



UNIVERSIDADE D
COIMBRA

João Miguel Gouveia Leal

**ADVERSARIAL ATTACKS TO CLASSIFICATION
SYSTEMS**

Dissertation in the context of the Master in Informatics Engineering, specialization in Intelligent Systems, advised by Professor João Nuno Gonçalves Costa Cavaleiro Correia and Professor Tiago Filipe dos Santos Martins and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September 2022

Faculty of Sciences and Technology
Department of Informatics Engineering

Adversarial Attacks to Classification Systems

João Miguel Gouveia Leal

Dissertation in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems advised by Professor João Nuno Gonçalves Costa Cavaleiro Correia and Professor Tiago Filipe dos Santos Martins and presented to the Faculty of Sciences and Technology / Department of Informatics Engineering.

September 2022



UNIVERSIDADE D
COIMBRA

This page is intentionally left blank.

Abstract

Adversarial samples are inputs corrupted with inconspicuous perturbations misclassified by a given target model. Adversaries create adversarial samples using various methods that depend on the information available about the target system. In a white-box scenario, adversaries have full access to the model, and in a black-box scenario, usually, only the output layer is accessible. Researchers have developed adversarial samples that can fool target models even when the adversary has almost no information about the target system. To construct classifiers robust to adversarial samples, many authors have proposed adversarial defenses, mechanisms intended to protect deep learning models from adversarial attacks. However, many of these defenses have been shown to fail, which asserts that building robust models is an extremely arduous and complicated task to achieve. Motivated by this, there have been developed frameworks that group various adversarial attacks to allow users to test their models, however, none of them provide a pipeline mechanism and lack enough information about the robustness of the tested models. Various frameworks have also stopped receiving support, leading to frameworks with antiquated attacks and similar attacks between them. In this dissertation, a new framework was developed with a pipeline mechanism that allows users to input their models and to choose from the currently, eight adversarial attacks. After executing the pipeline, each model obtains a score based on its performance against all of the images generated by the adversarial attacks allowing for a better understanding of the robust levels of those same models. To test the validity and capabilities of the framework, an experiment was performed using the pipeline mechanism with models trained using an image classification dataset and the eight supported adversarial attacks. The results obtained allow for a deeper understanding of the robustness of the models. The evaluation of a model shouldn't be based only on the accuracy of the model on the adversarial samples but should take into consideration the amount of perturbation that a sample needs to have to be able to fool the target classifier.

Keywords

Deep Learning, Adversarial Learning, Adversarial Attacks, Robustness, Performance Metrics

This page is intentionally left blank.

Contents

1	Introduction	1
1.1	Motivation and Objectives	1
1.2	Document Structure	1
2	Concepts & Definitions	4
2.1	Introduction to Deep Learning	4
2.2	Explaining Adversarial Attacks	7
2.3	Adversarial Threat Model	9
2.4	Robustness Metrics for Classifiers Evaluation	12
3	State-of-the-Art	14
3.1	Adversarial Attacks	14
3.1.1	Poisoning Attacks	14
3.1.2	Exploratory Attacks	15
3.1.3	Evasion Attacks	18
3.1.4	Real-World Applications	42
3.2	State-of-the-Art Frameworks	43
4	Development Plan	50
4.1	Work Plan	50
4.2	Methodology	51
5	The Framework	53
5.1	Software Requirements Specification	53
5.2	Risk Analysis	55
5.3	Executed Work Plan	57
5.4	Architecture and Implementation	58
5.5	Model Robustness Score	62
6	Experimentation	67
6.1	Settings	67
6.2	Results and Discussion	70
7	Conclusion and Future Work	77

This page is intentionally left blank.

Acronyms

- AI** Artificial Intelligence. 1
- BIM** Basic Iterative Method. 21, 22, 28
- BNN** Bayesian Neural Network. 31
- CLEVER** Cross Lipschitz Extreme Value for nEtnetwork Robustness. 12, 44
- CMA-ES** Covariance Matrix Adaptation Evolution Strategy. 34
- CNN** Convolutional Neural Network. 4–7, 42
- DDN** Decoupled Direction and Norm. 34, 38
- DL** Deep Learning. 1, 2, 4
- DNN** Deep Neural Network. 4, 31
- EAD** Elastic Net Attack. 28
- FGSM** Fast Gradient Sign Method. 20, 21, 24–28
- GPC** Gaussian Process Classifier. 31
- JSMA** Jacobian-based Saliency Map Attack. 20, 42
- L-BFGS** Limited Memory Broyden-Fletcher-Goldfarb-Shanno. 20, 34
- MI** Model Inversion. 17
- ML** Machine Learning. 1, 4, 9–12, 14, 15, 20, 21, 24, 30, 33, 34, 42, 43, 53
- NES** Natural Evolutionary Strategies. 29, 33, 40
- PGD** Projected Gradient Descent. 28, 33, 35, 39, 42
- SVM** Support Vector Machine. 14, 15

This page is intentionally left blank.

List of Figures

2.1	Schematic of a Deep Learning Model with 3 hidden layers. (Image credits:[2])	5
2.2	A visual representation of the gradient descent algorithm optimizing the cost function, where the ball represents the cost function and the arrow represents the direction of the gradient. The cost function is represented by C and the variables of the function by v_1 and v_2 . (Image credits: [3])	6
2.3	Architecture of a Convolutional Neural Network. (Image credits: [4])	7
2.4	Attack surface of a generic machine learning system and, of two machine learning systems used in the physical world (Image credits: [14])	10
3.1	A backdoor attack. The backdoor trigger is a pattern of pixels that appears on the bottom right corner of the image. (a) A benign network that correctly classifies its input. (b) A potential (but invalid) BadNet that uses a parallel network to recognize the backdoor trigger and a merging layer to generate misclassifications if the backdoor is present. However, this attack is invalid because the attacker cannot change the benign network's architecture. (c) A valid BadNet attack. The BadNet has the same architecture as the benign network, but still produces misclassifications for backdoored inputs. (Image credits: [21])	16
3.2	The training process of an attack model using the Shadow training technique proposed by Shokri et al. (Image credits: [22])	17
3.3	The reconstruction of the individual on the left using the reconstruction attack on several different machine learning models. (Image credits: [24]) . .	17
3.4	An example of an adversarial attack using the Fast Gradient Sign Attack. The image, after being perturbed, was misclassified by the model with a higher confidence level than the original image. (Image credits: Goodfellow et al. [6])	21
3.5	Saliency map of a 28x28 image. Large absolute values correspond to the pixels with the greatest impact on the decision of the network (Image credits: [26])	22
3.6	Approximation of the decision boundaries to find an adversarial sample. (Image credits: [28])	23
3.7	A perturbation generated using the UAP method. The perturbation, when applied to various images of a dataset, is capable of fooling the target model. (Image credits: [33])	25
3.8	The process of generating an adversarial sample using the stAdv attack. The green point is the location of the pixel in the benign sample. The red flow represents the displacement that the pixels were subjected to. The blue dot represents the position of the green dot in the adversarial sample (Image credits: [36])	26
3.9	The architecture of an AdvGAN. (Image credits: [37])	27

