Measuring The Transferability Of Adversarial Examples

Author - Deyan Petrov
Supervisor - Timothy Hospedales

MInf Project (Part 1) Report
Master of Informatics
School of Informatics
University of Edinburgh

2019

Abstract

Adversarial examples are of wide concern due to their impact on the reliability of contemporary machine learning systems. Effective adversarial examples are mostly found via white-box attacks. However, in some cases they can be transferred across models, thus enabling them to attack black-box models. In this thesis we evaluate the transferability of 3 adversarial attacks across 2 classes of models.
The evaluated attacks are the Fast Gradient Sign Method attack, the Basic Iterative Method attack, and the Carlini & Wagner attack.
The 2 classes of models are the VGG class(using VGG16, VGG19 and an ensemble of VGG16 and VGG19), and the Inception class(Inception V3, Xception, Inception Resnet V2, and an ensemble of the 3).
Moreover, the problems with the assessment of transferability in the current body of research are outlined and a technique for amending these problems is applied using L-Infinity clipping and SSIM evaluation.
Finally, the effectiveness of the Spatial Smoothing denoising defense is tested against the three attacks.

Acknowledgements

Thank you to my supervisor Timothy Hospedales for his support throughout the project, for providing me with different interesting project ideas and for helping me address some of the problems met in this work.

Thank you to family for their continuous emotional and financial support and my friends for helping me relax during stressful times.

Table of Contents

Chapter 1

Introduction

Image classification is the task of assigning an image a label based on the contents of the image. Convolutional neural networks (CNNs) [9] are neural networks that have been at the forefront of image classification tasks since the ongoing emergence of big data. However, Neural Networks are vulnerable to adversarial examples[24].
Adversarial examples are inputs created with the purpose of fooling a machine learning model. In the context of image classification, this means distorting an image which has been initially correctly classified, to be misclassified due to the distortion.
Adversarial examples pose various research questions, such as how neural networks and humans differ in classifying images. Adding noise to an image does not usually deceive a human annotator, however, neural networks can be easily fooled in many cases.
Furthermore, from a security perspective, adversarial examples can pose risks in a variety of machine learning applications, such as self-driving cars, drone control, facial recognition systems, speech recognition systems and others. For example, [3] elaborates on how bad actors in the heavily funded healthcare sector are able to take advantage of adversarial attacks for misdiagnoses, leading to unnecessary and expensive medical procedures.
Adversarial images are created via a method called adversarial attack. Adversarial attacks target a specific model, and look to change the input to this model, so that misclassification occurs. During the process of creating an adversarial input, informed attacks need to have access to the architecture and parameters of the model. This is called a ”white-box” setting(the model is a white box). Adding random noise without taking advantage of the available knowledge of the model will produce adversarial inputs of much lower quality.
Knowing the parameters and/or architecture of a model, however, is not always possible. A malicious hacker that wants to attack a web service, for example, will not typically know the inner details of the algorithm that the service is using. This is called a ”black-box” setting(the model is a black box). One way to attack the service in this black-box setting, is to run an adversarial attack on either a custom local model, or on a publicly available model, in a white-box way, and then send the adversarially created input over to the service. An example of this is the attack on the Clarifai.com service[11]. The property of an adversarial example trained on one system whose pa-

rameters and architecture are known, to transfer to another unknown system, is called transferability.
The research body of adversarial examples is growing rapidly and lots is done on inventing and developing offensive techniques for, and defensive techniques against attacks, both in the white-box and the black-box settings.
In this work we focus on the ”one-shot” black-box transferability of adversarial attacks. By this we mean that an attacker does not receive a feedback from the system, does not know the model architecure and the parameters and can modify a query only once.
Specifically, the focus is on measuring the transferability of different adversarial attacks for risk assessment across five standard image classification models - VGG16, VGG19, Inception V3, Xception and Inception Resnet V2, pre-trained on the ImageNet dataset, and across two ensembles. The idea is to check the transferability across similar models (VGG16 vs 19, and the Inception family) and across different models (VGGs vs Inceptions). Do small differences in the model architecture account for small differences in the transferability or is this not the case? Since the models are well-known, highly-performant, and publicly available, web services are able to, and have an incentive to adapt them to their specific use-cases via transfer learning. This could lead to overuse of the models in the public space and motivates the choice of models explored in this work.
In order to evaluate the transferability of the three different adversarial attacks, a small ImageNet subset is used (described in Chapter 4).
In the context of previous work, a fair amount of research has been done on our standard models to evaluate the effectiveness or transferability of adversarial attacks [19, 11]. However, this is usually done only for single or multiple parameter settings of the attacks. The parameters are usually chosen, so that the visual change in the attacked images is imperceptible. This ”parameter overfitting” may lead to unfair or inexhaustive comparisons due to insufficient values of the parameters being tested and the varying importances and roles of the parameters used in the different algorithms. This gives motivation for using a different way of assessing transferability(sec. 4.1)
Experiments were conducted for three attacks and one denoising defense using some of the adversarial attack frameworks available online.
Finally, adversarial images have been pasted in the Appendix section.

Report outline

Chapter 2 describes CNNs and the models used to test the attacks.
This chapter also explains the frameworks used for the attacks, and their capabilities. They are useful tools for developers to check their models for vulnerabilities and to try various defenses.
In Chapter 3 there are brief descriptions of the attacks and the defense used.
In Chapter 4 the metrics and methodology are discussed and the relevant results are interpreted.
Chapter 5 gives a summary of the report.
In Chapter 6 possible extensions or future work are discussed.

Chapter 2

Background

2.1 Convolutional Neural Networks

Convolutional Neural Networks(CNNs) are a standard when it comes to image classification problems. A typical CNN network consists of a series of convolutional and pooling layers stacked one after the other. Classic powerful CNN architectures consist of a long sequence or paralellization of such layers. For example, VGGNet uses 10 convolutional and 5 max pooling layers, Inception networks utilize branching convolution and pooling operations, and residual blocks allow for training very deep CNNs.

2.1.1 Convolutional Layers

Convolutional layers are filters with a fixed-sized width and height which iterate over windows of pixels of an image and calculate the dot product between the image pixel values and the filter values:

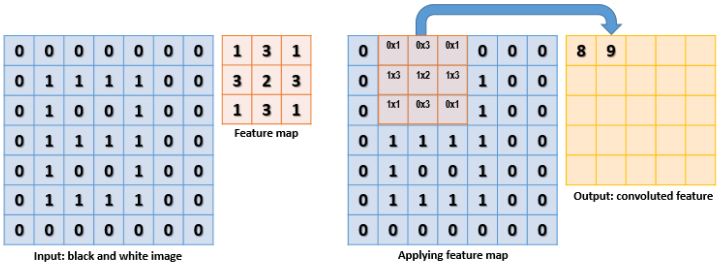


Figure 2.1: Convolutional Layer with a stride of 1.
https://docs.microsoft.com/en-us/cognitive-toolkit/tutorial2/tutorial2

The convolutional layers serve as feature extractors, and the resulting output from such a layer is called a feature map. The values of these filters are trainable parameters in the CNN.

2.1.2 Pooling Layers

A pooling layer is one which reduces the dimensionality of the feature maps by taking the most prominent features in a window(max-pooling), the average feature(average pooling) or by using another metric. It iterates over the matrix in the same way:

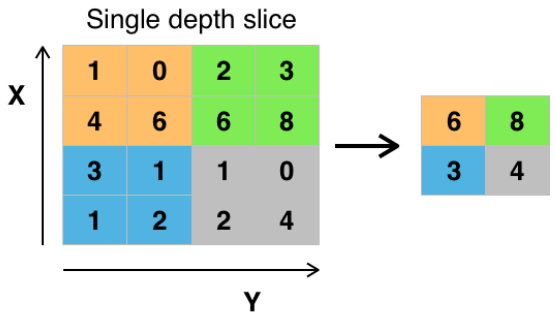


Figure 2.2: Max Pooling Layer with a stride of 2.
https://docs.microsoft.com/en-us/cognitive-toolkit/tutorial2/tutorial2

2.1.3 Activation Functions

Activation functions introduce nonlinearities for complex pattern recognition in neural networks. The activation functions are typically applied after a convolutional layer. The most commonly used activation function in CNNs is Relu, which is characterized by fast computation times and alleviates vanishing gradient problems. The networks described below use Relu as their activations.

$$f(x) = max(x, 0) \tag{2.1}$$

Softmax function is applied to the outputs of the classifiers to yield a probability for the 1000 ImageNet classes.

$$p(C_k|\mathbf{x}) = \frac{e^{\,a_k(\mathbf{x})}}{\sum_{j=1}^{K} e^{\,a_j(\mathbf{x})}} \tag{2.2}$$

2.1.4 Residual Blocks

Residual blocks allow for training of deep neural networks by alleviating problems such as vanishing gradients and allow for the same information of the image to be aggregated in different ways, increasing accuracy.

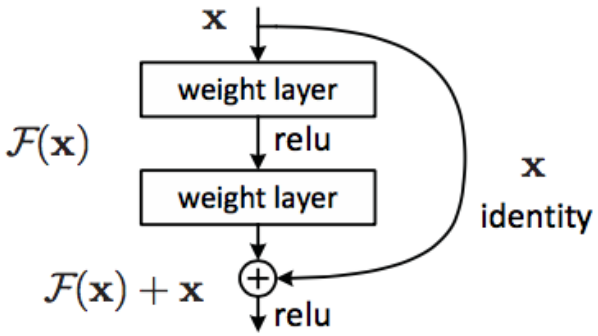


Figure 2. Residual learning: a building block.

Figure 2.3: Residual block
https://docs.microsoft.com/en-us/cognitive-toolkit/tutorial2/tutorial2

2.2 Standard Image Classification Networks

The following models were the default Keras implementations pre-trained on the ImageNet dataset.

2.2.1 VGG Family

The first evaluated family of models is the VGG family. The VGG16 architecture was first introduced in [18]. For accurate classification, the networks expect a RGB image to be preprocessed by flipping the R and B dimensions and subtracting 103.939 from the blue, 116.779 from the green, and 123.68 from the red channel.

2.2.1.1 VGG16

The VGG16 implementation in Keras consists of 5 blocks. The first 2 blocks use 2 consecutive convolutional layers of filter size 3, and 'same' padding. The last 3 blocks have the same specifics, but use 3 convolutions instead of 2. The activation function is Relu. The number of convolutional filters doubles each block. After the convolutions there is a Max Pooling layer of size 2 and stride 2. At the end of the blocks there are 2 fully connected layers with Relu activation of size 4096 and finally a Softmax layer for the 1000 ImageNet categories.

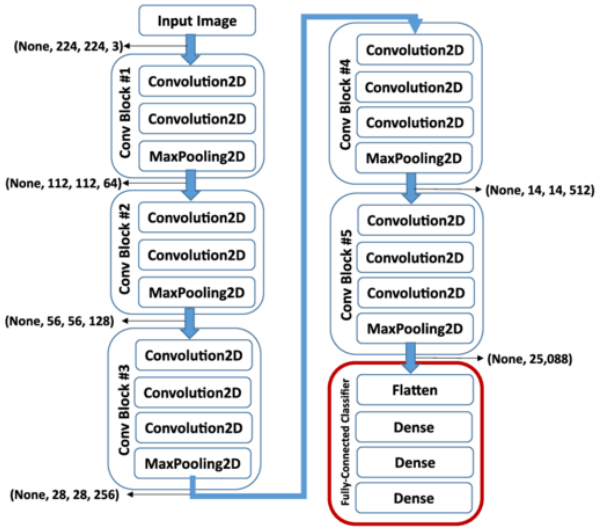


Figure 2.4: VGG16 Architecture
https://www.researchgate.net/profile/Kasthurirangan_Gopalakrishnan/publication/319952138/

The network has achieved 7.3% top-5 error rate on the 2014 ImageNet challenge.

2.2.1.2 VGG19

The VGG19 architecture has the same architecture as VGG16, but instead of 3, it has 4 convolutional layers in the last 3 blocks. It allows for deeper feature representations than VGG16, but does not yield much more accurate results than VGG16 on the ImageNet challenge.

An ensemble of the two models was also evaluated against the attacks.

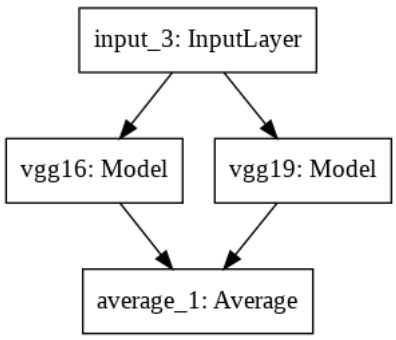
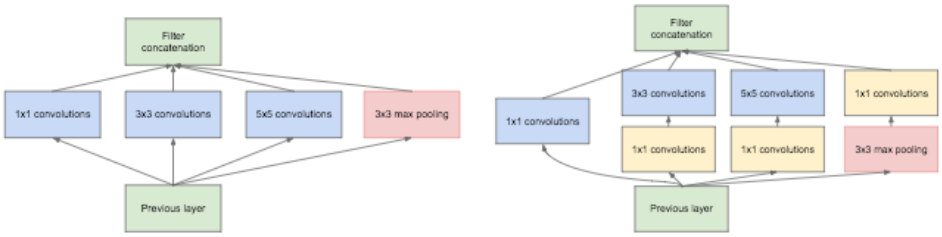


Figure 2.5: VGG Ensemble

2.2.2 Inception Family

GoogLeNet, also known as Inception-v1 was first introduced in 2014 [22] and exploited the idea of an ”inception modules”[10]. The inception modules are feature

maps combining contexts of different sizes to obtain different types of patterns, which removes the need for manual selection of filter sizes and reduces computational and memory cost via convolution dimensionality reductions.



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2.6: Inception modules from GoogLeNet's original paper [22]

The GoogLeNet architecture combines these modules and has 6.7% error rate on the 2014 ImageNet challenge and outperforms VGG16 despite being a lot smaller(55 MB vs 490 MB).

It uses auxiliary classifiers to combine losses and reduce overfitting.

The inception models' preprocessing require the pixels of an image to be divided by 255, subtracted by 2, and multiplied by 2, so the original [0-255] pixel range is mapped to [-1-1].

