SSIM (±0.005)	0.95	0.80	0.95	0.80	0.95	0.80
epsilon	0.08	0.45	**180**	**840**	5	23
avg transf atk	71.28	88.60	**73.79**	**89.71**	73.53	89.30
succ atk on generator	100.00	100.00	**97.20**	**98.00**	89.60	93.80
succ atk aft conv	100.00	100.00	**100.00**	**100.00**	100.00	100.00
total atks aft conv	100.00	100.00	**97.20**	**98.00**	89.60	93.80

Table 14.: Non-targeted PGD average SSIM, epsilons, success attack rate, success attacks after conversion and total successful attacks for each norm

Target PGD	L_∞		L_1		L_2	
SSIM (±0.005)	0.95	0.80	0.95	0.80	0.95	0.80
epsilon	**0.175**	**0.725**	2000.0	5800.0	60.0	165.0
avg transf atk	**42.56**	**63.44**	30.48	52.48	28.44	47.60
succ atk on generator	**100.00**	**100.00**	100.00	100.00	100.00	100.00
succ atk aft conv	**100.00**	**100.00**	100.00	100.00	100.00	100.00
total atks aft conv	**100.00**	**100.00**	100.00	100.00	100.00	100.00

Table 15.: targeted PGD average SSIM, epsilons, success attack rate, success attacks after conversion and total successful attacks for each norm

The targeted attacks show a similar pattern of behaviour to the non-targeted version, with the expected drop of transferability, see Figures in Section A.1.4.

3.4.7 *One Pixel/Few Pixels*

For this attack we use the implementation in Song (2019) based on Su et al. (2019). The parameters used for the differential evolution algorithm were: $maxiter = 200$ and $popsize = 400$, the only variable being the number of pixels perturbed. Due to this attack being computationally costly, instead of using 500 samples per parameter we used only 250.

In Figure 36 we can see adversarial samples for different numbers of pixels perturbed.

Figure 36.: Non-targeted Few Pixels adversarial attack sample for some number of pixels perturbed

Figure 37 shows the success attack rate on each model per number o pixels perturbed, their corresponding average SSIM and total successful adversarial samples (the samples

generated with this attack are already on the form of valid RGB images with $[0, 1]$ range) on model generator.

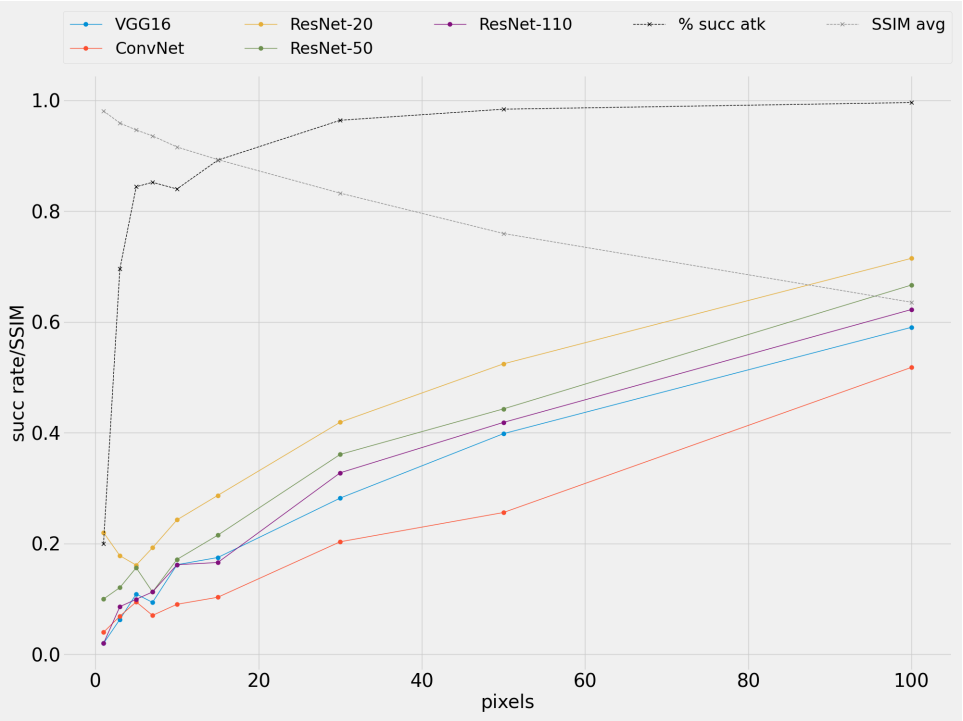


Figure 37.: Success attack rate of non-targeted Few Pixels attack on each model and their corresponding SSIM and total successful adversarial attacks on model generator per number of pixels perturbed

For SSIM=0.95 the number of pixels perturbed found was 5 pixels attaining an average transferability attack success rate of 12.42% and for SSIM=0.80 the number of pixels perturbed found was 37 achieving an average transferability attack success rate of 31.89%, Table 16.

Few Pixels attack	**Non-targeted**		**Targeted**	
SSIM($\pm$0.005)	0.95	0.80	0.95	0.80
# pixels	**5**	**37**	**5**	**37**
avg transf atk	**12.42**	**31.89**	_	_
target avg transf atk	_	_	**4.20**	**10.92**
succ atk on generator	**84.40**	**97.60**	**40.00**	**65.20**

Table 16.: Non-targeted Few Pixels attack average SSIM, number of pixels perturbed and success attack rate for targeted and non-targeted setting

When looking at the transferability rates, this attacks displays the same relative behaviour as PGD, with ConvNet being the most robust model. Nontheless its transferability is much lower than PGD.

Targeted Attack

For SSIM=0.95 the number of pixels perturbed found was 5 pixels attaining an average transferability targeted attack success rate of 4.20%, and for SSIM=0.80 the number of pixels perturbed found was 37 achieving an average transferability attack success rate of 10.92%. Table 16 presents the results for both SSIM indices, and Figure 99 displays the graph for all models and values of pixels perturbed. Figures in Section A.1.2 shows the same trend as the non-targeted attack version.

3.4.8 *Adversarial attack comparison*

In this section we start from the results attained on the previous sections for best parameters values and norms which resulted in best transferability success rate. These values will be used to realize a more comprehensive test. Now for each pair (attack method - SSIM) we generate 1000 samples (100 for each class) and use 5 different models for each architecture except for the ResNet-50 where we will use 4 (the 5th model is the model used to generate the samples). The final result will be the average of these runs.

In this section we refer to SSIM 0.95 as small perceptual change (S) and to SSIM 0.80 as large perceptual change (L) to keep the graphics and tables uncluttered.

Non-targeted transferability

In Figure 38 and Figure 39 we can see a sample for all the different non-targeted attacks considering both perceptual changes. All these samples were generated using the parameters and norms that attain best average misclassification accuracy found in the previous sections.

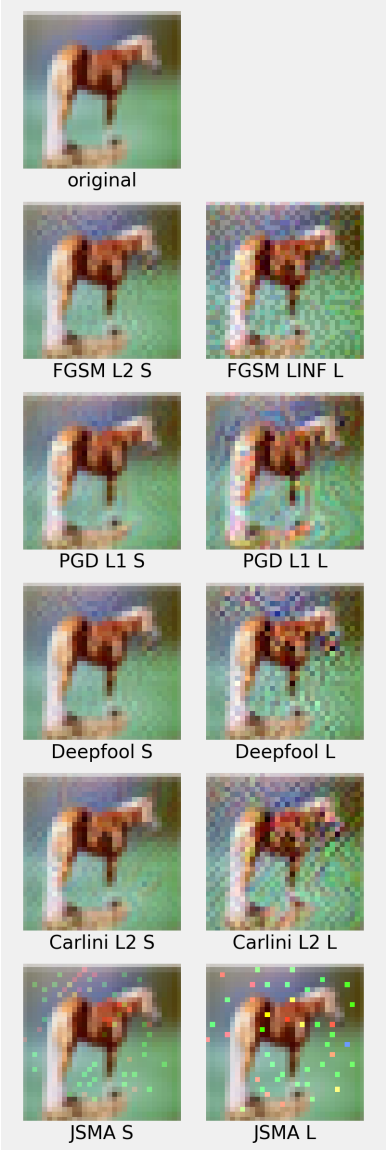


Figure 38.: Sample for non-targeted attacks (FGSM, PGD, Deepfool, Carlini and JSMA) within $[0.925, 0.975]$ and $[0.75, 0.85]$ SSIMs for S and L settings respectively

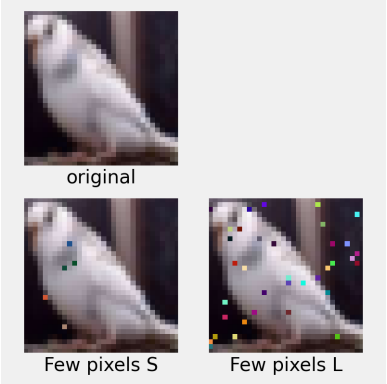


Figure 39.: Sample for non-targeted attack with One Pixel method within $[0.925, 0.975]$ and $[0.75, 0.85]$ SSIMs for S and L settings respectively

Based in Figure 38 and Figure 39 it is clear that both JSMA and Few Pixels produce clearly noticeable artifacts in the image, in the form of highly salient coloured pixels. The disturbance in the remaining methods resembles the existence of noise, not being so easily discernible for a human if they are in fact the result of adversarial attacks.

In Figure 40 we see the different attacks and the their transferability rate between models. From the 1000 samples generated for each attack setting we picked only the samples that were misclassified by the generator model (ResNet50 Generator) when converted to the range $[0, 1]$. These samples were then predicted by the other models to calculate the transferability rate. The percentage of misclassified samples per attack can be seen in Table 17.

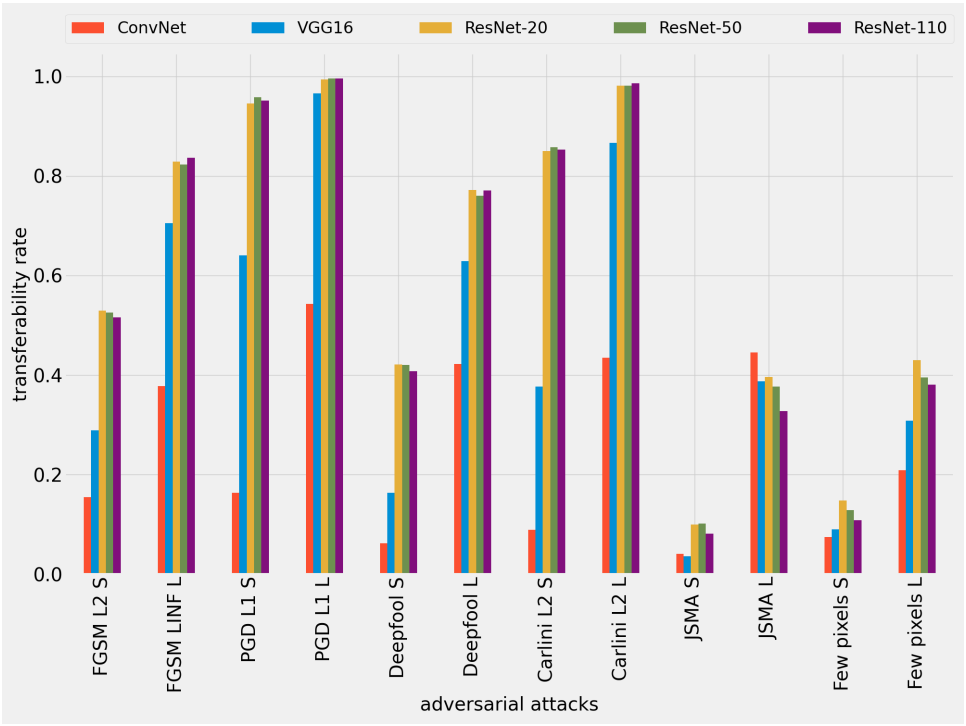


Figure 40.: Non-targeted attacks and corresponding transferability transfer rate.

nt atk	VGG16	ConvNet	R20	R50	R110	TA	I%	C%	F%
FGSM L_2 S	28.87	15.51	**52.99**	52.55	51.61	40.31	54.00	99.81	53.90
FGSM L_∞ L	70.51	37.79	82.89	82.31	**83.69**	71.44	83.20	99.04	82.40
PGD L_1 S	64.07	16.35	94.54	**95.83**	95.14	73.18	95.90	100.00	95.90
PGD L_1 L	96.58	54.27	99.43	**99.62**	99.59	89.90	97.80	100.00	97.80
Deepfool S	16.33	6.16	**42.09**	42.01	40.81	29.48	95.90	99.90	95.80
Deepfool L	62.93	42.27	**77.21**	76.00	77.07	67.09	88.20	99.09	87.40
Carlini L_2 S	37.70	8.88	85.02	**85.78**	85.30	60.54	100.00	100.00	100.00
Carlini L_2 L	86.65	43.50	98.12	98.17	**98.60**	85.01	99.90	100.00	99.90
JSMA S	3.64	4.08	9.95	**10.19**	8.10	7.19	100.00	77.00	77.00
JSMA L	38.72	**44.51**	39.57	37.66	32.77	38.65	100.00	23.50	23.50
Few pixels S	9.04	7.43	**14.77**	12.87	10.84	10.99	84.50	100.00	84.50
Few pixels L	30.81	20.89	**42.99**	39.52	38.11	34.46	97.10	100.00	97.10

Table 17.: CIFAR-10 non-targeted transferability table. I% is the initial attacks successful on generator, C% the percentage of attacks successful on generator after conversion to valid image, and F% final percentage of successful attacks on generator that were used to attack the other networks, this is a pure white box attack. Column TA is the transferability average of an attack across all networks. In bold are the highest values per attack, in green and blue cells are the highest transferability values of all attacks per model architecture for the L and S setting, respectively

From Figure 40 and Table 17 we can observe the following regarding the model architectures:

- Overall, the most robust architecture is ConvNet, possibly due to being the shallowest model, followed by VGG16, with the ResNet family being the most susceptible to the transferability of adversarial samples. This hints that architectural similarities may be relevant when considering the transferability of adversarial attacks.
- The only exception to the first point is JSMA L (large perturbation), where all architectures show a more similar transferability rate. However, note that the transferability of JSMA is small compared to most other methods;
- Within the ResNet family it is interesting to note that, although the adversarial samples were crafted in a ResNet-50, these models are not the least robust in some settings;
- It is worth noting that, while Few Pixels does not use any architectural knowledge, it follows the general transferability trend regarding architecture similarity;
- The attacks with best transferability overall are PGD and Carlini. On the other hand JSMA and Few Pixels are the less capable methods regarding transferability.

Considering the two worst methods regarding transferability JSMA and Few Pixels, it is noticeable that both adopted the same approach of affecting only a small number of pixels with a high perturbation. All other methods produce a more diffuse perturbation. This may hint that the latter approach is preferable regarding transferability.

Note that, as mentioned before, the SSIM values attained using these parameters are an average over the datasets. This implies that some samples have indexes values above or below that target values. This effect can be seen in Figure 41 presenting some Carlini L_2 samples crafted using the same parameters and their distortion level varies significantly.

This variability is directly related to the uncertainty of the level of perturbation obtained with an attack considering a fixed set of parameters. The higher the variability the less certain we can be of the amount of degradation present in an image.

In Table 18 we present more information regarding the obtained SSIMs of each attack method for the adversarial examples crafted on the generator.

We can see that the attacks with highest SSIM variability among samples are Deepfool and Carlini L_2 for L setting. On the other hand PGD is the more consistent method regarding the level of perturbation generated for an adversarial sample.

In order to evaluate the influence of the level of perturbation, we repeated the experiments picking only the adversarials that have SSIMs within $[0.925, 0.975]$ and $[0.75, 0.85]$ for the S setting and the L settings respectively. In Figure 42 we can see the results, the overall trends are the same as in Figure 40. The differences between these two experiments can be seen in Figure 43. The increase in transferability when using threshold selection is

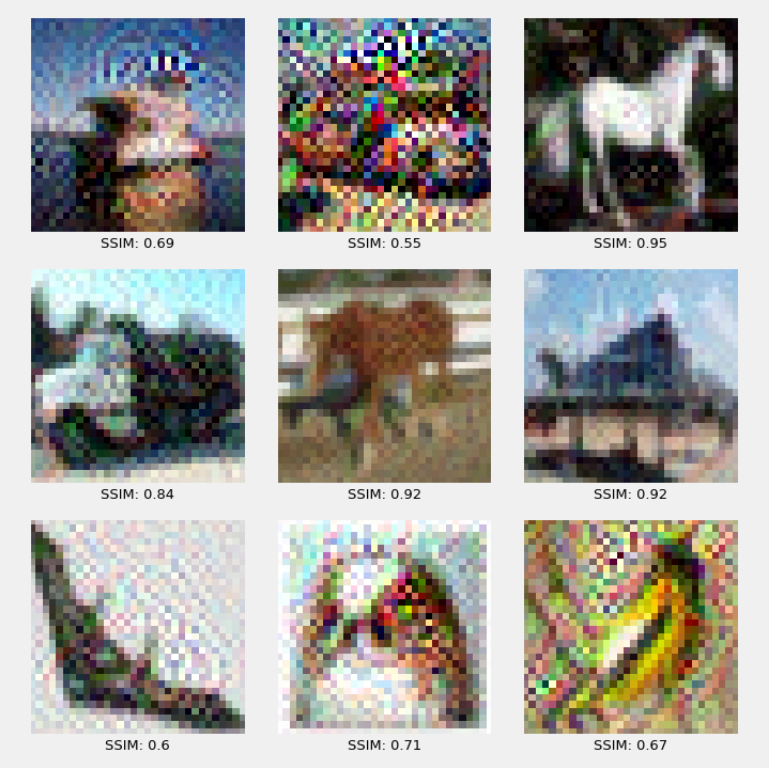


Figure 41.: CIFAR-10 Carlini L_2 samples using the same parameters ($k = 78$)

below 10% for most methods, the only exception being JSMA L where we have a significant increase in transferability on VGG16 and ConvNet models, with the latter obtaining an increase over 20%. Nevertheless, this does not change that JSMA relative performance regarding transferability still remains weak.

non-targeted adv SSIMs	min	max	avg	std	% within correct SSIMs
FGSM L_2 S	0.675	1.000	0.967	0.039	37.80
PGD L_1 S	0.815	1.000	0.949	0.027	75.50
Deepfool S	0.706	1.000	0.950	0.046	41.60
Carlini L_2 S	0.261	0.998	0.959	0.058	30.60
JSMA S	0.638	0.999	0.948	0.039	53.00
Few pixels S	0.774	0.997	0.944	0.028	70.40
FGSM L_∞ L	0.287	0.946	0.794	0.091	41.20
PGD L_1 L	0.414	1.000	0.807	0.078	51.10
Deepfool L	0.316	1.000	0.783	0.134	27.20
Carlini L_2 L	0.014	1.000	0.804	0.177	18.20
JSMA L	0.447	0.995	0.877	0.076	24.00
Few pixels L	0.412	0.942	0.801	0.070	54.10

Table 18.: SSIM statistics of crafting examples on model generator for each CIFAR-10 non-targeted adversarials attacks using the parameters found in Section 3.4.2

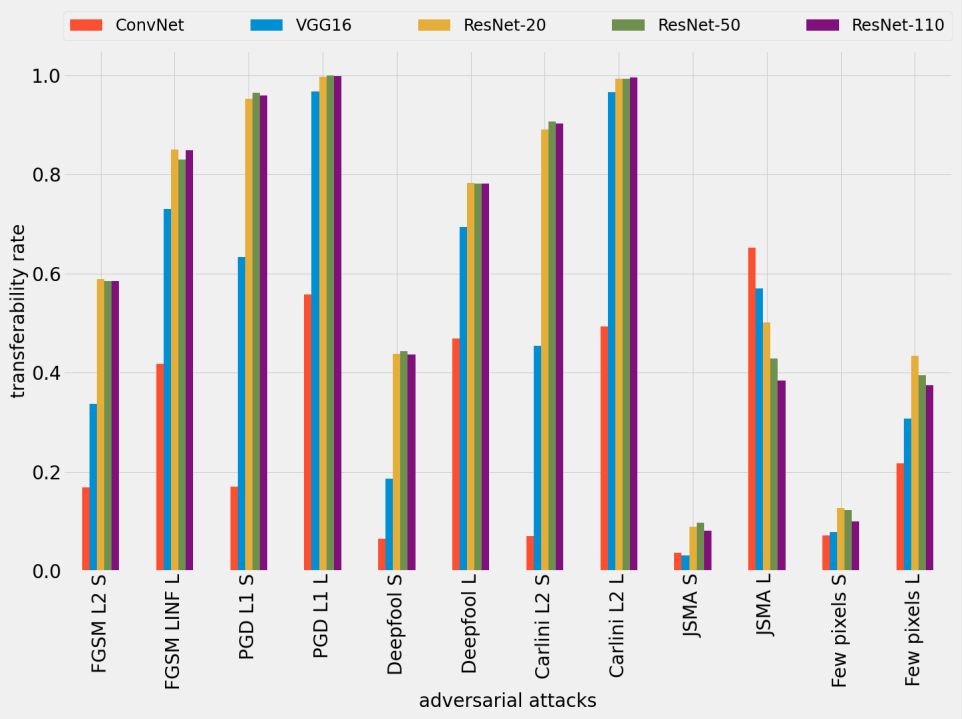


Figure 42.: Non-targeted attacks and corresponding transferability rate with adversarials samples that are within the threshold interval

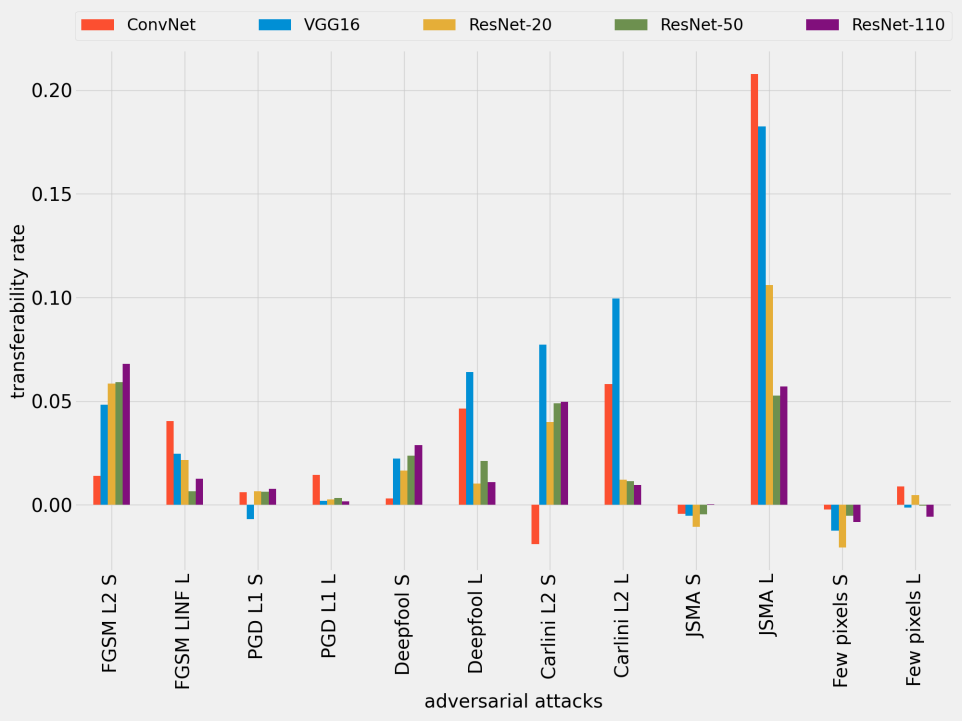


Figure 43.: Non-targeted attacks transferability rate difference between attacks within threshold and without restrictions

Concluding we can state that PGD is not only the most effective method regarding transferability on CIFAR-10, but it is also the method that provides the most consistent degradation when creating the adversarial samples.

Targeted transferability

In this setting we used all attacks methods in targeted environment. This excludes Deepfool because it does not have target behavior.

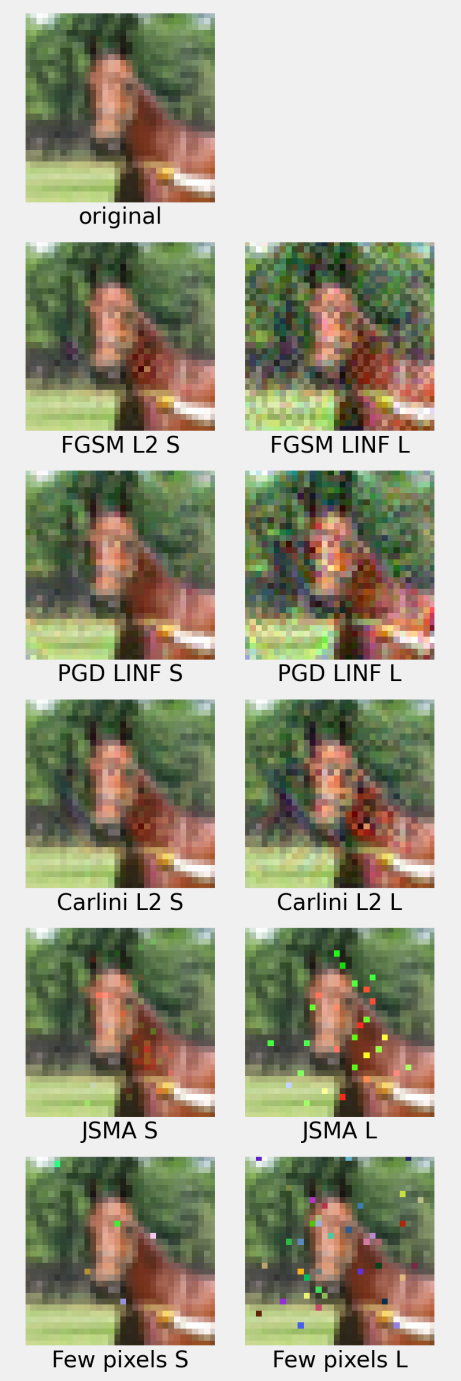


Figure 44.: Sample for targeted attacks within $[0.925, 0.975]$ and $[0.75, 0.85]$ SSIMs for S and L settings respectively

In Figure 44 we can see all the different targeted attacks considered, for each type of perturbation used: small (S) and large (L). All these samples were generated using the parameters and norms that attained best average misclassification accuracy, found in the

previous subsections.

As in the non-targeted attacks, JSMA and Few Pixels methods also show clearly visible artifacts in the form of coloured pixels. For the remaining methods, the perturbations follow the same noisy pattern as in the non-targeted case. This is to be expected as the methods do not have a fundamental algorithm change between non-targeted and targeted versions.

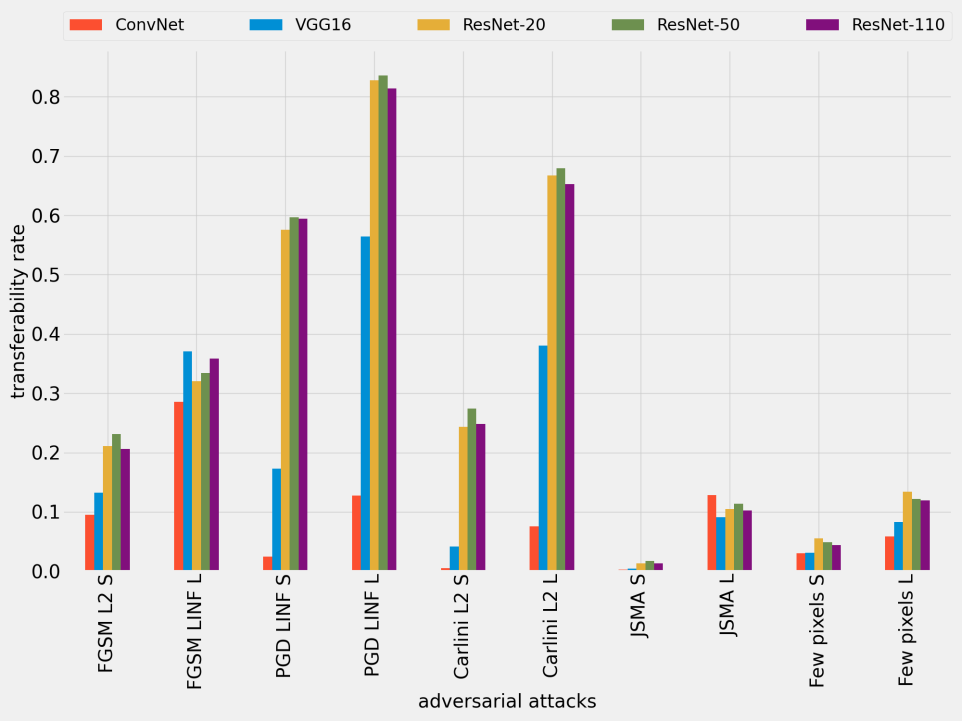


Figure 45.: Targeted attacks and corresponding transferability transfer rate.