3.10	The process of generating an adversarial image using the Boundary Attack, one of the first decision-based black-box attacks to be presented. (Image credits: [38])	28
3.11	The iterative process of finding an adversarial sample using the Projected Gradient Descent Attack. On the first try, the sample found has a low loss which means it will be a weak adversarial sample. In the second try, however, the sample lands in a region with high loss that results in a strong adversarial sample. (Image credits: https://towardsdatascience.com/known-your-enemy-7f7c5038bdf3)	29
3.12	The results of the Partial Information Attack on the Google Cloud Vision API. (Image adapted from: [44])	30
3.13	The autoencoder used by AutoZOOM to generate adversarial samples. (Image credits: [48])	32
3.14	The process of finding an adversarial image used by the Decoupled Direction and Norm attack. The adversarial sample is found when the projection of x leaves the original class boundary. (Image credits: [55])	35
3.15	The process of generating an adversarial sample using the POBA-GA attack. (Image credits: [56])	36
3.16	Adversarial images generated by the CornerSearch attack. The three variations of the attack produce samples with extremely low sparsity that are almost imperceptible. (Image credits: [57])	37
3.17	The process of finding an adversarial image using the LogBarrier attack. The starting sample is already adversarial and iteratively approaches the decision boundary of the original image class. (Image credits: [58])	38
3.18	Schematic of the Brendel & Bethge approach. Considering an input which a model either interprets as a dog (shaded region) or as a cat (white region). Given a clean dog image (solid triangle), the algorithm searches for the closest image classified as a cat. The attack starts from an adversarial image far away from the clean image and walks along the boundary towards the closest adversarial (middle). In each step, an optimization problem is solved to find the optimal descent direction along the boundary that stays within the valid pixel bounds and the trust region (right). (Image credits: [59])	38
3.19	A natural image (left), an adversarial image generated by an L_∞ attack (middle), and an adversarial sample generated by the shadow attack (right). The perturbation generated by the shadow attack is large while blending in the image. (Image adapted from: [60])	39
3.20	Visual representation of GeoDA. (Image credits: [62])	40
3.21	Visual representation of AdvFlow. (Image credits: [63])	41
3.22	Physical adversarial perturbations for traffic signs against LISA-CNN and GTSRB-CNN classifiers (Image credits: [78])	43
3.23	Impersonation and dodging attacks. The top row represents the impersonators using the adversarial glasses and the bottom row represents the attributed classification by the target classifier. (Image credits: [79])	44
3.24	The left person is detected by the classifier, however, the right person wearing the patch is ignored. (Image credits: [80])	45
4.1	Gantt chart of the development plan	51
5.1	MoSCoW diagram representing the software requirements of the framework	54
5.2	Risk assessment table	56
5.3	2nd Semester Work Plan	57
5.4	Framework file structure	58

5.5	A call to the pipeline class. The execute function has several inputs that are needed for the execution to start.	61
5.6	An example of a config file. In this example, the Bandits & Prior and the DDN attacks are going to be executed in the pipeline.	62
5.7	The <i>outputs</i> folder with examples of executions using the pipeline mechanism of the framework	63
5.8	UML diagram representing the main classes of the framework and the external libraries	64
6.1	CIFAR-10 samples used in the experimentation	68
6.2	Comparison between the original sample 4 (a), the adversarial image generated by the UAP attack for the ResNet18 (b), and the adversarial image generated by UAP for the MobileNet (c)	72
6.3	ResNet18 generated images (1-5)	73
6.4	MobileNet generated images (1-5)	74
1	ResNet18 generated images (6-10)	90
2	ResNet18 generated images (11-15)	91
3	ResNet18 generated images (16-20)	92
4	MobileNet generated images (6-10)	93
5	MobileNet generated images (11-15)	94
6	MobileNet generated images (16-20)	95

This page is intentionally left blank.

List of Tables

3.1	Taxonomy of Evasion Adversarial Attacks	19
3.2	Functionalities supported in the current state-of-the-art frameworks	46
3.3	Evasion Adversarial Attacks supported in the current state-of-the-art frameworks	47
5.1	List of attacks analyzed during the selection phase of the framework development	59
6.1	Test accuracy of models used in the experimentation	67
6.2	Settings of the attacks used in the experimentation	69
6.3	Model Robustness Values	71
6.4	L2 scores of the adversarial images generated for the ResNet18 and the MobileNet	75

This page is intentionally left blank.

Chapter 1

Introduction

Artificial Intelligence (AI) is an ever-growing field that has allowed computers to solve problems that are complex for humans but simple for machines. The main challenge of AI was solving complex problems that the human brain solves daily, for instance, recognizing a face in an image. Deep Learning (DL) models surged as a branch of Machine Learning (ML) capable of solving those problems. DL methods have gained immense interest and applications that span from self-driving cars, capable of identifying pedestrians and vehicles, to voice recognition systems capable of recognizing individuals by their voice. The use of DL in human's daily life has put forward the need of guaranteeing that those models are secure. It was discovered, however, that DL models are not secure and can be easily tricked into misclassification. Adversaries can create inputs with added perturbations that are inconspicuous to the human eye but force the model to produce an incorrect output.

1.1 Motivation and Objectives

The discovery of adversarial samples has motivated researchers to understand the robustness of DL models. In the latest years, the adversarial learning landscape has grown exponentially with a constant clash between developing adversarial attacks and adversarial defenses capable of increasing the adversarial robustness of classifiers. To help researchers understand if their model or defense is robust, frameworks that facilitate this process have been developed. However, most of those frameworks are still incomplete or have been partially abandoned. This dissertation aimed to develop a framework, oriented to the image classification domain, that is complete with all the tools needed to evaluate the robustness of a model. The framework was developed in Python and supports PyTorch, a well-known machine learning framework and currently supports eight adversarial attacks. The developed framework offers a pipeline mechanism that allows users to choose a variety of models and to attack them using any or all of the current supported adversarial attacks. In order to allow users to understand how robust their models are, a new robustness metric was developed that takes into consideration the behavior and performance of a model against adversarial images generated by an attack. This robustness metric is completely integrated into the pipeline mechanism which allows users to easily compare their models.

1.2 Document Structure

In addition to this introduction, the following chapters are organized as follows:

Chapter 2 introduces a brief presentation of concepts and definitions related to DL and adversarial attacks. An analysis of current robustness metrics for evaluating classifiers is also performed.

Chapter 3 presents the state-of-the-art that introduces the most relevant categories of adversarial attacks and a deeper dive on evasion adversarial attacks is made. Real world applications of adversarial attacks are analyzed to understand the reach and the effects that adversarial attacks can have on machine learning based systems. In the final section, the most well-known and used current state-of-the-art frameworks in the adversarial learning panorama are analyzed and its limitations are discussed.

Chapter 4 describes the development plan used in the first semester as well as the development planned for the second semester. In addition, the methodology that was expected to be followed in the second semester is also presented.

Chapter 5 extends the development of the framework which starts by the software requirements specification that was performed in the first semester as well as an analysis of the potential risks that were associated with developing the framework. The actual executed work plan of the second semester is compared to the original expected plan. The architecture and the implementation steps of the framework, that include the developed file system, the pipeline mechanism and the metric to evaluate the robustness of models are presented.

Chapter 6 exposes the experiment realized to test the capabilities of the framework, which includes the pipeline mechanism and the robustness metric. The results of that experimentation are analyzed and discussed.

Chapter 7 concludes the work of this dissertation and proposes changes to be made in the future.

This page is intentionally left blank.

Chapter 2

Concepts & Definitions

The following chapter starts by introducing important concepts about Deep Learning (DL) and DL applied to image recognition, which is the focus of this dissertation. A brief presentation of theories for the existence of adversarial samples is made which is then followed up with an introduction to the concept of threat model in machine learning security. Finally, an analysis of current metrics to evaluate the robustness of classifiers towards adversarial attacks is presented.

2.1 Introduction to Deep Learning

DL [1] is an approach of Artificial Intelligence that focuses on solving complex but intuitive problems for humans such as recognizing a face. DL allows computers to learn complex concepts from simpler ones without the need for human interference. DL models are used in various areas such as computer vision, natural language processing, and speech recognition. There is a multitude of different DL algorithms that are used according to their purpose. These include Convolutional Neural Networks (CNNs), Recurrent Neural Networks, Multilayer Perceptrons, and others, with all of them sharing the same foundation: neurons and hidden layers that are connected. A typical Deep Neural Network (DNN) can be seen in figure 2.1.