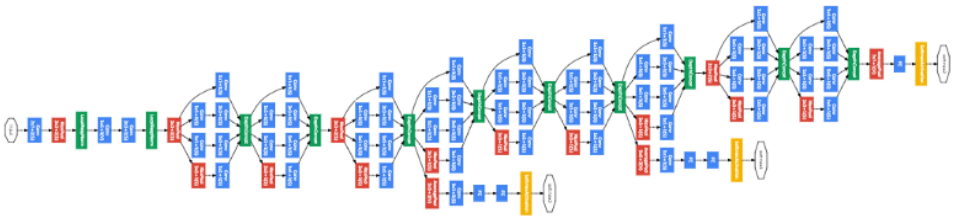


Figure 2.7: Inception-V1 architecture [22]

2.2.2.1 Inception-V3

Inception-V2 and V3 were introduced in the same paper[23]. In it, several improvements were done to the original Inception architecture. 5x5 convolutions were factorized to 2 3x3 convolutions, which sped up computation. Furthermore, they found that deeper representations lose information due to loss of dimensions and thus information. Therefore, they made the inception modules wider to keep the number of channels high. For version 3, RMSProp Optimizer and batch normalization[7] were added, 7x7 convolutions were factorized, and label smoothing was done on the class predictions to prevent the network from being too confident and avoid overfitting.

2.2.2.2 Xception

The Xception was introduced in [2] and replaces the Inception network convolutions with depthwise separable convolutions. It has roughly the same number of parameters as Inception-V3, but slightly outperforms it on the ImageNet dataset. It is based on the assumption that spatial and depthwise correlations can be decoupled. This is a strong assumption, leading to the name of the network Xception from ”Extreme Inception”. After performing a 1x1 convolution on a feature map, it uses n convolutional filters, where n is the number of channels of the input, and each filter iterates only over the specific channel, instead of using filters that iterate over all channels. The network also makes use of residual blocks and takes into account previous feature maps. Xception has slightly smaller memory requirements than Inception V3.

Figure 4: An “extreme” version of our Inception module, with one spatial convolution per output channel of the 1x1 convolution.

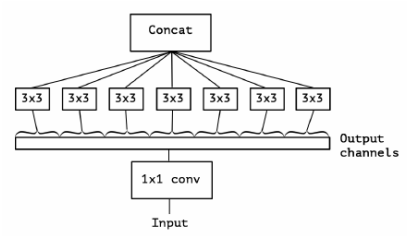


Figure 2.8: Xception module [2]

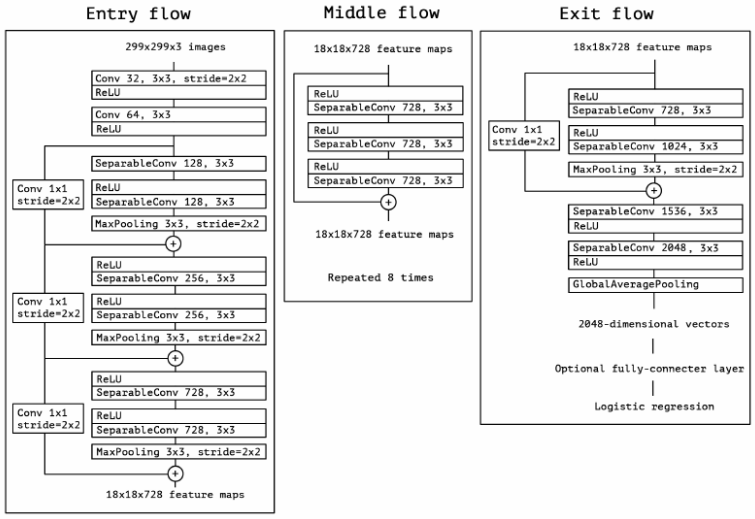


Figure 2.9: Xception architecture [2]

2.2.2.3 Inception Resnet V2

Our final model is Inception Resnet V2. It incorporates elements of Residual Networks [6] into the Inception V4 architecture. It has the same architecture as Inception Resnet V1 with the exception of the ”stem” (the part before the inception modules), but has

different hyperparameter settings. Both versions were introduced in the same paper [21]. Inception V4 and Inception Resnet V2 have similar computational complexity despite the latter being deeper. They both utilize reduction blocks which reduce the size of the output, and Block A is the same for both architectures. Inception V4 and Inception Resnet V2 also differ by the types of modules they use. The Inception Resnet versions incorporate residual connections in their modules.

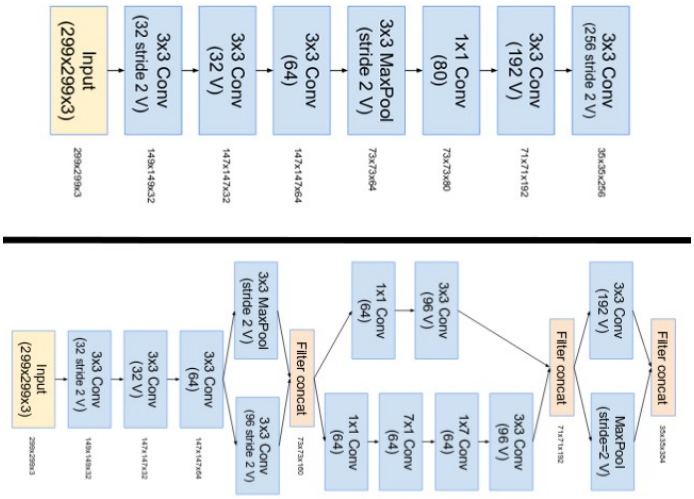


Figure 2.10: Top is the stem for Inception-Resnet V1, bottom is the stem for Inception-V4 and Inception Resnet V2. [21]

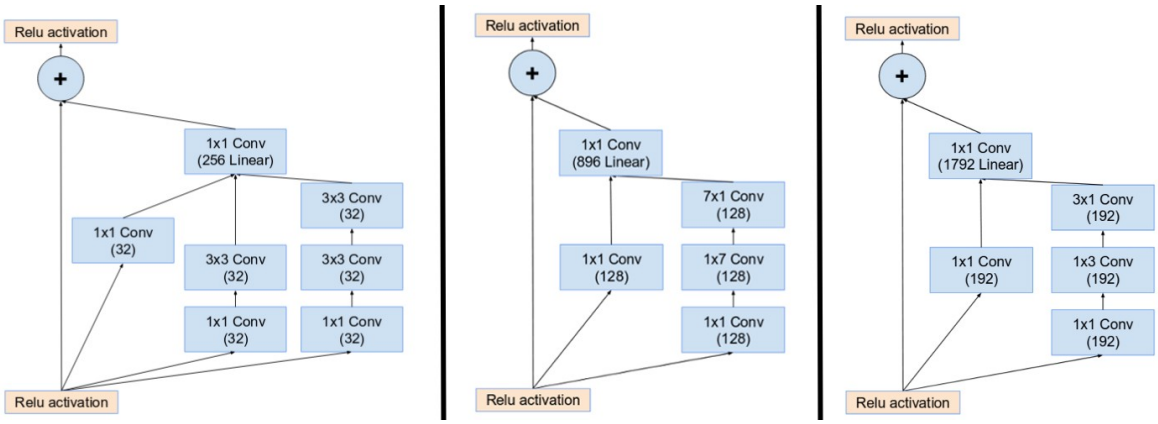


Figure 2.11: Inception modules A, B, and C for Inception Resnet. [21]

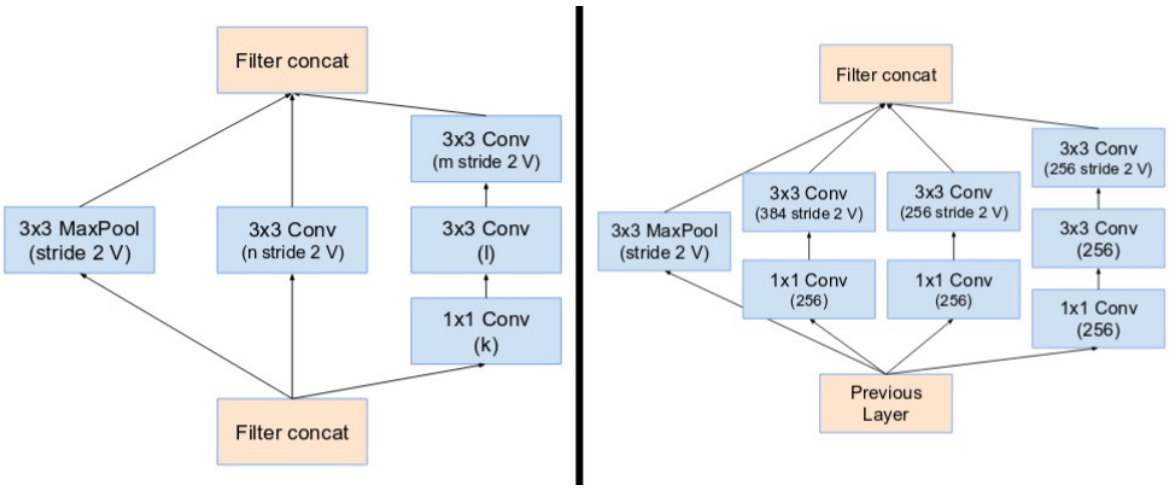


Figure 2.12: Reduction blocks for Inception Resnet. [21]

For network stability, activation scaling is used for residual layers deeper in the architecture.

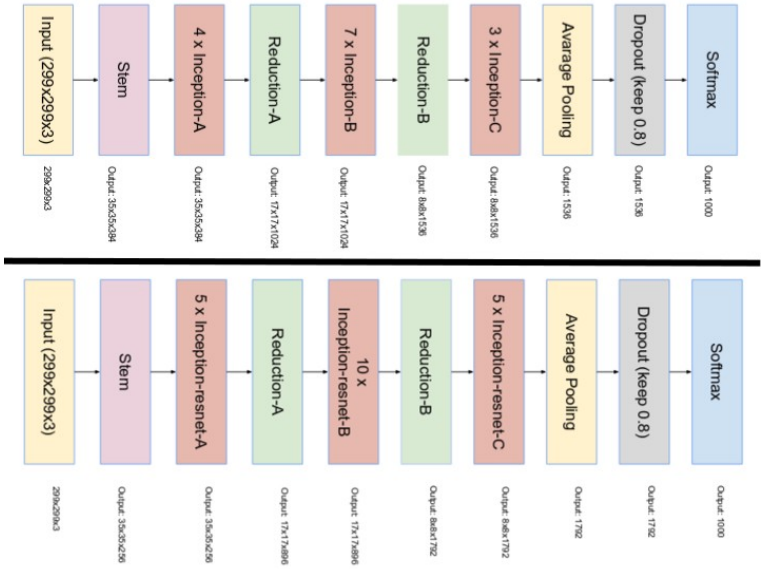


Figure 2.13: Top is architecture for Inception V4, bottom is architecture for Inception Resnet V2. [21]

Out of the three models, Inception Resnet V2 is the deepest and the differs the most from the other two. It requires twice the memory and computation compared to Inception V3. However, it outperforms Inception V3 on the ILSVRC 2012 image classification benchmark with a top-5 accuracy of 95.3%(with 93.9% for Inception V3) and top-1 accuracy of 80.4%(compared to 78.0%).

Finally, as before, the concatenation of these models was also evaluated by averaging their predictions(ensemble model is fed in ART with the argument $use\_logits = True$ and into the Foolbox with $predicts =$ "$logits$").

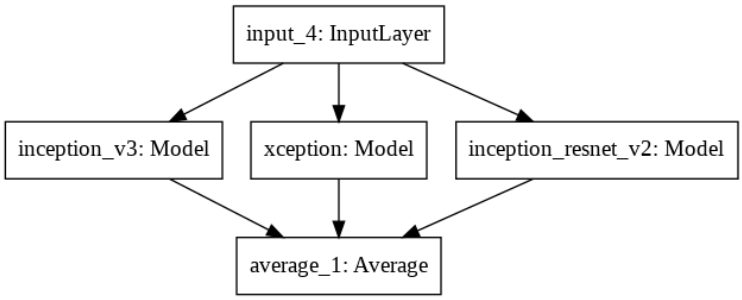


Figure 2.14: Inception Ensemble.

2.3 Adversarial Attack Frameworks

Three out-of-the-box adversarial attacks frameworks were tested in this work. They are still in development and the versions used for the experiments can be found on my **github repo**. The libraries were pulled from my repo onto Google Colab, which uses Tesla K80 GPU and Jupyter Notebooks for easy plotting of graphs and images.

2.3.1 Adversarial Robustness Toolbox

The Adversarial Robustness Toolbox (ART) [12] is a tool developed by IBM and supports a variety of attacks and defenses, such as adversarial training, defensive distillation [14], and denoisers. The library was used in its raw format for the experiments, and the version used is again, on my repo.

2.3.2 Foolbox

This library [16] supports a larger number of attacks than ART, but no defenses. Various attacks such as informed noise attacks, gradient based attacks, and attacks optimizing different norms are implemented. This library was modified slightly for the experiment setup on my repo.

2.3.3 Cleverhans

Cleverhans [13] is the most famous framework for adversarial attacks. It was also tested for training the images, however, memory problems were encountered during testing and only ART and Foolbox were used in the end.

Chapter 3

Adversarial Attacks and Defense

In this section we discuss the set of adversarial attacks evaluated in this project. The three frameworks supported a variety of adversarial attacks and defenses, however, due to slow training times which limited the parameter search, and our specific experiment setup, only three attacks were evaluated. Graphs were gathered for The Uniform Noise Attack and The Gaussian Blur attack from Foolbox at the initial stages of the project as well. However, they were not included in the report due to their more random nature. However, a more general evaluation on ensemble performance would have been useful by including these attacks as well, but this was not done due to time limitations. In the end, statistics were gathered for 3 attacks - the Fast Gradient Sign Method, the Basic Iterative Method,and the Carlini Wagner method.

3.1 Fast Gradient Sign Attack Method

The Fast Gradient Sign Method was first introduced in [4] and exploits the linear nature of neural networks. It is an attack which adds noise to the original input by taking the sign of the gradient of the loss function J of a trained model with respect to the inputs, and adding it to the original input. A distortion parameter ε controls how much the input is perturbed. The equation is:

$$x^{adv} = x + \varepsilon * sign(\nabla_x J(x, y_{true}))$$

where x^{adv} is the calculated adversarial image, x is the original image, y_{true} is the true label of the image, and $\nabla_x J$ is the Jacobian of the loss function with respect to the image.

It is a fast and reliable method to find adversarial examples, although not the most widely used for finding realistic looking adversarials, as high-success rate attacks yield more noticeable perturbations than other more advanced methods.