In Figure 45 we can see the different target attacks and their transferability rate between models, the main takeaways being:

- Compared to the non-targeted attacks, the transferability is smaller as expected. Recall that we only consider an attack as successful when the model misclassifies the adversarial sample on the target class;
- Once more, PGD gets the best overall transferability rate;
- Regarding the architectures, ConvNet is the architecture that achieves the lowest transferability rate, with only once going above the 20% mark, the only exception being JSMA L where the ConvNet is the architecture with highest transferability rate by a small margin;
- Again the ResNet family has the highest transferability rates, suggesting that a similarity in architecture can be a significant factor in attack transferability rate;

- JSMA and few pixels attack remains the worst attacks regarding transferability.

The percentage of samples classified as target class per attack can be seen in Table 19.

target atks	VGG16	ConvNet	R20	R50	R110	TA	I	C	F
FGSM L_2 S	13.24	9.51	21.07	**23.11**	20.62	17.51	23.00	97.83	22.50
FGSM L_∞ L	**37.06**	28.53	32.06	33.46	35.88	33.40	13.90	97.84	13.60
PGD L_∞ S	17.26	2.44	57.58	**59.65**	59.42	39.27	100.00	100.00	100.00
PGD L_∞ L	56.42	12.74	82.78	**83.62**	81.40	63.39	100.00	100.00	100.00
Carlini L_2 S	4.14	0.54	24.30	**27.40**	24.82	16.24	100.00	100.00	100.00
Carlini L_2 L	38.00	7.55	66.76	**67.93**	65.27	49.10	94.10	100.00	94.10
JSMA S	0.45	0.28	1.35	**1.72**	1.35	1.03	100.00	71.20	71.20
JSMA L	9.12	**12.84**	10.49	11.40	10.20	10.81	100.00	20.40	20.40
Few pixels S	3.06	3.01	**5.53**	4.85	4.42	4.17	41.20	100.00	41.20
Few pixels L	8.31	5.83	**13.42**	12.14	11.90	10.32	66.90	100.00	66.90

Table 19.: CIFAR-10 targeted transferability table. I% is the initial attacks successful on generator, C% the percentage of attacks successful on generator after conversion, and F% final percentage of successful attacks on generator that were used to attack the other networks, this is a pure white box attack. Column TA is the transferability average of an attack across all networks. In bold are the highest values per attack, and in green and blue cells are the highest transferability values of all attacks per model architecture, for the L and S setting, respectively

FGSM obtains the best transferability rate for ConvNet in both *S* and *L* settings, while being the least successful method in generating adversarial samples for the generator model.

Apart from ConvNet result with FGSM, these results follow the trends found with the non-targeted attacks, with PGD dominating in all the other architectures, although with a lower transferability rate as expected.

3.5 GTSRB: Adversarial attacks

This section details all the experiments performed on models trained with GTSRB. It starts by describing the specific procedure details for this dataset in Section 3.5.1. From Section 3.5.2 to Section 3.5.5 the parameter search for each of the attacks involved are detailed. This search is performed using a single network for each architecture. Finally, Section 3.5.6 presents the final aggregate test results, using all models, for the two SSIM indices.

3.5.1 *Procedural details and attack parameters*

The overall procedural details for GTSRB are similar to those presented in 3.4.1 for the CIFAR-10 dataset. In this section we only present the details which are specific for the traffic sign dataset.

For this dataset we used the following architectures: VGG16, ConvNet, ResNet-20 and ResNet-50. For each architecture we trained 5 models. Table 20 presents the average accuracies for each architecture.

	VGG16	**ConvNet**	**ResNet-20**	**ResNet-50**
GTSRB	98.79 ± 0.20	98.81 ± 0.08	98.94 ± 0.32	98.86 ± 0.31
# train params	15008875	2736943	275211	762411

Table 20.: Average accuracy and standard deviation on test set for all models trained on GTSRB dataset and their corresponding number of trainable parameters.

Each model has been trained using the same training parameters presented in Section 3.4.1 with one exception, the VGG16: instead of using a LearningRateScheduler callback it uses a ReduceLROnPlateau during training, the callback parameters can be seen in Table 21.

	VGG16 ReduceLROnPlateau callback
monitor	val loss
factor	0.2
patience	5
min lr	0.0001

Table 21.: VGG16 model callback used for training on GTSRB dataset

Considering that in the original training set each class contains multiple 30 images sequences depicting the same sign, we removed 20% of these 30 images sequences from each class to build a validation set.

We applied the same preprocessing as done with CIFAR-10, i.e., we subtracted each image with the training set mean per channel.

The parameters used for dynamic data augmentation to train the networks can be seen in Table 22.

The attacks chosen for this test on GTSRB dataset are: FGSM, Deepfool, Carlini L_∞ and PGD; we excluded JSMA and Few Pixels attacks due to their poor performance on the CIFAR-10 experiment. To perform the attacks we randomly picked 430 images that were all correctly predicted by all the models to perform the attacks. We searched for the parameters using the same methodology used on CIFAR-10.

For the target attacks we calculate the closest and farthest classes for each class using the SSIM metric. To determine these classes we calculate the average image for each class using the training dataset and then we calculate the SSIM against each other average image

	ImageDataGenerator
rotation_range	10
width_shift_range	0.1
height_shift_range	0.1
shear_range	0.15
zoom_range	0.15
fill_mode	nearest

Table 22.: GTSRB data augmentation

class. In Figure 46 we can see a part of the confusion matrix of pairs of mean images of some classes corresponding to each category of traffic signs and their SSIM metric. Table 23 shows the corresponding closest and farthest classes for each class by name.

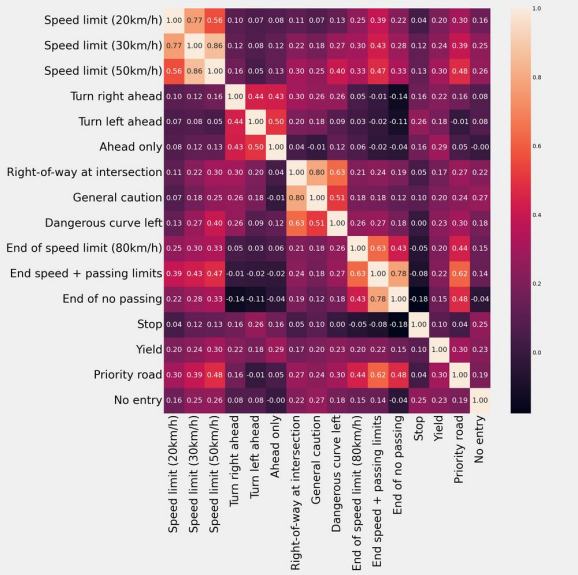


Figure 46.: Confusion matrix of pairs of mean images of some classes corresponding to each type of traffic signs and their SSIM metric

3.5.2 *FGSM*

FGSM has a single parameter ϵ, supports both targeted and non-targeted attacks, and both versions were tested considering the three error norms: L_∞, L_1, and L_2.

As for CIFAR-10, for each norm a different range of ϵ values had to be explored

Non-targeted L_∞ norm

With the L_∞ norm, the ϵ explored range was $[0.01, 0.085]$, initially in 0.005 increments. An example of an adversarial sample for some ϵ values considered is shown in Figure 47 .

cls id	**original cls**	**closest cls**	**farthest cls**
0	Speed limit (20km/h)	Speed limit (30km/h)	Keep right
1	Speed limit (30km/h)	Speed limit (50km/h)	Keep right
2	Speed limit (50km/h)	Speed limit (30km/h)	Turn left ahead
3	Speed limit (60km/h)	Speed limit (80km/h)	End of no passing
4	Speed limit (70km/h)	Speed limit (50km/h)	Turn left ahead
5	Speed limit (80km/h)	Speed limit (120km/h)	End of no passing
6	End of speed limit (80km/h)	End no passing veh >3.5 tons	Stop
7	Speed limit (100km/h)	Speed limit (120km/h)	Stop
8	Speed limit (120km/h)	Speed limit (100km/h)	Stop
9	No passing	Speed limit (50km/h)	No entry
10	No passing veh over 3.5 tons	Speed limit (80km/h)	Turn left ahead
11	Right-of-way at intersection	Road narrows on the right	Ahead only
12	Priority road	End speed + passing limits	Turn left ahead
13	Yield	No passing	Stop
14	Stop	No vehicles	End of no passing
15	No vehicles	No passing	End of no passing
16	Veh >3.5 tons prohibited	No passing	No entry
17	No entry	Speed limit (70km/h)	Veh >3.5 tons prohibited
18	General caution	Right-of-way at intersection	Ahead only
19	Dangerous curve left	Double curve	Stop
20	Dangerous curve right	Road work	Speed limit (20km/h)
21	Double curve	Dangerous curve left	Stop
22	Bumpy road	Road work	Speed limit (20km/h)
23	Slippery road	Double curve	Stop
24	Road narrows on the right	Right-of-way at intersection	Stop
25	Road work	Road narrows on the right	Speed limit (20km/h)
26	Traffic signals	Pedestrians	Ahead only
27	Pedestrians	Right-of-way at intersection	Stop
28	Children crossing	Right-of-way at intersection	Stop
29	Bicycles crossing	Road work	End of no passing
30	Beware of ice/snow	Dangerous curve right	Ahead only
31	Wild animals crossing	Double curve	Stop
32	End speed + passing limits	End of no passing	Stop
33	Turn right ahead	Keep left	End of no passing
34	Turn left ahead	Keep right	End of no passing
35	Ahead only	Turn left ahead	End of no passing
36	Go straight or right	Keep right	Beware of ice/snow
37	Go straight or left	Turn right ahead	End of no passing
38	Keep right	Go straight or right	Speed limit (20km/h)
39	Keep left	Turn right ahead	End of no passing
40	Roundabout mandatory	Priority road	Stop
41	End of no passing	End speed + passing limits	Stop
42	End no passing veh >3.5 tons	End of speed limit (80km/h)	Stop

Table 23.: Closest and farthest class by SSIM metric for each class.

Figure 48 shows the success attack rate on each model per ϵ, their corresponding average SSIM, and total percentage of successful adversarial samples on model generator after converting to valid images.

After some tuning the SSIM index 0.95 was found with $\epsilon = 0.012$ attaining an average success attack rate of 21.57%. For SSIM index 0.80 the value found was $\epsilon = 0.037$ attaining an average success attack rate of 38.63%.

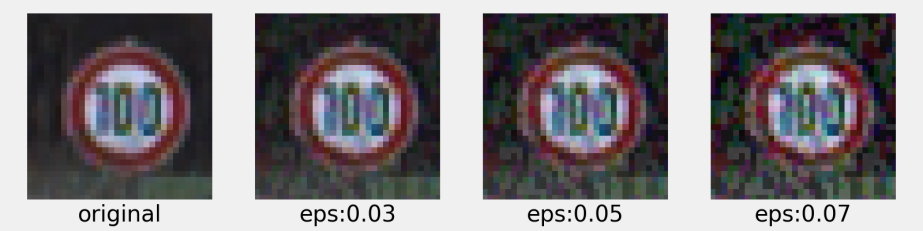


Figure 47.: Non-targeted FGSM L_∞ norm adversarial sample per ϵ

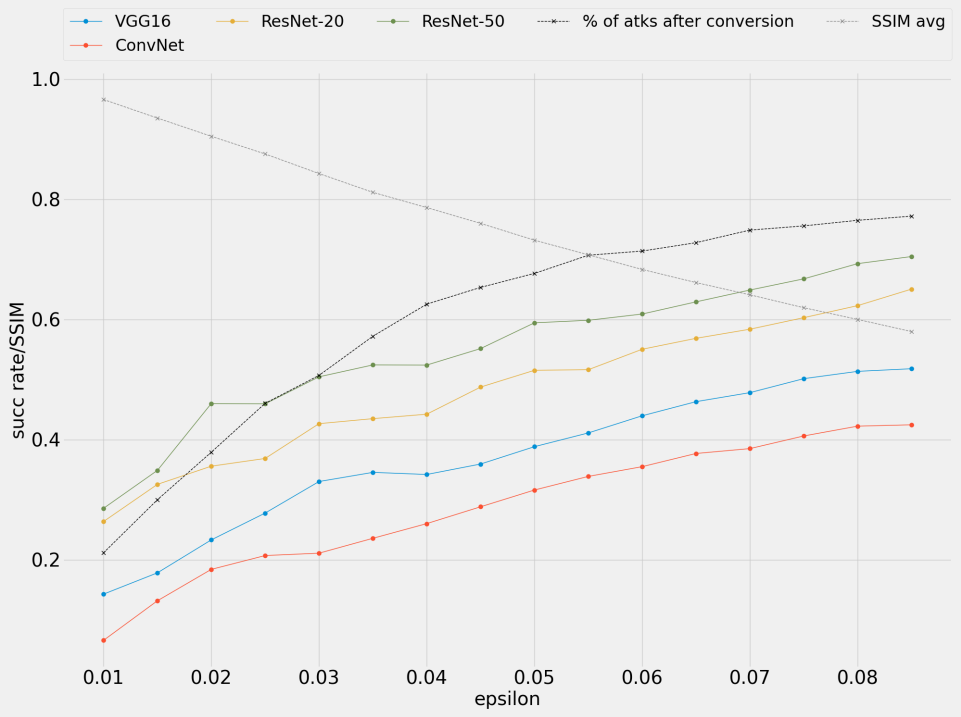


Figure 48.: Success attack rate of non-targeted FGSM L_∞ norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

Non-targeted L_1 norm

With the L_1 norm, initially the ϵ explored range was $[10, 85]$, in 5.0 increments. An example of an adversarial sample for some ϵ considered is shown in Figure 49 .

Figure 50 shows the success attack rate on each model per ϵ, their corresponding average SSIM, and total percentage of successful adversarial samples on model generator after converting to valid images.

For SSIM index 0.95 the epsilon found was $\epsilon = 18$ attaining an average success attack rate of 42.50%. For SSIM index 0.80 the epsilon found was $\epsilon = 72$ attaining an average success attack rate of 61.75%.

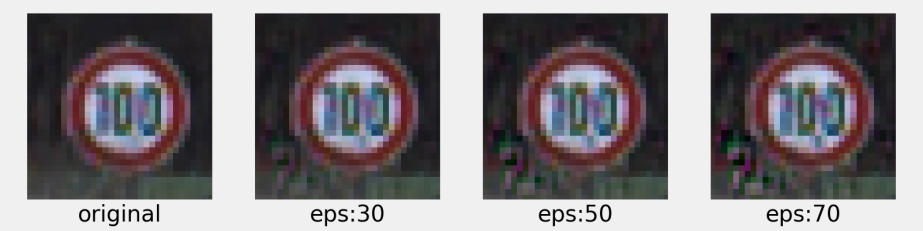


Figure 49.: Non-targeted FGSM L_1 norm adversarial sample for some epsilon

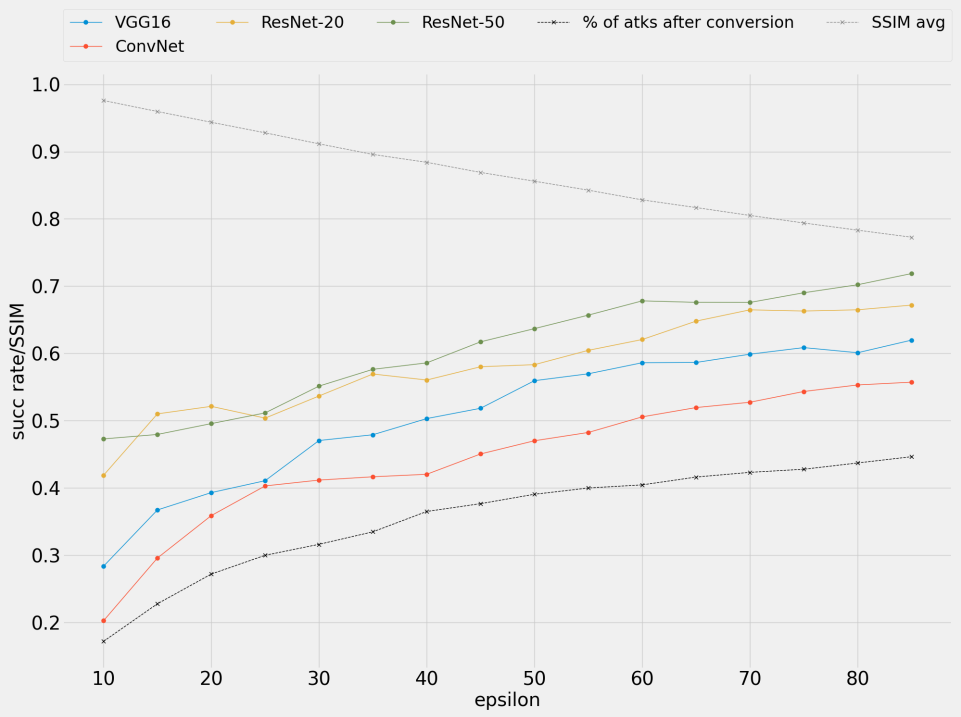


Figure 50.: Success attack rate of non-targeted FGSM L_1 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

Non-targeted L_2 norm

When using the L_2 norm, initially the ϵ explored range was $[0.25, 4]$, in 0.25 increments. An example of an adversarial sample for some ϵ considered is shown in Figure 51 .

Figure 51.: Non-targeted FGSM L_2 norm adversarial sample for some epsilon

Figure 52 shows the success attack rate on each model per ϵ, their corresponding average SSIM, and total percentage of successful adversarial samples on model generator after converting to valid images. After some tuning the SSIM index 0.95 we found was $\epsilon = 0.75$ attaining an average success attack rate of 43.92%. For SSIM index 0.80 the value found was $\epsilon = 3.0$ attaining an average success attack rate of 61.97%.

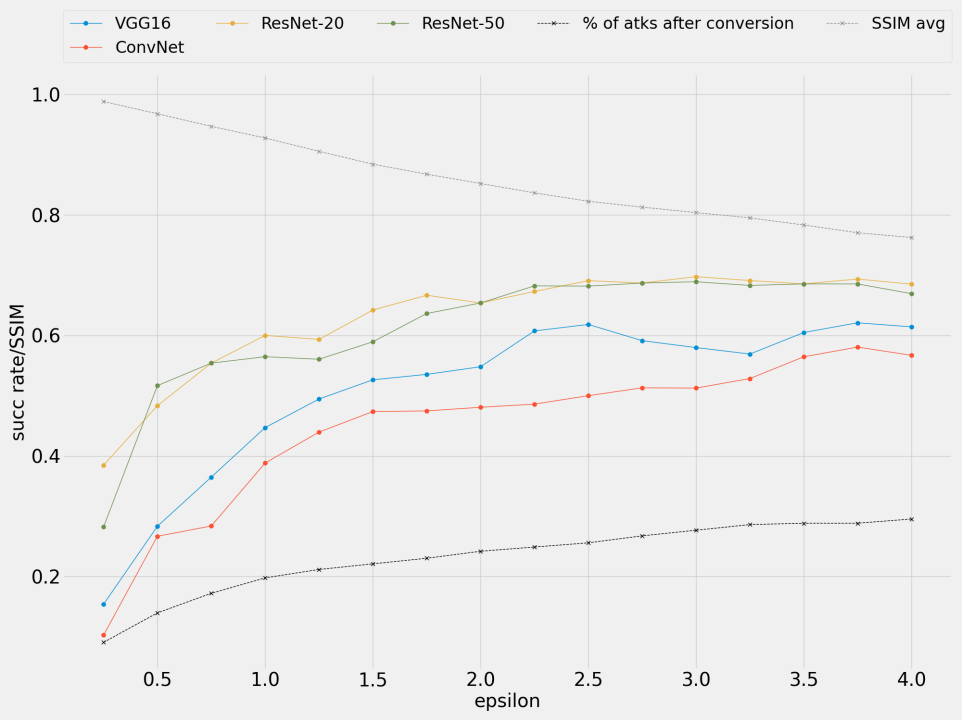


Figure 52.: Success attack rate of non-targeted FGSM L_2 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

Non-targeted summary

The graphics of the 3 norms shows the same trends found in Section 3.4.2 where ConvNet and VGG16 remain the most resistance architectures against transferability of adversarials. However, the transferability rates are significantly lower than for CIFAR-10.

Table 24 reports the results of success attack rate per SSIM and epsilon for each norm. As can be seen the norm that achieved highest average transferability for both $SSIM = 0.80$ and $SSIM = 0.95$ was L_2. This norm will be the norm used in the attacks reported in Section 3.5.6.

FGSM	L_∞		L_1		L_2	
SSIM ($\pm$0.005)	0.95	0.80	0.95	0.80	0.95	0.80
epsilon	0.012	0.037	18	72	**0.75**	**3.00**
avg transf atk	21.57	38.63	42.50	61.75	**43.92**	**61.97**
succ atk on generator	24.19	60.23	25.81	43.49	**17.21**	**28.14**
succ atk aft conv	98.08	98.46	99.10	97.86	**100.00**	**98.35**
total atks aft conv	23.72	59.30	25.58	42.56	**17.21**	**27.67**

Table 24.: FGSM non-targeted average SSIM, epsilons, success attack rate, success attacks after conversion and total successful attacks for each norm

Targeted attack

A similar search for the ϵ values required to obtain SSIM indices was performed for each norm considering targeted attacks: closest and farthest classes. In Table 25 shows the results of successful target attack rate per SSIM and epsilon for each norm for the closest class, the norm that archived highest average transferability for $SSIM = 0.80$ was L_2 and for $SSIM = 0.95$ was the L_1, the graphs related to these searches can be found in Figure 100, Figure 101 and Figure 102. All the norms show the same behaviour found in the non-targeted version, with ConvNet being the most resistant architecture against transferability of adversarials.

Similarly, Table 26 shows the results for the target attack on the farthest class, the norm that achieved highest average transferability for $SSIM = 0.80$ was L_∞, for $SSIM = 0.95$ no norm attained any transferability. The graphs related to these searches can be found in Figure 106, Figure 107 and Figure 108. Note that these targeted attacks do not have show any transferability in the S setting for all norms, and L setting attains very small transferability rates. The farthest class, as expected, has smaller transferability rates than the closer ones.

3.5.3 *Deepfool*

As mentioned in Section 3.4.3, this attack doesn't have norms, and is purely a non-targeted attack. The attack has a single parameter: ϵ. An example of an adversarial sample for some considered ϵ values is shown in Figure 53.

FGSM closest cls	L_∞		L_1		L_2	
SSIM(+-0.005)	0.95	0.80	0.95	0.80	0.95	0.80
epsilon	0.012	0.060	**18**	98	0.60	**3.25**
avg transf atks	12.90	10.19	**23.80**	20.94	22.29	**21.00**
succ atks on generator	14.65	26.51	**19.53**	28.60	19.30	**29.53**
succ atks aft conv	98.41	94.74	**98.81**	95.12	100.00	**98.43**
total atks aft conv	14.42	25.12	**19.30**	27.21	19.30	**29.07**

Table 25.: FGSM targeted attack for closest class average SSIM, epsilons, success attack rate, success attacks after conversion and total successful attacks for each norm

FGSM farthest cls	L_∞		L_1		L_2	
SSIM(+-0.005)	0.95	0.80	0.95	0.80	0.95	0.80
epsilon	0.013	**0.031**	17.0	47.0	0.55	1.45
avg transf atks	0.00	**4.49**	0.00	2.88	0.00	0.00
succ atks on generator	4.19	**9.07**	1.86	6.74	1.63	5.81
succ atks aft conv	100.00	**100.00**	100.00	89.66	100.00	92.00
total atks aft conv	4.19	**9.07**	1.86	6.05	1.63	5.35

Table 26.: FGSM targeted attack for farthest class average SSIM, epsilons, success attack rate, success attacks after conversion and total successful attacks for each norm

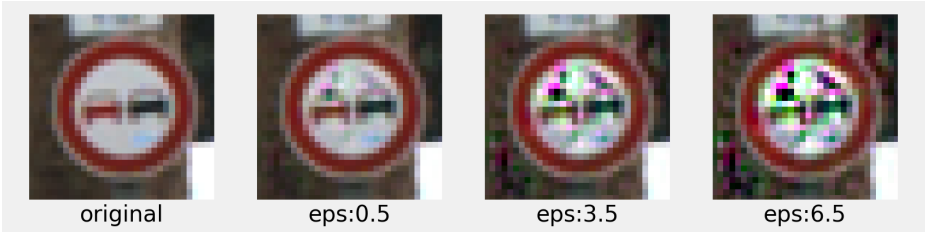


Figure 53.: Deepfool adversarial sample for some epsilon

Figure 54 shows the success rate on each model per ϵ, their corresponding average SSIM and total percentage of successful adversarial samples on model generator after converting to valid images. For SSIM= 0.95 we found $\epsilon = 0.5$ attaining an average success transferability rate of 13.14%. For SSIM = 0.80 the value found was $\epsilon = 3.1$ attaining an average success transferability rate of 51.77%, these results are shown in Table 27. The results in Figure 54 are similar to those obtained in CIFAR-10, see Section 3.4.3.

Another interesting note is that there is no significant change in transferability between CIFAR-10 and GTSRB trained models, as opposed to what occurred when attacking with FGSM, see Section 3.5.2.

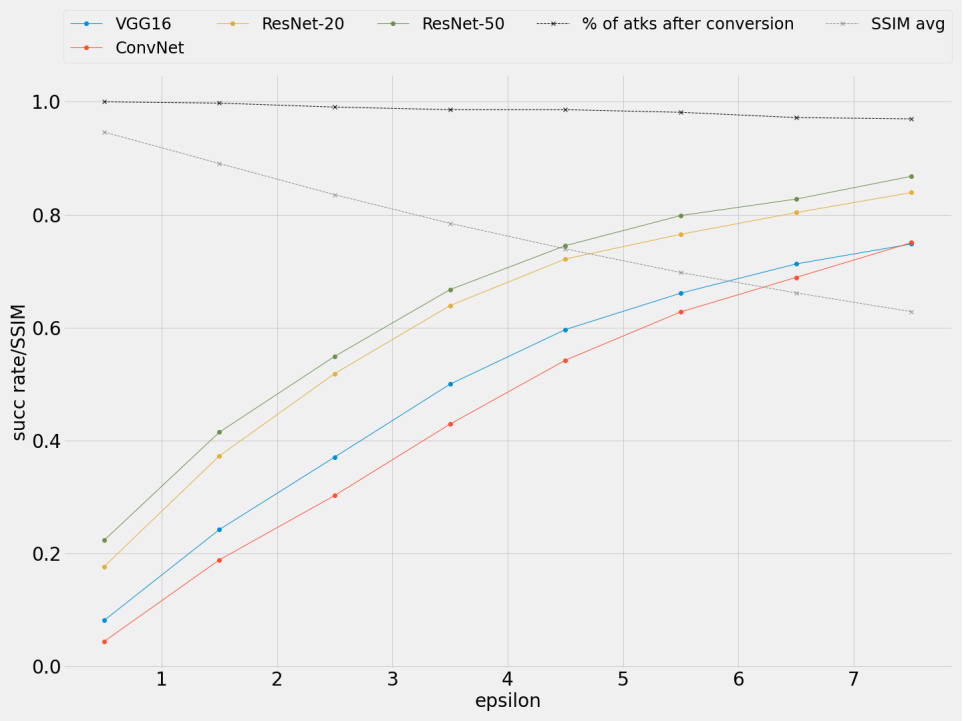


Figure 54.: Success attack rate of Deepfool on each model per ϵ

	Deepfool	
SSIM($\pm$0.005)	**0.95**	**0.80**
epsilon	**0.5**	**3.1**
avg transf atk	**13.14**	**51.77**
succ atk on generator	**100.00**	**98.60**
succ atk aft conv	**100.00**	**100.00**
total atks aft conv	**100.00**	**98.60**

Table 27.: Deepfool average SSIM, epsilons and transferability success rate

3.5.4 *Carlini*

Similarly, to Section 3.4.5 only L_2 was used in this test. The parameter to search is confidence k, the other values are kept as constant as in Section 3.4.5.

Figure 55 shows adversarial sample for different confidence k.

Figure 56 shows the success attack rate on each model per confidence k, their corresponding average SSIM and total percentage of successful adversarial samples on model generator after converting to valid images.