As mentioned before, Machine Learning (ML) models can be used for various tasks. A task in ML refers to the way that a model should process an example. From the several tasks, the most relevant one for the context of this dissertation is classification. In this task, the model is asked to specify to which classes the input corresponds. The model after processing the input outputs a category that best classifies that input. To evaluate the performance of the classifier model, it is often used the accuracy rate which corresponds to the proportion of examples that were correctly classified by the model. The proportion of incorrectly classified inputs is denominated error rate. The objective of the model is to obtain the best accuracy possible which in result diminishes the error rate. In an effort to evaluate the performance of the model, it is often used a test set that contains samples that the model hasn't had access to during the training process.

The training process of a model can be categorized into three main categories: supervised learning, unsupervised learning, and reinforcement learning [1]. The focus will be on supervised learning since it is the main process used for image classification. During the training process, the model experiences a dataset that is full of samples and each sample contains features that represent the individual sample. Additionally, in a supervised

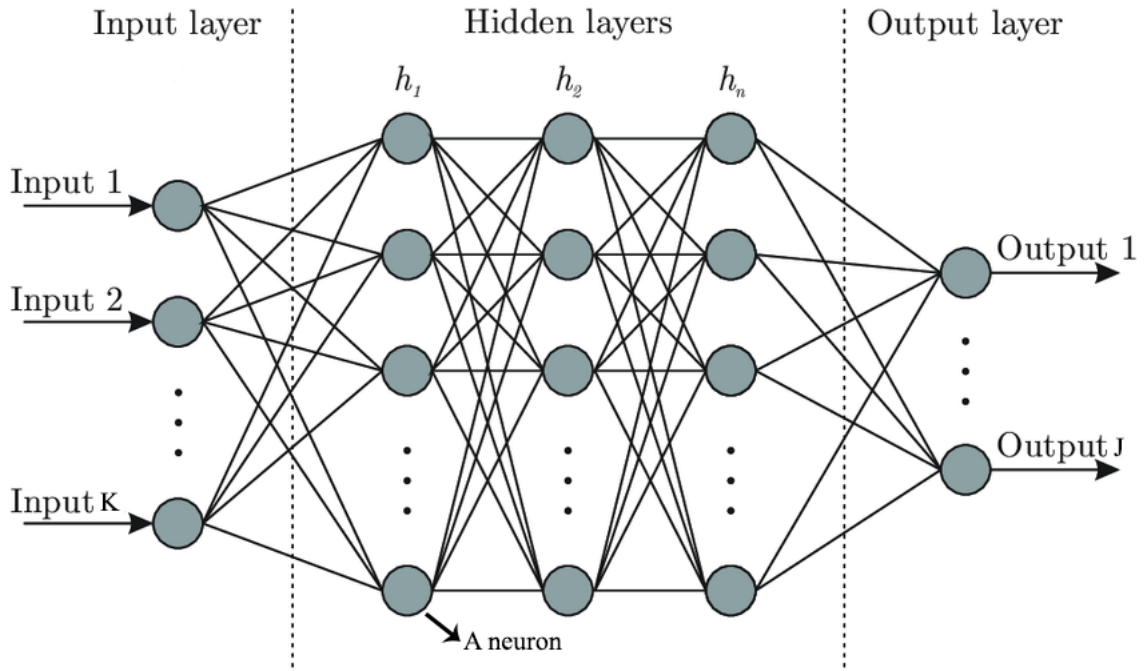


Figure 2.1: Schematic of a Deep Learning Model with 3 hidden layers. (Image credits:[2])

learning scenario, each example is associated with a label. To illustrate, in the task of recognizing photographs, the training dataset of photographs also needs to have the class that each photograph is part of.

A neural network has multiple hidden layers and each layer has various neurons that have a weight and a bias associated with them [1]. It also has an input layer that is designed depending on its application. To illustrate, a CNN is usually designed to classify images and for that has an input layer designed to take advantage of a 2D input. The output layer outputs the values that were calculated in the hidden layers. If the neural network was designed to, for example, classify images of handwritten digits, the output layer would consist of a vector with 10 values, each one with a probability associated with a digit. The aforementioned weights and biases of a neural network need to be computed so that the classifier is capable of classifying inputs. To do that, it is used a cost function, also known as loss function. The cost function can be represented as:

$$C(w, b) \equiv \frac{1}{2n} \sum_x ||y(x) - a||^2 \quad (2.1)$$

Where x is the input, $y(x)$ is the correct output of x , w represents all the weights in a neural network, b represents all the biases and a is the vector of outputs, for all training inputs, and its value depends on x , w , and b . After inspecting the cost function, it is clear that if the value is low, the classifier is correctly classifying most of the inputs and if the value is high, the model is not performing as desired. This means that minimizing the cost function results in better and more capable models. In order to optimize the weights and biases of the neural networks, it is used an optimization method being the gradient descent, one of the most preferred. The gradient descent algorithm, repeatedly, computes the gradient value of the cost function, or in other words, the vector of partial derivatives of the cost function with respect to the weights and the biases. The gradient in combination with a small constant known as learning rate, attempts to decrease the value of the cost function. A very simplified version of this algorithm can be seen as a ball falling down a valley and a visual representation can be seen in figure 2.2.

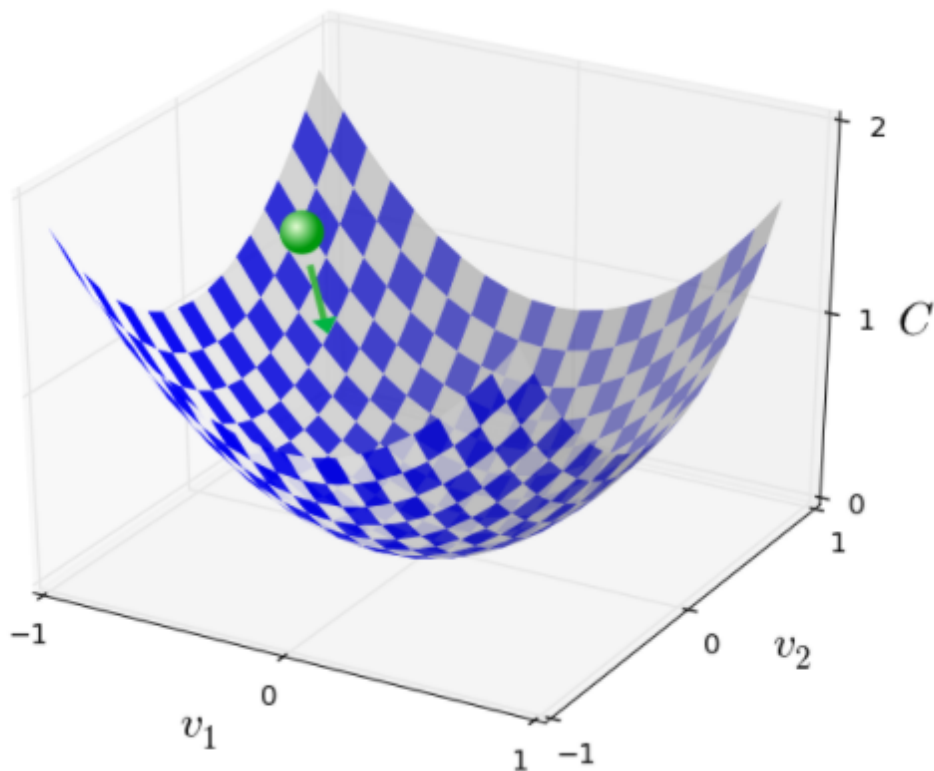


Figure 2.2: A visual representation of the gradient descent algorithm optimizing the cost function, where the ball represents the cost function and the arrow represents the direction of the gradient. The cost function is represented by C and the variables of the function by v_1 and v_2 . (Image credits: [3])

CNNs [1] are a type of neural network that is used to analyze grid-like data such as images, which can be seen as a 2D grid of pixels. CNN uses a process called convolution, which involves putting the input pixels through a filter to create a feature map. There are three main properties of the convolution process: sparse interactions, parameter sharing, and equivariant representations. Sparse interactions can be achieved by using a filter that is smaller than the input image. More importantly, when processing an image with millions of pixels, filters enable the extraction of the most important information, such as edge detection, considerably improving network performance. When a weight is used more than once, it is referred to as parameter sharing which reduces the number of parameters that need to be calculated, which leads to a decrease in memory needs and increases the computational performance. Translation equivariance is caused by parameter sharing. This means that performing the convolution and then moving a pixel one unit to the right is the same as applying the convolution and then moving a pixel one unit to the right. However, this only applies to translations; equivariance does not apply to rotations and scaling is not a natural property of convolutions. A convolutional layer [1] consists of performing several convolutions that result in linear activations which are then passed through a non-linear function. The output is then run through a pooling function. This final step is crucial in making sure that the representation is invariant to small translations, especially when it is more important to identify that a feature is present other than exactly where it is. This allows the network to be robust to small pixel translations and increases the network efficiency. It is significant to consider that there are many variations of CNNs, however, all share the same building blocks presented in this section. A visual representation of a

CNN can be seen in figure 2.3.