3.2 Iterative Fast Gradient Sign Attack Method

The Iterative Fast Gradient Sign Method takes T gradient steps of magnitude $\alpha = \varepsilon/T$ instead of a single step. It is also called Basic Iterative Method [8] in ART and does not apply clipping after each iteration, although there is an argument for that, which is the algorithm described in [8]. The equations are

$$x_0^{adv} = x$$
$$x_{t+1}^{adv} = x_t^{adv} + \alpha * sign(\nabla_x J(x_t^{adv}, y_{true}))$$

where x_t^{adv} is the adversarial found at iteration t.

3.3 Carlini and Wagner Attack Method

The Carlini and Wagner [1] method is currently one of the strongest adversarial attacks. It was used to break defensive distillation(a method of defense that was able to reduce the success rate of previous attacks' ability to find adversarial examples from 95% to 0.5%). It achieved success rate of 100% on both distilled and undistilled networks for all three norms of the attack. The attack solves the optimization problem of minimizing the distance D between the original image and the adversarial.

$$\begin{aligned} &\text{minimize} \quad D(x, x+\mu) \\ &\text{subject to} \quad f(x+\mu) \leq 0 \end{aligned} \tag{3.1}$$

The optimization function f is

$$f(x+\mu) = max(max\{Z(x+\mu)_i : i \neq t\} - Z(x+\mu)_t, -k)$$

where Z is the softmax output for the most probable class different from the targeted class.

The k parameter controls the confidence with which the misclassification occurs. The attack uses the Adam optimizer and so this allows for the tuning of the learning rate. Several other parameters are available in the frameworks, which did not seem to affect the attack success as much as the learning rate(most contributory) and the confidence constant.

3.4 Spatial Smoothing Defense Method

Spatial smoothing, also known as "blur" or "median filtering" is a standard denoising technique, which runs a filter window over the pixels of the image and selects the median value from that window. It is a simple defense that does not require retraining the models, or training a new model. It has as a parameter the filter size. The median filter is "essentially squeezing features out of the sample by making adjacent pixels more similar"[26].

Chapter 4

Experiments and Evaluation

A large body of literature has so far concentrated on finding attacks and parameters with high success rate for low perturbations. These evaluations may sometimes find adversarial examples with 1 pixel L-infinity distance from the original image(i.e. each pixel of the adversarial is within 1 of the original image). Many papers compare attacks by, for example, listing a very small number of L2/L-infinity norm thresholds for the perturbations and reporting accuracy percentages.
Though useful for evaluation of minimal perturbations, these comparisons are not universally practical due to the wide-spread image formats using integer representation. Converting an adversarially found numpy floating point array to an image is not possible, except if very specific and not widely spread floating point image formats, such as *.tiff* are used. These formats provide high-quality images and are usually employed in niche domains(e.g. X-ray imaging). A majority of common formats such as the *.png* and *.jpg* format use 8-bit integer representation for their color channels and pixel values[0-255]. Many papers take this into account and measure minimal perturbations from images rounded to integer values, though some do not mention it explicitly.
This means that until floating point image formats are widely used, an attacker has to round the pixel values, which could potentially overturn the adversarial training process and make the prediction of the white-box-attacked classifier correct again, which makes evaluation of minimal perturbations not valuable in some cases. The effects of rounding the values may vary. For this work, metrics were gathered on valid images that were mapped back to their original [0-255] ranges. In order to examine transferability fairly, a fair metric is needed. An attack might be more transferable than another just because it changes the image more, not because the changes are necessarily better or stronger. An initial idea for a fair metric was to pick several thresholds for some L-norm distance from the original image and compare the transferability across these thresholds.
This is the method used by many works in the literature. However, a problem with this is that the ranges of perturbations found by different attacks can vary by orders of magnitude, which makes picking appropriate thresholds to evaluate transferability problematic. Table 4.1 shows normalized MSE distances for iterations from runs for some of the different tested attacking algorithms for a sample image using the Foolbox. One can see that different attacks start finding adversarials at different distances(first

entries of the table), some start to converge, and others take different-sized steps.

Saliency Map	Blended Noise	DeepFool	Iterative Gradient	Contrast Reduction
6.07e-04	3.49e-02	7.96e-06	1.09e-03	9.48e-02
6.16e-04	3.50e-02	1.02e-05	1.24e-03	9.50e-02
6.20e-04	3.52e-02	1.04e-05	1.40e-03	9.52e-02
6.21e-04	3.54e-02	1.18e-05	1.54e-03	9.54e-02
6.22e-04	3.55e-02	1.19e-05	1.73e-03	9.56e-02
6.25e-04	3.57e-02	1.19e-05	1.87e-03	9.58e-02
6.29e-04	3.58e-02	1.19e-05	2.05e-03	9.60e-02
6.30e-04	3.60e-02	1.19e-05	2.22e-03	9.62e-02
6.34e-04	3.62e-02	1.19e-05	2.40e-03	9.63e-02
6.39e-04	3.63e-02	1.19e-05	2.57e-03	9.64e-02
6.44e-04	3.65e-02	1.19e-05	2.77e-03	9.66e-02
6.49e-04	3.67e-02	1.19e-05	2.96e-03	9.68e-02
6.54e-04	3.68e-02	1.19e-05	3.16e-03	9.69e-02
...	...	...	...	...
6.58e-02	9.43e-02	1.19e-05	9.62e-02	9.70e-02
6.58e-02	9.46e-02	1.19e-05	9.63e-02	9.72e-02
6.58e-02	9.48e-02	1.19e-05	9.64e-02	9.74e-02
6.58e-02	9.51e-02	1.19e-05	9.65e-02	9.76e-02
6.59e-02	9.54e-02	1.19e-05	9.67e-02	9.78e-02
6.59e-02	9.56e-02	1.19e-05	9.68e-02	9.80e-02
6.59e-02	9.59e-02	1.19e-05	9.70e-02	9.82e-02
6.59e-02	9.62e-02	1.19e-05	9.70e-02	9.84e-02
6.59e-02	9.64e-02	1.19e-05	9.72e-02	9.86e-02
Interrupted	Interrupted	Interrupted	Interrupted	Interrupted

Table 4.1: Normalized MSE for adversarials across iterations

A way to amend this problem is to restrict the comparison to a family of similar algorithms which modify the image in comparable ranges. However, this method is too restrictive in general as new attacks emerge, and as mentioned thresholds for existing attacks are difficult to choose.
Another method is to let an attack run its course and then post-process the adversarial image by clipping it to the closest point on a ball of a given radius away from the original image. This method is how the experiments were set up. For the experiments, a dataset of 496 images, all classified correctly with 0.98 confidence or more than all 5 classifiers, was composed. The idea behind this is that images as far away from the decision boundaries were preferred in order to discourage the effectiveness of random attacks.
The images were trained for the seven classifiers in a white-box way using an untargeted attack(the attack changes the image to any other class instead of a specific target class). Then each of the models was attacked in a black-box manner by simply taking the accuracy of the predictions of the classifiers for the clipped trained images.
The experiment setup and technical details are as follows. The dataset was created by

gathering 5 random classes of images which were passed(total of 497 images 100 images for each class) through the VGG or Inception preprocessing functions. They were center-cropped to have dimensions of 224x224x3 and were ensured to have correct and confident predictions for all five models after the preprocessing. During the course of this work, the Keras version was updated to not allow 224x224 images for the Inception models, but only 299x299. Older Keras versions work for this(e.g. 2.1.5).
For the training stage, the images were passed through the preprocessing functions for the VGG or Inception models. Then, the attacking algorithms were run, and were left to ”wonder around”. After, reverse preprocessing was performed, the images were clipped to minimum of 0, and maximum of 255, in case they were out of the allowed range after the reverse processing operation(as the API of the attacks did not allow for different color channels to have different minimum and maximum values, which is the case for the VGG preprocessed numpy arrays). It was observed that VGGs have a tendency to make the add noise to the images, and Inceptions have the tendency to grey out the images. Then, the images were clipped again to the different L-infinity norms, as described below. Finally, the images were rounded to the nearest integer and results are gathered. The clipping procedure used was numpy's function ”clip”.
For this setup a strong attack which fools most of the images in the source model had to be found right after preprocessing. Parameter search for the attacks was done manually, due to slow training times that discouraged standard parameter search procedures. Different L-infinity clipping ranges - 0, 5, 10, 15, until 150 (therefore each pixel must be at a maximum of 0, 5, 10, 15, ... 150 distance from the original image - L-infinity norm) were used in the presented results. In addition to evaluating the adversarial accuracy using the standard networks, an interesting question is how transferable adversarial examples are when on trained on an ensemble of models. Can an attacker merge the 5 models in parallel and in this way fool each 5 of them?
This question, however, was not answered, because it was not clear how to resolve the different preprocessing methods used for the 2 families of models(the VGG family and the Inception family). In order to do this, 2 versions of the images had to be fed into the network, which was not allowed in the toolboxes' API. In addition to this, 2 versions of the adversarials would have had to have been trained, and experiments would have had to have been conducted for both versions of the input. This means that this type of investigation is not as valuable as there may be many different preprocessing methods and different versions of the adversarials would need to be evaluated on a case-by-case basis. Rather, a more useful investigation would be to use CNNs from different families of models, but that use the same preprocessing, like VGGs and ResNets.
Instead, ensembles of the VGGs(VGG16 and VGG19) and Inceptions(Inception V3, Xception and Inception Resnet V2) were used, as mentioned. The ensembles also classified all images correctly in the beginning due to each image being classified correctly with more than 0.98 confidence than all 5 models.
For the Median Filtering(Spatial Smoothing) defense, results are also shown for different number of window sizes. The filter is applied on the final adversarial images (preprocessed, attacked, postprocessed, clipped to min of 0, max 255, clipped to the L-infinity range, then rounded). Therefore the defense is again evaluated on the different L-infinity ranges for different window sizes. The plots in sec 4.1 show the different defensive capabilites of the networks for the various attacks - how one specific model

defends against the other networks. On the x axis is the clip range, and on the y the percentage of accurately classified images.
The tables shown before the plots for each attack show the different classifiers' accuracies in percentages right after the preprocessing and the attack end, and before the postprocessing operations.

4.1 L-Infinity Accuracies

4.1.1 FGSM - ART

For the L-Infinity FGSM implementation, ART was used with the parameter $\varepsilon = 100$.

VGG16:	0.0%
VGG19	0.6%
Inception V3	0.0%
Xception	0.0%
Inception Resnet V2	0.0%
VGG Ensemble	0.0%
Inception Ensemble	0.0%

Table 4.2: FGSM - Classifier accuracy before postprocessing (smaller numbers indicate a more successful attack)

4.1.2 I-FGSM - ART

The parameters used were a step size of 1, and number of iterations of 100 to match FGSM's distortion. The attack optimizes L-Infinity as well.

VGG16:	1.609%
VGG19	2.01%
Inception V3	0.0%
Xception	0.0%
Inception Resnet V2	0.0%
VGG Ensemble	1.207%
Inception Ensemble	0.0%

Table 4.3: I-FGSM Accuracy before postprocessing

4.1.3 Carlini Wagner - Foolbox

For this L2-norm version of the attack, the Foolbox was used(as the ART L-Infinity version was too slow) with parameters *max iterations* $= 100$, *learning rate* $= 2$, and

confidence = 50. The rest of the parameters were left to default at *initial const* = $1e-2$ and *binary search steps* = 5.

VGG16:	1.609%
VGG19	2.01%
Inception V3	0.20%
Xception	0.0%
Inception Resnet V2	0.80%
VGG Ensemble	1.00%
Inception Ensemble	23.34%

Table 4.4: Carlini and Wagner with learning rate 2. Accuracy before postprocessing.

Due to the low success rate of the attack for the Inception Ensemble, the learning rate was increased to 7. Plots are shown for both learning rates.

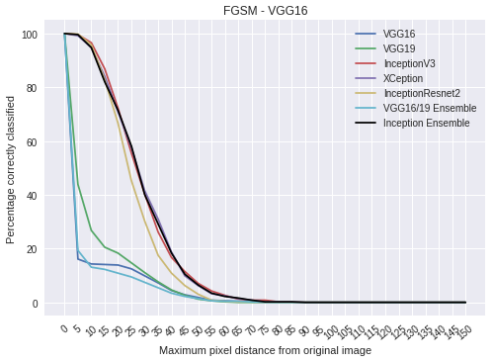
VGG16	0.0%
VGG19	0.0%
Inception V3	0.0%
Xception	0.0%
Inception Resnet V2	0.0%
VGG Ensemble	0.0%
Inception Ensemble	0.603%

Table 4.5: Carlini and Wagner with learning rate 7. Accuracy before postprocessing.

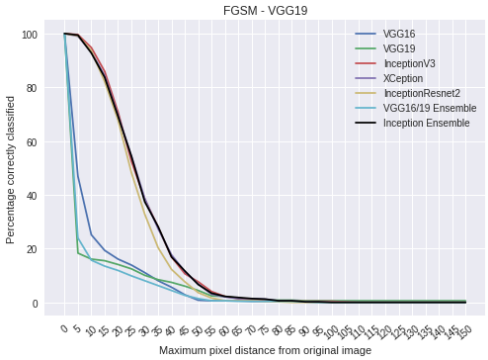
4.1.4 Spatial Smoothing Defense - ART

The Spatial Smoothing defense can be applied on each image set of different clipping range for the 3 attacks. Therefore, in addition to the attacking plots, three sets of plots are shown for the three(using C&W with LR 7) different attacks, where the defense is applied with different window sizes(2-8), after which the results are rounded, and results are gathered for the defensive accuracy of the VGG16 model. This will show whether this defense is effective against the 3 attacks in bringing down the percentage of images which successfully fool the VGG model.

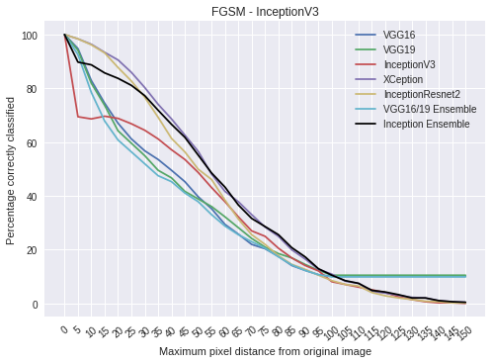
The results can be found below.