For SSIM=0.95 the k found was 35 attaining an average attack success rate of 32.36%, and for SSIM=0.80 the k is 57 achieving an average attack success rate of 64.34%, these results

Figure 55.: Non-targeted Carlini L_2 adversarial sample for some confidence values k

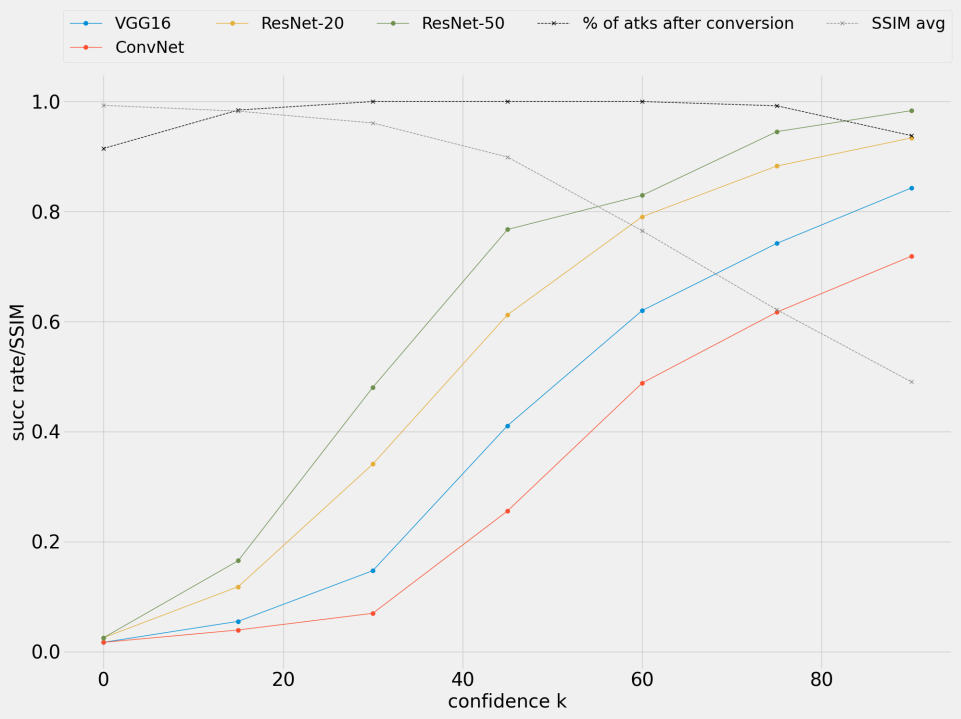


Figure 56.: Non-targeted Carlini L_2 norm success attack rate on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per k

are shown in Table 28. Figure 56 shows the same trend seen in Figure 29 where ConvNet is the more robust model regarding adversarials transferability followed by VGG16.

Targeted

A similar parameters search was performed for the target attack closest and farthest classes. Table 29 shows the results: for the closest class target attack the k found for the SSIM 0.95 and 0.80 were 7 and 41 with average transfer attack success rate of 6.20% and 51.47% respectably; for the farthest class the k found for SSIM 0.95 and 0.80 were 3 and 15 attaining an average transfer attack success rate of 1.20% and 6.15% respectably. The graphics related to these searches are shown in Section A.2.3. These graphics shows the same trend

	Carlini L_2	
SSIM($\pm$0.005)	0.95	0.80
k	**35**	**57**
c	**1**	**1**
avg transf atk	**32.36**	**64.34**
succ atk on generator	**100.00**	**100.00**
succ atk aft conv	**100.00**	**100.00**
total atks aft conv	**100.00**	**100.00**

Table 28.: Non-targeted Carlini L_2 norm average SSIM, k, c, success attack rate, success attacks after conversion and total successful attacks

where ConvNet is the more resistant architecture against transferability followed closely by VGG16.

Carlini L_2	**closest cls**		**farthest cls**	
SSIM ($\pm$0.005)	0.95	0.80	0.95	o.8o
k	**7**	**41**	**3**	**15**
c	**1**	**1**	**1**	**1**
avg transf atks	**6.20**	**51.47**	**1.20**	**6.15**
succ atks on generator	**100.00**	**26.36**	**100.00**	**94.57**
succ atks aft conv	**100.00**	**100.00**	**96.90**	**100.00**
total atks aft conv	**100.00**	**26.36**	**96.90**	**94.57**

Table 29.: Carlini L_2 targeted for closest and farthest classes, average SSIM, k, c, success attack rate, success attacks after conversion and total successful attacks

3.5.5 *PGD*

As mentioned in Section 3.4.6 PGD has a single parameter epsilon. In this test we allowed this algorithm to run for 100 iterations, and for each epsilon tested we defined the step size per iteration as $\frac{epsilon}{100}$. All three norms (L_∞, L_1, and L_2) have been tested in both targeted and non-targeted versions.

Non-targeted L_∞ norm

In Figure 57 we can see adversarial samples for selected epsilon values. Figure 58 shows the success attack rate on each model per ϵ, their corresponding average SSIM and total percentage of successful adversarial samples on model generator after converting to valid images.

For SSIM index 0.95 the epsilon found was $\epsilon = 0.03$ attaining an average success attack rate of 23.96%, and for SSIM index 0.80 the epsilon found was $\epsilon = 0.13$ attaining an average success attack rate of 61.34%.

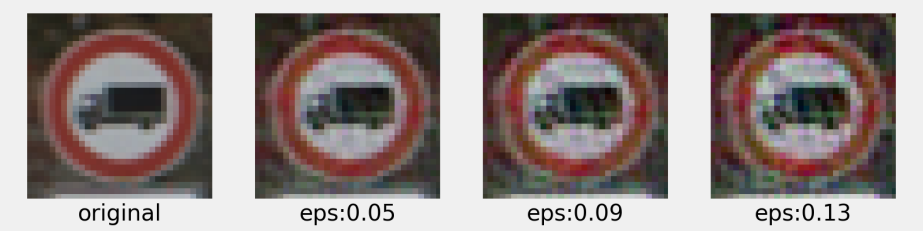


Figure 57.: Non-targeted PGD L_∞ norm adversarial sample for some epsilon

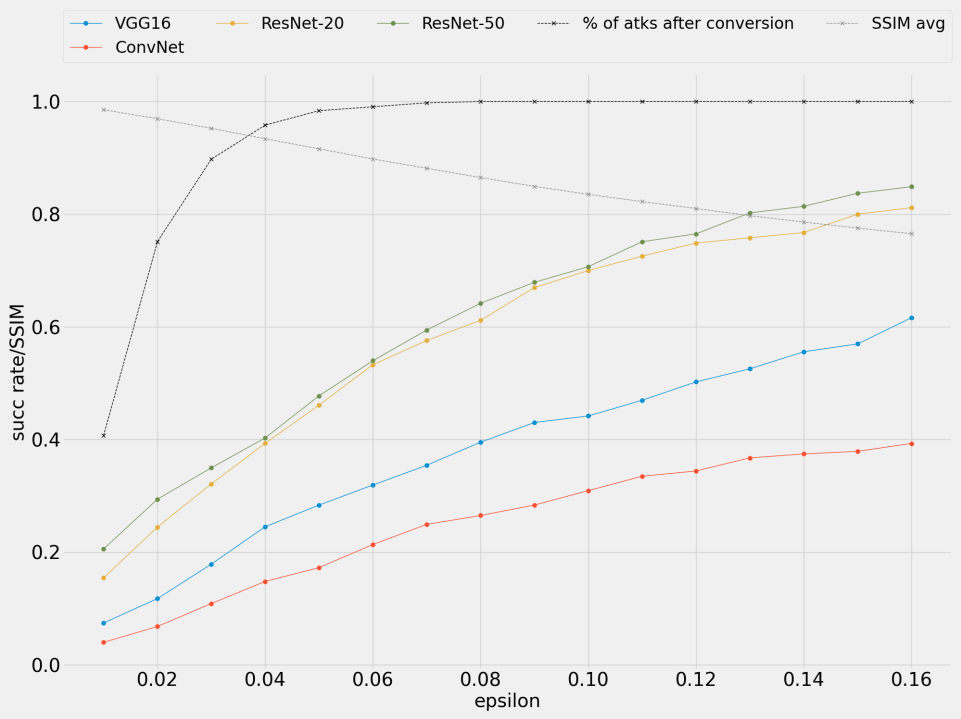


Figure 58.: Success attack rate of non-targeted PGD L_∞ norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

Non-targeted L_1 norm

In Figure 59 we can see adversarial samples for selected epsilon values. Figure 60 shows the success attack rate on each model per ϵ, their corresponding average SSIM and total percentage of successful adversarial samples on model generator after converting to valid images.

For SSIM index 0.95 the epsilon found was $\epsilon = 47$ attaining an average success attack rate of 48.03%, and for SSIM index 0.80 the epsilon found was $\epsilon = 240$ attaining an average success attack rate of 76.49%.

Figure 59.: Non-targeted PGD L_1 norm adversarial sample for some epsilon

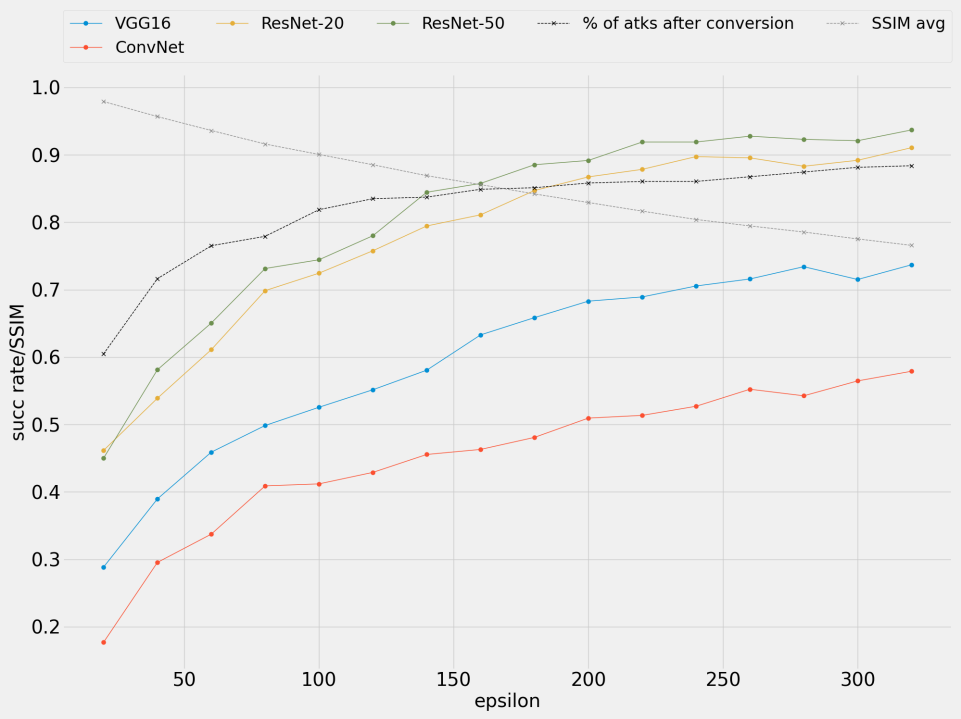


Figure 60.: Success attack rate of non-targeted PGD L_1 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

Non-targeted L_2 norm

In Figure 61 we can see adversarial samples for selected epsilon values. Figure 62 shows the success attack rate on each model per ϵ, their corresponding average SSIM and total percentage of successful adversarial samples on model generator after converting to valid images.

For SSIM index 0.95 the epsilon found was $\epsilon = 1.5$ attaining an average success attack rate of 51.77%, and for SSIM index 0.80 the epsilon found was $\epsilon = 8.0$ attaining an average success attack rate of 78.39%.

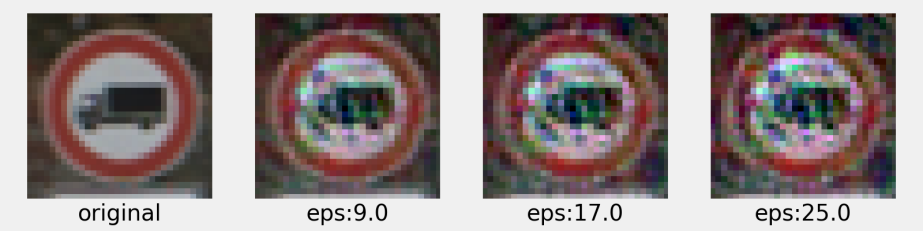


Figure 61.: Non-targeted PGD L_2 norm adversarial sample for some epsilon

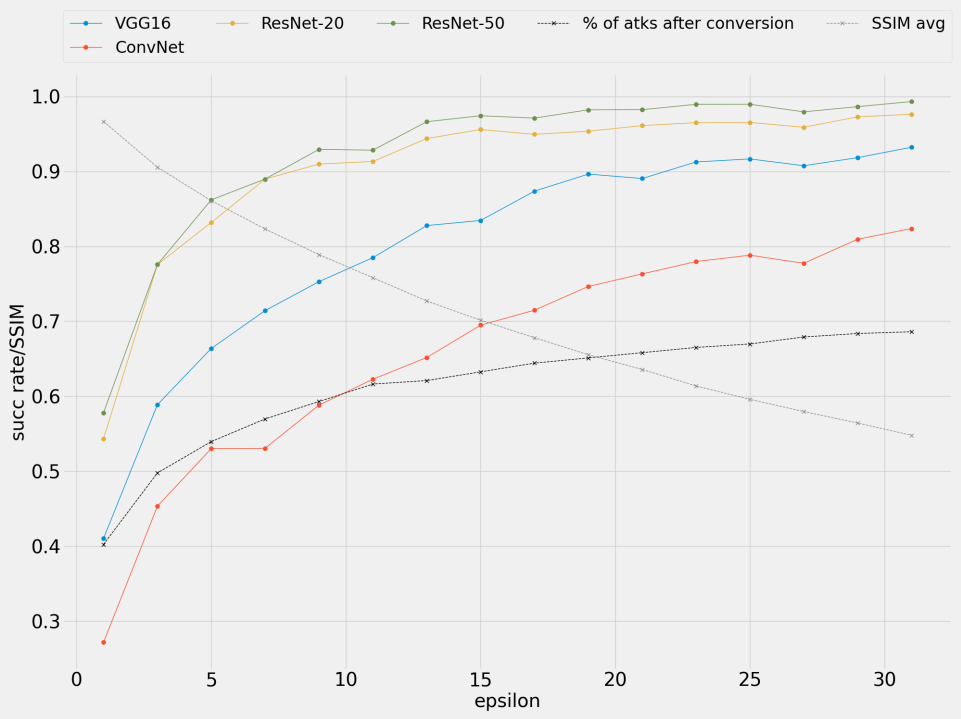


Figure 62.: Success attack rate of non-targeted PGD L_2 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

In Table 30 are the results of success attack rate per SSIM for each norm. As we can see the norm that achieved highest average transferability for SSIM=0.80 and SSIM=0.95 was the L_2 norm, this norm will be used for the corresponding SSIM for the full scale attacks.

Summary for non-targeted attacks

All non-targeted attacks exhibit the same relative behavior as other non-targeted attacks referred before in this chapter where ResNet models family group have similar behavior and are more susceptible to transferability of adversarials while ConvNet remains the architecture more robust to these followed by VGG16.

PGD	L_∞		L_1		L_2	
SSIM ($\pm$0.005)	0.95	0.80	0.95	0.80	0.95	0.80
epsilon	0.03	0.13	47	240	**1.5**	**8.0**
avg transf atk	23.96	61.34	48.03	76.49	**51.77**	**78.39**
succ atk on generator	90.93	100.00	74.19	86.05	**49.30**	**63.49**
succ atk aft conv	98.72	100.00	99.69	100.00	**100.00**	**100.00**
total atks aft conv	89.77	100.00	73.95	86.05	**49.30**	**63.49**

Table 30.: PGD non-targeted average SSIM, epsilons, success attack rate, success attacks after conversion and total successful attacks for each norm

Targeted attack

A similar search for the parameters was performed for each norm considering the two types of target attacks: closest and farthest classes. Table 31 shows the results of successful target attack rate per SSIM and epsilon for each norm for the closest class. The norm that achieved highest average transferability for SSIM= 0.80 was L_1 and SSIM= 0.95 was L_2. Similarly, Table 32 shows the results for the target attack on the farthest class, the norm that achieved highest average transferability for SSIM=0.80 and 0.95 was L_1, these will be the norms and parameters used for the full scale test. The graphics related to theses targeted parameters search can be found in Section A.2.4. All these targeted attack follow the same trend as non-targeted setting.

PGD closest cls	L_∞		L_1		L_2	
SSIM(+-0.005)	0.95	0.80	0.95	0.80	0.95	0.80
epsilon	0.04	0.23	180	**2750**	**24**	104
avg transf atks	11.04	26.57	21.05	**46.05**	**24.77**	45.64
succ atks on generator	97.44	100.00	100.00	**100.00**	**100.00**	100.00
succ atks aft conv	98.33	100.00	100.00	**100.00**	**100.00**	100.00
total atks aft conv	95.81	100.00	100.00	**100.00**	**100.00**	100.00

Table 31.: PGD targeted attack for closest class average SSIM, epsilons, success attack rate, success attacks after conversion and total successful attacks for each norm

PGD farthest cls	L_∞		L_1		L_2	
SSIM(+-0.005)	0.95	0.80	0.95	0.80	0.95	0.80
epsilon	0.04	0.24	**80**	**2000**	6	82
avg transf atks	0.39	2.67	**1.14**	**9.09**	0.93	8.95
succ atks on generator	90.23	100.00	**97.67**	**99.77**	100.00	100.00
succ atks aft conv	97.94	100.00	**99.05**	**100.00**	100.00	100.00
total atks aft conv	88.37	100.00	**96.74**	**99.77**	100.00	100.00

Table 32.: PGD targeted attack for farthest class average SSIM, epsilons, success attack rate, success attacks after conversion and total successful attacks for each norm

3.5.6 *Adversarial attack comparison*

This section uses the results attained in the search for the best parameters values and norms which resulted in best transferability success rate. For each pair (attack method - SSIM) we generated 860 samples using these parameters. From the 860 (20 for each class) samples generated for each attack setting we picked only the samples that were misclassified by the generator model when converted to a valid image (in the range [0,1]). These samples were then used to attack the other models and calculate the transferability rate. Similarly, as in Section 3.4.8 we use 5 different models for each architecture (VGG16, ConvNet, ResNet-20) except for the ResNet-50 where we will use 4 (the 5th model is the one used to generate the samples). The final result will be the average of these models per architecture.

Non-targeted transferability

Figure 63 and Figure 64 show a sample for all the different types of non-targeted attacks considering both perceptual changes. All these samples were generated using the parameters and norms that attained best average transferability found in the previous sections.

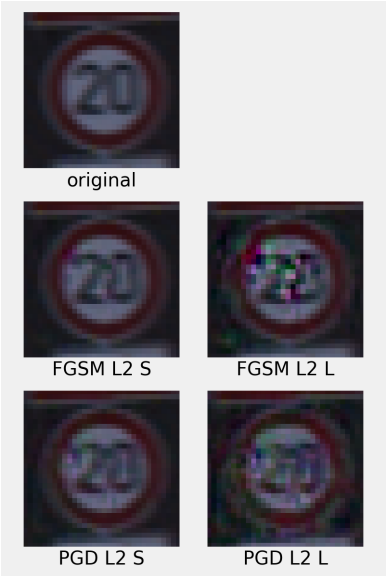


Figure 63.: Sample for non-targeted attacks (FGSM and PGD) within $[0.925, 0.975]$ and $[0.75, 0.85]$ SSIMs for S and L settings respectively

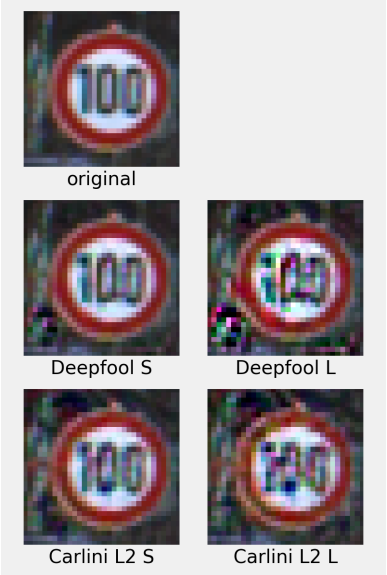


Figure 64.: Sample for non-targeted attacks (Deepfool and Carlini) within $[0.925, 0.975]$ and $[0.75, 0.85]$ SSIMs for S and L settings respectively

In Figure 65 we see the different attacks and their transferability rate between models. The percentage of misclassified samples per attack is presented in Table 33

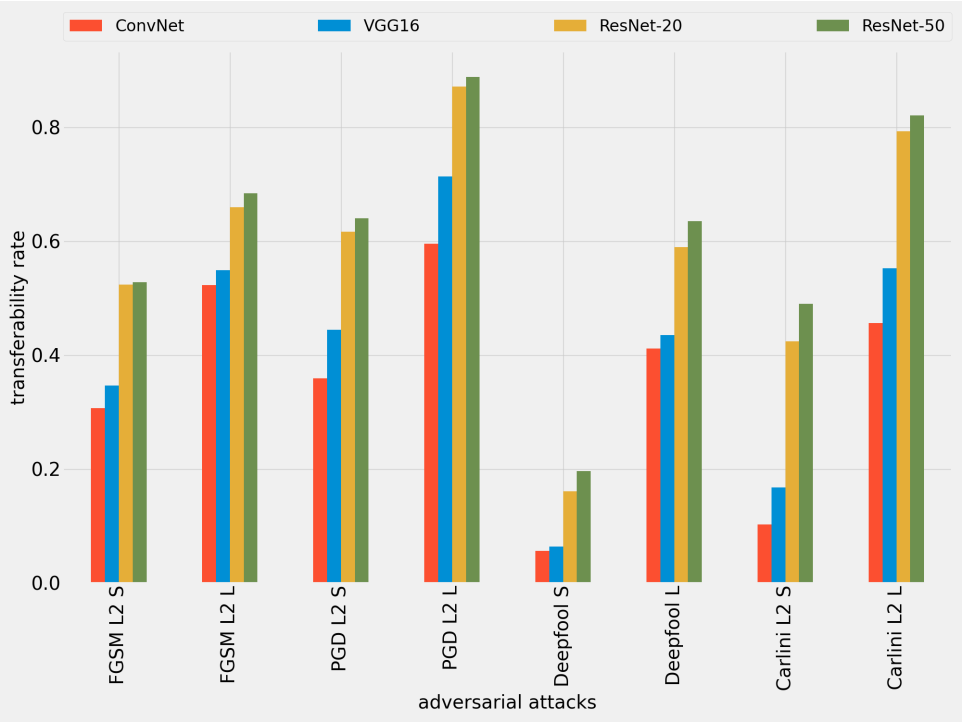


Figure 65.: Non-targeted attacks and corresponding transferability transfer rate.

nt attack	VGG16	ConvNet	R20	R50	TA	I%	C%	F%
FGSM L_2 S	34.65	30.70	52.39	**52.82**	42.64	16.51	100.00	16.51
FGSM L_2 L	54.89	52.34	65.96	**68.40**	60.40	27.91	97.92	27.33
PGD L_2 S	44.43	35.91	61.70	**64.05**	51.52	47.91	99.76	47.79
PGD L_2 L	71.42	59.56	87.18	**88.86**	76.75	63.14	100.00	63.14
Deepfool S	6.43	5.61	16.14	**19.68**	11.97	100.00	99.53	99.53
Deepfool L	43.55	41.14	58.96	**63.49**	51.78	98.26	100.00	98.26
Carlini L_2 S	16.81	10.30	42.42	**49.01**	29.64	100.00	100.00	100.00
Carlini L_2 L	55.23	45.67	79.28	**82.09**	65.57	100.00	100.00	100.00

Table 33.: GTSRB non-targeted transferability table. I% is the initial attacks successful on generator, C% the percentage of attacks successful on generator after conversion to valid image, and F% final percentage of successful attacks on generator that were used to attack the other networks, this is a pure white box attack. Column TA is the transferability average of an attack across all networks. In bold are the highest values per attack and in green and blue cells are the highest transferability values of all L and S attacks respectably, per model architecture

From Figure 65 and Table 33 we can observe the following regarding model architectures and the different attack methods:

- Overall the most robust model against transferability is ConvNet, followed by VGG16, with the ResNet family being the most susceptible to the transferability of adversarial samples. This is similar to the results found in Section 3.4.8;
- Within the ResNet family we can see only a slight decrease in robustness for ResNet-50, which is to be expected as the adversarial samples were crafted also on a ResNet-50;
- The attack with best transferability for each architecture and best average among all architectures is PGD for both L and S settings;
- The attack with worse transferability for each architecture and worst average among all architectures is Deepfool for both L and S settings.

Closest classes target transferability

In this experiment the target considered is the closest class as presented in Table 23.

Figure 66 and Figure 67 show samples for all the different type of attack methods considering both perceptual settings. All these samples were generated using the parameters and norms that attained best average target classification accuracy in Section 3.5.1.

Figure 66.: Sample for closest class targeted attacks within $[0.925, 0.975]$ and $[0.75, 0.85]$ SSIMs for S and L settings respectively

Figure 67.: Sample for closest class targeted attacks within $[0.925, 0.975]$ and $[0.75, 0.85]$ SSIMs for S and L settings respectively

In Figure 68 are shown the different attacks and their target transferability rate between models. Results are numerically presented in Table 34.

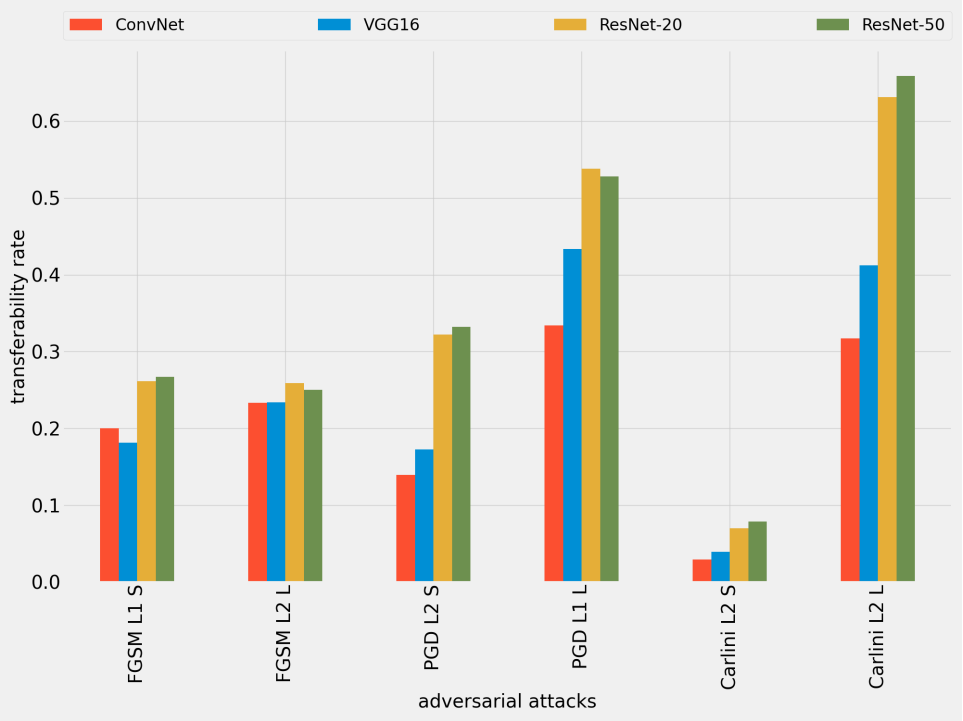


Figure 68.: Targeted transferability for closest class attacks.

cls targeted atks	VGG16	ConvNet	R20	R50	TA	I%	C%	F%
FGSM L_1 S	18.12	20.00	26.17	**26.68**	22.74	17.56	98.68	17.33
FGSM L_2 L	23.39	23.31	**25.91**	25.00	24.40	30.58	96.58	29.53
PGD L_2 S	17.26	13.95	32.21	**33.20**	24.15	100.00	100.00	100.00
PGD L_1 L	43.33	33.42	**53.81**	52.79	45.84	100.00	100.00	100.00
Carlini L_2 S	3.92	2.94	6.99	**7.87**	5.43	99.88	99.88	99.77
Carlini L_2 L	41.21	31.74	63.16	**65.89**	50.50	28.72	100.00	28.72

Table 34.: GTSRB targeted for closest class transferability table. I% is the initial attacks successful on generator, C% the percentage of attacks successful on generator after conversion to valid image, and F% final percentage of successful attacks on generator that were used to attack the other networks. Column TA is the transferability average of an attack across all networks. In bold are the highest values per attack and in green and blue cells are the highest transferability values of all L and S attacks respectably, per model architecture

From Figure 68 and Table 34 we can observe the following regarding model architectures and the different attack methods:

- Overall, the most robust architecture is ConvNet followed by VGG16, with the ResNet family being the most susceptible to the transferability of targeted adversarial samples. This behavior is consistent with the CIFAR-10 experiments and with GTSRB non-targeted experiment;

- Note that, although both FGSM S and L attained similar results transferability wise, implying that the level of distortion is not a major factor, the percentage of successful attacks on the model generator is very low, thereby providing a diminished statistical significance to these results.
- One curious observation is that Carlini L_2 attained both worst and highest targeted transferability rates for S and L settings, respectably.

As expected this targeted attack attained worse transferability rates than their non-targeted counterpart.

Farthest classes target transferability

This experiment also uses targeted attacks but now we used the farthest class as the target.

Figure 69 displays samples for all different types of attacks methods considering both perceptual settings.