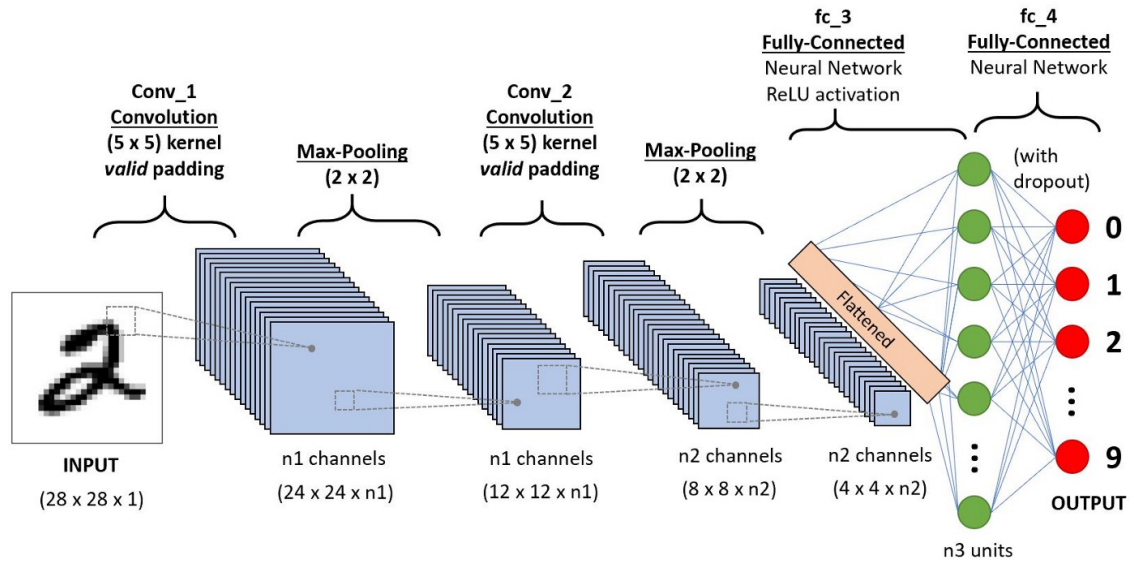


Figure 2.3: Architecture of a Convolutional Neural Network. (Image credits: [4])

2.2 Explaining Adversarial Attacks

Szegedy et al. [5] discovered a property concerning the stability of neural networks when inputs have small, almost undetectable perturbations. The authors discovered that adding a perturbation to an image, almost undetectable to the human eye, could change the behavior of the classifier model. Additionally, the same perturbed image could be misclassified by different neural network models. These perturbed samples were called *adversarial samples* by the same authors. The process of finding perturbations that are imperceptible but, at the same time capable of changing the behavior of the target model can be seen as an adversarial attack.

Since the discovery of adversarial attacks, various authors have tried to understand and explain the existence of these occurrences. Some of the most known and discussed theories are briefed in the following sections.

The linearity present in most classifiers could also be related to the origin of adversarial samples. Goodfellow et al. [6] demonstrate that the linear behavior in high-dimensional spaces is sufficient to explain adversarial examples. This behavior was shown to be present not only in linear but also in non-linear classifiers. Non-linear models, such as sigmoid networks, are constructed in a way that facilitates their optimization, this leads to them spending most of their time in the more linear regime. This leads to the same weak robustness present in linear classifiers.

The generalization of adversarial attacks in different models with different configurations, discovered by Szegedy et al. [5] was also proposed as being a result of the linearity present in models. This property implies that it should be possible to attack a neural network without having any previous knowledge about the configurations and using only attacks that were shown to be effective in deceiving other classifiers.

Gilmer et al. [7] hypothesize that high-dimensional geometry was responsible for the existence of adversarial examples. To investigate this hypothesis, a synthetic high-dimensional dataset was constructed to facilitate the construction of adversarial examples since the data distribution and the decision boundary of the model are known. The focus of the study was the relationship between two measures of the testing error set of a model: the under-the-data distribution of the error set and the average distance to the nearest error. The experiments showed that the chosen points from the data distributions were correctly classified by the model however, they were very close to the decision bound of another class. This means that the models are extremely prone to being wrong if a small perturbation was to be inserted in the inputs. The authors infer that to increase the adversarial robustness of the models, the test error needs to be minimized.

Tanay et al. [8] argue that adversarial examples exist when the classification boundary of a model lies close to the submanifold of sampled data, and present a mathematical analysis of this new perspective in the linear case. The authors argue that the linearity of models in high dimensional settings argument presented by Goodfellow et al. [6] is insufficient to fully explain the adversarial samples phenomena. They demonstrate that increasing the dimensionality of the problem does not make the problem of the adversarial examples worse. Additionally, the authors show that for some classes of linear classifiers, it was impossible to generate adversarial samples.

The boundary tilting perspective is the new solution presented by the authors to explain adversarial samples. Essentially, a submanifold of sampled data when intersected by a class boundary close to it suffers from adversarial examples. The behavior of low and high-dimensional spaces can also be explained by this perspective. In higher dimensions, the probability that a perturbation will move in the direction of the boundary is low which makes the samples more robust to adversarial samples. On lower dimensions, on the other hand, the perturbations are more likely to cross the boundary which makes samples in a low dimension space more prone to adversarial examples.

Schmidt et al. [9] studied if current training datasets are capable of training classifiers robust to adversarial samples. They identify that, as adversarial examples only occupy a small measure in the data distribution, the standard notion of generalization cannot be applied. Instead, a new notion of adversarial robust generalization needs to be used. To understand the differences between benign and adversarial robustness generalization, they used two distributional models. The results show that even for simple data distribution, the sample complexity of adversarial robust generalization is significantly larger than that of standard benign generalization. They hypothesized that current approaches may be unable to attain higher adversarial accuracy on more complex datasets such as CIFAR-10 due to the size limitation of the dataset. This means that, training a classifier with a dataset that has a low number of samples isn't sufficient for the model to be robust to adversarial examples. Models that use less complex datasets to train such as MNIST, were also shown to be more robust networks. The authors also propose that adversarial defenses need to be adapted to the dataset that is used to train the classifier.

Bubeck et al. [10] study the phenomenon of adversarial examples as a computational constraint problem. The authors show that classifiers in high dimensions are prone to adversarial examples not due to information-theoretic limitations but rather to computational constraints. The authors present evidence supporting a hypothesis for and a hypothesis against the existence of adversarial examples. The first hypothesis is *"Identifying a robust classifier from limited training data is possible but computationally intractable"* and the second hypothesis is *"Identifying a robust classifier requires too much training data"*. The evidence in favor of the first hypothesis is that there are robust classification tasks that

are easy in terms of information but computationally intractable under the most powerful current generation computational models available. The second hypothesis is rejected by the authors based on evidence that shows that if a robust classifier exists then it can be found with relatively few training examples under a standard assumption on the data distribution. It is important to mention that the assumptions are based on the data distribution used by the authors which is a uniform data distribution and for that reason, the results are inconsistent with the natural distribution of images.