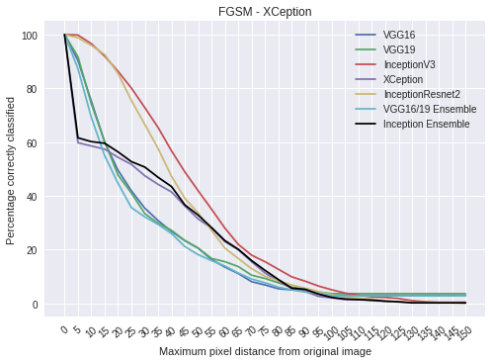
VGG16 attacked by FGSM



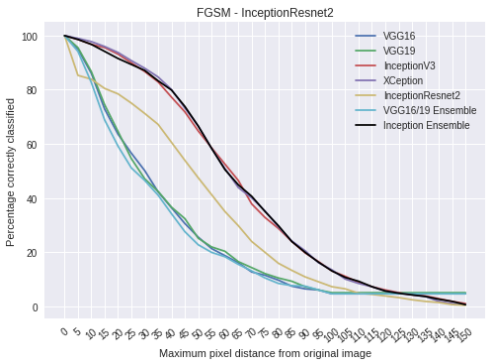
VGG19 attacked by FGSM



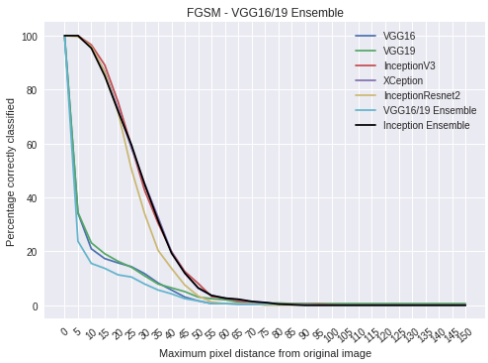
Inception V3 attacked by FGSM



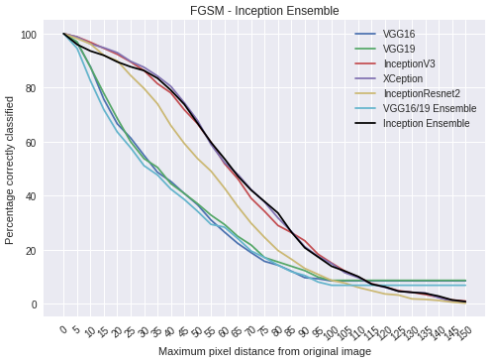
Xception attacked by FGSM



Inception Resnet V2 attacked by FGSM

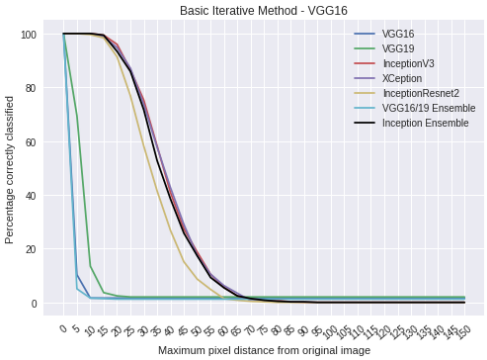


VGG Ensemble attacked by FGSM

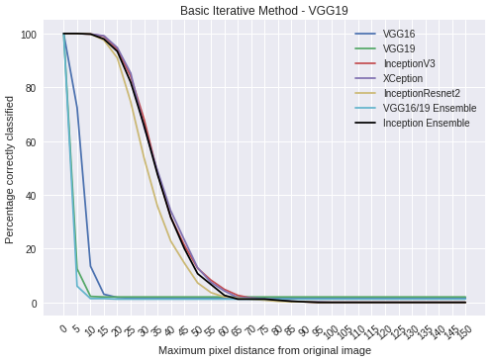


Inception Ensemble attacked by FGSM

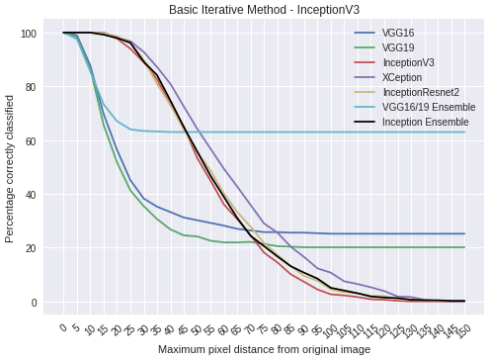
Figure 4.1: Defensive accuracies for FGSM using L-Infinity metric



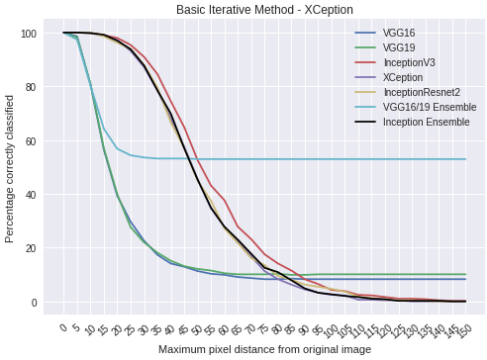
VGG16 attacked by I-FGSM



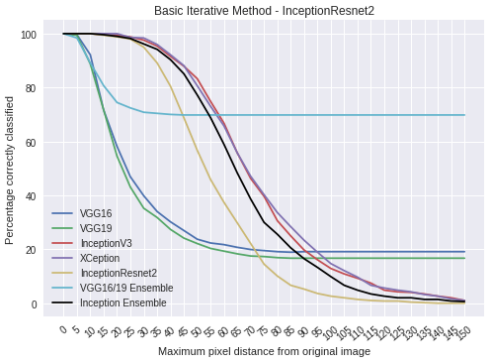
VGG19 attacked by I-FGSM



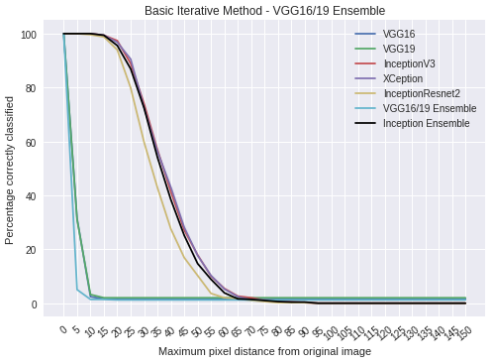
Inception V3 attacked by I-FGSM



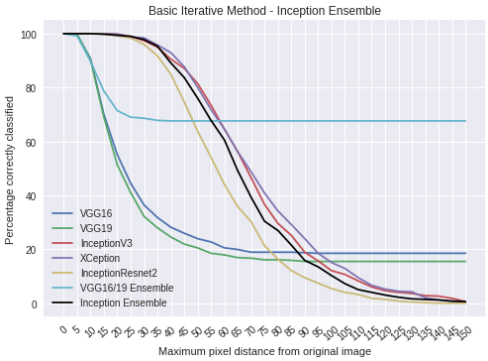
Xception attacked by I-FGSM



Inception Resnet V2 attacked by I-FGSM

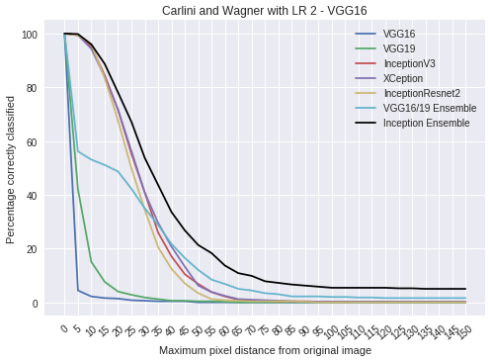


VGG Ensemble attacked by I-FGSM

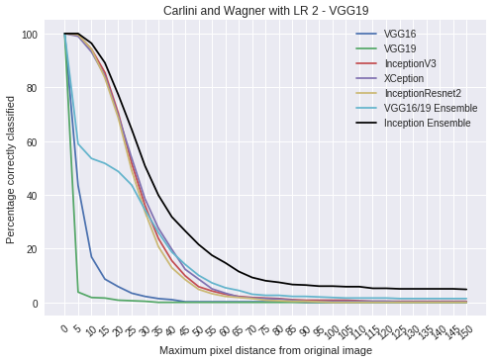


Inception Ensemble attacked by I-FGSM

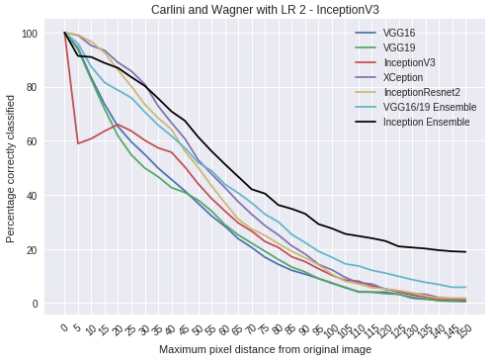
Figure 4.2: Defensive accuracies for I-FGSM using L-Infinity metric



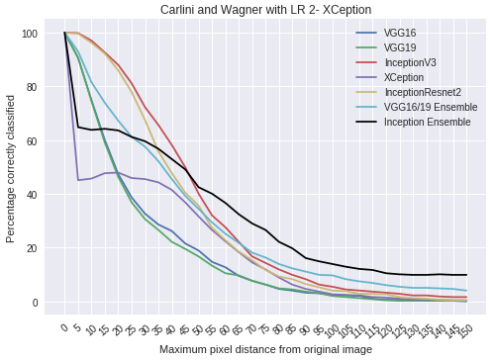
VGG16 attacked by C&W



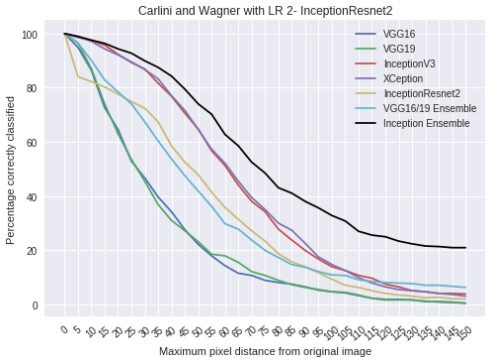
VGG19 attacked by C&W



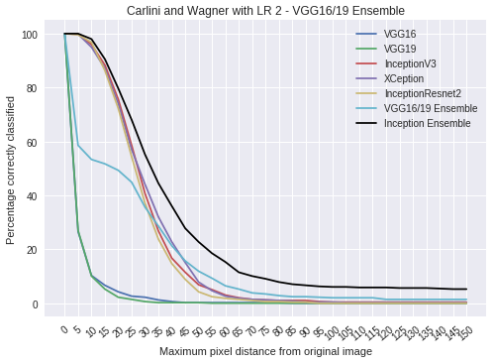
Inception V3 attacked by C&W



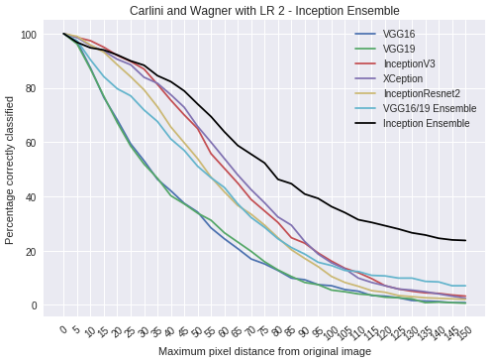
Xception attacked by C&W



Inception Resnet V2 attacked by C&W

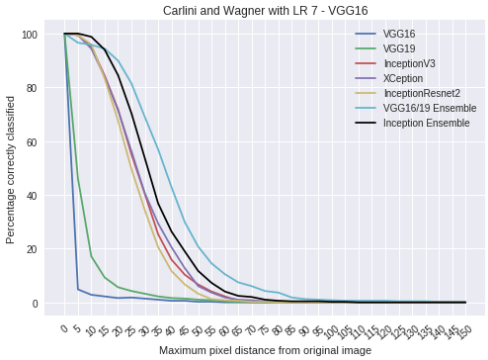


VGG Ensemble attacked by C&W

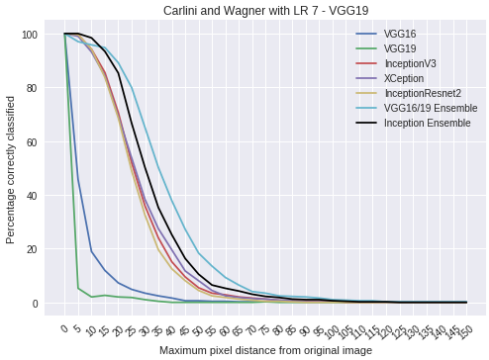


Inception Ensemble attacked by C&W

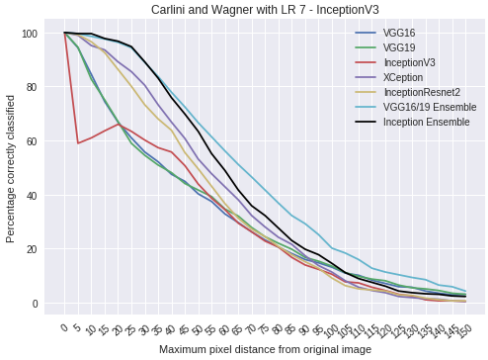
Figure 4.3: Defensive accuracies for C&W with LR 2 using L-Infinity metric