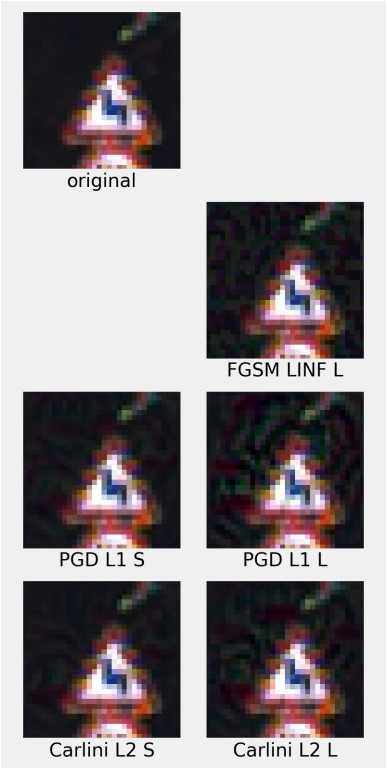


Figure 69.: Sample for farthest class targeted attacks within $[0.925, 0.975]$ and $[0.75, 0.85]$ SSIMs for S and L settings respectively

Figure 70 shows the different attacks and their targeted transferability rate between models. Numerical results are presented in Table 35.

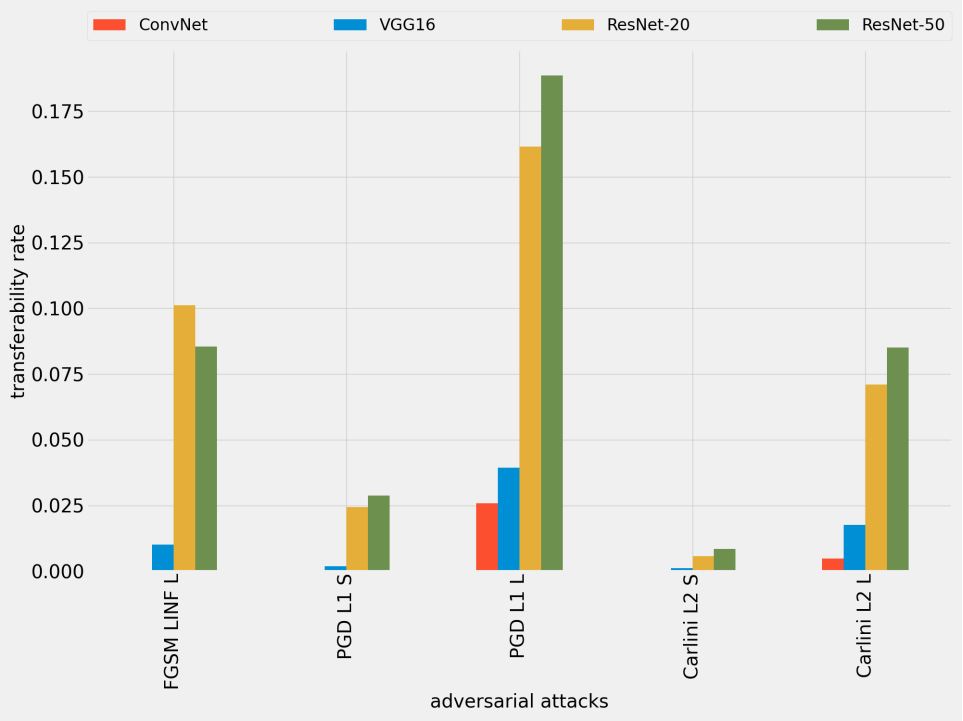


Figure 70.: Targeted for farthest class attacks and corresponding transferability transfer rate.

In Figure 70 and Table 35 we can observe the following regarding model architectures and the different attack methods:

- Most noticeable in this graph is the severe drop in transferability across all methods;
- As in previous experiments, the most robust model is ConvNet with attacks transferability never surpassing 3%. Most variants did not even to get a single successful transfer;
- The ResNet family all have similar behavior and were the most susceptible models to the targeted attacks;
- The attacks with highest targeted transferability rates were PGD L_1 for both S and L settings.

These targeted attack attained much worse transferability rates than the closest targeted attacks in Table 3.5.6. This is expected considering that theses classes are structurally the least similar, thus small perturbations applied to the image should have more difficulty to flip the classification to the expected targeted class. In particular, no successful attacks were found for FGSM for the S setting.

This difference in results between using the closest and farthest classes hints that the difference in SSIM values among the classes has an impact on the transferability rate for

targeted attacks, i.e., the farthest the original class is from the target class the harder it is to build a targeted sample that successfully transfer to another model.

far targeted atks	VGG16	ConvNet	R20	R50	TA	I%	C%	F%
FGSM L_∞ L	1.01	0.00	**10.13**	8.54	4.92	9.30	98.75	9.19
PGD L_1 S	0.19	0.00	2.44	**2.87**	1.38	98.02	99.05	97.09
PGD L_1 L	3.93	2.58	16.16	**18.87**	10.39	100.00	100.00	100.00
Carlini L_2 S	0.12	0.00	0.58	**0.84**	0.38	100.00	96.86	96.86
Carlini L_2 L	1.76	0.48	7.09	**8.50**	4.46	96.40	100.00	96.40

Table 35.: GTSRB targeted for farthest class transferability table. I% is the initial attacks successful on generator, C% the percentage of attacks successful on generator after conversion to valid image, and F% final percentage of successful attacks on generator that were used to attack the other networks, this is a pure white box attack. Column TA is the transferability average of an attack across all networks. In bold are the highest values per attack and in green and blue cells are the highest transferability values of all L and S attacks respectably, per model architecture

3.6 ATTACK ANALYSIS

In the non-targeted attack setting we saw that the best attack for the two different datasets was the PGD for the $SSIM = 0.80$, with Carlini L_2 being the second best followed by FGSM and Deepfool respectively, this tendency is present in both datasets.

In the target attack setting, although the targets are different in each dataset (CIFAR-10 is true class + 1; GTSRB closest and farthest classes) the same behavior was seen across all experiments, with PGD being consistently the most successful attack in both variants. The relative order is mostly the same as found in the non-targeted experiments.

Another interesting point is that overall the decrease in transferability in GTSRB relative to CIFAR-10. This can be observed even when considering the closest target class in GTSRB.

Although the selection of the closest class in GTSRB is a more favourable targeted attack scenario than target class selection for CIFAR ($true_cls + 1$), it is interesting to observe that the ResNet family has a higher transferability rate in the CIFAR-10 case.

This increase could be caused by:

- The images from GTSRB have a higher structural similarity between them when compared to CIFAR-10. This implies that the models must pay more attention to small details in the images, which can make them more resistant to attacks. Hence, to craft an attack, more distortion maybe required for GTSRB;
- The useful content for image classification in GTSRB is concentrated in the central area of the image, therefore, distortions in the background area should have little impact on the classification;

- Models trained for GTSRB have a much higher accuracy than the CIFAR-10 models, which could help to prevent transferability.

Note, however, that further study is required to validate any of these hypotheses.

Regarding the model architectures the ResNet family were consistently the models with highest transferability rates by a significant margin. This hints that architecture similarity may have an impact in transferability.

On the other hand the ConvNet was the architecture with lowest transferability rates in all the experiments by a significant margin, in some cases attaining 0%, this could be due to this architecture being the shallowest.

The hypothesis that there is a relationship between shallowness and transferability rate, is consistent with the results, although, as is, it requires further study to check if there is a significant correlation.

4

DEFENSE EVALUATION

In this chapter we will evaluate two types of defenses: Adversarial Training and Defensive Distillation. The test methodology is similar to the one used in Chapter 3. Both CIFAR-10 and GTSRB datasets were used. The attack methods are the ones used in Section 3.5, namely: FGSM, PGD, Deepfool and Carlini.

Initially, the settings for each of the defenses are defined for each defense (Section 4.1 and Section 4.2).

Afterwards, reports on the performance of the defenses are presented in Section 4.4 for CIFAR-10 and Section 4.5 for GTSRB.

The chapter ends with concluding remarks based on the test results, presented in Section 4.6.

4.1 ADVERSARIAL TRAINING

One defense against adversarial attacks is Adversarial Training introduced in Goodfellow et al. (2015), see Section 2.2.1. Adversarial Training consists in training the model with adversarial samples added to the original the training set. This defense methodology aims to make the model insensitive to perturbations present in the adversarial samples. This allows the network to be more resistant to external adversarial attacks (attacks with different algorithms and/or magnitude of perturbation).

Adversarial training can be realized in two ways:

- Prior to training, generate adversarial samples for each sample of the training set, and during training mix these samples with the original training set;
- Dynamically, in each training step, generate new adversarial samples from original training samples and mixed them with the original samples.

The first version has a major drawback: the adversarial samples are constant from start to the end of training. On the other hand, in the second version adversarial samples are evolving alongside training. As the weights of the model change, the adversarial samples

crafted by the model will change too. This is a more effective train method because the model is trained under different perturbations per training step, making the model more robust and insensitive to a higher range of perturbations. The downside of this version it is that is computational more expensive as it requires crafting new adversarial samples at each training step.

For this experiment we opted to train the model using adversarial samples generated dynamically in each training step. Specifically for each training step there is a 20% probability of all images from the batch to be adversarial samples, otherwise all images from the batch are unchanged. We chose this split of 20% heuristically. We trained 5 models for each norm (L_1, L_2 and L_∞) with adversarial samples generated with non targeted PGD algorithm. Adversarial samples are generated with ϵ sampled for each batch randomly from a uniform distribution that lies between 0 and the corresponding norm ϵ value for $SSIM = 0.80$ (this values were found in Section 3.4.6 and Section 3.5.5). All parameters can be found in Table 36.

	PGD attack					
dataset	**CIFAR-10**			**GTSRB**		
norm	L_∞	L_1	L_2	L_∞	L_1	L_2
eps	$[0, 0.45]$	$[0, 840]$	$[0, 23]$	$[0, 0.13]$	$[0, 240]$	$[0, 8]$
eps per iter	$\epsilon/5$					
iterations	10					

Table 36.: PGD parameters used in Adversarial Training

We used the ResNet-50 architecture for the adversarial training. The training hyperparameters, data-augmentation and preprocessing are as described in Chapter 3.

4.2 Defensive Distillation

Defensive Distillation (Section 2.2.2) is an alternative to Adversarial Training, being first proposed in Papernot et al. (2016). This method aims to smooth the decision boundary of the model so that a larger perturbation will be needed for an adversarial attack to succeed. Commonly, the aim of an adversarial attack is to find the smallest perturbation to make the model misclassify. By smoothing the decision boundary this minimal perturbation must increase to achieve the misclassification.

This defense consists in initially training a model with hard labels (one hot encode categorical labels) using a softmax function with temperature T as can be seen in Equation 26

$$F(X) = \left[\frac{e^{z_i(X)/T}}{\sum_{l=0}^{N-1} e^{z_l(X)/T}} \right]_{i \in 0..N-1} \tag{26}$$

For this experiment we used the ResNet-50 architecture, with temperature $T = 100$ (Papernot et al. (2016), Carlini and Wagner (2017)). Five teacher networks were trained and the teacher with the median accuracy in the test set was picked to teach five students networks. This was done for CIFAR-10 and GTSRB datasets, the training procedures are identical to those described in Chapter 3.

4.3 DEFENSES EXPERIMENTS SETTINGS

For this experiment we trained 5 defensive distilled models and 5 adversarial trained models for each of the 3 norms (L_∞, L_1, L_2) plus the 4 ResNet-50 already used in the Section 3.4.8 and Section 3.5.6. In total, for both CIFAR-10 and GTSRB tests, we used 24 models to be attacked, and used the same Resnet-50 generator used in Section 3.4.8 and Section 3.5.6 to craft the adversarial attacks.

To analyze the effect of different norms on adversarial trained models and defensive distillation we attacked with all three norms (L_1,L_2 and L_∞) for FGSM and PGD plus all other attacks used in Section 3.5.6.

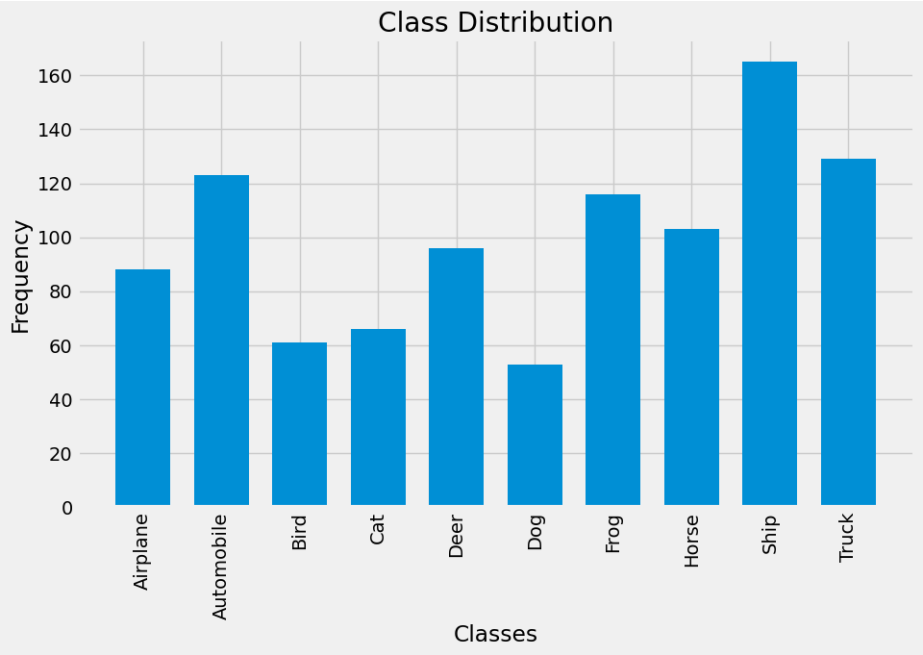


Figure 71.: Histogram of CIFAR-10 samples used to craft adversarial attacks.

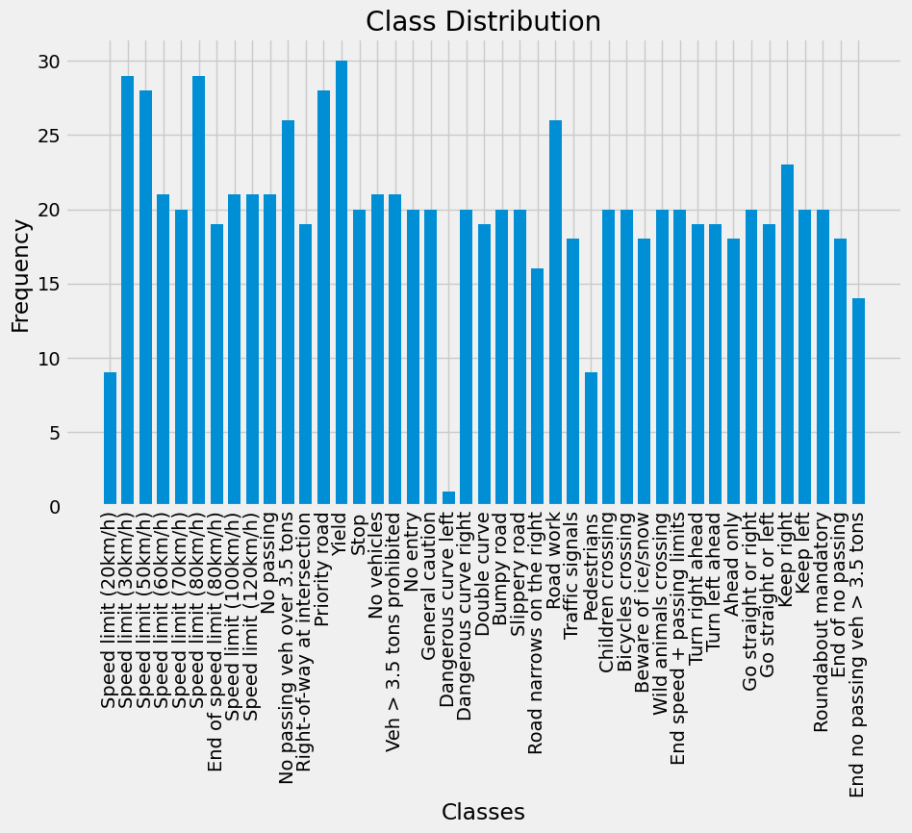


Figure 72.: Histogram of GTSRB samples used to craft adversarial attacks.

Following the same methodology in Section 3.4.8 and Section 3.5.6 we generated, across all classes, 1000 samples for CIFAR-10 and 860 samples for GTSRB datasets. Due to the large number of models with varying test accuracy it was not possible to get an equally balanced sample distribution per class well classified by all models as in Section 3.4.8 and Section 3.5.6. Figures Figure 71 and Figure 72 show the sample class histogram for each dataset.

4.4 Adversarial Defenses experiments on CIFAR-10

Before we start the defense evaluation per se, we tested how the trained models fared in the test set of CIFAR-10. As can be seen in Table 37 the defended models present a significant decrease in accuracy in the test set. This is particularly critical for Adversarial Trained models, where the drop in accuracy is between 6% and 10%.

Regardless of the level of success obtained with the defensive methods, it is clear that will come with a cost in accuracy in regular test data. In this section we will evaluate this level of defense.

	R50	**R50 distil**	**R50 adv L_∞**	**R50 adv L_1**	**R50 adv L_2**
CIFAR-10	91.52 ± 0.13	88.07 ± 0.18	81.77 ± 4.72	84.19 ± 3.45	84.93 ± 2.37

Table 37.: Defensive models and normal ResNet-50 average accuracies on CIFAR-10 test set and corresponding standard deviation

For this experiment we used the same methodology as in Section 3.4. We gathered 1000 samples correctly predicted by all the networks (including 5 ResNet-50 with Defensive Distillation and 5 ResNet-50 with Adversarial Training for each of the 3 norms). Hence, this is a different set of samples from the one used in Section 3.4. With these samples we used the model generator to craft the adversarial samples using the same parameters found in Section 3.4 and only kept the adversarial samples that misclassified the generator to attack the other models.

This experiment has 2 setups:

- non-targeted environment: attack the defensive models with non-targeted adversarial samples;
- target environment: attack the defensive models with targeted adversarial samples aiming to make the models misclassify the samples as a specific class.

4.4.1 *Non-targeted environment*

In this setup we compare transferability of the non-targeted adversarial samples against the defensive models. In this comparison we used ResNet-50 models from Section 3.4 plus 20 new defensive models (considering 2 defensive methods and three norm variations in adversarial training).

As in Section 3.4, we picked only the samples that were misclassified by the generator model when converted to valid images ($[0, 1]$ range). Table 38 presents the percentage of samples used for each combination of attacking method/norm/SSIM. Theses samples were then fed to all the defensively trained models in order to evaluate the transferability rate.

Figure 73 shows the different attacks and their transferability rate between models. The percentage of misclassified samples per attack can be seen in Table 38.

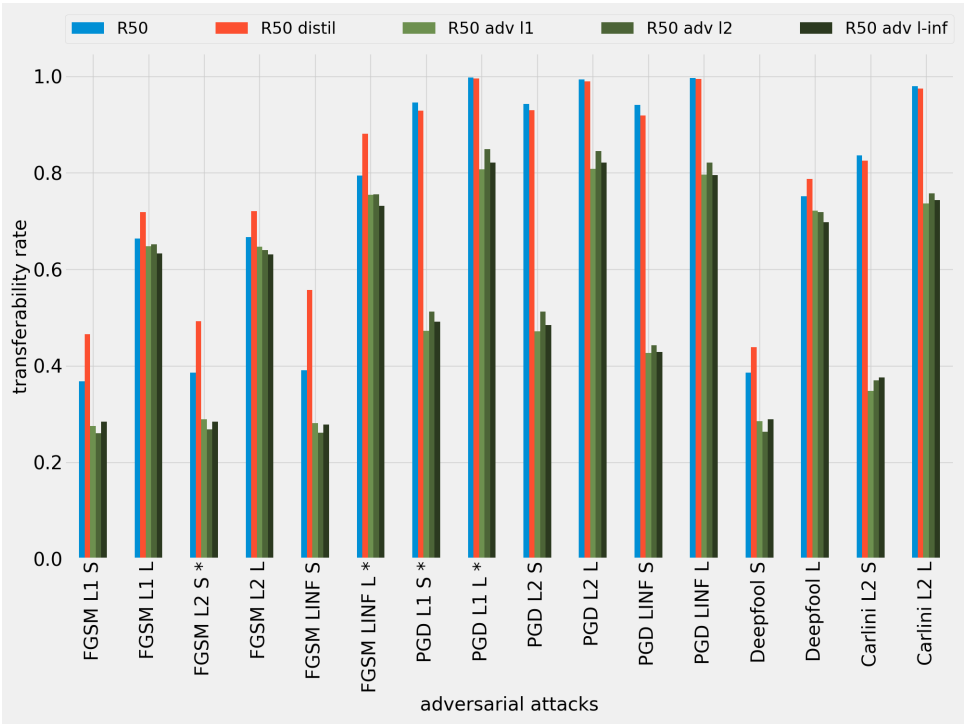


Figure 73.: CIFAR-10 non-targeted attacks and corresponding transferability transfer rate. Attacks with * are those with norms that attained best transferability rates on Section 3.4

From Figure 73 and Table 38 we can observe the following regarding the defensive methods:

- Defensive distillation shows a higher transferability rate than the original ResNet-50 for a significant number of attacks. This is true considering the attacks crafted with FGSM and Deepfool where the performance of the distilled networks is significantly worse than all the other tested models;
- Adversarial training is more robust in these tests, with a significant decrease in transferability rate, across all attacks, when considering the S setting. For the L setting the defense is not as effective, having a higher robustness gain when compared to the non-defensive model;
- Although PGD was used to train the adversarial models, as an attacking option it remains the stronger attacking method.

Considering the norms used in the adversarial training, there does not seem to be a significant difference regarding the norms.

Also regarding the norms, but considering the attack methods, FGSM and PGD, there is also no noticeable difference apart from a slight advantage FGSM L_∞ L.

Comparing the transferability rates between the ResNet-50 and the defensive models, it is clear that, although there is a reduction when using Adversarial Training, none of the methods provides a true defense in the context of this experiment.

Nevertheless, apart from FGSM (in all settings) and Deepfool (L), all other attacking methods suffer substantially in their transferability rate when confronted with Adversarial Trained models. Naturally, this increase in robustness is more noticeable when considering the S setting.

On the other hand, there is a significant trade-off regarding the accuracy obtained in legitimate samples as shown in Table 37.

nt atks	R50	R50 distil	R50 adv l_1	R50 adv l_2	R50 adv l_∞	I%	C%	F%
FGSM L_1 S	36.85	**46.53**	27.53	26.03	28.48	48.3	99.59	48.1
FGSM L_1 L	66.42	**71.91**	64.79	65.19	63.27	54.5	98.9	53.9
FGSM L_2 S *	38.57	**49.25**	28.89	26.82	28.48	38.7	100.0	38.7
FGSM L_2 L	66.71	**72.07**	64.66	63.99	63.08	41.8	99.52	41.6
FGSM L_∞ S	39.09	**55.72**	28.17	26.13	27.8	71.9	99.44	71.5
FGSM L_∞ L *	79.47	**88.15**	75.43	75.53	73.17	81.1	99.26	80.5
PGD L_1 S *	**94.57**	92.86	47.28	51.27	49.15	90.7	100.0	90.7
PGD L_1 L *	**99.73**	99.51	80.75	84.95	82.13	93.7	100.0	93.7
PGD L_2 S	**94.28**	93.02	47.14	51.27	48.47	76.5	100.0	76.5
PGD L_2 L	**99.35**	98.98	80.81	84.5	82.16	84.4	100.0	84.4
PGD L_∞ S	**94.1**	91.9	42.64	44.3	42.84	100.0	100.0	100.0
PGD L_∞ L	**99.62**	99.48	79.68	82.14	79.5	100.0	100.0	100.0
Deepfool S	38.56	**43.91**	28.53	26.31	28.94	95.8	99.9	95.7
Deepfool L	75.17	**78.75**	72.14	71.88	69.73	87.3	99.08	86.5
Carlini L_2 S	**83.62**	82.52	34.82	37.02	37.6	100.0	100.0	100.0
Carlini L_2 L	**98.0**	97.44	73.64	75.72	74.38	100.0	100.0	100.0

Table 38.: CIFAR-10 non-targeted transferability table. I% is the initial percentage of attacks successful on generator, C% is the percentage of successful attacks on generator after conversion to valid image, and F% is the final percentage of successful attacks on generator that were used to attack the other networks, this is a pure white box attack. In bold are the highest values per attack and in green and blue cells are the highest transferability values of all L and S attacks respectably per, model architecture. Attacks with * are those with norms that attained best transferability rates in Section 3.4

4.4.2 *Targeted environment*

In this setup we compare transferability of the targeted adversarial samples against the defensive models. We test the transferability of misclassification for the target class. All the attack samples generated used the same norms and parameters found in Section 3.4.

Recall that we only consider a successful transfer if the sample gets classified with the target class, i.e. a sample being misclassified in any other class is not considered a successful transferred attack.

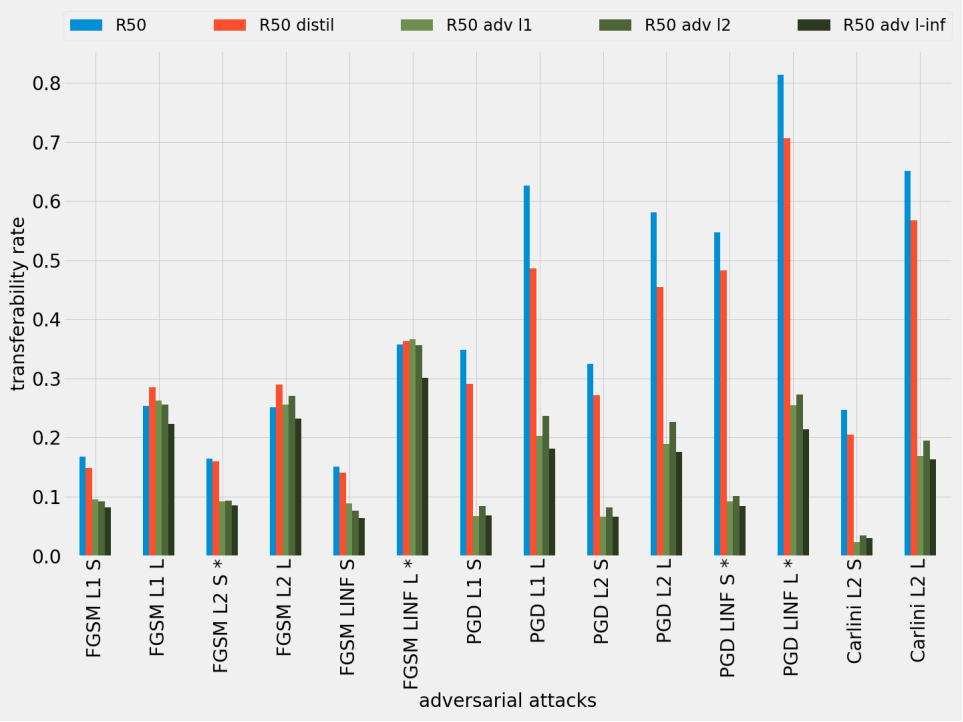


Figure 74.: CIFAR-10 targeted attacks and corresponding transferability transfer rate. Attacks with * are those with norms that attained best transferability rates on Section 3.4

From Figure 74 and Table 39 it is clear that the trend observed in the non-targeted attacks can also be observed in the targeted attacks, namely:

- Comparing to the non-targeted environment, the distilled models manage to achieve some success with all methods except FGSM. However, Adversarial Training is still the most robust method by a comfortable margin;
- Considering Defensive Distillation, the attack with the highest transferability rate is PGD L_∞ for both in S and L. For Adversarial Training models the highest transferability rate is obtained with FGSM L_∞ L. All defensive methods performed poorly in all FGSM attacks specially on L settings, in some instances performed worse than the normal models.
- Adversarial trained models performed very well against PGD and Carlini L_2. Although the adversarial trained models where trained using different L norms it seems that didn't help to improve their performance against the same L norm attacks, all ad-

versarial trained models performed similarly, this behavior was also seen on the non target setting.

Although Adversarial Training achieves very robust results, reducing very significantly the transferability rate (apart from attacks with FGSM), the trade-off in accuracy is still worth taking into consideration when considering deploying these defenses.