Ilyas et al. [11] demonstrated that the current explanations for the existence of adversarial attacks are still unable to explain the behavior of those attacks. They put forward a new perspective for the occurrence of adversarial attacks - adversarial vulnerability is a consequence of the current supervised learning model. The phenomenon of adversarial samples occurs because classifiers are trained to maximize the accuracy and to do that they use every signal that is present in the data. Furthermore, the authors find that datasets admit features that may seem imperceptible but are highly predictive in which classifiers will rely on to maximize their accuracy, which were called *non-robust features*. This phenomenon results in models being weak towards adversarial examples as it is easy to manipulate the mentioned non-robust features. Their hypothesis also tries to explain the transferability property presented by Szegedy et al. [5]. Adversarial examples are transferable between classifiers because even if they are different models, they will most likely rely on non-robust features to maximize their classification accuracy. In the current supervised learning, perceptible and imperceptible features are learned in the same way by classification models even if those same features are different to the human eye.

Despite the importance that adversarial attacks have gained recently, there is still a lot of work to do. A universally accepted explanation as to why adversarial examples can brittle networks is still unclear but as the importance of this subject grows, more explanations are presented for the existence of adversarial samples. Even though there is no explanation, the relevance of adversarial learning has increased exponentially in the last few years.

Although there isn't an explanation for adversarial attacks, many authors have started to propose mechanisms to defend classifiers against adversarial attacks. These mechanisms are called adversarial defenses, however, many defenses that have been proposed are incapable of defending the models [12]. Since the focus of this dissertation is on adversarial attacks, exploring the advances of adversarial defenses is out of the scope.

2.3 Adversarial Threat Model

It is clear that adversarial attacks are a threat to ML security since they can alter the behavior of classifier systems with very subtle changes to an input. It is, therefore, important to understand how different adversaries can attack a model depending on the information that they have about the system. The following section is organized based on the work done by Papernot et al. [13] and will delineate the current landscape of machine learning security.

The security level of an ML system is measured concerning the adversarial goals and capabilities of the adversary threat model. The main threat models affect classifiers in two distinct ways: poisoning attacks focus on attacking and altering the training phase of the model, while evasion attacks focus on the testing phase of the model.

More and more applications are using ML as a foundation for their systems and while those systems can vary from each other, they can be viewed as a generalization of a data

processing pipeline. In a very primitive way, an ML system works in the following manner: (1) a collection of input data is obtained from the physical world, (2) then it is processed and converted into digital format, (3) the data, now digital, is forwarded to the system to be processed and, (4) it finally outputs a result. If the ML system is being used in a system in the physical world, (5) the output can have an action that has a real impact. The aforementioned process can be seen in figure 2.4.

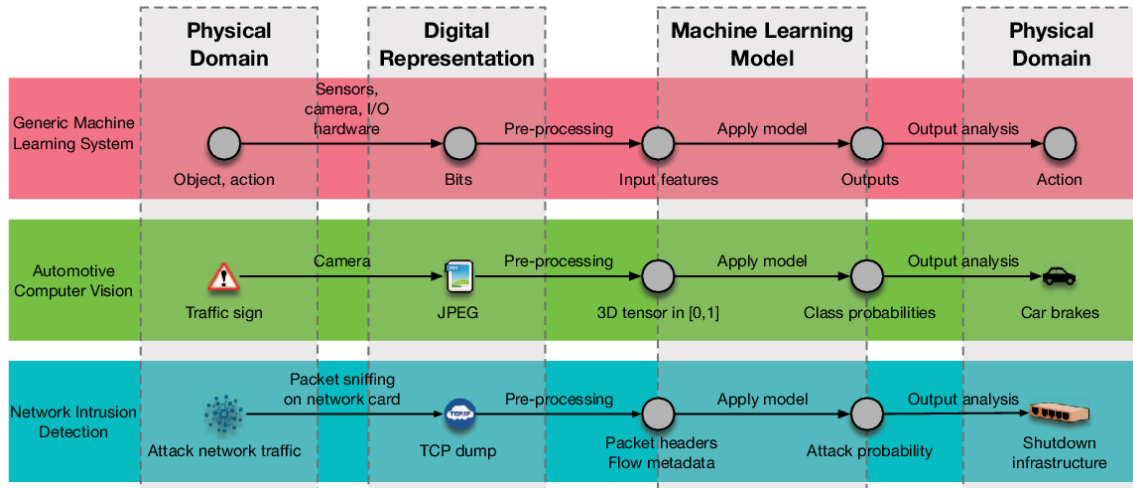


Figure 2.4: Attack surface of a generic machine learning system and, of two machine learning systems used in the physical world (Image credits: [14])

The adversarial capabilities of an adversary can be viewed as the amount of information the adversary has about the target and also, as the actions that the adversary can perform based on that information. The capabilities of an adversary can be broadly classified into two major groups: testing phase capabilities and training phase capabilities.

During the training phase of ML models, adversaries try to learn or corrupt the ML system. The attacks could be focused on the ML algorithm or in the training data itself. The adversary can have different levels of control over the model and the dataset. Usually, stronger attacks are the ones capable of altering the ML model and the complete dataset, and weaker attacks normally can only add corrupt samples to the training dataset. There are three main strategies of poisoning attack use:

- (1) **Data injection:** In data injection, the adversary doesn't have access to the training data nor to the learning algorithm, he can insert adversarial samples to the training dataset.
- (2) **Data modification:** In this scenario, the adversary doesn't have access to the learning algorithm of the target model, however, has full access to the training dataset. The adversary directly alters the training data before it is used in the training process of the model.
- (3) **Logic corruption:** These types of attacks can directly alter the ML algorithm of the target model. This is the most harmful type of attack as the adversary can change the behavior of the model in a way that it becomes unable to learn. Despite being extremely harmful, it is extremely hard to execute.

Attacks performed during the testing, or the inference, phase of the life of the model, don't tamper with the training phase of the ML model. In testing phase attacks, the adversary tries to feed the neural network with perturbed inputs to have the classifier misclassify them or, in other cases, only tries to collect information about the model. These types

of attacks can be separated into two broad categories that are related to the amount of information that the adversary has on the target: white-box attacks and black-box attacks.

In white-box attacks, the adversary has total information about the target classifier. This information could be the training dataset, the training data distribution, the type of neural network of the classifier, the configurations of the internal layers, the architecture of the model, etc. Usually, in a white-box attack scenario, the adversary uses attacks that are based on the gradient of the loss function of the neural network. State-of-the-art white-box attacks use backpropagation with the intent to compute the gradient to then construct the attack.

In black-box attacks, the adversary usually has no knowledge or very little knowledge about the target classifier. In most cases, carefully constructed adversarial examples are sent to the input of the classifiers, and the adversarial attack is constructed according to the output. Black-box attacks can be further broken down into three different categories.

In a non-adaptive black-box attack scenario, the adversary knows the training data distribution of the target model. The adversary creates a substitute target model and trains the substitute with samples generated from the data distribution to shape the substitute to be closer to the target model. After the substitute model is created and trained, the adversary generates white-box attacks based on the substitute model and tries to attack the target model using those adversarial samples.

In adaptive black-box scenarios, the adversary has access to the input and the output of the target model and can query carefully created inputs and obtain the output from the target. The adversary then proceeds to create a substitute model and trains that model with the input/output samples that were fed to the target model. Analogous to what happens in a non-adaptive black-box scenario, white-box attacks are constructed based on the substitute model and are then queried to the target model.

Strict black-box attacks are the most common type of scenario. In this setting, the adversary only has access to the input and output of the target and can only perform queries to obtain the output from the target classifier. Most types of strict black-box attacks usually construct attacks based on a computed approximation of the gradient values of the target model.

The goals of the adversary are also an important step of the adversarial threat model. Four main goals can drive an adversarial attack: confidentiality, integrity, availability, and privacy.