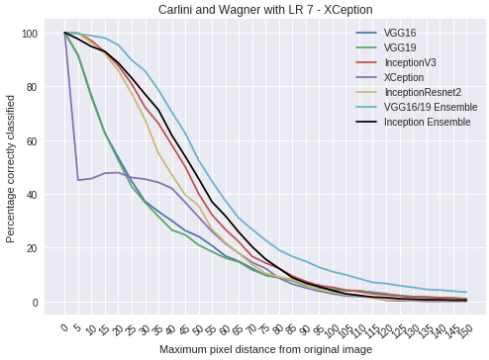
VGG16 attacked by C&W



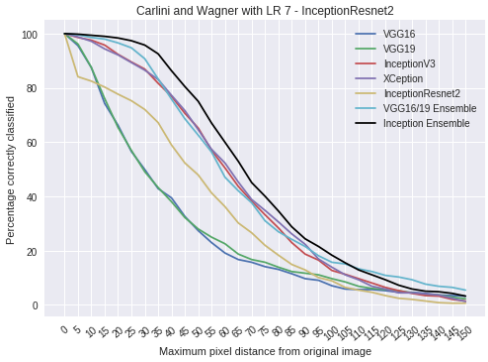
VGG19 attacked by C&W



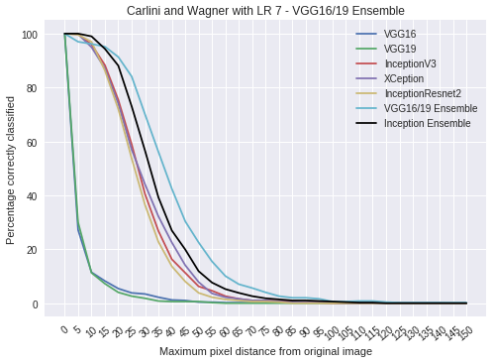
Inception V3 attacked by C&W



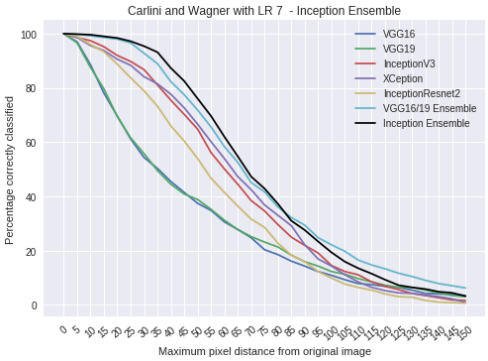
Xception attacked by C&W



Inception Resnet V2 attacked by C&W

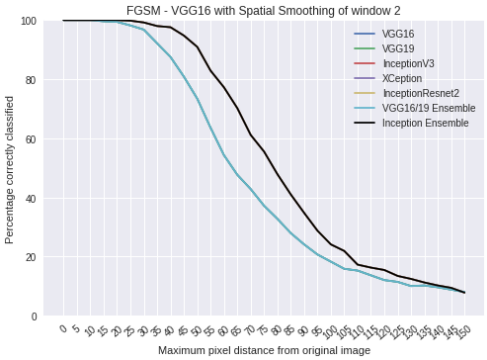


VGG Ensemble attacked by C&W

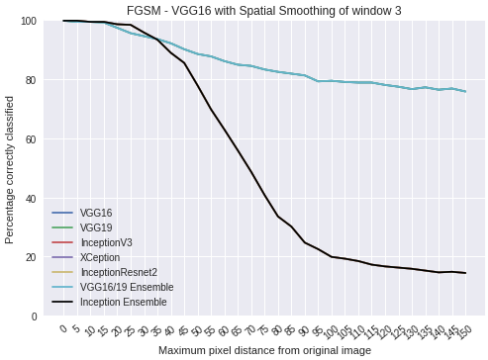


Inception Ensemble attacked by C&W

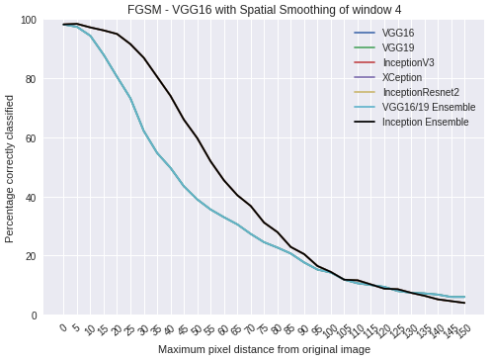
Figure 4.4: Defensive accuracies for C&W with LR 7 using L-Infinity metric



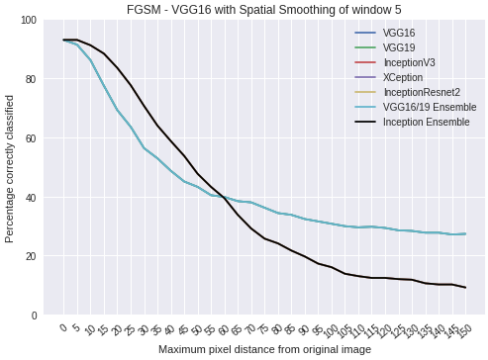
Spatial Smoothing of Window 2



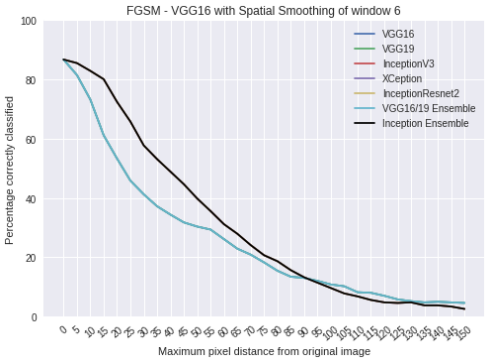
Spatial Smoothing of Window 3



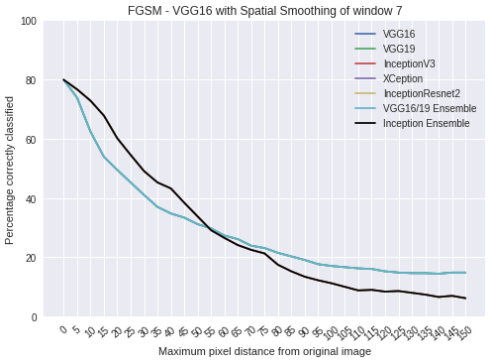
Spatial Smoothing of Window 4



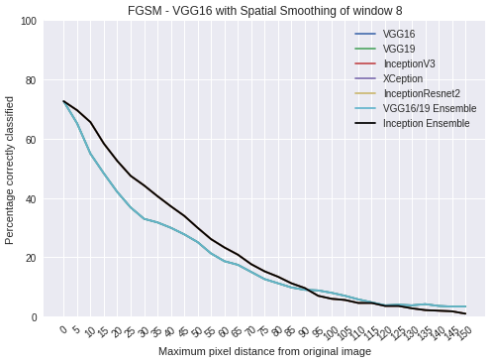
Spatial Smoothing of Window 5



Spatial Smoothing of Window 6

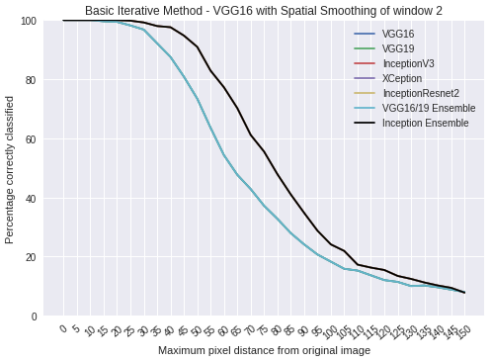


Spatial Smoothing of Window 7

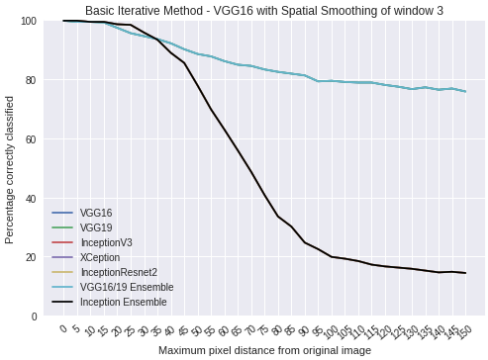


Spatial Smoothing of Window 8

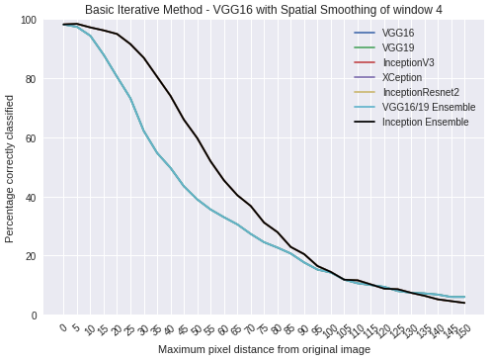
Figure 4.5: Defensive accuracies for VGG16 attacked by FGSM with Spatial Smoothing using L-Infinity metric



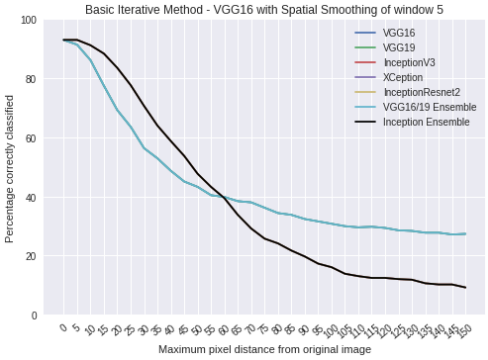
Spatial Smoothing of Window 2



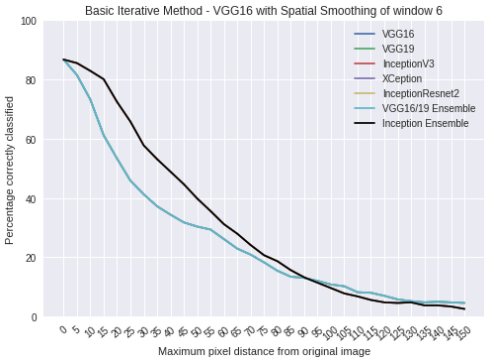
Spatial Smoothing of Window 3



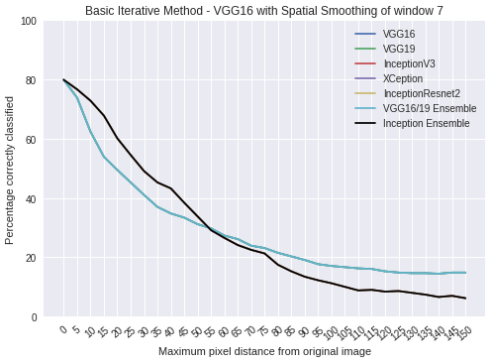
Spatial Smoothing of Window 4



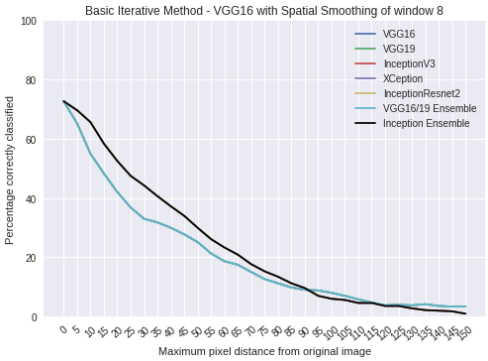
Spatial Smoothing of Window 5



Spatial Smoothing of Window 6

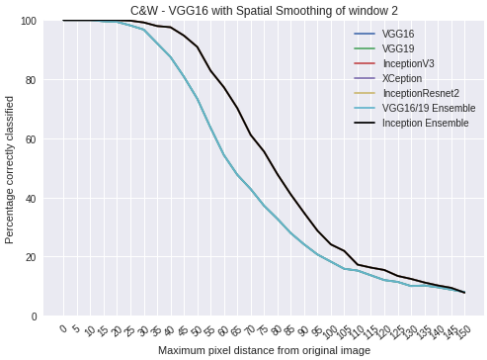


Spatial Smoothing of Window 7

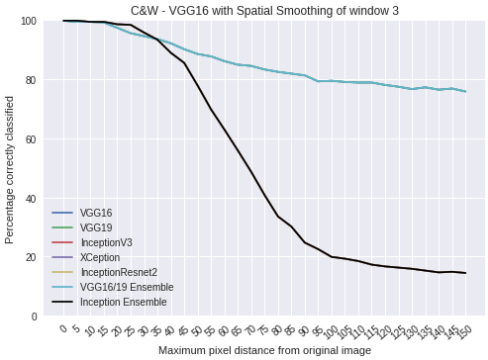


(a) Spatial Smoothing of Window 8

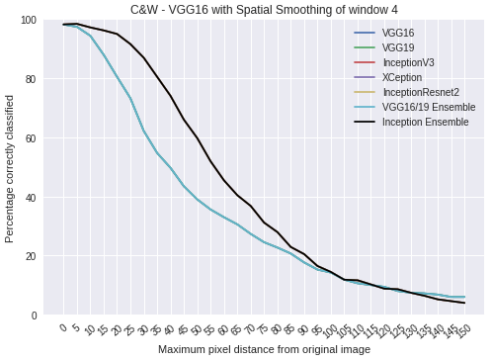
Figure 4.6: Defensive accuracies for VGG16 attacked by I-FGSM with Spatial Smoothing using L-Infinity metric



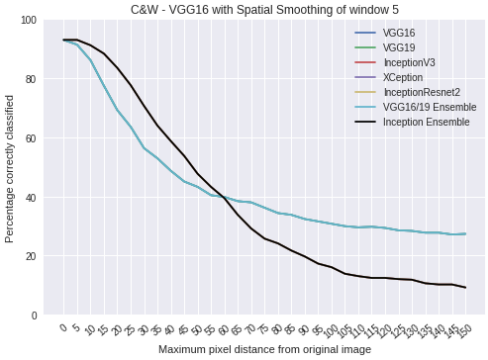
Spatial Smoothing of Window 2



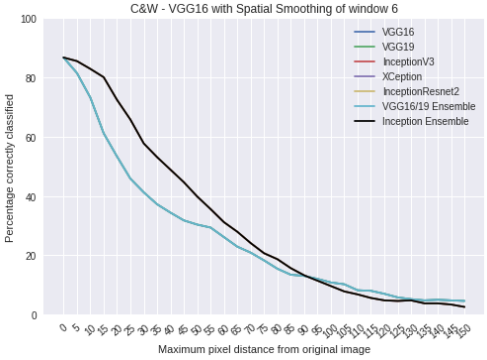
Spatial Smoothing of Window 3



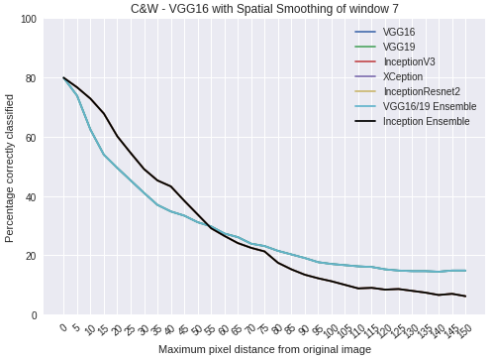
Spatial Smoothing of Window 4



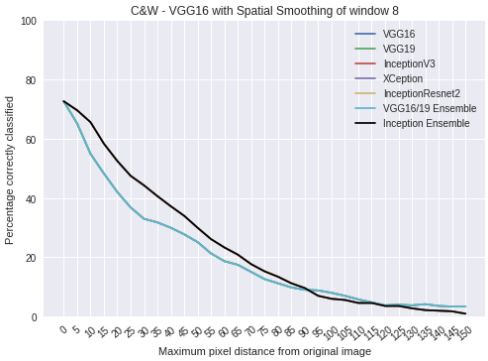
Spatial Smoothing of Window 5



Spatial Smoothing of Window 6



Spatial Smoothing of Window 7



(a) Spatial Smoothing of Window 8

Figure 4.7: Defensive accuracies for VGG16 attacked by C&W with LR 7 with Spatial Smoothing using L-Infinity metric

4.2 Short Analysis and Visual Metrics Motivation

We observe that the spatial smoothing defense is able to effectively reduce the success rate for all of the attacks(fig. 4.5, 4.6, 4.7). Moreover, the smoothing graphs for all 3 attacks are the same, which implies that this defense is equally effective against the 3 attacks in this setting. We also find that the defense ”squeezes” the adversarials found by similar models onto the same representation, which is why there are only 2 lines visible in the plots(the lines for VGG16, VGG19, and the VGG Ensemble overlap, and the lines for Inception V3, Xception, Inception Resnet 2, and the Inception Ensemble overlap).
However, due to the blurring of the image, the distortion induced by the smoothing might be problematic in some applications. Furthermore, since the L-Infinity metric is not indicative of noticeable visual perturbation(since an adversarial which has 1 pixel distorted by a value of 100 is the same as an adversarial which has all of its pixels distorted by a value of 100 under the L-Infinity metric), we investigate different visual metrics in order to replot the results with metrics which are more correlated with human perception. The L-Infinity plots are kept for reference on how accuracy percentages are gathered for the results shown in sec. 4.4 and analysis is deferred until sec. 4.5.