Regarding the attacking methods, FGSM is the most effective in the targeted mode, as opposed to the non-targeted version, where it has the lowest transferability rate in almost every setting.

targeted atks	R50	R50 distil	R50 adv l_1	R50 adv l_2	R50 adv l_∞	I%	C%	F%
FGSM L_1 S	**16.82**	14.88	9.57	9.19	8.15	21.5	98.14	21.1
FGSM L_1 L	25.36	**28.57**	26.29	25.57	22.29	15.3	91.5	14.0
FGSM L_2 S *	**16.47**	15.98	9.25	9.35	8.5	21.7	98.62	21.4
FGSM L_2 L	25.18	**28.92**	25.61	27.05	23.17	15.0	92.67	13.9
FGSM L_∞ S	**15.06**	14.06	8.92	7.63	6.43	25.3	98.42	24.9
FGSM L_∞ L *	35.78	36.33	**36.7**	35.6	30.09	11.2	97.32	10.9
PGD L_1 S	**34.88**	29.04	6.74	8.38	6.78	100.0	100.0	100.0
PGD L_1 L	**62.62**	48.66	20.28	23.62	18.08	100.0	100.0	100.0
PGD L_2 S	**32.42**	27.18	6.58	8.2	6.64	100.0	100.0	100.0
PGD L_2 L	**58.12**	45.44	18.96	22.64	17.54	100.0	100.0	100.0
PGD L_∞ S *	**54.7**	48.28	9.18	10.14	8.44	100.0	100.0	100.0
PGD L_∞ L *	**81.37**	70.68	25.44	27.22	21.42	100.0	100.0	100.0
Carlini L_2 S	**24.63**	20.54	2.36	3.4	3.0	100.0	100.0	100.0
Carlini L_2 L	**65.11**	56.77	16.92	19.45	16.35	95.5	99.79	95.3

Table 39.: CIFAR-10 targeted transferability table. I% is the initial percentage of attacks successful on generator, C% the percentage of attacks successful on generator after conversion to valid image, and F% final percentage of successful attacks on generator that were used to attack the other networks, this is a pure white box attack. In bold are the highest values per attack, and in green and blue cells are the highest transferability values of all L and S attacks respectably per model architecture. Attacks with * are those with norms that attained best transferability rates on Section 3.4

4.5 ADVERSARIAL DEFENSES EXPERIMENTS ON GTSRB

Similarly, to what was done for CIFAR-10, we tested how the trained models fared in the test set of GTSRB.

Although not as severe as in CIFAR-10, the level of defense will come with a cost in accuracy in regular test data, see Table 40. The relative ranking in accuracy remains the same as in CIFAR-10 (Table 37)), with Defensive Distillation models performing better on non-adversarial samples than Adversarial Trained models.

	R50	**R50 distil**	**R50 adv L_∞**	**R50 adv L_1**	**R50 adv L_2**
GTSRB	98.86 ± 0.31	98.76 ± 0.21	97.18 ± 0.28	94.93 ± 1.64	93.98 ± 1.11

Table 40.: Defensive models and normal ResNet-50 average accuracies on test set and corresponding standard deviation for GTSRB

The purpose of this section is to evaluate this level of defense in GTSRB. For this experiment we used the same methodology as in Section 3.5. After collecting the samples well predicted by all the networks (generator, 4 ResNet-50, 5 defensive distilled and 5 adversarial trained models for each norm), we used the generator to craft the adversarial samples, and kept only the adversarial samples that misclassified the generator to attack the other models. This experiment has three setups:

- non-targeted environment: attack the defensive models with non-targeted adversarial samples, misclassified by the generator;
- closest target environment: attack the defensive models with targeted adversarial samples aiming to make the models misclassify the samples as the closest class (see Section 3.5);
- farthest target environment: similar to the closest target environment, but the target class is the farthest class.

4.5.1 *Non-targeted environment*

In this setup we compare transferability of the non-targeted adversarial samples against the defensive models.

As in Section 3.5 we picked only the samples that were misclassified by the generator model when converted into valid images ($[0, 1]$ range). Theses samples were then predicted by the other models to calculate the transferability rate.

Figure 75 shows the different attacks and their transferability rate between models. The percentage of misclassified samples per attack can be seen in Table 41.

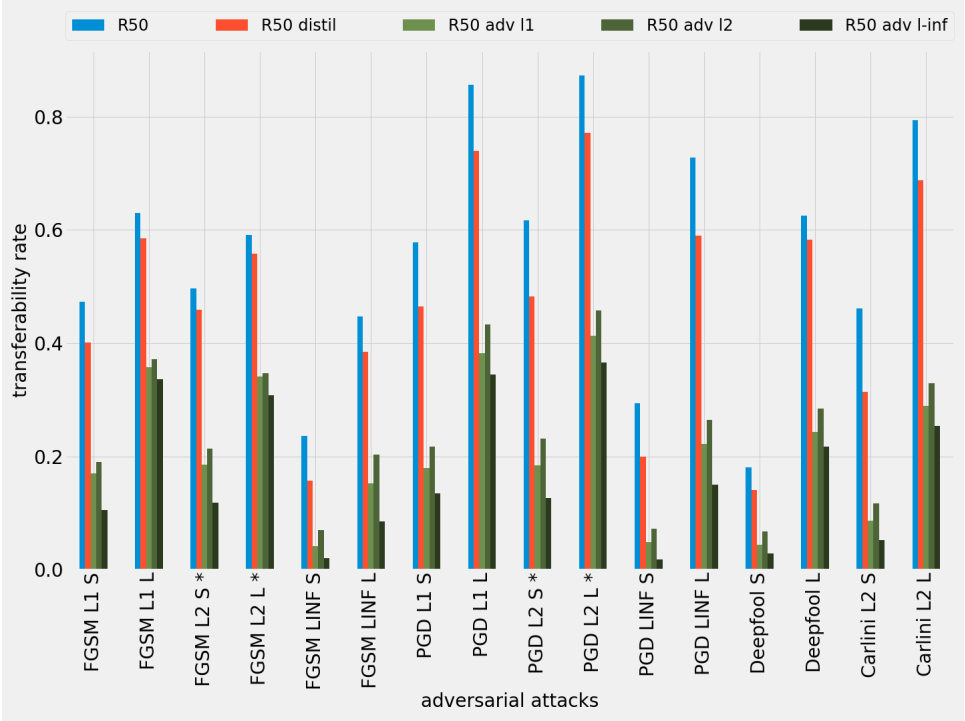


Figure 75.: GTSRB non-targeted attacks and corresponding transferability rate on normal and defensive trained models. Attacks with * are those with norms that attained best transferability rates on Section 3.5

From Figure 75 and Table 41 we can observe the following regarding the defenses:

- Although Distillation performed consistently better compared with the non defensive models in this dataset, Adversarial Training still outperforms it significantly;
- Regarding the norms used in the defensive methods, there is now a clearer distinction, with L_∞ appearing to be more robust. Notably, this norm is also the one that leads to a lower decrease in accuracy, making it an easy choice for training adversarial models with PGD.
- On the other hand L_2 trained models performed consistently worse, this behavior was also seen in Section 4.4.1 although at a small degree in most attacks.
- PGD is once again the most effective against adversarial trained models. Although the adversarial trained models were trained using adversarials crafted with PGD, the attacks with highest transferability rate were still PGD, despite adversarial trained models reduced the transferability by more than half in most cases when compared to non defensive models.

Comparing the defense performance with GTSRB against CIFAR-10 it is clearly that defenses performed much better with far more significant reductions in the transferability

rate. This could be due to the fact that the dataset is more homogeneous, with a much smaller intra-class diversity, and also lower inter-class distinction. Further research with a larger range of datasets is required to confirm, or deny, this hypothesis;

nt atks	R50	R50 distil	R50 adv l_1	R50 adv l_2	R50 adv l_∞	I%	C%	F%
FGSM L_1 S	**47.24**	40.13	17.01	18.96	10.52	18.02	99.35	17.91
FGSM L_1 L	**63.01**	58.44	35.76	37.13	33.58	38.02	98.17	37.33
FGSM L_2 S *	**49.71**	45.88	18.59	21.41	11.76	10.0	98.84	9.88
FGSM L_2 L *	**59.06**	55.79	34.15	34.74	30.76	20.0	99.42	19.88
FGSM L_∞ S	**23.65**	15.68	4.19	7.03	2.03	17.56	98.01	17.21
FGSM L_∞ L	**44.66**	38.5	15.24	20.26	8.46	53.37	98.91	52.79
PGD L_1 S	**57.74**	46.43	17.96	21.71	13.44	70.7	99.84	70.58
PGD L_1 L	**85.65**	73.9	38.25	43.31	34.4	83.49	100.0	83.49
PGD L_2 S *	**61.68**	48.19	18.41	23.13	12.64	42.56	99.45	42.33
PGD L_2 L *	**87.26**	77.12	41.23	45.77	36.62	60.47	100.0	60.47
PGD L_∞ S	**29.36**	19.92	4.84	7.2	1.82	91.4	97.84	89.42
PGD L_∞ L	**72.76**	58.95	22.23	26.4	15.05	100.0	100.0	100.0
Deepfool S	**18.11**	14.11	4.35	6.71	2.87	100.0	99.53	99.53
Deepfool L	**62.47**	58.24	24.35	28.46	21.66	98.26	99.64	97.91
Carlini L_2 S	**46.1**	31.44	8.67	11.67	5.21	100.0	100.0	100.0
Carlini L_2 L	**79.39**	68.7	28.95	32.88	25.35	100.0	100.0	100.0

Table 41.: GTSRB non-targeted transferability table. I% is the initial attacks successful on generator, C% the percentage of attacks successful on generator after conversion to valid image, and F% final percentage of successful attacks on generator that were used to attack the other networks, this is a pure white box attack. In bold are the highest values per attack and in green and blue cells are the highest transferability values of all L and S attacks respectably per, model architecture. Attacks with * are those with norms that attained best transferability rates on Section 3.5

.

4.5.2 *Targeted environment*

In this setup we compare transferability of the targeted adversarial samples against the defensive models. We test the transferability of misclassification for the target to closest class and farthest class. All the attack samples generated used the same norms and parameters found in Section 3.5. An attack is considered successful if and only if the model classifies the adversarials as the target class.

Closest classes target attack

In this setting we performed targeted attacks using the closest class, found using SSIM metric in Section 3.5, as the target class.

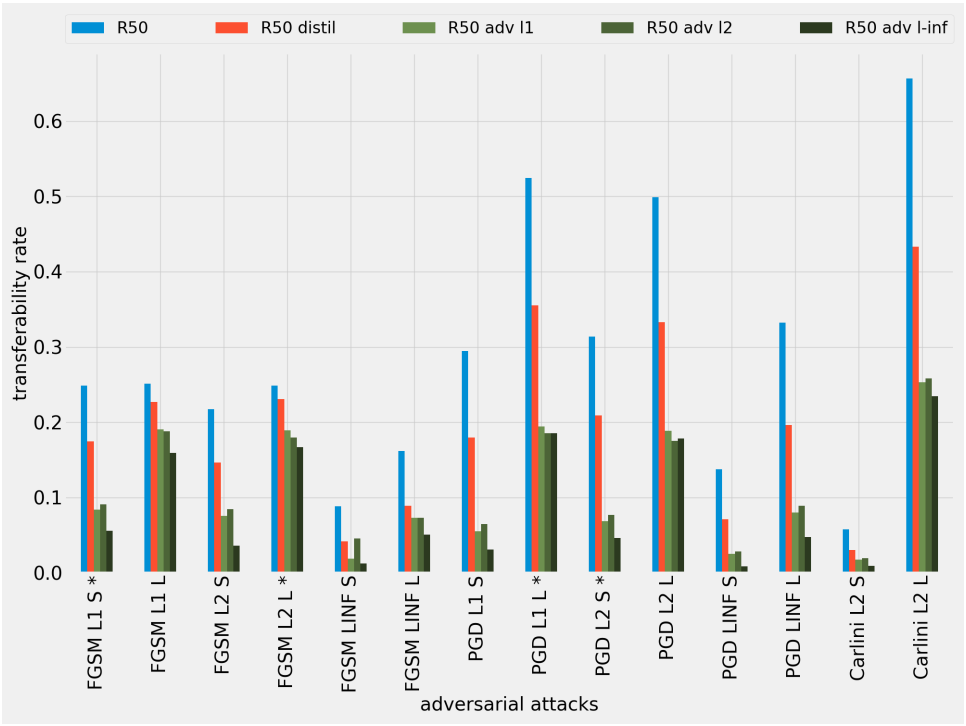


Figure 76.: GTSRB targeted to closest class attacks and corresponding transferability rate on normal and defensive trained models. Attacks with * are those with norms that attained best transferability rates on Section 3.5

From Figure 76 and Table 42 we can observe the following regarding the defenses:

- As opposed to the similar test on CIFAR-10, defensive distillation increased robustness against all attacks
- As in previous tests, adversarial training is the best defensive model method with highest robustness improvement against all attacks
- Defensive models performed poorly against FGSM L attacks, for L_1 and L_2 norms, while against others attack the gain in robustness was significant. This behaviour can also be observed in Section 4.4.2.
- Adversarial trained models gained significant robustness against PGD as expected due to the fact that these models were trained with adversarial crafted with this algorithm, this is consistent with the trend found in Section 4.5.1.
- A large variability can be observed the results obtained with Carlini, being the least successful attack in the S setting, while being the most successful attack in the L setting.

cls target atks	R50	R50 distil	R50 adv l_1	R50 adv l_2	R50 adv l	I%	C%	F%
FGSM L_1 S *	**24.83**	17.47	8.4	9.07	5.6	17.44	100.0	17.44
FGSM L_1 L	**25.11**	22.67	19.05	18.79	15.95	28.84	93.55	26.98
FGSM L_2 S	**21.74**	14.66	7.58	8.45	3.6	18.72	100.0	18.72
FGSM L_2 L *	**24.89**	23.06	18.95	17.99	16.68	27.91	95.42	26.63
FGSM L_∞ S	**8.85**	4.17	1.88	4.58	1.25	11.4	97.96	11.16
FGSM L_∞ L	**16.21**	8.9	7.29	7.29	5.08	29.07	94.4	27.44
PGD L_1 S	**29.48**	17.95	5.51	6.47	3.09	100.0	100.0	100.0
PGD L_1 L *	**52.44**	35.53	19.44	18.53	18.51	100.0	100.0	100.0
PGD L_2 S *	**31.37**	20.88	6.86	7.7	4.6	100.0	100.0	100.0
PGD L_2 L	**49.88**	33.3	18.86	17.53	17.86	100.0	100.0	100.0
PGD L_∞ S	**13.74**	7.1	2.52	2.84	0.88	96.74	98.2	95.0
PGD L_∞ L	**33.2**	19.63	7.98	8.91	4.74	100.0	100.0	100.0
Carlini $L2$ S	**5.76**	3.04	1.73	1.94	0.94	100.0	99.42	99.42
Carlini $L2$ L	**65.7**	43.31	25.34	25.79	23.46	30.93	100.0	30.93

Table 42.: GTSRB targeted to closest class transferability table. I% is the initial percentage of attacks successful on generator, C% the percentage of attacks successful on generator after conversion to valid image, and F% final percentage of successful attacks on generator that were used to attack the other networks, this is a pure white box attack. In bold are the highest values per attack, and in green and blue cells are the highest transferability values of all L and S attacks respectably per model architecture. Attacks with * are those with norms that attained best transferability rates on Section 3.5

Farthest classes target attack

In this setting we performed targeted attacks using the farthest class, found using SSIM metric in Section 3.5, as the target class.

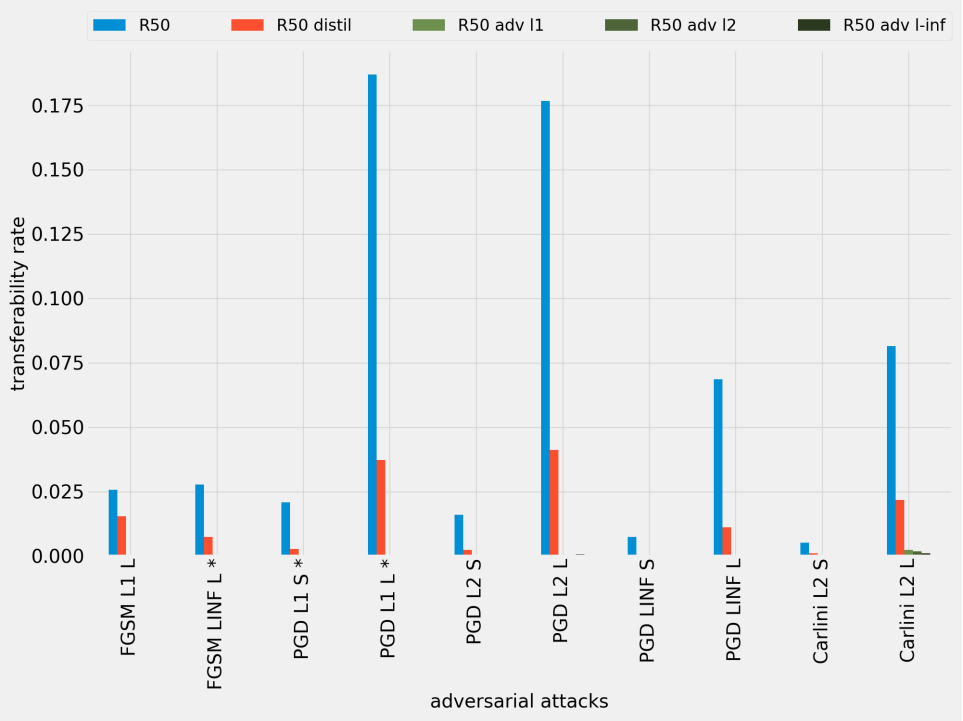


Figure 77.: GTSRB targeted to farthest class attacks and corresponding transferability rate on normal and defensive trained models. Attacks with * are those with norms that attained best transferability rates on Section 3.5

From Figure 77 and Table 43 we can observe the following regarding the defenses:

- Although the transferability rates on targeted for farthest class are very small compared with closest targeted attacks, the relative performance between models remains the same: ResNet-50 models are more vulnerable to the transferability, followed by the defensive distilled models and lastly the adversarial trained models;
- All attacks in the S version transferred poorly to other models never surpassing 2.5%. With Defensive Distillation transferability being always below 0.5% and 5% for all S and L attacks respectably. Adversarial trained models were almost completely impervious to all adversarials with highest transferability being 0.24% with Carlini L_2 L;

4.6 DEFENSE ANALYSIS

Results from the non-targeted experiments on CIFAR-10 and GTSRB datasets, show that Adversarial Training provides significant robustness against transferability attacks on both datasets. On the other hand, for some attacks, Defensive Distillation results in an increase

far target atks	R50	R50 distil	R50 adv l_1	R50 adv l_2	R50 adv l_∞	I%	C%	F%
FGSM L_1 L	**2.56**	1.54	0.00	0.00	0.00	4.88	92.86	4.53
FGSM L_∞ L *	**2.78**	0.74	0.00	0.00	0.00	6.4	98.18	6.28
PGD L_1 S *	**2.09**	0.26	0.00	0.00	0.00	98.72	98.82	97.56
PGD L_1 L *	**18.69**	3.72	0.05	0.00	0.05	100.0	100.0	100.0
PGD L_2 S	**1.6**	0.23	0.00	0.00	0.00	100.0	100.0	100.0
PGD L_2 L	**17.67**	4.12	0.05	0.05	0.07	100.0	100.0	100.0
PGD L_∞ S	**0.74**	0.03	0.00	0.00	0.00	88.95	96.86	86.16
PGD L_∞ L	**6.87**	1.12	0.00	0.00	0.00	100.0	99.88	99.88
Carlini L_2 S	**0.5**	0.1	0.02	0.00	0.00	100.0	97.91	97.91
Carlini L_2 L	**8.15**	2.17	0.24	0.17	0.1	97.33	100.0	97.33

Table 43.: GTSRB targeted to farthest class transferability table. I% is the initial percentage of attacks successful on generator, C% the percentage of attacks successful on generator after conversion to valid image, and F% final percentage of successful attacks on generator that were used to attack the other networks, this is a pure white box attack. In bold are the highest values per attack, and in green and blue cells are the highest transferability values of all L and S attacks respectably per model architecture. Attacks with * are those with norms that attained best transferability rates on Section 3.5

of transferability in CIFAR-10. While in GTSRB this does not occur, it still allows for significantly higher transferability rates than Adversarial Training.

Even though the adversarial samples were crafted with PGD, it is still the most successful attack method, which attests once again to its effectiveness for adversarial attacks.

The target environment experiments show a significant decrease in transferability as expected. When considering closest and farthest classes in GTSRB it becomes clear that the similarity of the classes, measured with SSIM, has a strong impact in transferability.

Regarding the level of perturbation (S and L) it is clear that the smallest level of perturbation leads to a higher robustness in both methods as expected.

Comparing the performance amongst datasets is harder as there are too many factors at stake. Both the inter-class and intra-class variability are significantly different, and the accuracies obtained in the test set are also substantially dissimilar.

In CIFAR-10 the accuracy obtained with all models is very low compared to the accuracy for the GTSRB test set.

CIFAR-10 classes have a very high intra-class variability, whereas in GTSRB the variability within each class is significantly lower. Furthermore, inter-class variability is also lower in the traffic sign dataset.

Both variations are related to the boundaries of the classes in the sample space, and the accuracies on the test set for each dataset also reflect this variability, with models trained on CIFAR-10 having a much lower accuracy than models trained in GTSRB.

In order to evaluate the impact of the accuracy of the trained models, we have performed another test on GTSRB. This time, the models were early stopped when an accuracy similar

to that obtained with CIFAR-10 was attained. This results in relatively poorly trained models considering the accuracy previously obtained in GTSRB. The goal is to evaluate how the accuracy of the model influences its transferability and its defense capabilities.

Figure 78 presents the transferability rates for non-targeted GTSRB where the adversarial samples are crafted with a high accuracy ResNet-50.

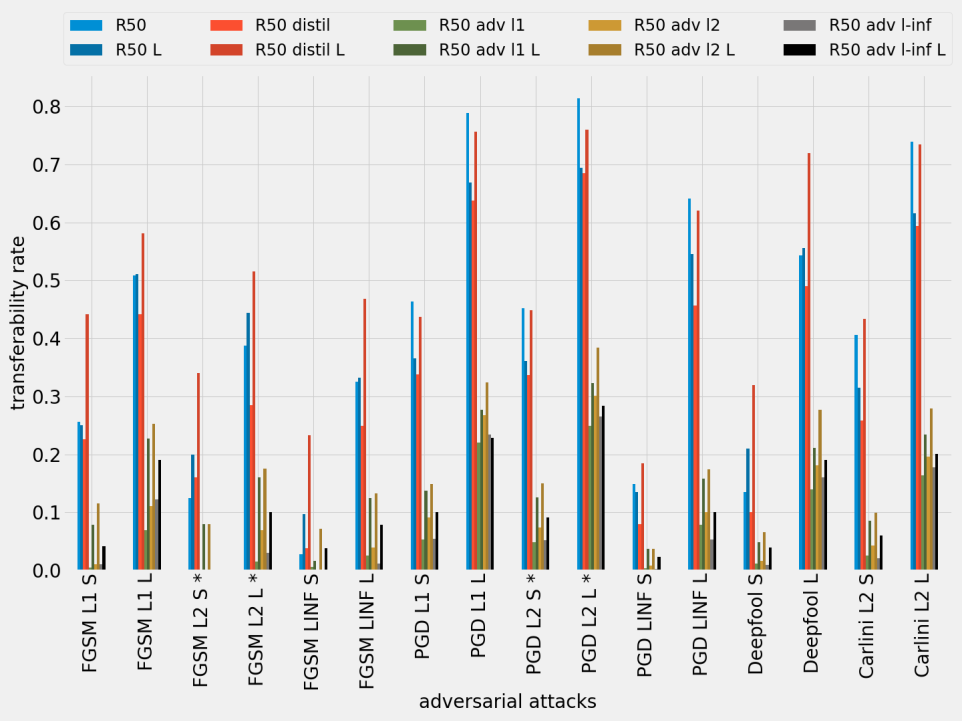


Figure 78.: GTSRB non targeted attacks generated by a ResNet-50 well trained, and corresponding transferability rate on normal and defensive trained models with high and low accuracies. Models with L letter in front of the architecture name are low accuracy models that imitate CIFAR-10 test accuracy. Attacks with * are those with norms that attained best transferability rates on Section 3.5

In this chart we can observe that the transferability in lower accuracy models is consistently higher than on well trained models.

Note that the set of adversarial samples is now different from the one used in Section 4.5.1, since all samples used to craft the adversarial attacks must be well classified by a larger set of models.

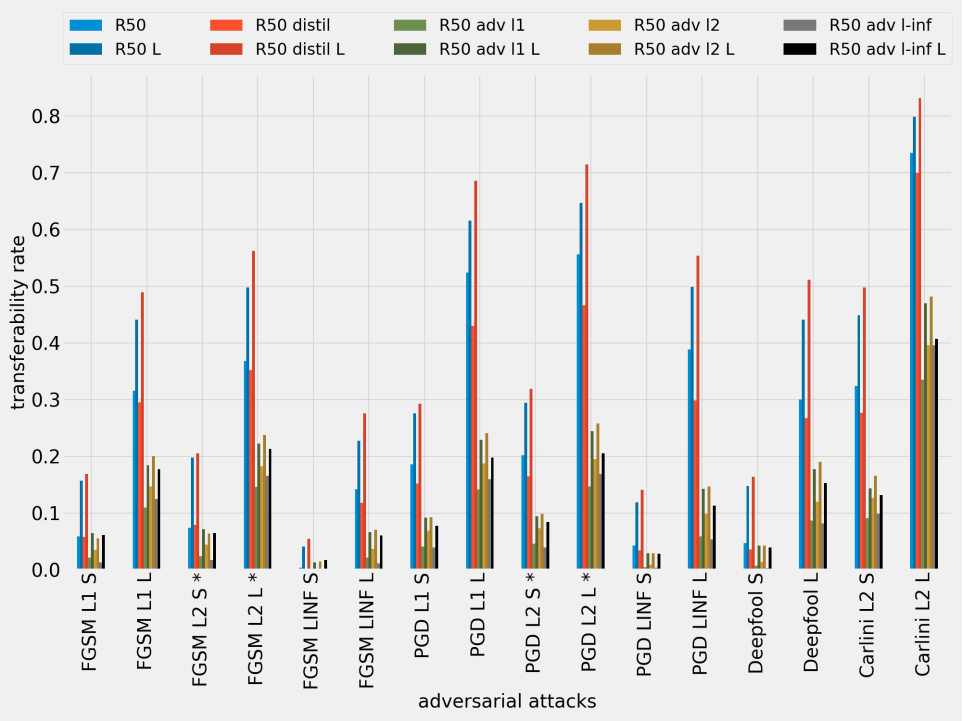


Figure 79.: GTSRB non targeted attacks generated by a ResNet-50 with low accuracy, and corresponding transferability rate on normal and defensive trained models with normal and low accuracies. Models with L letter in front of the architecture name are low accuracy models that imitate CIFAR-10 test accuracy. Attacks with * are those with norms that attained best transferability rates on Section 3.5

Figure 79 presents a similar experiment, but this time the adversarial samples were crafted in the low accuracy ResNet-50.

As in the previous chart we can also observe that the transferability in lower accuracy models is consistently higher than on well trained models.

These two experiments suggest that the higher the accuracy of the model the more robust it is.

Taking into account the accuracy of the model used to craft the adversarial samples we observe distinct behaviours for different attacks. Comparing Figure 78 to Figure 79 we can observe that while FGSM and Carlini show overall an increase in transferability, for PGD an increase of robustness for the defensive methods can be observed.

Hence, this suggests that the accuracy of the generator model used to craft the adversarial samples has an impact on the robustness of the defensive models. However, due to having different results for different attacking methods, the effect of such an impact is not universal.

Note that the set of adversarial samples used in each situation is different, which also makes it harder to establish a direct comparison.

5

CONCLUSION

Nowadays convolutional neural networks are present in all kinds of fields and markets, from facial recognition to unlocking mobile phones to autonomous driving cars. Although this technology is widespread, the discovery of adversarial examples came to prove that not only these systems are not entirely secure but the attacks that create misclassification can be undetectable to the human eye. This can lead to serious security breaches such as impersonating other people identities, to misclassification of traffic signs in autonomous driving cars. With the appearance of these problems it became a necessity to devise techniques that aimed to improve resistance against these adversarial attacks.

We focused our tests on transferability, i.e., the creation of an adversarial sample in a model that causes misclassification on another model. To test the relative performance of the attacks we used SSIM to quantify the level of perturbation. Therefore, we were able to test all the attacks with the same level of perturbation.

To evaluate transferability across a range of models we selected models from the same architecture family of the adversarial sample generator, as well as significantly different models. We also used two datasets with different features to evaluate how the dataset itself affects transferability.

Our results show that some attacks have high levels of transferability on non-targeted attacks, with PGD being the clear winner in all tests performed.

Considering targeted attacks, we see the transferability rates drop to much lower levels, as expected. For targeted attacks we also showed that using a sample closer to the target class as a base to generate the adversarial sample, based on SSIM, provides a higher transferability rate.

Regarding the architecture similarity of the tested models, our results hint that this may impact transferability.

Finally, considering the datasets, we noticed a direct correlation between the accuracy of the trained models and their robustness to transferability. However, since the composition of the datasets differs significantly we can't conclude that accuracy is the only relevant

factor. Intra and inter-class variability can also play a role, and further testing is required to evaluate its significance.

Regarding defensive methods, in our assessment we found that, on non-targeted environment, Adversarial Training improved significantly the robustness against transferability attacks on both datasets. Defensive Distillation on the other hand performed much worse on both targeted and non-targeted settings, and in some instances it even performed worse than normal trained models.