In a scenario in which an adversary is not a trusted user of a model, it may try to access the model or the training data of the model. In these types of scenarios, the adversary is breaching the confidentiality and the privacy of the model and its data. In addition, models keep information about the training data set. In certain scenarios, an attacker might realize an attack to obtain information about the training data. If the model that is being attacked used individuals' data to learn, those individuals' privacy is being threatened and the confidentiality of the data is also being put at risk.

The integrity and availability could be also the target of an adversary. In this case, the target would be the output values of the model. Attacks with the intent of disrupting the integrity of the model are especially dangerous as they affect the accuracy of the model. Changes in the accuracy of models, depending on the function that the ML model has, could result in catastrophic results. The integrity of the target model can be altered depending on the objective of the adversary. The adversarial goals, related to the integrity of the model, can be classified in four main ways:

- (1) Confidence reduction: The adversary tries to lower the confidence score of a prediction made by the target.
- (2) Misclassification: The adversary will try to change the output class label of an input.
- (3) Targeted misclassification: In this scenario, the adversary feeds multiple different inputs and tries to alter the output class of those inputs to a specific target class chosen by the adversary.
- (4) Source/Target misclassification: In this scenario, the adversary tries to create a specific misclassification output in a specific input.

Availability-focused adversarial attacks try to make the target model outputs inconsistent or unreliable. Availability attacks, similar to integrity attacks, will usually focus on poisoning the model during the training phase or in querying the model, during the testing phase, with adversarial samples. If a system is incapacitated to perform its actions due to the ML outputs being unavailable, availability attacks can be categorized as a form of denial of service attacks. While the adversary goals may be similar to the ones in integrity attacks, availability-focused attacks can input such a large size of perturbed inputs that the machine becomes unstable.

2.4 Robustness Metrics for Classifiers Evaluation

Adversaries are capable of attacking ML systems under various conditions, using attacks at the training phase but also at the testing phase. How does one evaluate the capability of classifiers, that are trained for the same task, to resist adversarial attacks? This is an extremely important question that has yet to receive a concise and universally agreed answer.

The current landscape of adversarial robustness of a neural network model mostly settles on the misclassification ratio of the samples generated by an adversarial attack. Despite this metric being the most used, it doesn't provide enough important information about the robustness of a classifier. This section provides some metrics that authors use to evaluate the adversarial robustness of classifiers.

Weng et al. [15] was the first to propose a new attack-agnostic metric for evaluating the robustness of a neural network classifier called Cross Lipschitz Extreme Value for nEtwork Robustness (CLEVER). The authors were motivated by the fact that the adversarial robustness of a model was entangled with the adversarial attack used which could lead to a biased analysis. This metric stems from the lower bound of an adversarial attack which is the least amount of perturbation that an unperturbed sample can receive in order to be misclassified by the target model for any perturbation bound of an attack ℓ_p where $p \geq 1$. The authors convert the lower attack bound into a local Lipschitz constant estimation problem where the constant is estimated using Extreme Value Theory. This metric is attack-agnostic, applicable to any neural network classifier, and was shown to be computationally feasible even for larger networks. The experiments performed show that the robustness of a classifier using the CLEVER metric was aligned with the robustness measured by adversarial attacks that used ℓ_2 and ℓ_∞ norms.

Dong et al. [16] propose methodologies, other than simply the misclassification ratio of the adversarial images, to evaluate the robustness of a classifier. They put forward two robustness curves to analyze the robustness and resistance of a classifier against an attack. The first curve is the *attack success rate vs. perturbation budget curve* where the

perturbation budget is the amount of perturbation added to the adversarial image. The second curve is the *attack success rate vs. attack strength curve* where the attack strength is represented by the number of queries that the attack makes to the target model. The authors also suggest differentiating the attack success of an attack depending on whether the attack is targeted or untargeted.

It is clear that the adversarial learning panorama is still in its early stages and that a universal metric for evaluating adversarial robustness, after a thorough search of the relevant literature, hasn't been defined. This is a subject that will be further researched in this dissertation.

Chapter 3

State-of-the-Art

This chapter is a representation of the initial research performed in the first iteration of this dissertation. The initial sections introduce the most common categories of adversarial attacks with an emphasis on evasion adversarial attacks since it is the focus of this dissertation. The last section exhibits the current state-of-the-art frameworks that help researchers evaluate the adversarial robustness ML models.

3.1 Adversarial Attacks

This section addresses the most relevant categories of adversarial attacks currently in the literature. The main focus is evasion attacks since this is the type of attack that is to be used in the framework to develop. However, due to the importance that the other attacks have in the ML landscape they are also briefly addressed.

3.1.1 Poisoning Attacks

Poisoning attacks are directly related to adversarial threat models that affect the training phase of a model. Adversaries using poisoning attacks are, essentially, trying to alter the training phase of the model by changing the training dataset to modify the decision boundary of the target model.

Similar to what was mentioned in training phase capabilities, poisoning attacks can be further classified into three main groups: Data injection attacks, Data manipulation attacks, and Logic corruption attacks. Since this type of attack won't be the focus of this dissertation, there is only going to be presented an overview of poisoning attacks.

In data manipulation attacks, the adversary has only access to the training data and can perform two varieties of attacks: label manipulation and input manipulation.

In a label manipulation scenario, the adversary only has the capability of modifying the training labels. Then, he needs to obtain and perturb the most vulnerable labels given that he knows the learning algorithm. The base approach is to perturb labels randomly selected from a subset of the original training data. Biggio et al. [17] presented that randomly flipping at least 40% of the training labels resulted in a degradation of Support Vector Machine (SVM) classifiers.

In an input manipulation scenario, the adversary has access to the learning algorithm

and the input features of training points, in addition to its labels. As the knowledge of the target model is greater, the adversary is capable of developing more advanced and capable attacks. Attacks focused on input manipulation have a procedure of generating perturbations similar to the process of evasion attacks. This results in some authors also using techniques of generating evasion attacks to generate input manipulation attacks. This type of scenario can also be different depending on whether the learning algorithm is online or not.

In online scenarios, such as the one present in Kloft et al. [18], the training data of the model is obtained at regular intervals. This is an easy target for the adversary because poisoned points are found by calculating a simple linear programming problem. In offline scenarios, Biggio et al. [19] introduced an attack that identifies poisoning points based on the gradient ascent method. Including these inputs in the training, the data set resulted in the degradation of the accuracy of an SVM classifier during the test phase of the model.

A data injection attack is a more limited attack than a data manipulation or a logic corruption attack because the adversary is only capable of inserting perturbed samples in the training dataset.

The work presented by Chen et al. [20] represents a backdoor poisoning attack with a face recognition system as the target. A backdoor attack focuses on creating a backdoor on the target model that can later be exploited by the adversary. The attacker can then create inputs with a backdoor trigger to be wrongly predicted as a label chosen by the attacker. The scenario of this work assumes that the adversary has no knowledge of the model and the training set and it is only allowed to input a small sample size of perturbed samples. The restrictions of this work allow for a more real-world situation in which the adversary has very limited conditions and can only input a very small number of samples in the training set. Despite the number of samples being very low, around 50 samples, the results of the work show a success rate of 90%.

In a situation where an adversary obtains control over a machine learning algorithm of a system, there could be dangerous repercussions, especially if the ML system is being used in the physical world. A logic corruption attack has that exact objective: to obtain control over a system. In Gu et al. [21] the authors developed a logic corruption attack called BadNet. This attack is a backdoor attack, similar to the one presented before, however, the knowledge and the procedure are different. In the proposed scenario, the training process of a neural network system is outsourced to an untrusted third-party entity, which is a scenario that is currently gaining traction. The returned model then comes with a backdoor trigger that can be activated when in contact with specific inputs that the attacker may input. This means that the machine learning model when in contact with supposedly regular inputs could misclassify those inputs. Despite being an extremely powerful attack, the BadNet attack needs to be implemented in such a way that it doesn't fail two requirements. First, it can only modify the model in a way that doesn't completely change the original neural network structure and doesn't reduce the classification accuracy of the validation test set. And second, the backdoor present in the classifier needs to be capable to differentiate backdoor triggered inputs from regular inputs. A way of implementing the BadNet attack can be seen in figure 3.1.