4.3 Visual Metrics

This section presents the metrics considered to reorganise the results found so far.

4.3.1 Inception Score

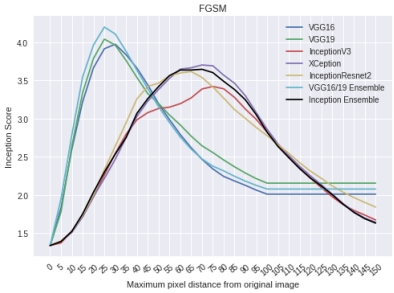
The first tested visual metric is the Inception Score (IS), which was first introduced in [17] and serves as a measure for the performance of Generative Adversarial Networks (GAN). The IS takes a list of images as input and returns a score. The IS was measured using **this github implementation** where the lists of the clipped images of the different various L-Infinity clipping ranges were passed through the scoring function. After examining the images, the perturbation for the clipping ranges appeared gradual, but there was no pattern to the IS(fig. 4.8). Therefore the metric was not chosen for revision of the results.

4.3.2 Mean Absolute Distance (MAD)

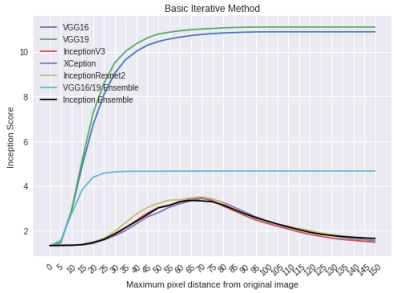
Commonly used metrics in the literature to measure minimum perturbations of adversarial examples include the Mean Squarred Distance(MSD) and Mean Absolute Distance(MAD). MAD was chosen as another potential metric for revising the results (fig. 4.9) and measures the mean absolute difference between the clipped images and the originals.

4.3.3 Structural Similarity Index (SSIM)

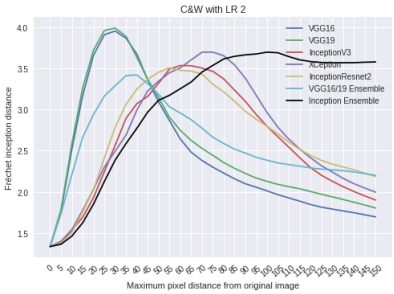
The final metric tested was the SSIM(fig. 4.10) and can be used to measure the quality degradation between 2 images, commonly reported from 1(no degradation) to 0 or from 100 to 0. This measure has been mentioned in the image analysis literature to correlate more with human perception than MAD[25], therefore it was chosen for revision of the results (4.4). **Scikit-image** was used for the calculation of the SSIM.



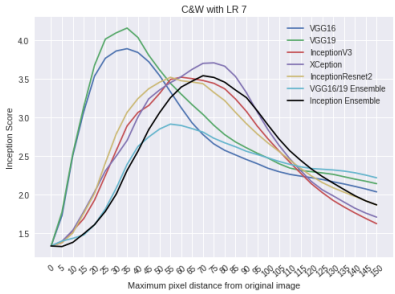
Inception scores for FGSM



Inception scores for I-FGSM

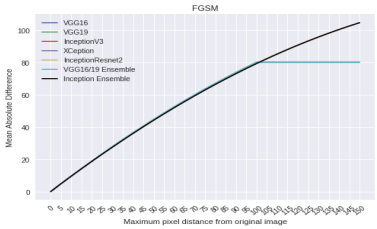


Inception scores for C&W with LR 2

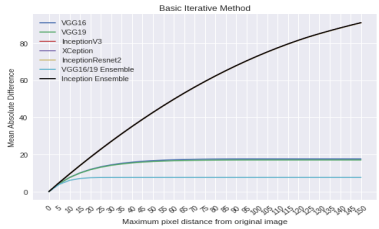


Inception scores for C&W with LR 7

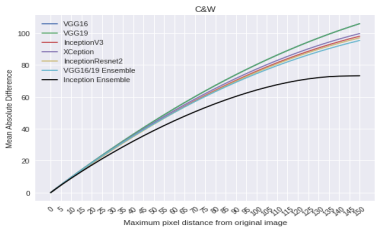
Figure 4.8: Inception scores for the clipped images for the 4 attacks



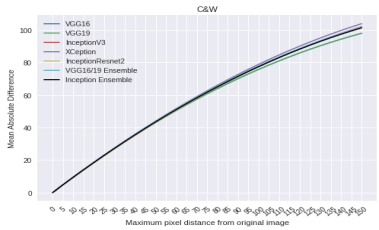
MAD for FGSM



MAD for I-FGSM

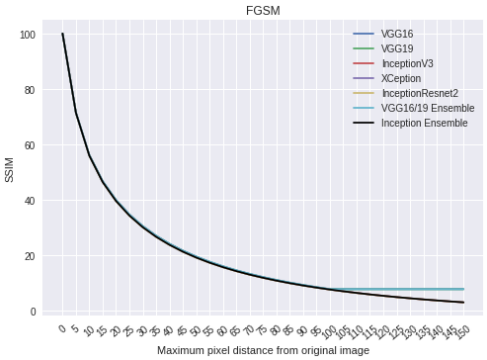


MAD for C&W with LR 2

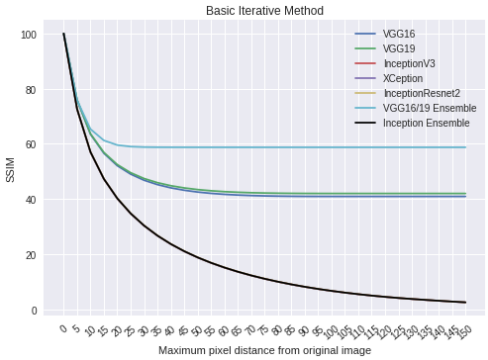


MAD for C&W with LR 7

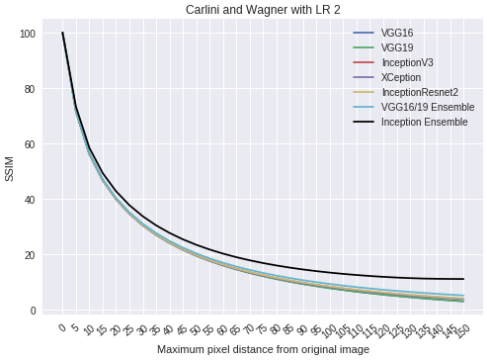
Figure 4.9: Mean Absolute Distance(MAD) for the clipped images for the 4 attacks



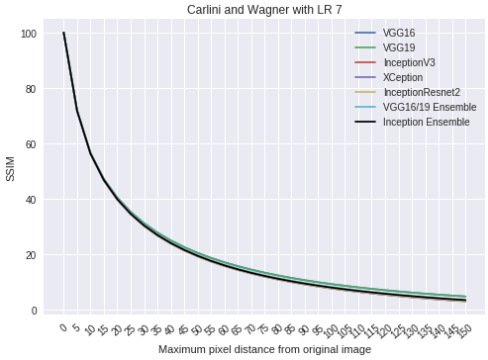
SSIM for FGSM



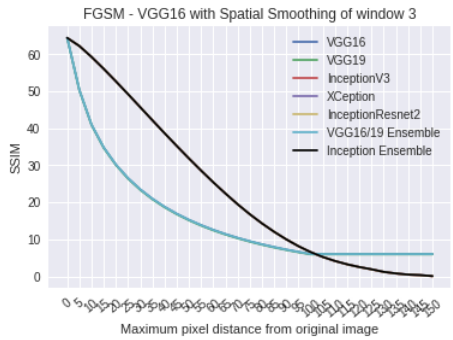
SSIM for I-FGSM



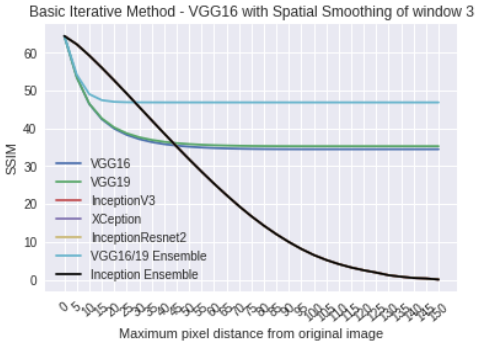
SSIM for C&W with LR 2



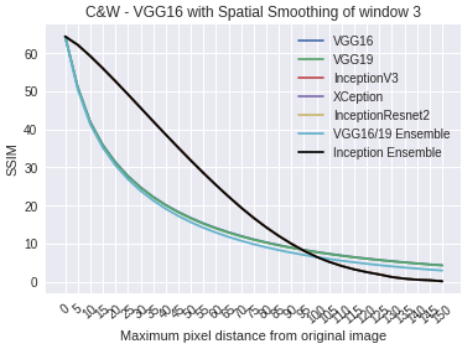
SSIM for C&W with LR 7



SSIM for FGSM with Smoothing window 3



SSIM for I-FGSM with Smoothing window 3

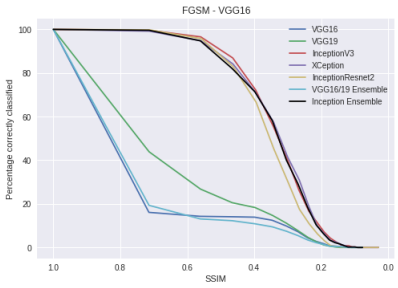


SSIM for C&W LR 7 with Smoothing window 3

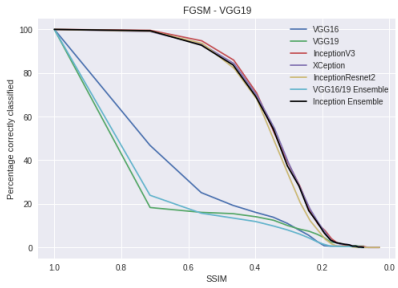
Figure 4.10: Average Structural Similarity Index(SSIM) for the clipped images for the 4 attacks and the Spatial Smoothing defense

4.4 SSIM Accuracies

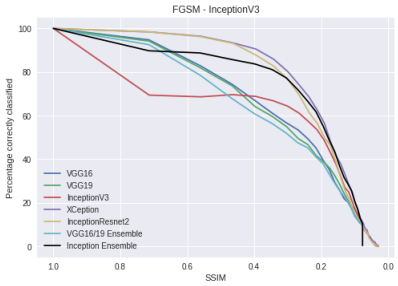
The L-Infinity plots are revised so that the x-axis shows the SSIM from 1 to 0.



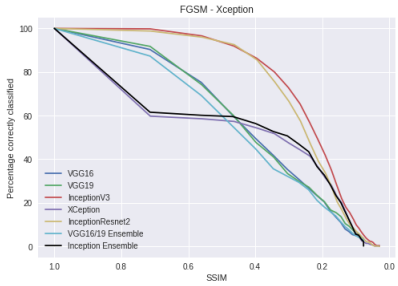
VGG16 attacked by FGSM



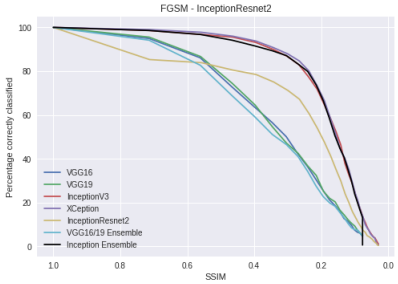
VGG19 attacked by FGSM



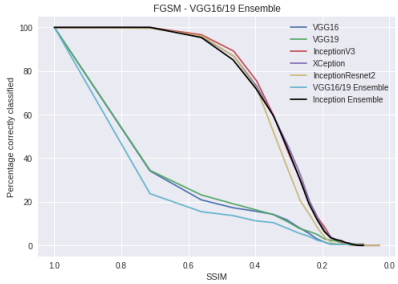
Inception V3 attacked by FGSM



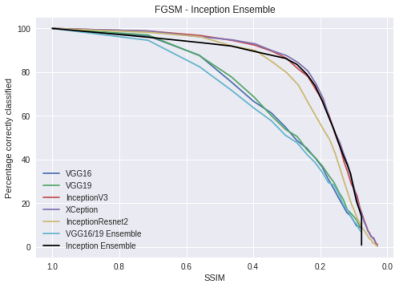
Xception attacked by FGSM



Inception Resnet 2 attacked by FGSM

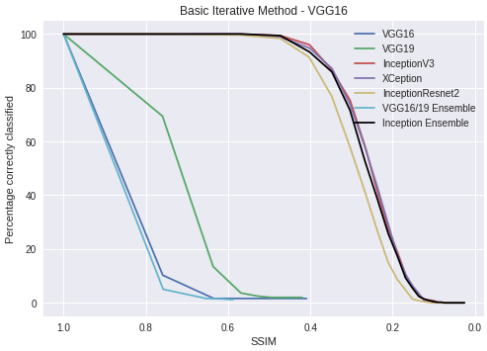


VGG Ensemble attacked by FGSM

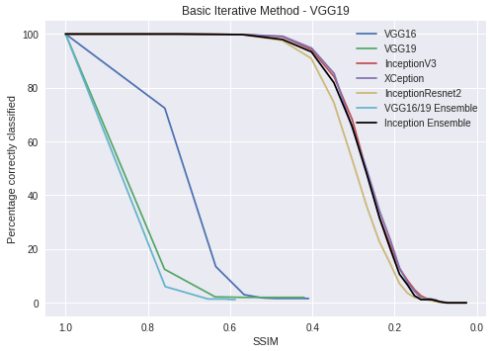


Inception Ensemble attacked by FGSM

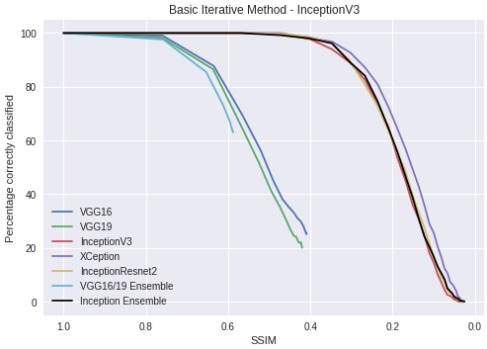
Figure 4.11: Defensive accuracies for FGSM using SSIM to calibrate the x-axis