As expected, the defensive methods worked far better in the targeted tests, with SSIM playing an important role. The defense models provided a higher robustness when the sample used to craft the adversarial sample was from a class that is further away from the target class.

Concerning the datasets both inter-class and intra-class variability seem to affect the performance of defensive methods. The accuracy of the models also appears to influence the robustness of the methods.

5.1 Prospect for future work

While in this work we have done a significant number of experiments, there are a lot of open questions at the end. These questions emerged when we got the results of our tests and were able to analyse them. The present work can be seen as the beginning of a long journey into the adversarial world.

Avenues for future work can be derived from the analysis performed both on the attacks and defenses.

Concerning defenses, there are still questions pending regarding the transferability effectiveness of an attack:

- To what extent is architecture similarity relevant?
- Is shallowness pertinent?
- Does intra and/or inter-class variability have an impact?

Regarding defense robustness the following questions also require further work:

- How much does the level of perturbation in Adversarial Training affect its robustness and accuracy?
- How does intra and/or inter-class variability impact on the robustness of the defenses?
- how does the accuracy of the models affect the transferability and defense robustness?

Finally, there are other defensive and attack methods, see Chakraborty et al. (2018), that were not explored in this thesis due to time and hardware constraints. It would be interesting to evaluate them in the context of the questions above.

BIBLIOGRAPHY

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. *arXiv:1608.04644v2 22 Mar 2017*, 2017.

Anirban Chakraborty, Manaar Aalm, VishalL Dey, Anupam Chattopadhay, and Debdeep Mukhopadhay. Adversarial attacks and defences: A survey. *arXiv:1810.00069v1 [cs.LG] 28 Sep 2018*, 2018.

Francois Chollet. Trains a resnet on the cifar10 dataset. URL https://keras.io/zh/examples/cifar10_resnet/.

Francois Chollet. *Deep Learning with Python*. Manning Publications, 2017. ISBN 978-1-61729-443-3.

Yonatan Geifman. cifar10vgg. https://github.com/geifmany/cifar-vgg/blob/master/cifar10vgg.py, 2018.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

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

GTSRB. German traffic sign repository benchmark, 2010. URL https://benchmark.ini.rub.de/.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv:1512.03385v1 10 Dec 2015*, 2015.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531v1 [stat.ML] 9 Mar 2015*, 2015.

Andrew Ilyas, Shibani Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry. Adversarial examples are not bugs, they are features. In *NeurIPS*, 2019.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

A. Kurakin, Ian J. Goodfellow, and S. Bengio. Adversarial machine learning at scale. *ArXiv*, abs/1611.01236, 2017a.

A. Kurakin, Ian J. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *ArXiv*, abs/1607.02533, 2017b.

Fei-Fei Li, Ranjay Krishna, and Danfei Xu. Cs231n: Convolutional neural networks for visual recognition. URL http://cs231n.stanford.edu/.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv:1706.06083v4 19 Jun 2017*, 2017.

S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, 2016. doi: 10.1109/CVPR.2016.282.

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. URL https://arxiv.org/pdf/1807.01069.

Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learningin adversarial settings. *1st IEEE European Symposium on Security & Privacy, IEEE 2016*, 2015.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *arXiv:1511.04508v2 [cs.CR] 14 Mar 2016*, 2016.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning, 2017.

Jaewoo Song. one-pixel-attack-keras. https://github.com/Hyperparticle/one-pixel-attack-keras/blob/master/1_one-pixel-attack-cifar10.ipynb, 2019.

Jiawei Su, Danilo Vasconcellos Vargas, , and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *arXiv:1710.08864v7 [cs.LG] 17 Oct 2019*, 2019.

Christian Szegedy, W. Zaremba, Ilya Sutskever, Joan Bruna, D. Erhan, Ian J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2014.

Antonio Torralba, Rob Fergus, and Bill Freeman. 80 million tiny images, 2020. URL http://groups.csail.mit.edu/vision/TinyImages/.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 13, NO. 4*, 2004.

Rey Wiyatno and Anqi Xu. Maximal jacobian-based saliency map attack. *arXiv:1808.07945v1 23 Aug 2018*, 2018.

Youlixx. tf-jsma-nt.py. https://github.com/probabilistic-jsmas/probabilistic-jsmas/blob/master/attacks/tf_jsma_nt.py, 2020.

Yinghua Zhang, Yangqiu Song, Jian Liang, Kun Bai, and Qiang Yang. Two sides of the same coin: White-box and black-box attacks. *ArXiv*, 2008.11089v1, 2020. URL https://arxiv.org/abs/2008.11089.

A

ATTACK EVALUATION GRAPHS

In this section we present the graphics that show the search parameters for all attacks and their norms for both CIFAR-10 and GTSRB, and their corresponding settings: targeted and non-targeted attacks.

For the sake of completeness, even graphics included in the main text are repeated in here for convenience.

A.1 CIFAR-10 SEARCH PARAMETERS

A.1.1 *FGSM*

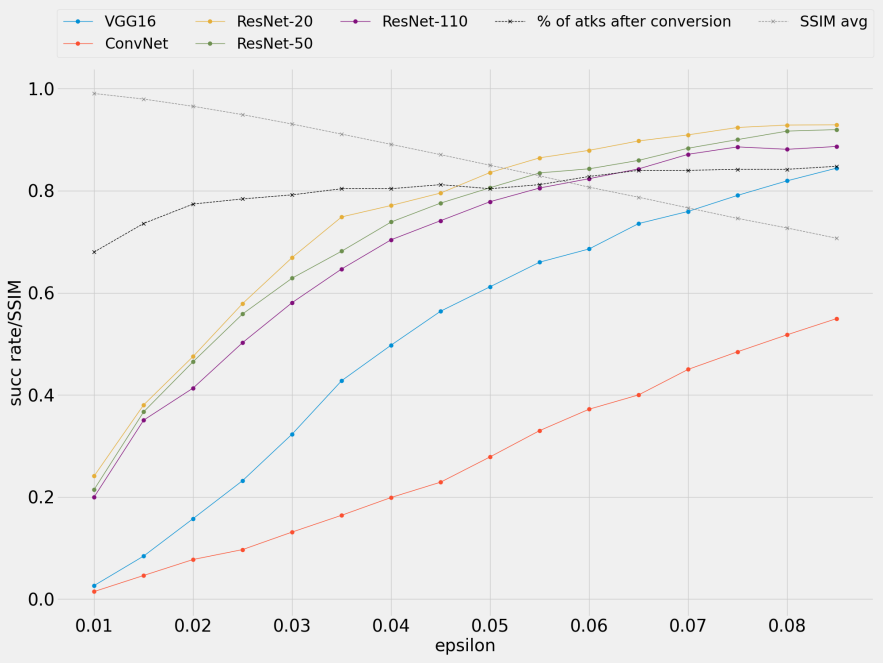


Figure 80.: Success attack rate of non-targeted FGSM L_∞ norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

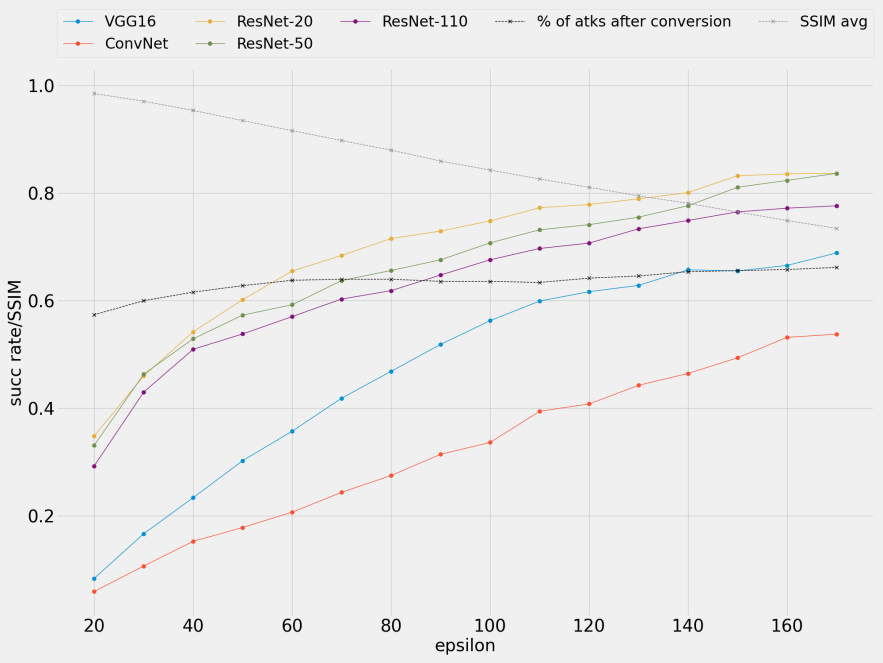


Figure 81.: Success attack rate of non-targeted FGSM L_1 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

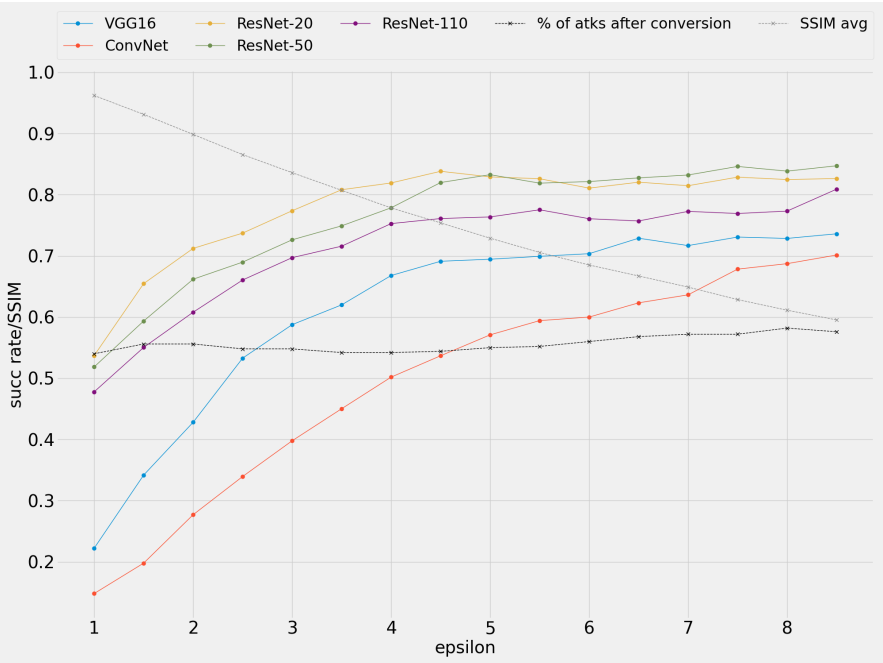


Figure 82.: Success attack rate of non-targeted FGSM L_2 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

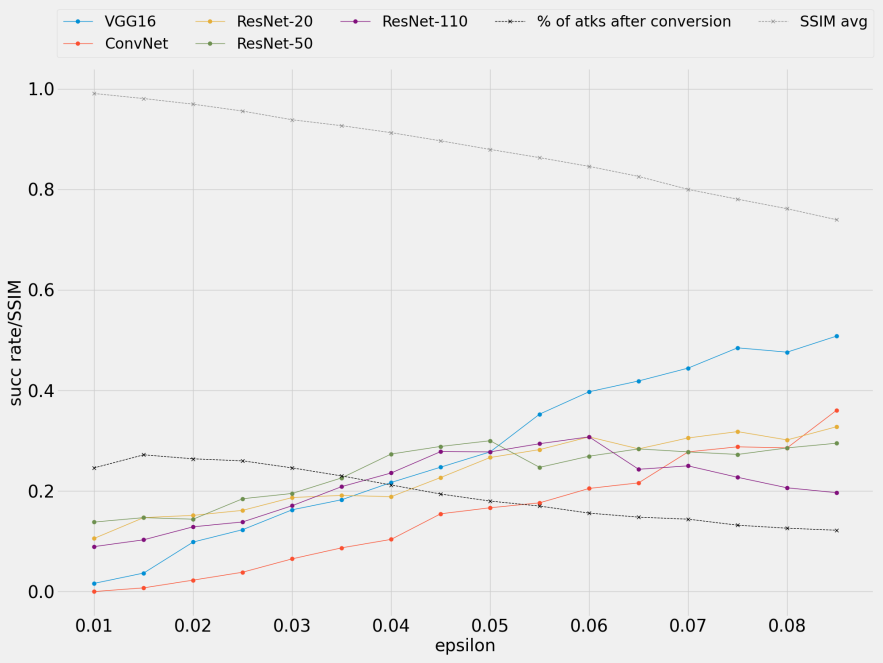


Figure 83.: Success attack rate of targeted FGSM L_∞ norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

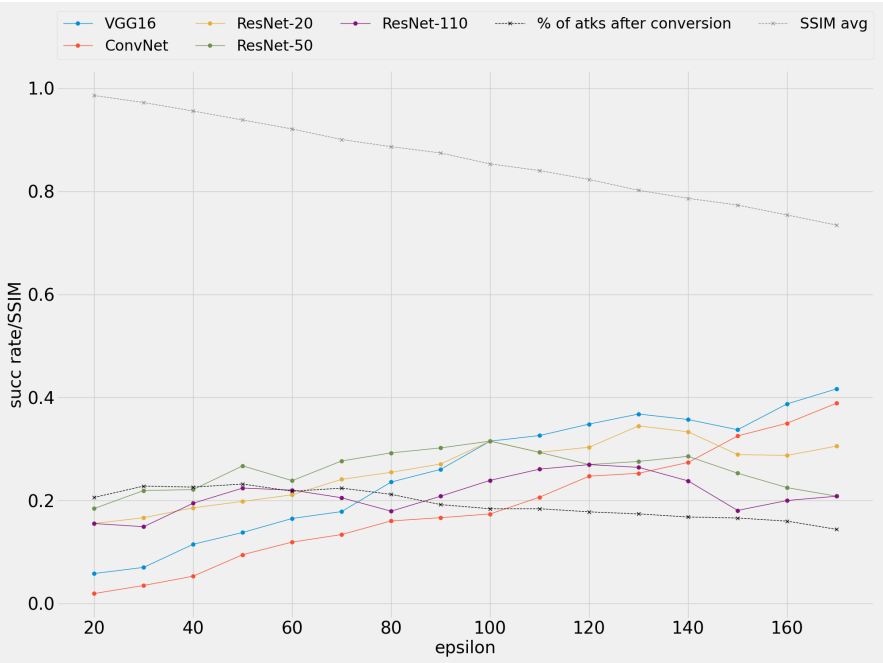


Figure 84.: Success attack rate of targeted FGSM L_1 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

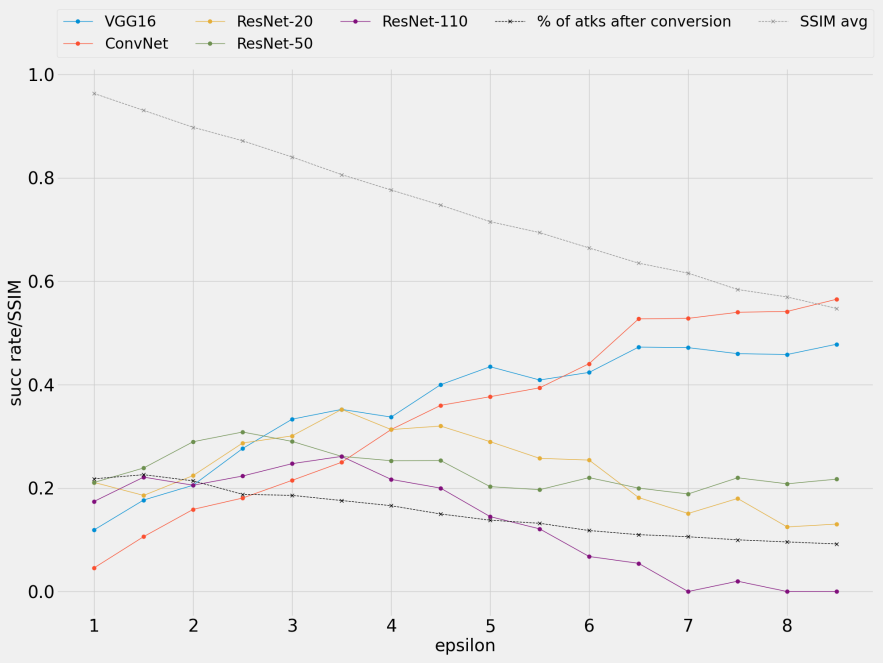


Figure 85.: Success attack rate of targeted FGSM L_2 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

A.1.2 *Deepfool*

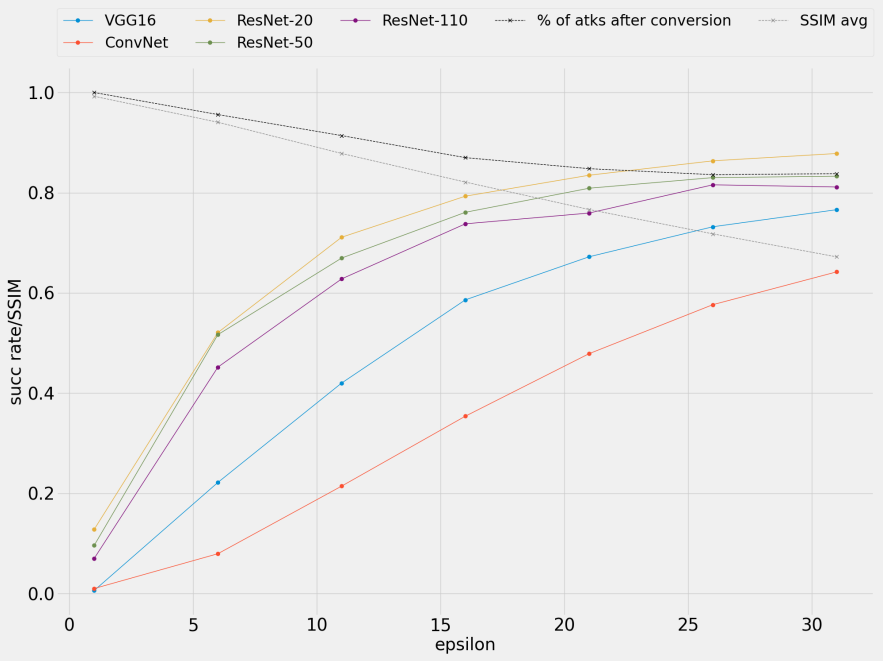


Figure 86.: Deepfool success attack rate on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

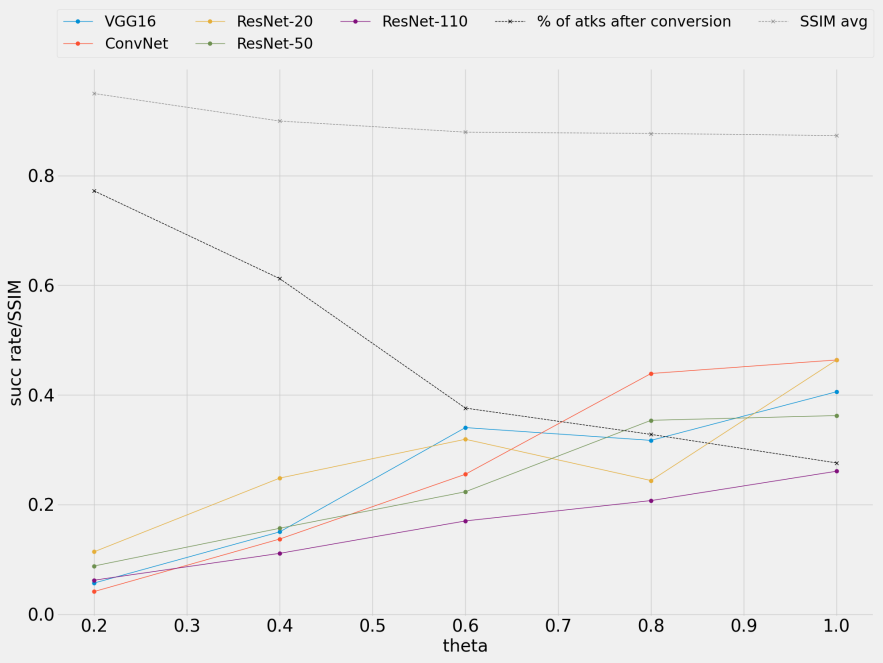


Figure 87.: NT-JSMA success attack rate on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per θ and $\gamma = 20\%$

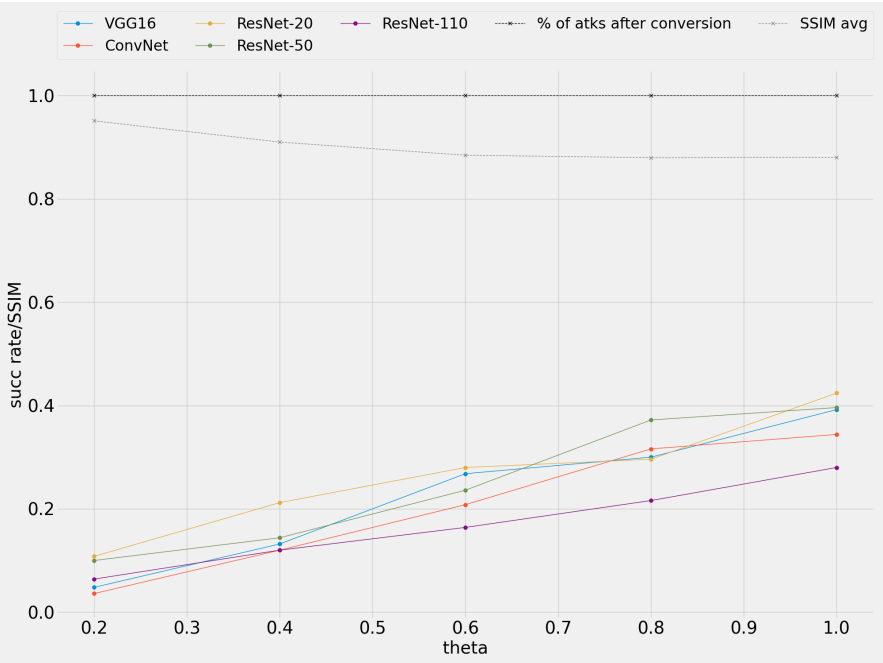


Figure 88.: NT-JSMA success attack rate on each model and their corresponding SSIM and total successful adversarial attacks on model generator without converting to a valid image per θ and $\gamma = 20\%$

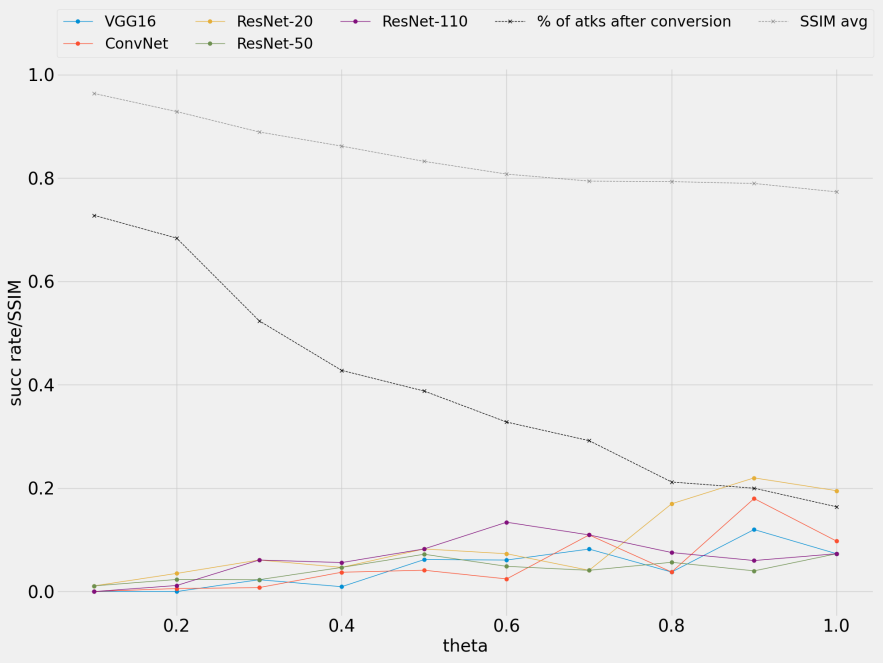


Figure 89.: Targeted JSMA success attack rate on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per θ and $\gamma = 20\%$

A.1.3 *Carlini*

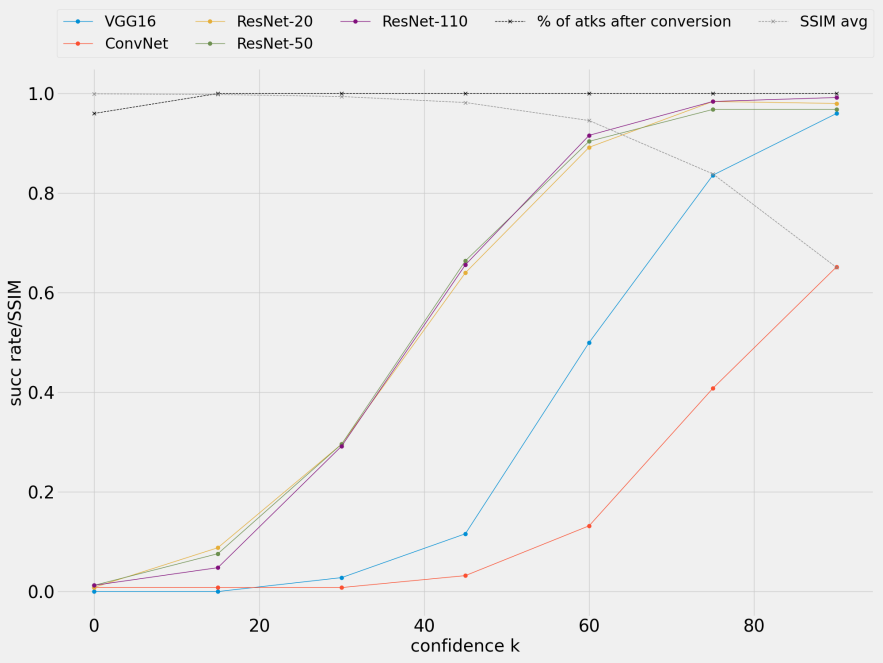


Figure 90.: Non-targeted Carlini L_2 norm success attack rate of on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per k

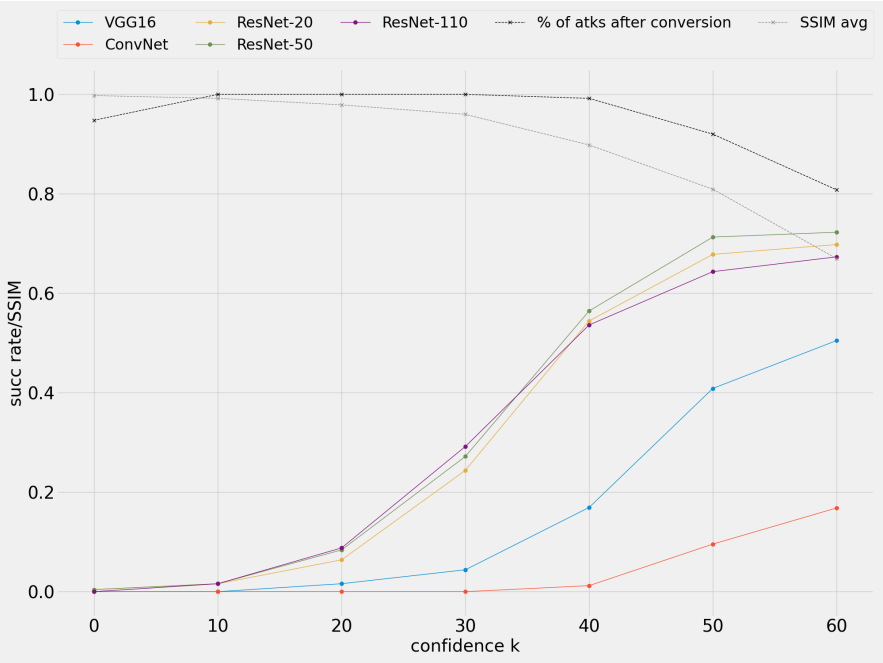


Figure 91.: Targeted Carlini L_2 adversarial sample for some confidence k

A.1.4 *PGD*

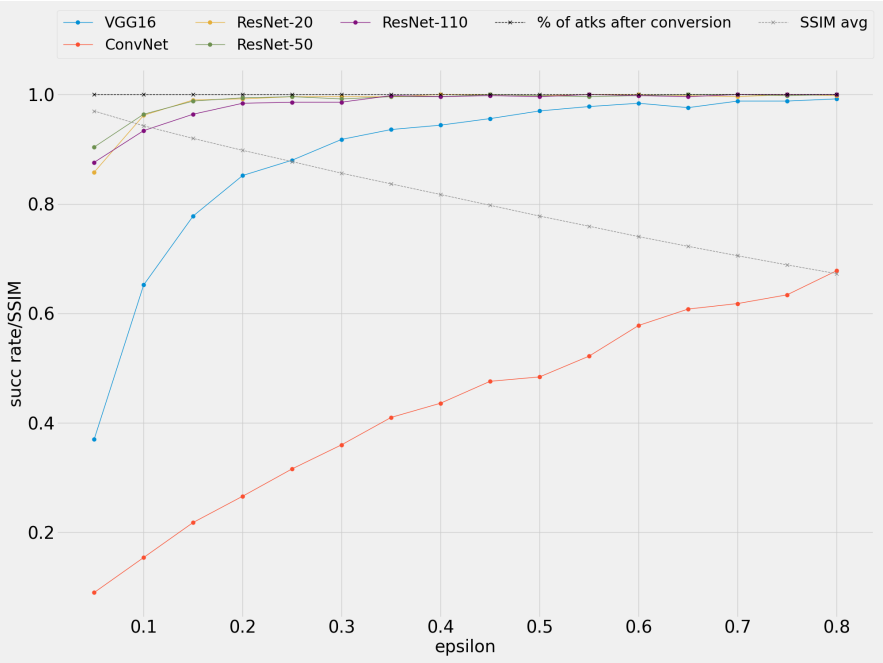


Figure 92.: Success attack rate of non-targeted PGD L_∞ norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