3.1.2 Exploratory Attacks

Exploratory attacks are similar to evasion attacks in the capabilities of the adversary - in both scenarios, the adversary only has testing phase capabilities. However, the goal of

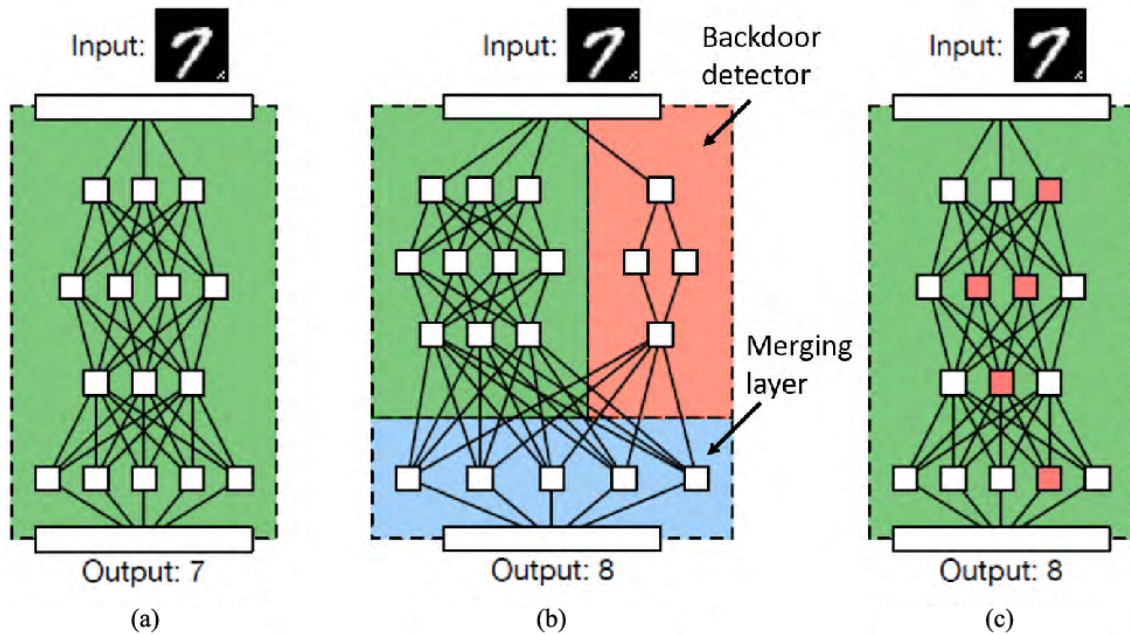


Figure 3.1: A backdoor attack. The backdoor trigger is a pattern of pixels that appears on the bottom right corner of the image. (a) A benign network that correctly classifies its input. (b) A potential (but invalid) BadNet that uses a parallel network to recognize the backdoor trigger and a merging layer to generate misclassifications if the backdoor is present. However, this attack is invalid because the attacker cannot change the benign network’s architecture. (c) A valid BadNet attack. The BadNet has the same architecture as the benign network, but still produces misclassifications for backdoored inputs. (Image credits: [21])

an exploratory attack focuses on breaching the privacy or the confidentiality of the model and, of the users that provided information to the training dataset of the model. The two current primary exploratory attacks are addressed in this section.

Membership inference attacks have been a fundamental question in the machine learning field. The objective of this attack is to identify if a certain sample was present in the training dataset of a model. This type of attack can be a breach of an individual’s privacy, not just of the ones that contributed to the training dataset but also of those that belong to the same population and may even be unaware that their privacy may be at risk. Shokri et al. [22] studied membership inference attacks in a black-box scenario. Despite the setting being more difficult for the attacker, it is also more probable to happen in the physical world. In this scenario, the adversary can only feed samples to the target and observe the output result. The approach consists in using an attack model to distinguish if a data sample was present in the training dataset or not. The training process of the attack model consists of a technique developed by the authors called shadow training. Shadow training dwells in creating multiple shadow models with similar behavior to the target model, the shadow models are trained with similar training data between them and that training data is also relatively similar to the one used by the model being attacked. Each shadow model has an "in" training dataset and an "out" testing dataset. The combination of the "in" and "out" datasets of the shadow models are used as the training dataset of the attack model. During training, the attack model needs to classify the samples given, essentially turning into a binary classification problem. After being trained, the attack model is ready to classify samples as being present or not in the training dataset of the target model. The

training process can be observed in figure 3.2. The experimentation of the shadow model resulted in 90% of membership inference accuracy against a Google-trained model.

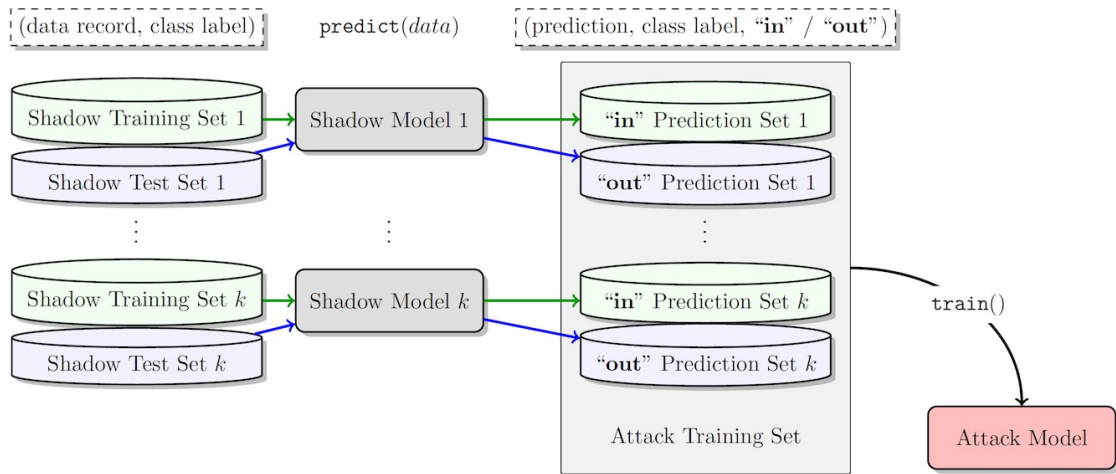


Figure 3.2: The training process of an attack model using the Shadow training technique proposed by Shokri et al. (Image credits: [22])

Fredrikson et al. [23] demonstrated a Model Inversion (MI) attack that is capable of estimating aspects of the genotype of an individual in a black-box scenario. Later, Fredrikson et al. [24] expanded that work into commercial ML-as-a-service APIs. The authors studied MI attacks in both white-box and black-box scenarios, depending on the information that the attacker has on the target model. The authors developed MI attacks targeted for facial recognition models. The two attacks only assume that the adversary has access to the trained model but has no access to the training dataset. The first attack tries to produce an image of a person using only a label or the name of that person and was named *Reconstruction*. The second attack tries to reconstruct an image of a person using the blurred face of that same person and was named *Deblurring*. While the focus of the first attack is to identify the person’s facial characteristics, the focus of the second attack is to identify if the subject present in the blurred image was a record of the training dataset of the model or not. The developed white-box algorithm for MI was tested on three different classifiers: a softmax regression, a multilayer perceptron network, and a stacked denoising autoencoder network. The reconstruction attack results can be seen in figure 3.3.