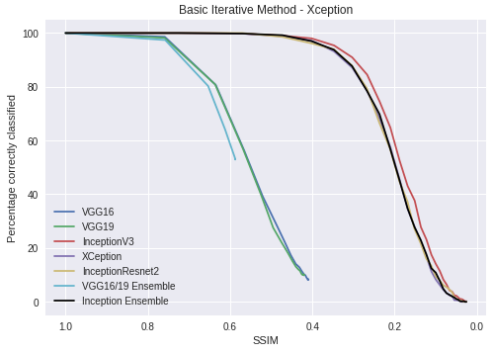
VGG16 attacked by I-FGSM



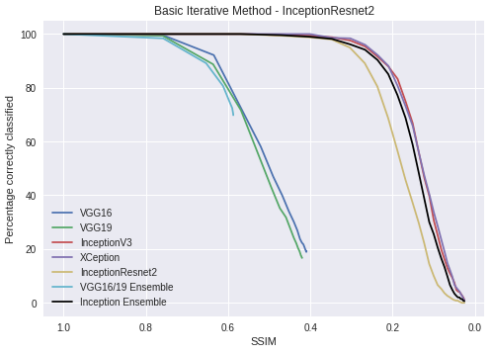
VGG19 attacked by I-FGSM



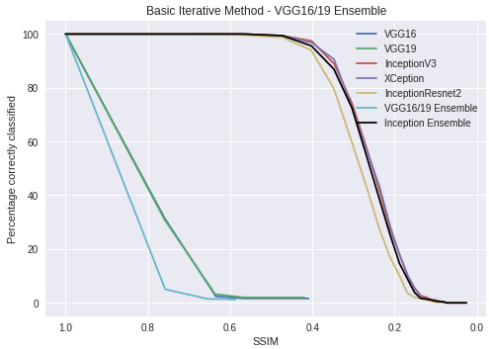
Inception V3 attacked by I-FGSM



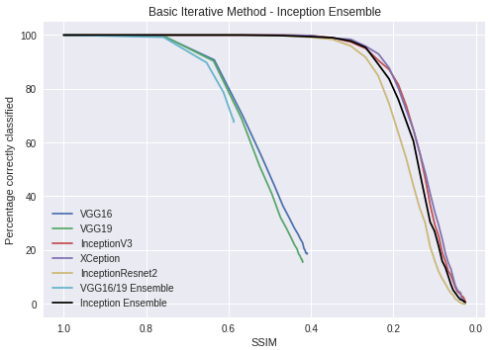
Xception attacked by I-FGSM



Inception Resnet 2 attacked by I-FGSM

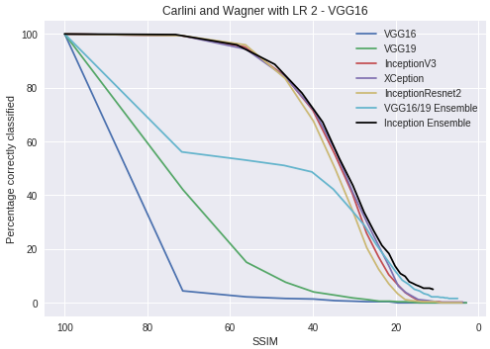


VGG Ensemble attacked by I-FGSM

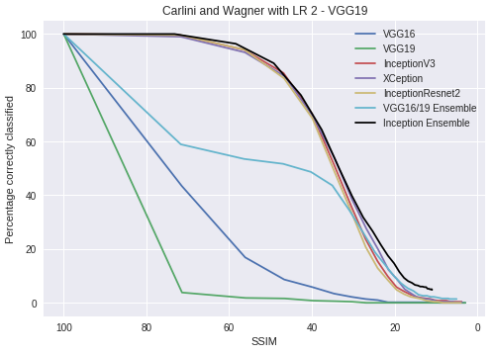


Inception Ensemble attacked by I-FGSM

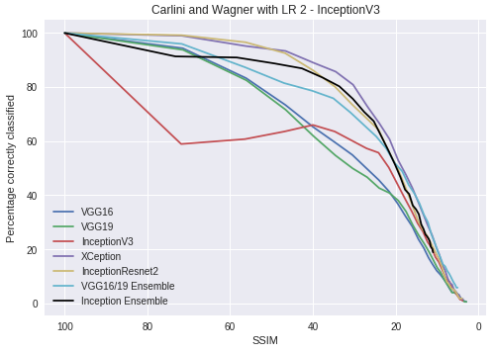
Figure 4.12: Defensive accuracies for I-FGSM using SSIM to calibrate the x-axis



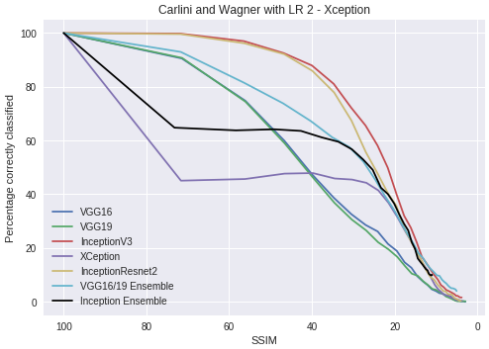
VGG16 attacked by C&W with LR 2



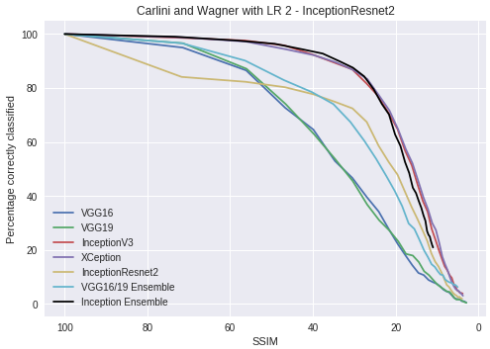
VGG19 attacked by C&W with LR 2



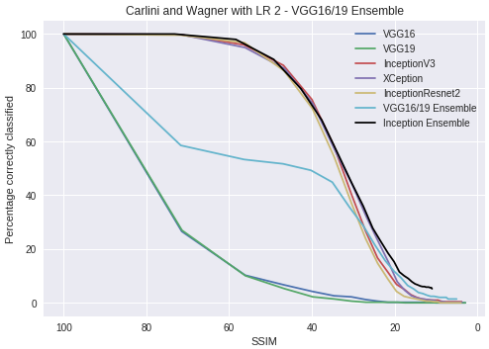
Inception V3 attacked by C&W with LR 2



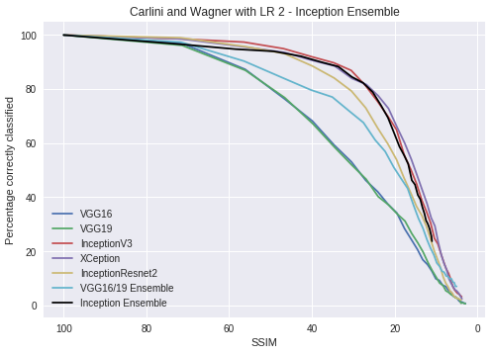
Xception attacked by C&W with LR 2



Inception Resnet 2 attacked by C&W with LR 2

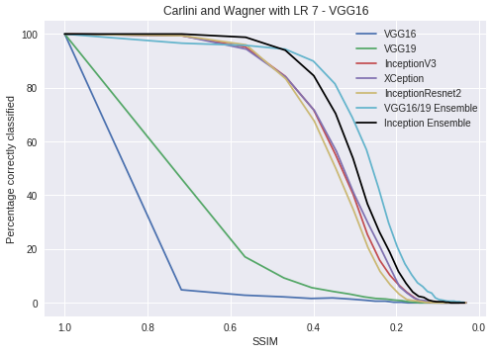


VGG Ensemble attacked by C&W with LR 2

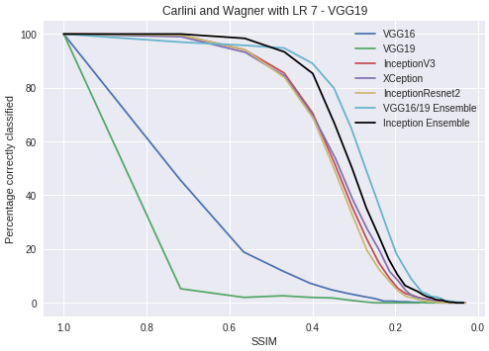


Inception Ensemble attacked by C&W with LR 2

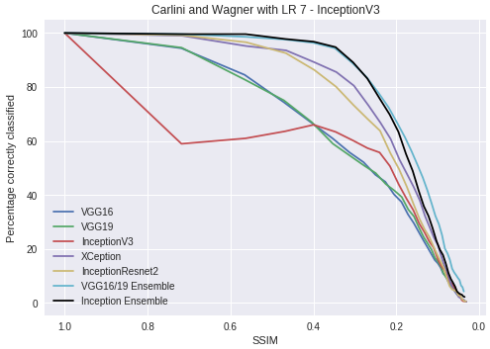
Figure 4.13: Defensive accuracies for C&W with LR 2 using SSIM to calibrate the x-axis



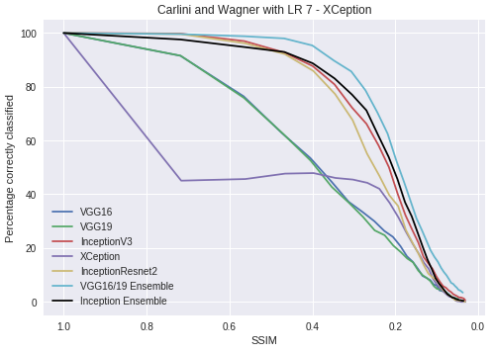
VGG16 attacked by C&W with LR 7



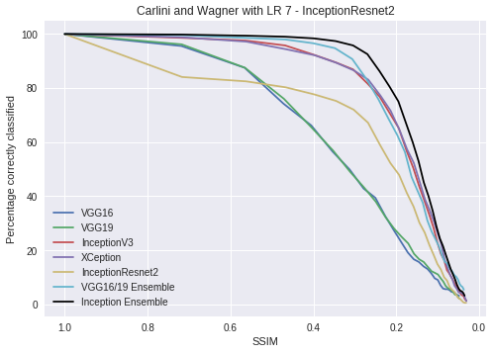
VGG19 attacked by C&W with LR 7



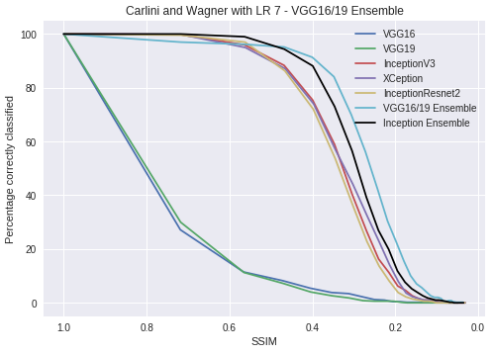
Inception V3 attacked by C&W with LR 7



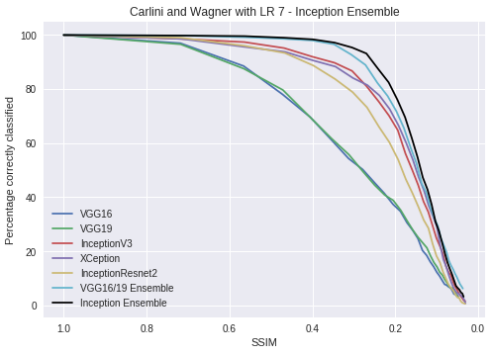
Xception attacked by C&W with LR 7



Inception Resnet 2 attacked by C&W with LR 7

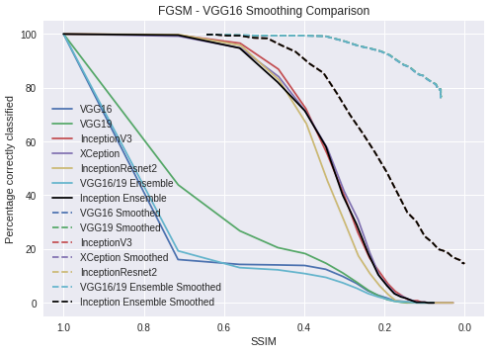


VGG Ensemble attacked by C&W with LR 7

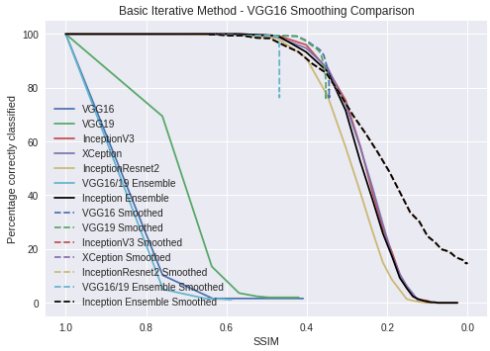


Inception Ensemble attacked by C&W with LR 7

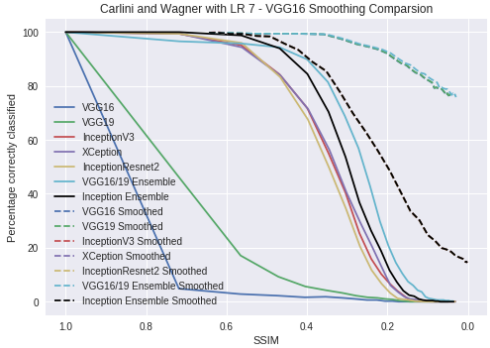
Figure 4.14: Defensive accuracies for C&W with LR 7 using SSIM to calibrate the x-axis