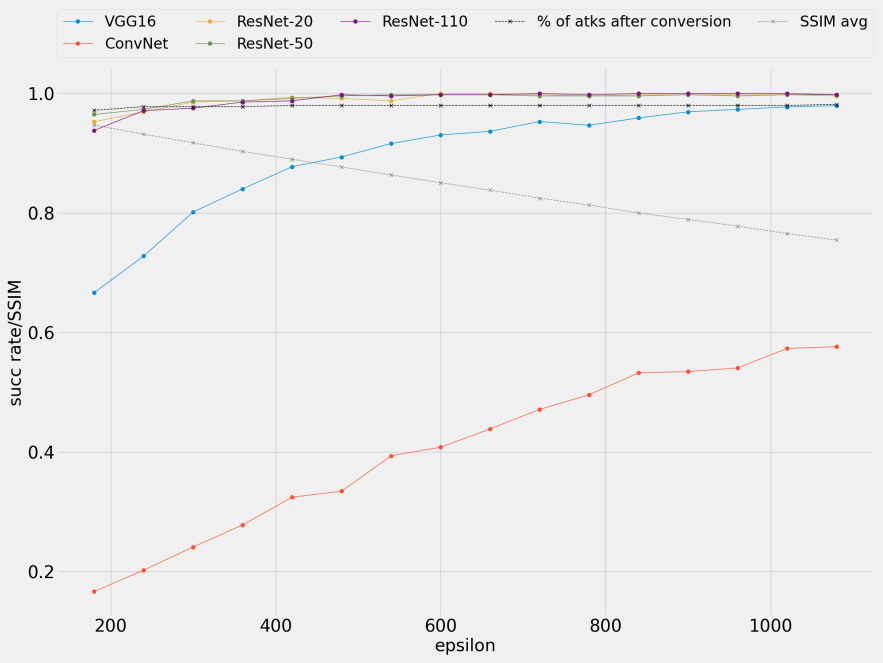


Figure 93.: Success attack rate of non-targeted PGD L_1 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

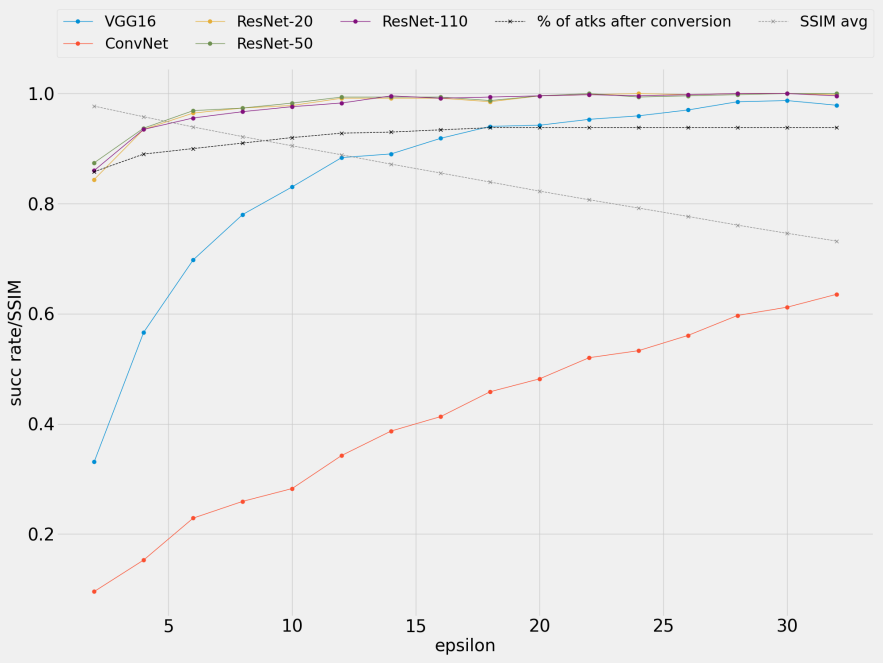


Figure 94.: Success attack rate of non-targeted PGD L_2 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

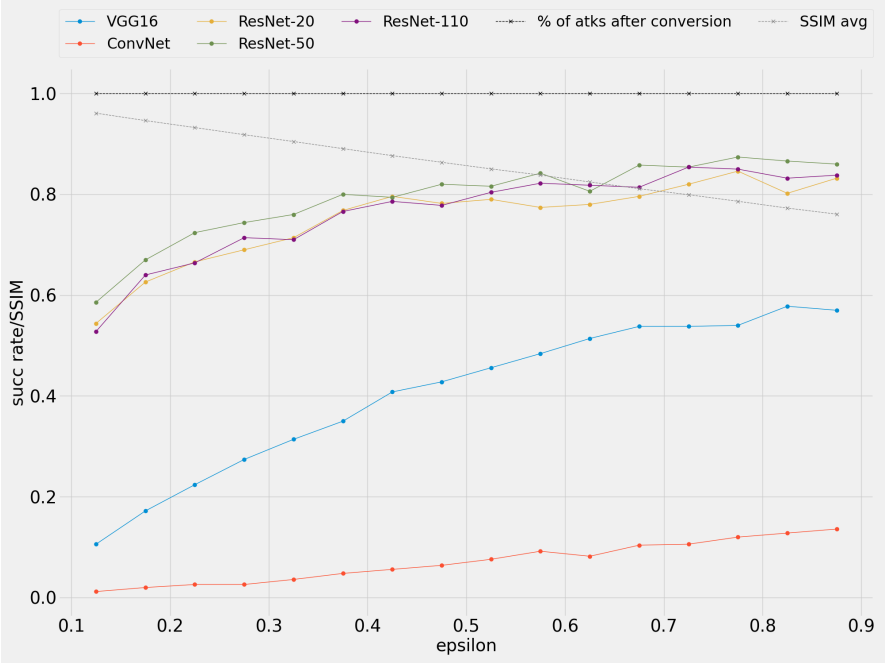


Figure 95.: Success attack rate of targeted PGD L_∞ norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

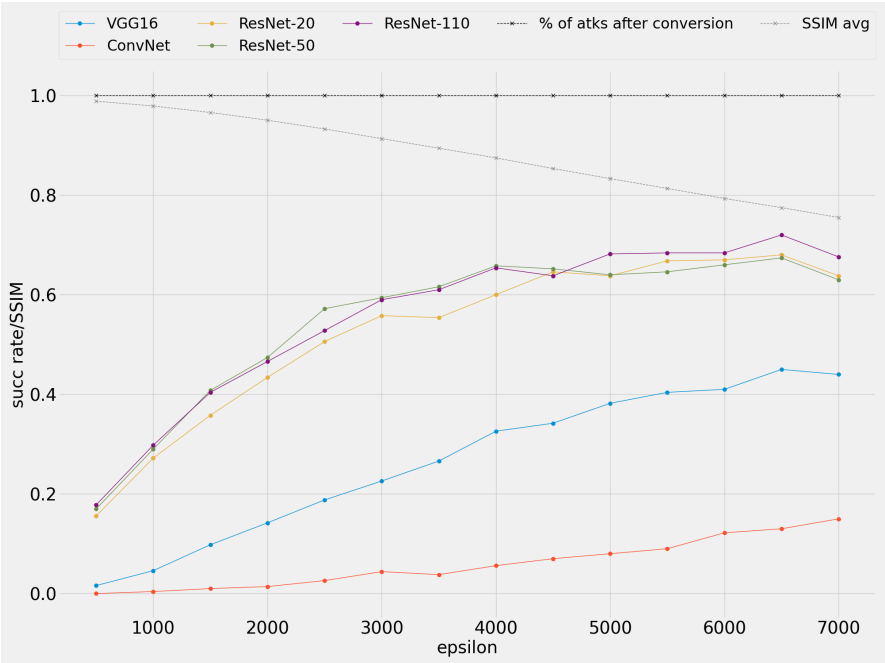


Figure 96.: Success attack rate of targeted PGD L_1 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

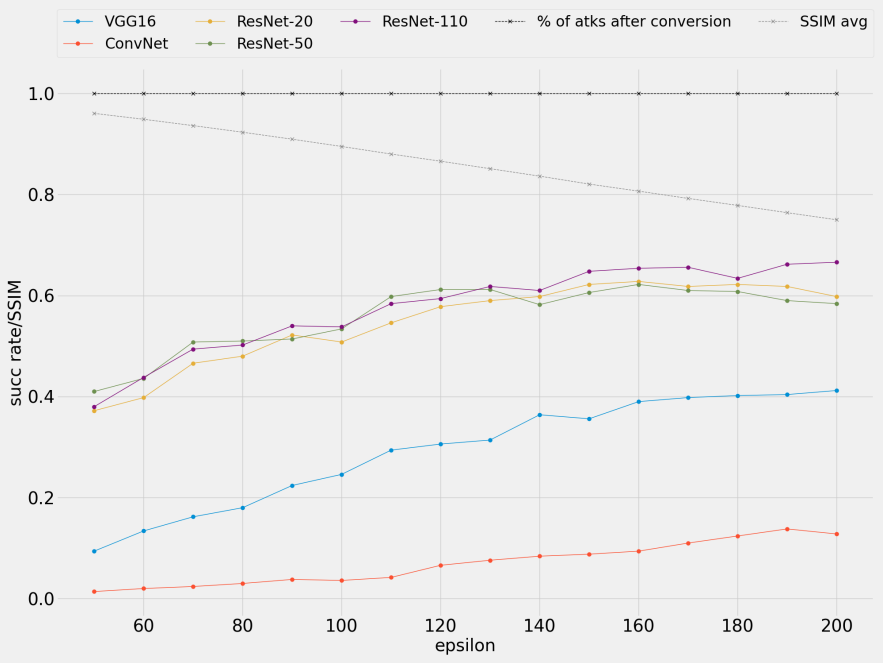


Figure 97.: Success attack rate of targeted PGD L_2 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

A.1.5 *One pixel/Few pixels*

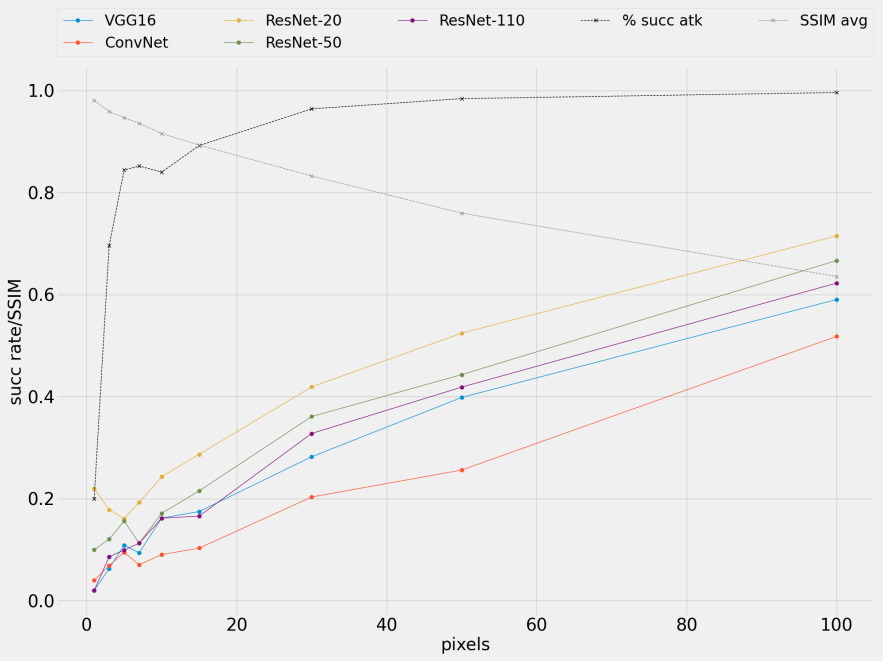


Figure 98.: Success attack rate of non-targeted Few pixels attack on each model and their corresponding SSIM and total successful adversarial attacks on model generator per number of pixels perturbed

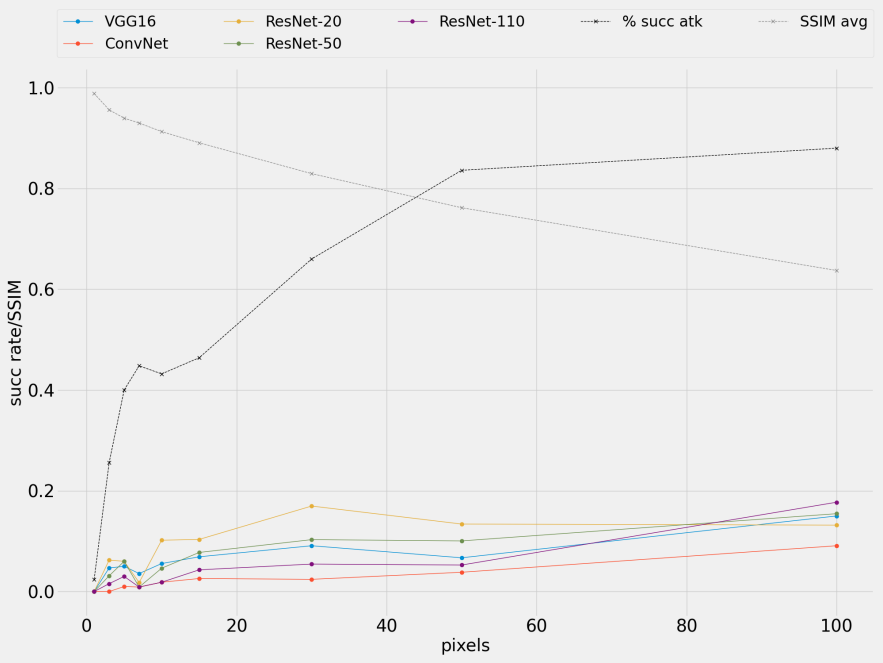


Figure 99.: Success attack rate of targeted Few pixels attack on each model and their corresponding SSIM and total successful adversarial attacks on model generator per number of pixels perturbed

A.2 GTSRB SEARCH PARAMETERS

A.2.1 *FGSM*

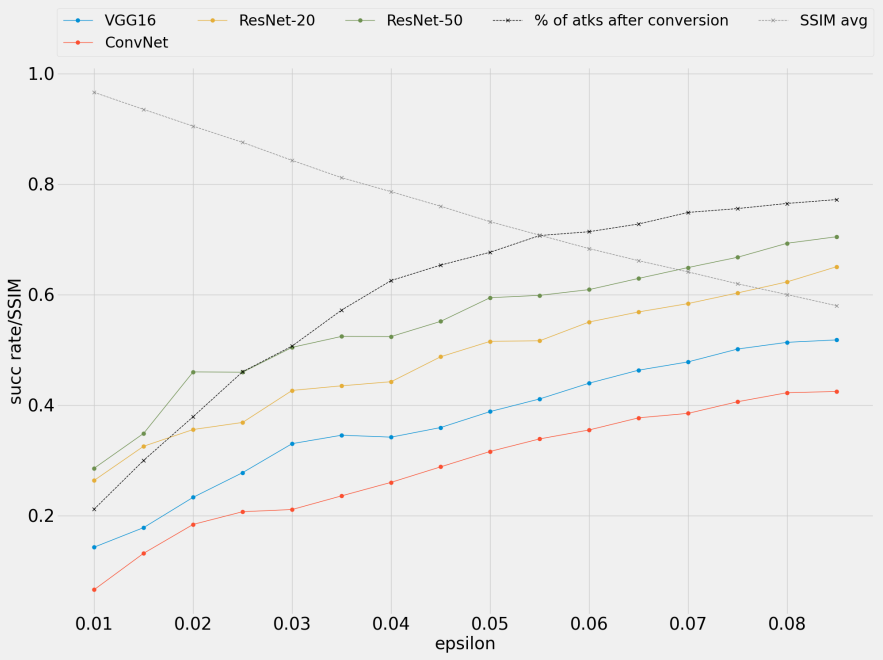


Figure 100.: Success attack rate of non-targeted FGSM L_∞ norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

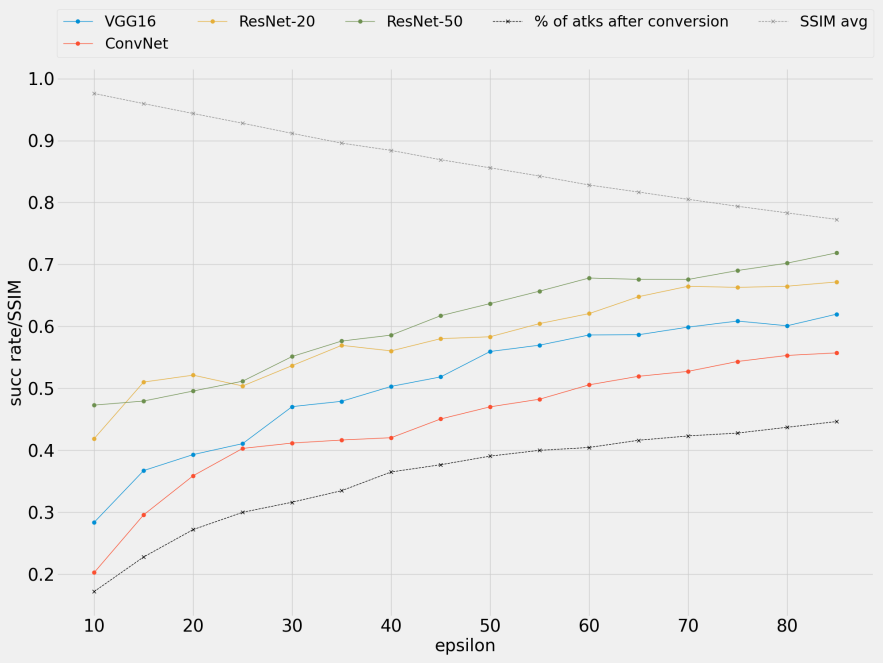


Figure 101.: Success attack rate of non-targeted FGSM L_1 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

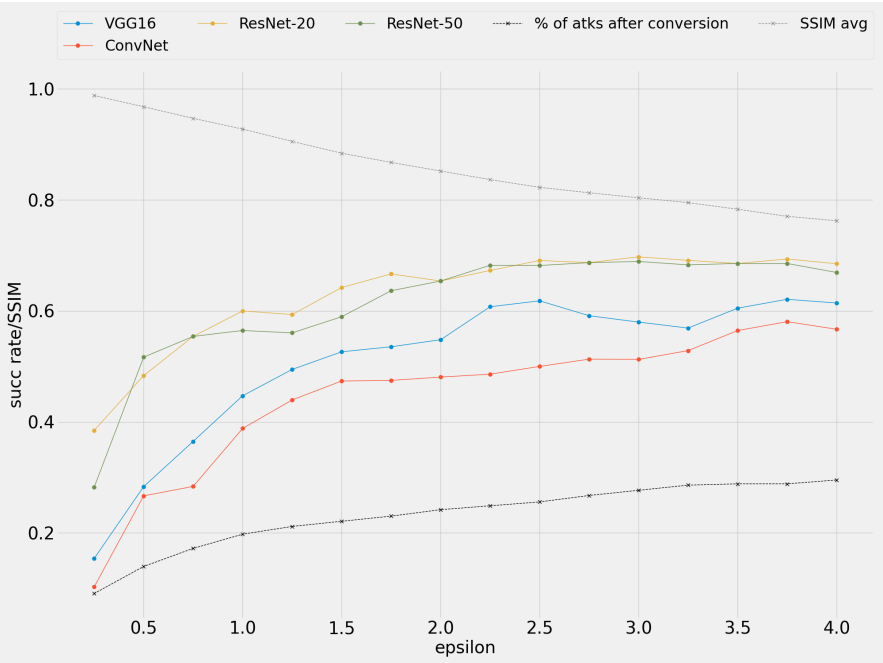


Figure 102.: Success attack rate of non-targeted FGSM L_2 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

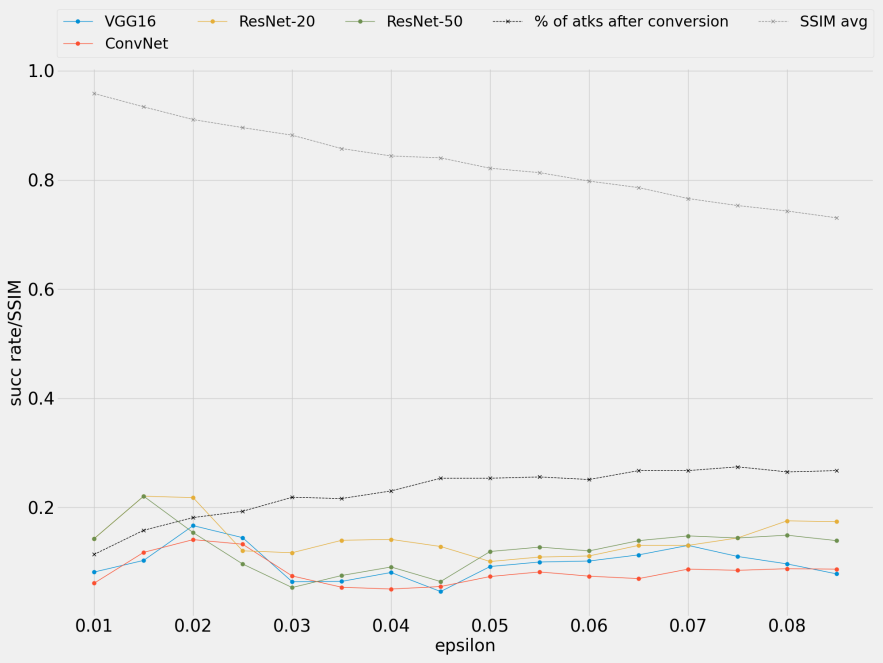


Figure 103.: Success attack rate of targeted for closest class FGSM L_∞ norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

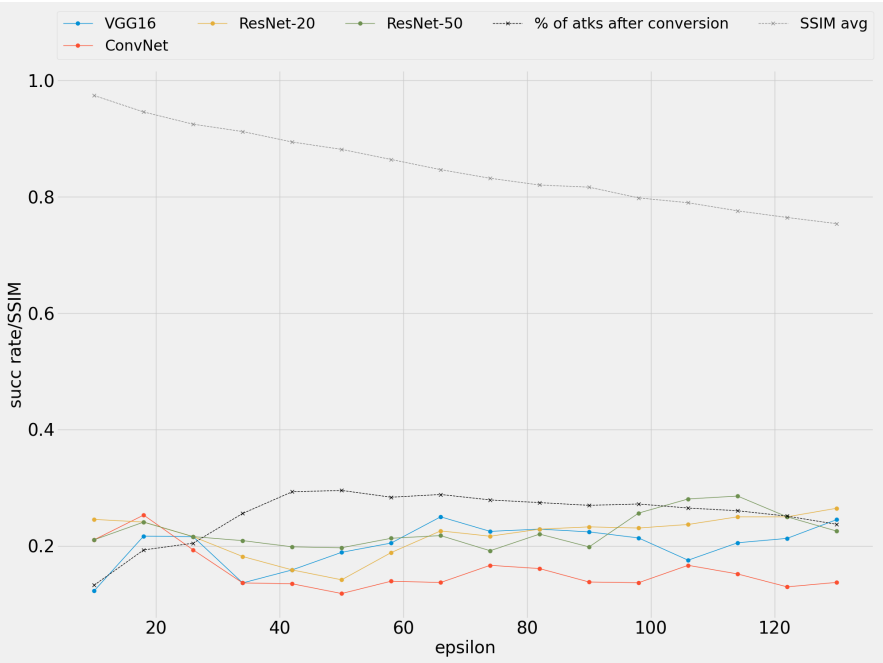


Figure 104.: Success attack rate of targeted for closest class FGSM L_1 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

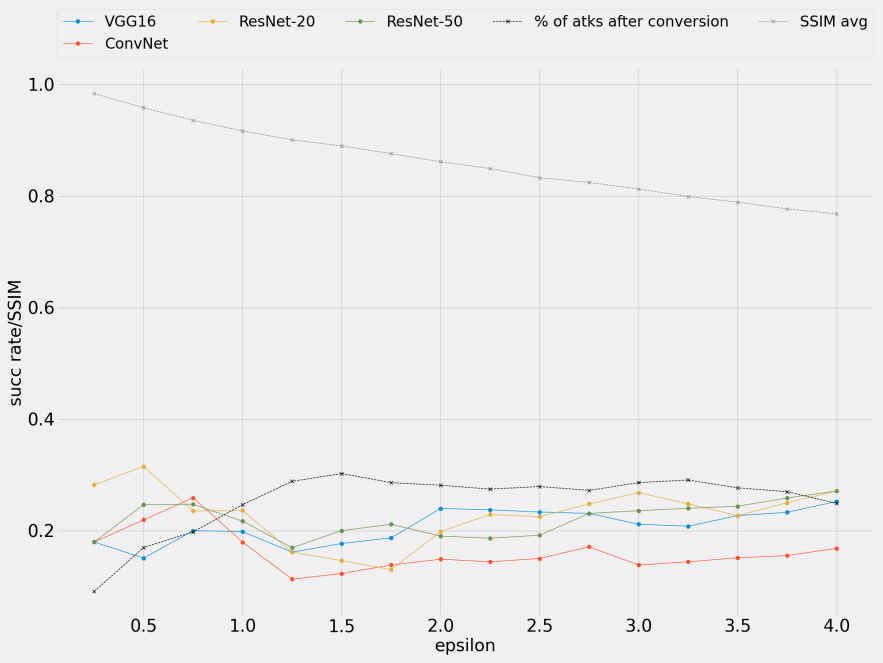


Figure 105.: Success attack rate of targeted for closest class FGSM L_2 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

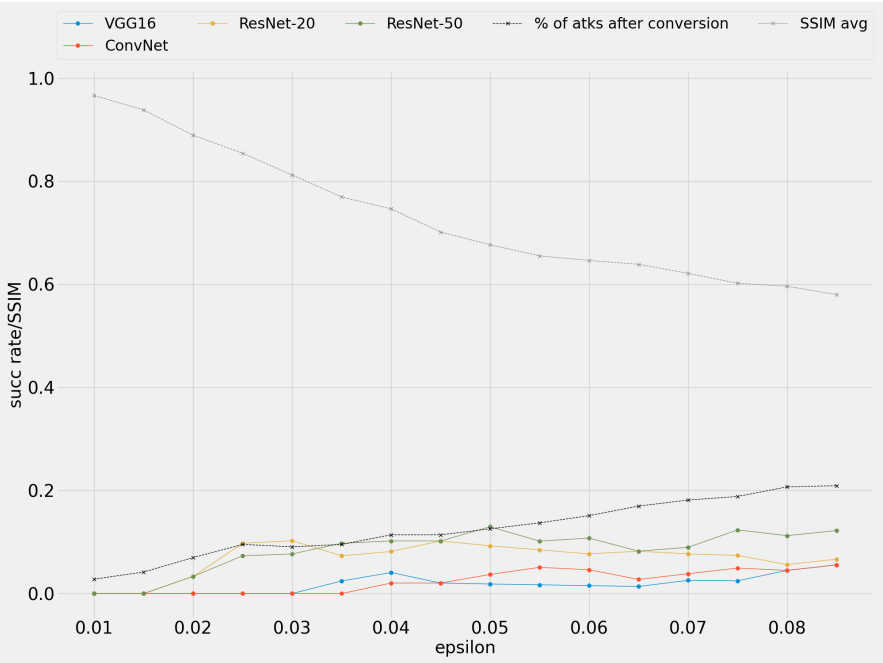


Figure 106.: Success attack rate of targeted for farthest class FGSM L_∞ norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

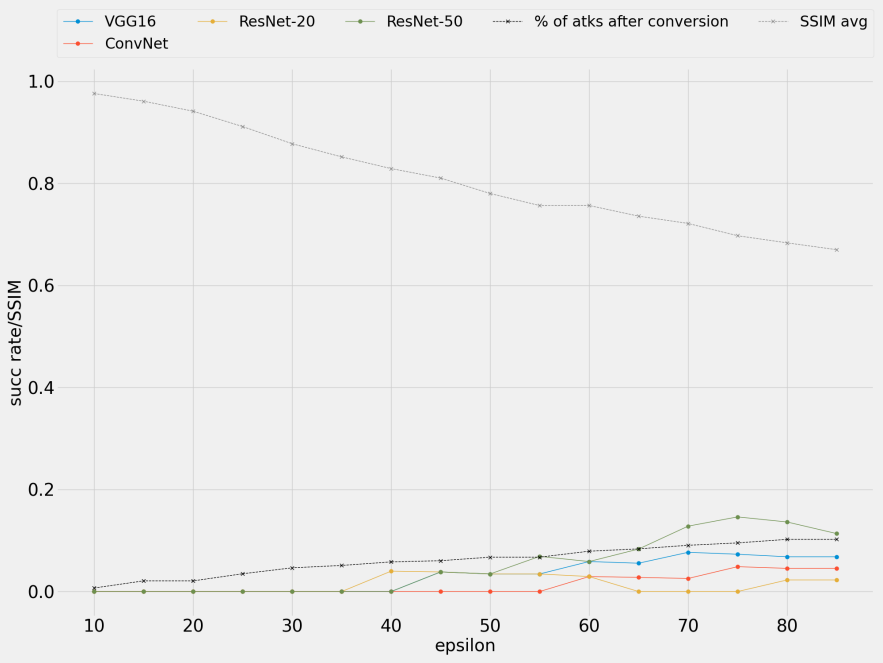


Figure 107.: Success attack rate of targeted for farthest class FGSM L_1 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

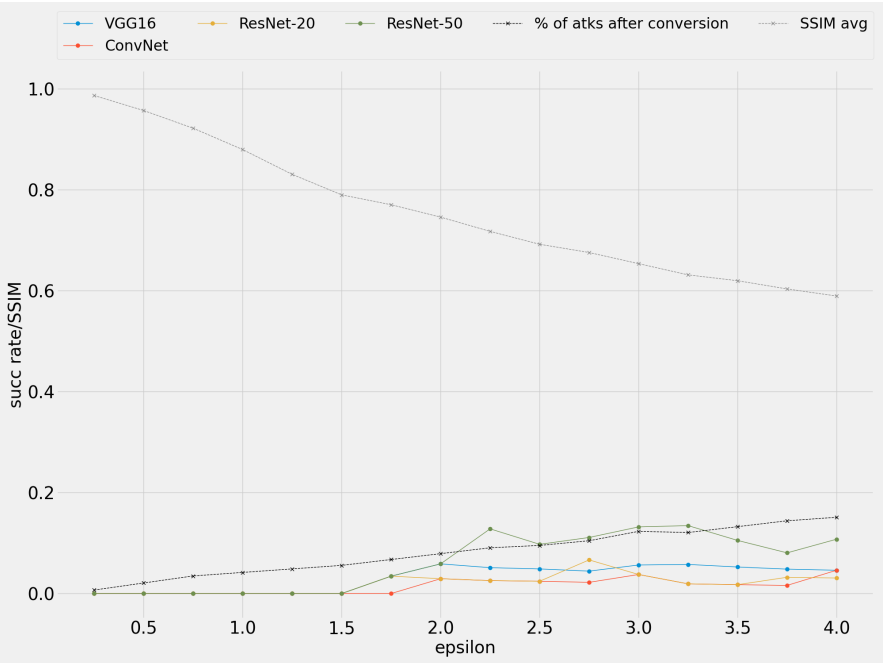


Figure 108.: Success attack rate of targeted for farthest class FGSM L_2 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

A.2.2 *Deepfool*

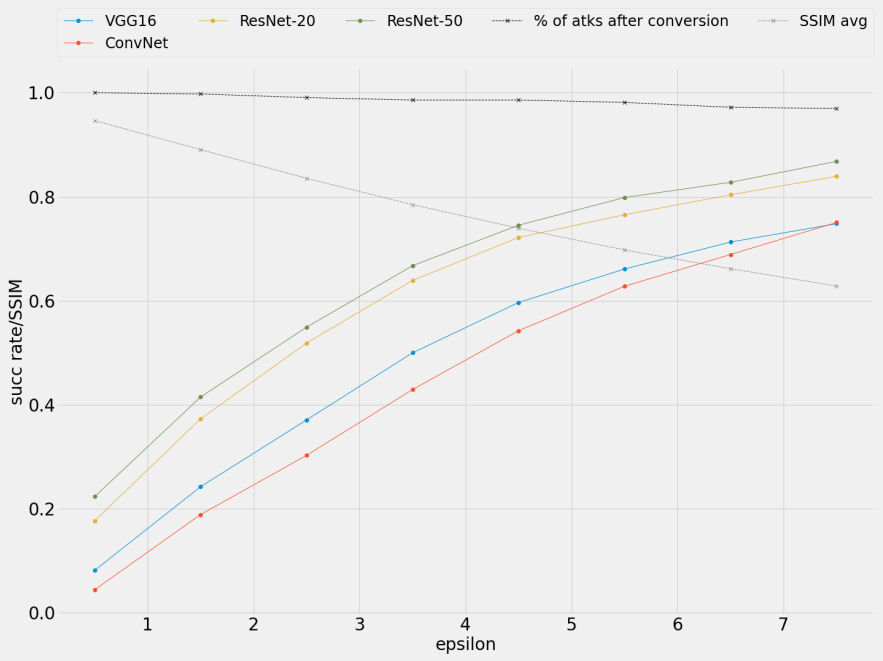


Figure 109.: Success attack rate of Deepfool on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

A.2.3 *Carlini*

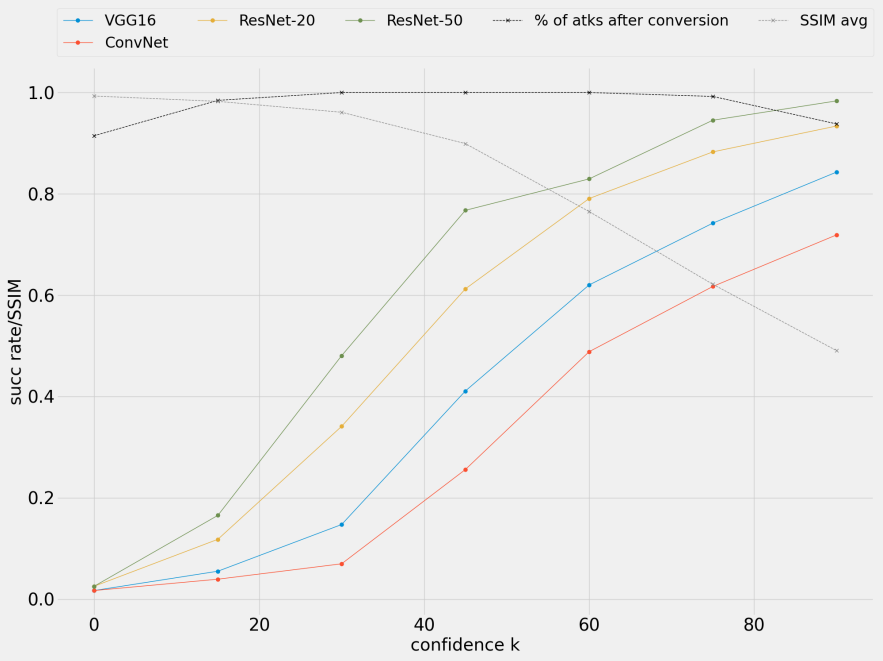


Figure 110.: Non-targeted Carlini L_2 norm success attack rate on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per k

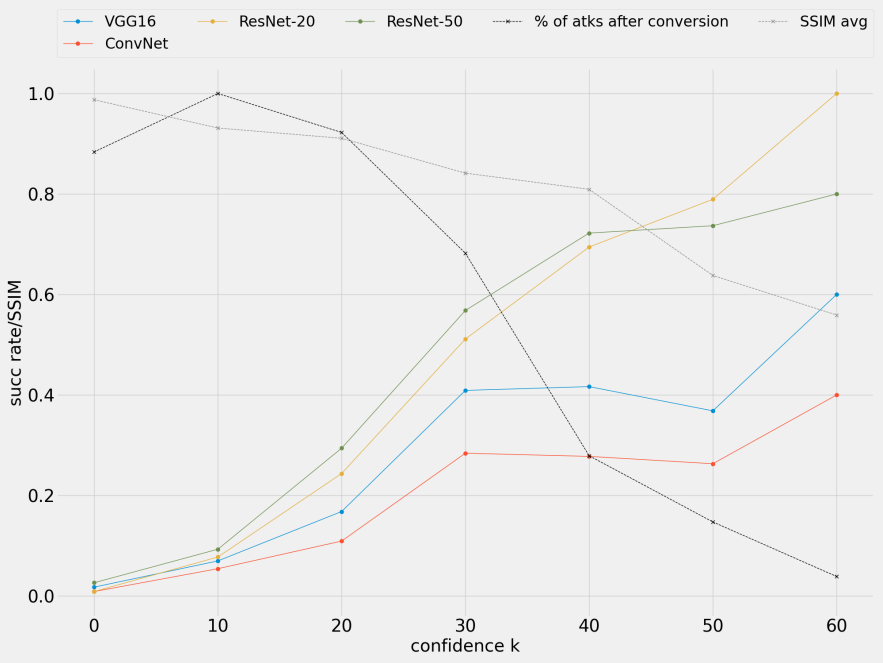


Figure 111.: Targeted Carlini L_2 for closest class success attack rate on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per k

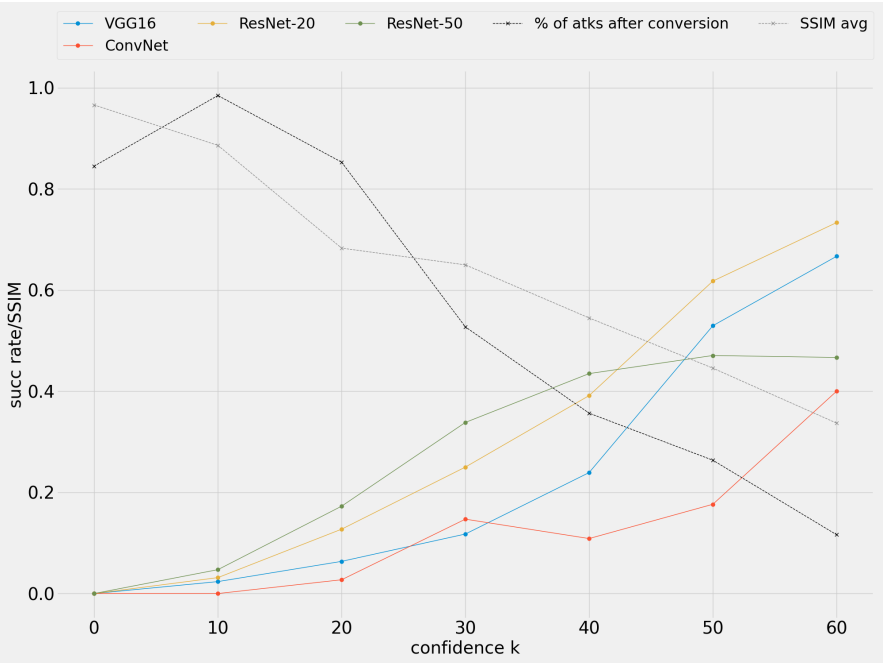


Figure 112.: Targeted Carlini L_2 for farthest class success attack rate on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per k

A.2.4 *PGD*

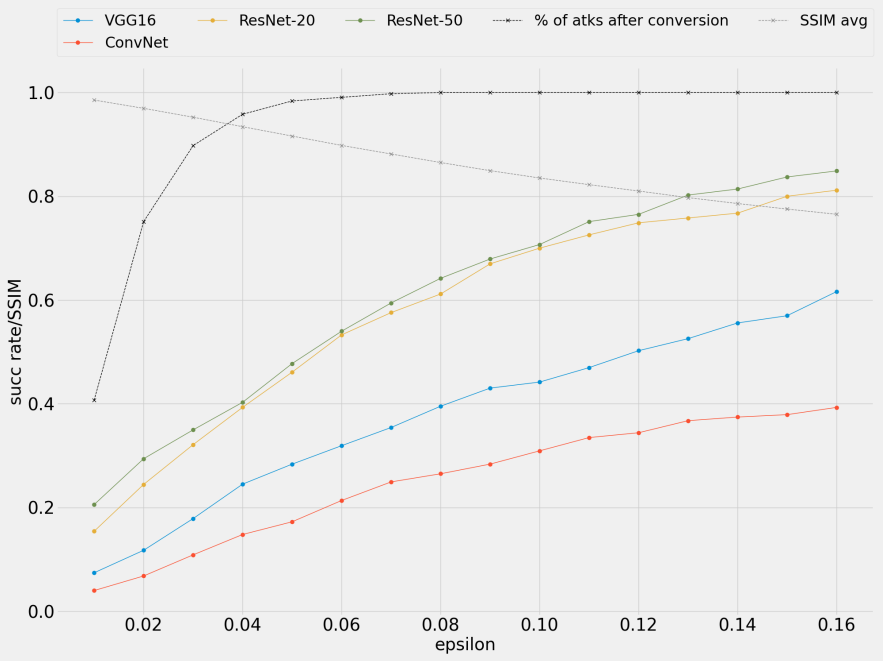


Figure 113.: Success attack rate of non-targeted PGD L_∞ norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

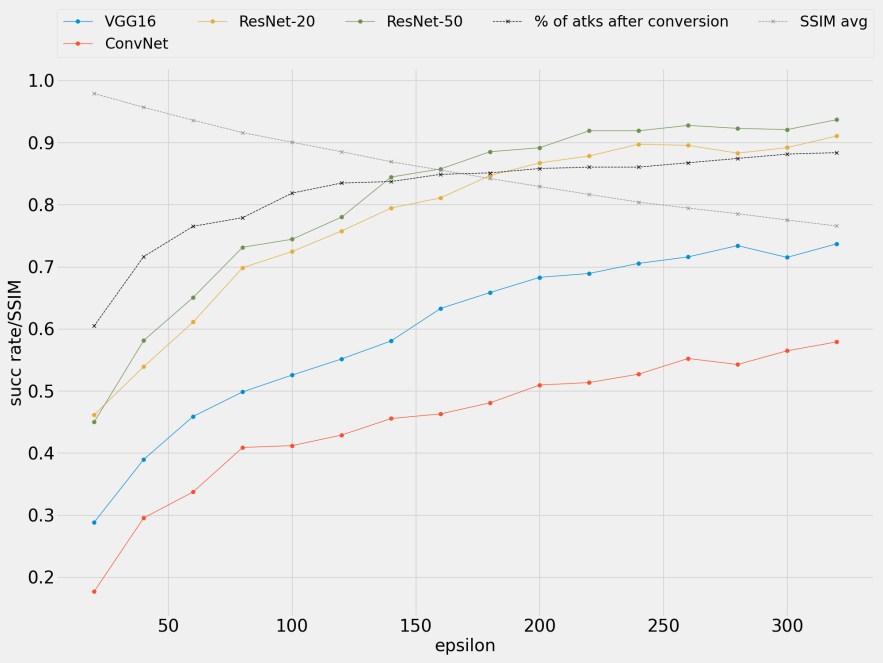


Figure 114.: Success attack rate of non-targeted PGD L_1 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

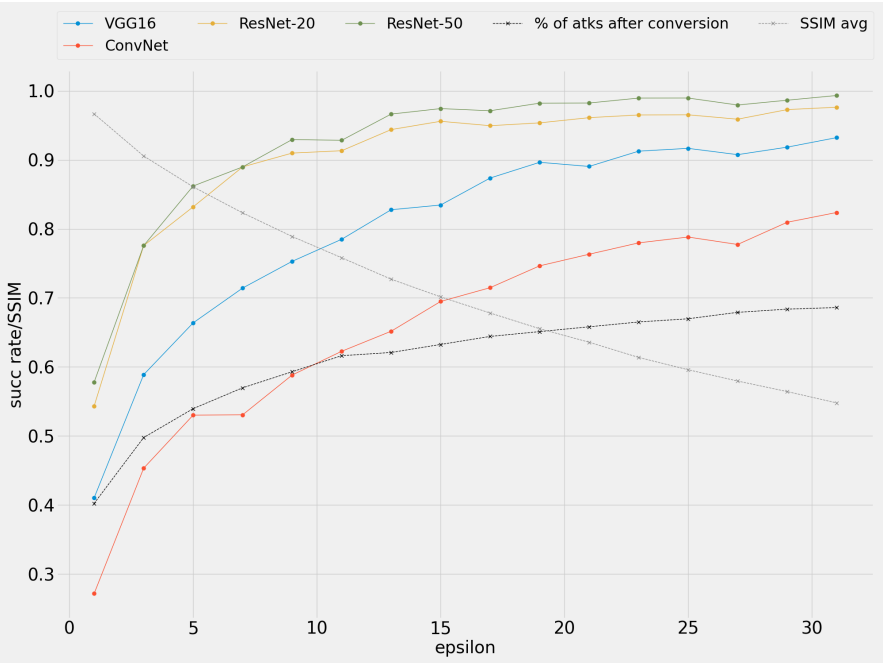


Figure 115.: Success attack rate of non-targeted PGD L_2 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

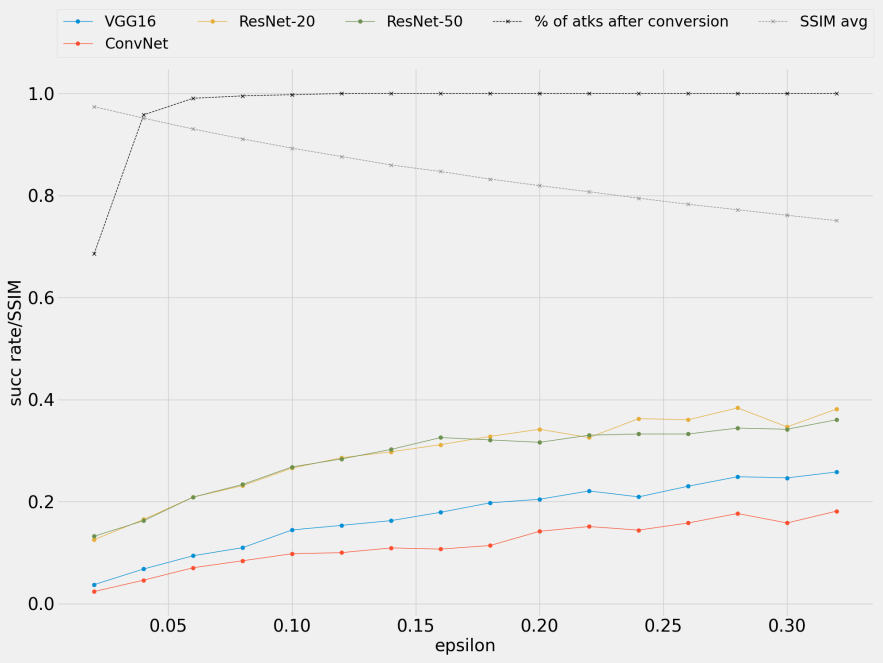


Figure 116.: Success attack rate of targeted for closest class PGD L_∞ norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

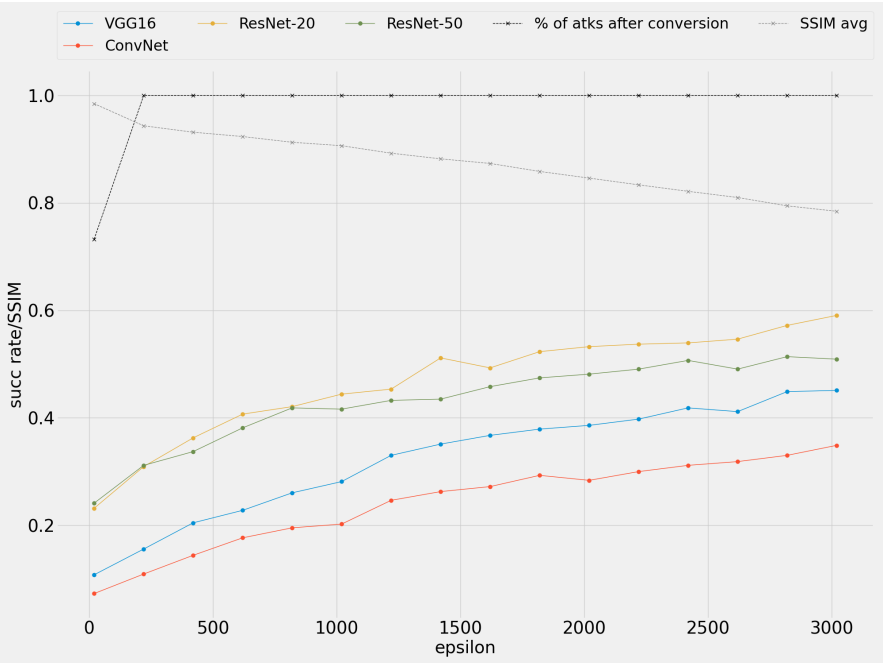


Figure 117.: Success attack rate of targeted for closest class PGD L_1 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

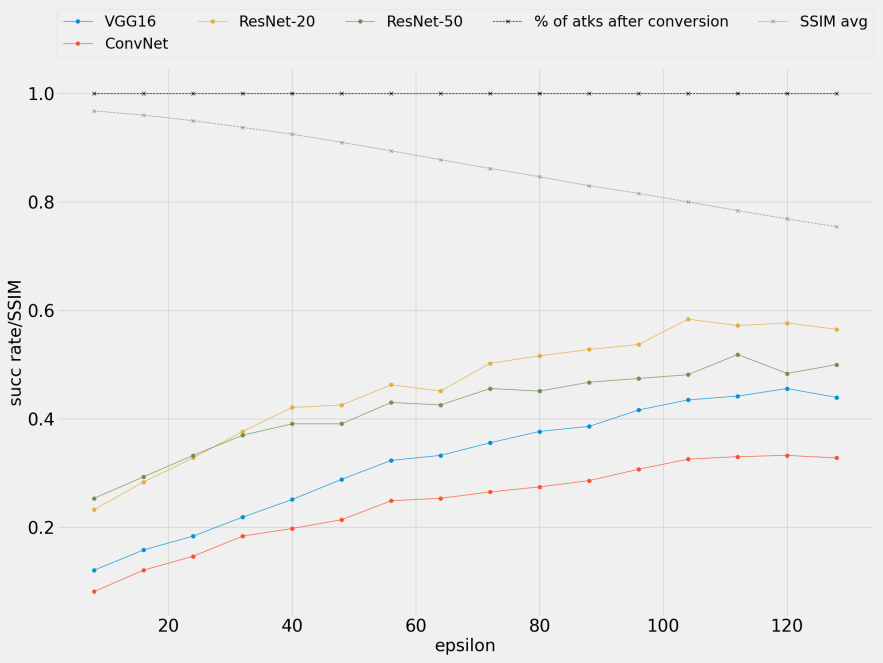


Figure 118.: Success attack rate of targeted for closest class PGD L_2 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

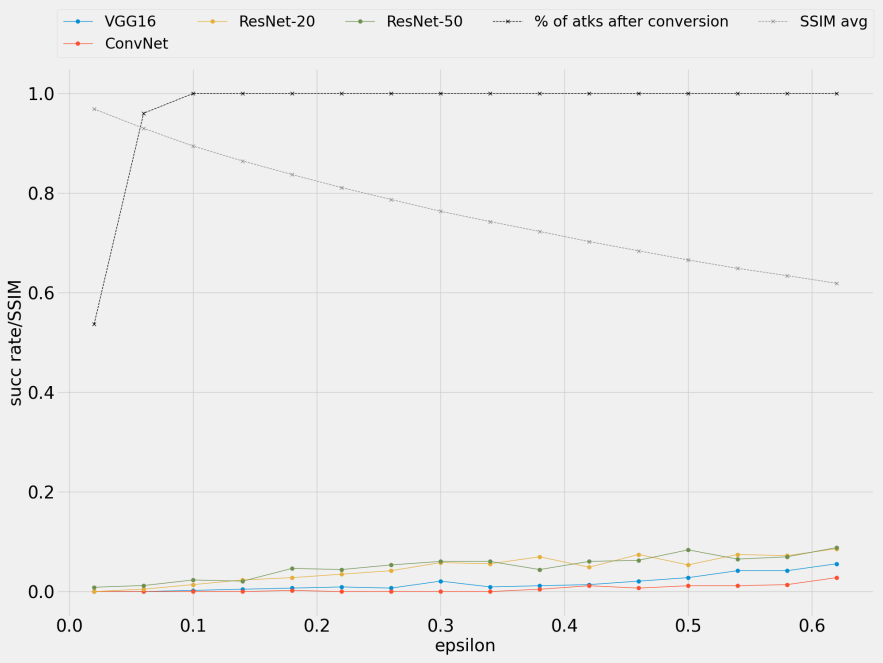


Figure 119.: Success attack rate of targeted for farthest class PGD L_∞ norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

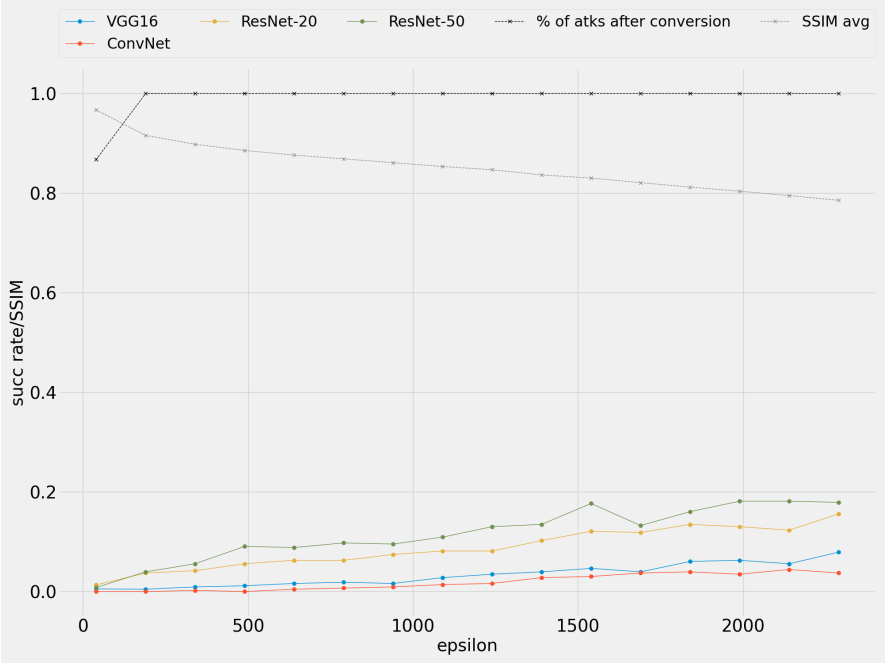


Figure 120.: Success attack rate of targeted for farthest class PGD L_1 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ

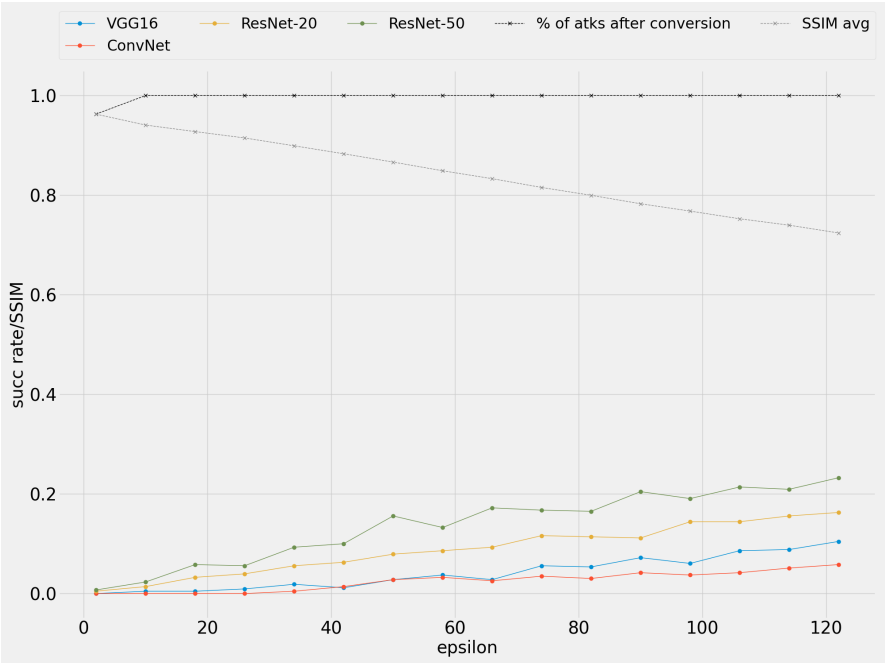


Figure 121.: Success attack rate of targeted for farthest class PGD L_2 norm on each model and their corresponding SSIM and total successful adversarial attacks on model generator after converting to a valid image per ϵ