VGG16 attacked by Smoothed FGSM



VGG16 attacked by Smoothed I-FGSM



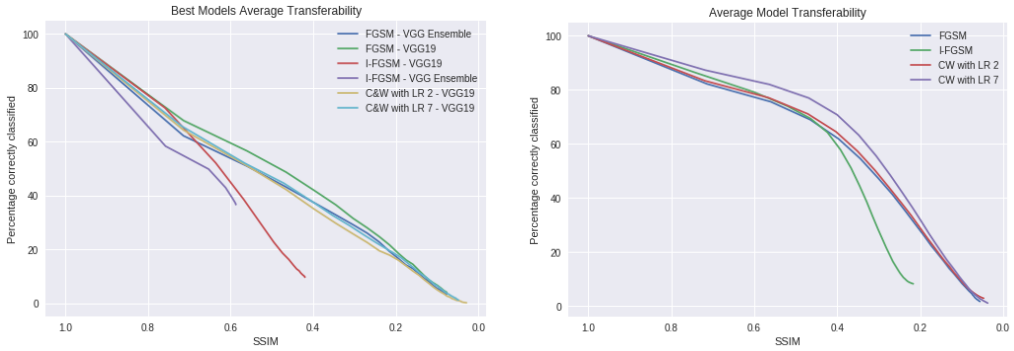
VGG16 attacked by Smoothed C&W with LR 7

Figure 4.15: Comparison of accuracies between attacks with no defense and attacks defended by the Spatial Smoothing method of window 3

4.5 Discussion

Using SSIM analysis we are able to gather results which are more correlated with human perception than when using the L-Infinity metric. However, due to the fact that the pixels of an image are changed in more-or-less uniform ranges(check Appendix), the L-Infinity plots and the SSIM plots are consistent with each other, although differences in performances are more pronounced in the SSIM plots. When assessing attacks such as the Single-pixel attack[20] or ”local” attacks, differences between the L-Infinity and SSIM results would be more substantial. Therefore SSIM is the superior method of evaluation of adversarial attacks in the general case, although it is still weak in evaluating certain classes of attacks such as adversarial scaling or rotations.
We found that, with the exception of the ensembles, for all attacks, similar architectures have similar fooling capacities. The adversarials found by similar architectures appear very similar as well.
For the FGSM attack, the most powerful and transferable model is the VGG-Ensemble, followed by VGG19/16, the Inception Ensemble, Inception Resnet V2, and Inception V3 and Xception respectively. For the I-FGSM attack, the VGG-Ensemble is the most transferable as well for small perturbations. However, due to its quick convergence

it is unable to change the image enough to to generalize to the Inception family for larger ranges of perturbation. After the VGG Ensemble, the remaining most transferable models are VGG19, VGG16, followed by the Inception family. We observe that using an ensemble of Inception models does not improve the success of the attack for both the Inception and the VGG models. For the C&W method with learning rate 2, we observe that the ensemble method does not always bring improvement in performance as well. Furthermore, images trained on VGG16 and 19 are able to fool the ensemble more than images trained on the ensemble itself. However, under this parameter setting, there is improvement in using the Inception Ensemble, when fooling the Inception networks. The most transferable models for this attack are again VGG19, VGG16, followed by the VGG Ensemble, the Inception Ensemble, Inception Resnet V2, and finally Xception and Inception V3. For C&W with LR 7, the ensembles perform the worst of all models. VGG19 and 16 are again the top performing models, followed by Inception Resnet V2, Xception and Inception V3.
Evaluating the Spatial Smoothing defense with window of size 3(chosen by examining the L-Infinity plots), we observe that it is a valid method of defense against black-box attacks. However, it has a very high initial cost under the visual perturbation metric(SSIM of the original images is reduced to about 0.62 after blurring). More denoising defenses can be compared in this way in order to determine which one brings the best accuracy under minimal SSIM loss.
In order to compare the ”best-case” transferability across the attacks, we present the average transferability of the strongest attacking model for each attack across all models, including the source.
In order to compare the average-case transferability, results are averaged across the other models as well, and the averaged results are averaged over each attack. The SSIM ranges of the models for each attack are averaged as well:



Best Models Average Transferability All Models Average Transferability

Figure 4.16: Comparisons of the Transferability of the Three Attacks

The strongest models across the different attacks exhibit similar transferability. For large SSIM values, the I-FGSM VGG Ensemble performs the best, followed by the FGSM Ensemble, Carlini and Wagner's VGG19, FGSM's VGG19, and finally I-FGSM's VGG19. However, in order to increase the percentage misclassified under minimal

SSIM loss, the most appropriate method of attack is to use I-FGSM with VGG19. This is evident by visualisation of the images as well(check Appendix), where the I-FGSM images appear the ”cleanest”. The average transferability for all attacks shows similar results to the best models transferability, most likely influenced by the best models of I-FGSM.

Chapter 5

Conclusion

Transferable adversarial examples are a potential risk to a variety of applications employing machine learning methods by using the examples created using a local model to attack a remote service.
In this work we evaluate the transferability of adversarial examples using the setting of strong class-untargeted attacks for the source models, which introduce a high visual perturbation to the images. We measure the accuracy of the classifications of the source and target models using different sets of L-Infinity clipped images.
The results are presented with respect to both L-Infinity, and SSIM as metrics for visual perturbation. We find that Inception Score is not a reliable way of measuring the quality of adversarial images.
We observe that for all attacks, the VGG models are a stronger attacker than the Inception models.
Furthermore, we find that applying a simple denoising preprocessing defensive step to the adversarial images reduces the effectiveness of the attack at the cost of loss of image quality. We find that this defense ”squeezes” the adversarials found by similar models onto the same representation.
Moreover, ensembles have been used before in the Clarifai.com black-box attack with high success rate[11]. We find that using ensembles of parallel models by averaging their predictions does not improve the attack success for all attack methods with a random parameter setting for the attack.
This method for evaluation of transferability is not exhaustive, but significantly reduces the search space of possible parameters for the attacks by restricting them to only those which make the source misclassify most of the images, without worrying about high visual perturbation, which is amended by the clipping mechanism. Evaluating the algorithms in the presented way could lead to a more systematic and quantifiable way to compare attack tranferability than the methods currently used in the literature.

Chapter 6

Possible future work/extensions

The field of adversarial attacks is rich and there are a lot of options for work in the second part of the MInf project and although no concrete plan has been made yet, these are some ideas for future work:

The current study can be extended by considering more existing adversarial attacks and defenses and comparing their performances alongside the examined methods.

Another option is to attempt to devise a novel method of adversarial attack or defense, and test its performance using a public benchmark or the current dataset. For example, the Lipschitz Regulariser [5] is a new regularizer, which has not been studied yet as a defense against adversarial attacks. By comparing the success rate of different attacks on several neural networks trained with and without the regularizer, this method can be tested for robustness against attacks.

A third option is to gather results on a different dataset or perform a class-based analysis of different attacks. For example, as mentioned in [3], potentially there may be enormous monetary incentives for fraud by using adversarial examples in the healthcare sector. By examining the performance of adversarial attacks with the same attack strength on different classes of images on several neural networks each retrained on the different image sets(e.g. chest X-rays vs Fundoscopy, vs Endoscopy images), one could measure whether some classes of images have geometric properties that make them more vulnerable to attacks, independent of network architecture. Subsequently, for these use-case, one would be more encouraged to keep the human in the loop or consider stronger defenses.

The field of adversarial attacks is not restricted to images. Noise can be added to audio and textual data as well, therefore a similar analysis can be done on these types of data.

Finally, there is also some work done on adversarial examples for robot/agent control in the literature[15]. One could examine how continuous control agents (e.g. humanoid walking) react to different kinds of disturbances and how to improve the robustness of the agents against adversarial noise.

Bibliography

[1] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016.

[2] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.

[3] Samuel G. Finlayson, Isaac S. Kohane, and Andrew L. Beam. Adversarial attacks against medical deep learning systems. *CoRR*, abs/1804.05296, 2018.

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

[5] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael Cree. Regularisation of neural networks by enforcing lipschitz continuity. 04 2018.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[8] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[9] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, pages 319–, London, UK, UK, 1999. Springer-Verlag.

[10] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013.

[11] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *CoRR*, abs/1611.02770, 2016.

[12] Maria-Irina Nicolae, Mathieu Sinn, Tran Ngoc Minh, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Ian M. Molloy, and Benjamin Edwards. Adversarial robustness toolbox v0.2.2. *CoRR*, abs/1807.01069, 2018.

[13] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang,

Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg, Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, and Rujun Long. Technical report on the cleverhans v2.1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2018.

[14] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *CoRR*, abs/1511.04508, 2015.

[15] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. *CoRR*, abs/1712.03632, 2017.

[16] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox v0.8.0: A python toolbox to benchmark the robustness of machine learning models. *CoRR*, abs/1707.04131, 2017.

[17] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.

[18] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv e-prints*, September 2014.

[19] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is robustness the cost of accuracy? - A comprehensive study on the robustness of 18 deep image classification models. *CoRR*, abs/1808.01688, 2018.

[20] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *CoRR*, abs/1710.08864, 2017.

[21] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

[22] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[23] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[24] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

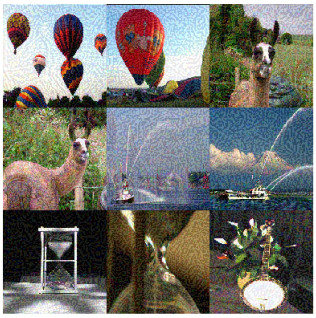
[25] Zhou Wang, Alan Bovik, Hamid Rahim Sheikh, and Eero Simoncelli. Image quality assessment: From error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13:600 – 612, 05 2004.

[26] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *CoRR*, abs/1704.01155, 2017.

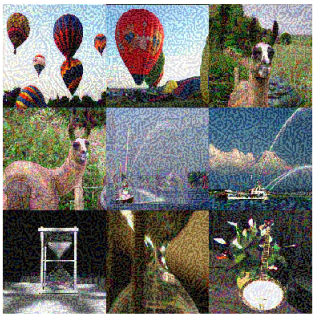
Appendix

Original images

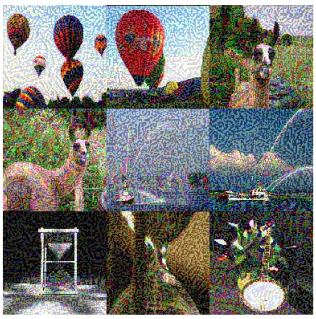
Clip range - 10, avg. transferability - 49%



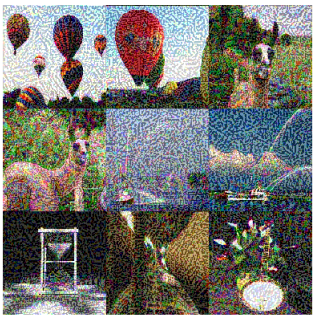
Clip range - 20, avg. transferability - 63%



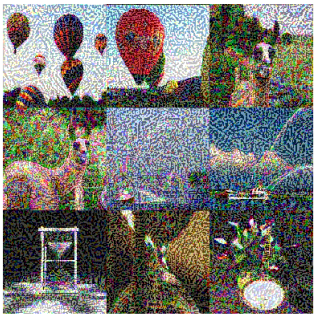
Clip range - 30, avg. transferability - 71%



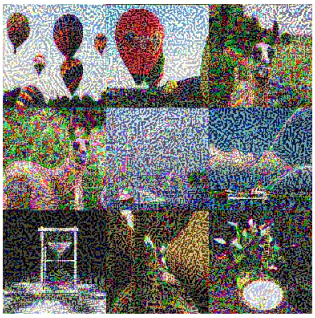
Clip range - 40, avg. transferability - 77%



Clip range - 50, avg. transferability - 83%



Clip range - 60, avg. transferability - 87%



Clip range - 70, avg. transferability - 90%

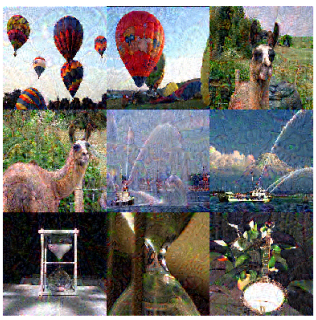
Figure 6.1: Images for the FGSM VGG-Ensemble of different L-Infinity clipping ranges and the average transferability(rounded) of these ranges across all models, including the source

Original images

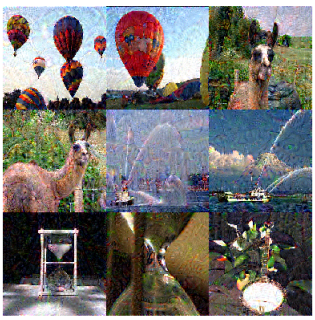
Clip range - 10, avg. transferability - 48%

Clip range - 20, avg. transferability - 70%

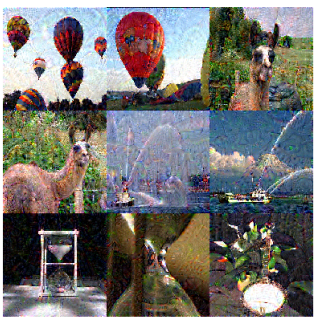
Clip range - 30, avg. transferability - 81%



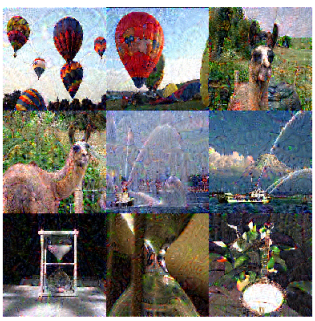
Clip range - 40, avg. transferability - 86%



Clip range - 50, avg. transferability - 88%



Clip range - 60, avg. transferability - 89%



Clip range - 70, avg. transferability - 89%

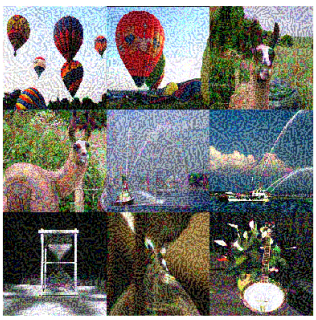
Figure 6.2: Images for the I-FGSM VGG19 of different L-Infinity clipping ranges and the average transferability(rounded) of these ranges across all models, including the source

Original images

Clip range - 10, avg. transferability - 48%

Clip range - 20, avg. transferability - 62%

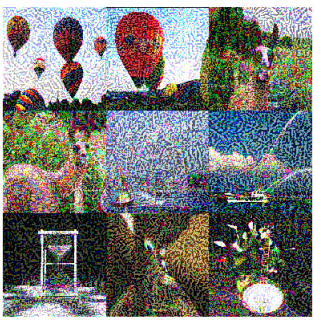
Clip range - 62%, avg. transferability - 71%



Clip range - 40, avg. transferability - 77%

Clip range - 50, avg. transferability - 81%

Clip range - 60, avg. transferability - 85%



Clip range - 70, avg. transferability - 88%

Figure 6.3: Images for the C&W with LR 7, VGG19 of different L-Infinity clipping ranges and the average transferability(rounded) of these ranges across all models, including the source