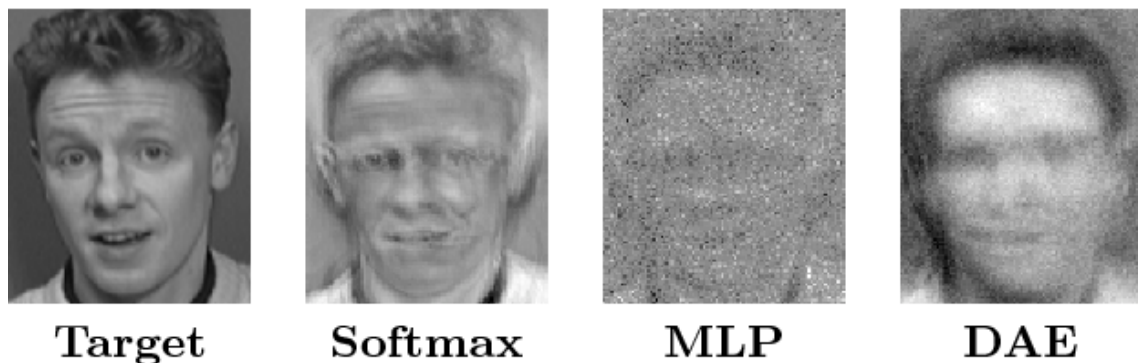


Figure 3.3: The reconstruction of the individual on the left using the reconstruction attack on several different machine learning models. (Image credits: [24])

3.1.3 Evasion Attacks

Evasion attacks are the most common type of adversarial attack. In this setting, the attacker tries to feed adversarial examples to the classifier during the testing period in order to cause misclassification or confidence reduction. This attack is directly correlated with the model of adversarial threat where the adversary has testing phase capabilities.

Essentially, the creation of an adversarial sample can be defined as:

$$f_{\theta}(x') \neq y$$

where x' is the adversarial sample, f_{θ} is the classifier with parameters θ and y is an incorrect label. The adversarial sample also needs to satisfy the following equation:

$$\|x' - x\|_p < \epsilon$$

where, once again, x' is the adversarial sample, x is the original sample and ϵ is the max amount of perturbation the original sample can receive under the norm p . The total amount of perturbation needs to be minimal so that the adversarial sample created is still recognizable to the human eye but, at the same time, can fool the target model. This distance metric can be seen as a metric that quantifies the similarity between the original image and the adversarial sample created. Currently, the main norm that is used to measure the amount of perturbation allowed in an adversarial sample is the p -norm, however, authors have been proposing different ways of constraining a perturbation such as the Wasserstein distance.

There are four main p norms that are currently taken into consideration when constructing adversarial attacks:

- (1) L_0 norm: This distance measures the total number of pixels that have been changed in the adversarial sample x' . The number of pixels to be modified isn't limited.
- (2) L_1 norm: The distance measures the Manhattan distance between the adversarial sample and the original sample. This distance calculates the absolute value difference between the same pixel in the adversarial sample and the original image and then sums it over all pixels.
- (3) L_2 norm: This distance measures the Euclidean distance between the adversarial sample and the original sample. As the standard euclidean distance is the root-mean-square, the distance can still be small when small changes have been made to many pixels.
- (4) L_{∞} norm: This distance bound the maximum amount of change each pixel of the sample can suffer. However, there is no limit to the number of pixels that can change.

Most of the work related to adversarial examples has focused on the p -norm distances. Wong et al. [25] presented a new way of generating adversarial attacks bounded by the Wasserstein distance to the original inputs. The Wasserstein distance is, essentially, the cost of moving pixel mass from one pixel to another. In this distance metric, the cost of the move increases with the distance.

Despite these norms being the most prevalent in the literature, other authors have proposed other metrics to constraint the differences between an original sample and the adversarial sample.

The following section will detail some of the most relevant evasion attacks. The full list is represented in the table 3.1. To simplify the threat model, the attacks are classified, with regards to the adversary knowledge, in black-box or white-box. It is important to refer

that due to the vast quantity of different attacks that are currently in the literature, it's impossible to represent them all.

Table 3.1: Taxonomy of Evasion Adversarial Attacks

Year	Method	Knowledge	Specificity	Perturbation
2014	L-BFGS [5]	W	T,U	L_2
2015	FGSM [6]	W	T,U	L_∞
2016	JSMA [26]	W	T,U	L_0
2016	Hot/Cold [27]	W	T	L_2
2016	DeepFool [28]	W	U	L_2, L_∞
2017	BIM (I-FGSM) [29]	W	T,U	L_∞
2017	ILLCM [29]	W	T	L_∞
2017	Substitute Model [30]	B	T,U	ND
2017	C&W [31]	W	T,U	L_0, L_2, L_∞
2017	UPSET & ANGRI [32]	B	T	L_2
2017	UAP [33]	W	U	L_2, L_∞
2017	ZOO [34]	B	T,U	L_2
2018	R+FGSM [35]	W	T,U	L_∞
2018	stAdv [36]	W	T,U	L_{flow}
2018	AdvGAN [37]	W,B	T,U	L_2
2018	Boundary Attack [38]	B	T,U	L_2
2018	BPDA [39]	W	T,U	ND
2018	PGD [40]	W	T,U	L_∞
2018	EAD [41]	W	T,U	L_1
2018	ATNs [42]	W,B	T,U	ND
2018	MI-FGSM [43]	W	T,U	L_∞
2018	Limited Queries and Info [44]	B	U	L_∞
2018	Opt-Attack [45]	B	T,U	L_2
2018	SPSA [46]	B	U	L_∞
2018	HCLU [47]	W	T,U	L_2
2019	AutoZOOM [48]	B	T,U	L_1, L_2, L_∞
2019	One-Pixel [49]	B	T,U	L_0
2019	Bandits & Prior [50]	B	U	L_2, L_∞
2019	Sparse L1 Descent (SLIDE) [51]	W	U	L_1
2019	NATTACK [52]	B	U	L_2, L_∞
2019	SimBA [53]	B	T,U	L_0
2019	Wasserstein [25]	W	U	Wasserstein
2019	Threshold & Few-pixel [54]	B	T,U	L_∞, L_0
2019	DDN [55]	W	T,U	L_2
2019	POBA-GA [56]	B	T,U	$Z(A_i^t)$
2019	CornerSearch [57]	B	T,U	L_0
2019	LogBarrier [58]	W	U	L_2, L_∞
2019	Brendel & Bethge [59]	W	T,U	L_0, L_1, L_2, L_∞
2020	Shadow [60]	W	U	L_2, L_∞
2020	HopSkipJump [61]	B	T,U	L_2, L_∞
2020	GeoDA [62]	B	U	L_1, L_2, L_∞
2020	AdvFlow [63]	B	T,U	L_2
2020	Square [64]	B	T,U	L_2, L_∞
2020	GreedyFool [65]	W	T,U	L_0

Continued on next page

Table 3.1 – continued from previous page

Year	Method	Knowledge	Specificity	Perturbation
End of Table 3.1				

B: Black-box
W: White-box
T: Targeted
U: Untargeted
ND: Not-defined

Limited Memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) The attack developed by Szegedy et al. [5] to demonstrates that ML models could be fooled by inputs with very small perturbations. The authors formulate the problem of computing a perturbation as an optimization problem. As this is a hard problem, the problem is solved by transforming it into a box-constrained formulation. This transformation is made using the Limited Memory Broyden-Fletcher-Goldfarb-Shanno algorithm. The objective is to find x' , using the L_2 norm (Euclidean distance), that minimizes

$$c\|x - x'\|_2 + \mathcal{L}(\theta, x', l) \text{ such that } x' \in [0, 1] \quad (3.1)$$

where $x - x'$ is the perturbation, x is the benign sample, x' is the adversarial sample, $\mathcal{L}(\theta, x', l)$ is the loss function of the model (e.g. cross-entropy) and l is the target misclassification label. The value of c is increased until an adversary sample is found. This attack was capable of fooling, that is, attacking with success, at the time, state-of-the-art ML models such as the AlexNet [66] and QuocNet [67].

Fast Gradient Sign Method (FGSM) Goodfellow et al. [6], following their hypothesis for the existence of adversarial samples, proposed a new way of generating perturbations. Opposite to the expensive method proposed in [5], this new method was faster since it only performs a one-step gradient update of the loss function of the model. More so, the perturbation bound used, contrary to the L-BFGS, is the max norm which allows a greater perturbation in the image. An adversarial sample generated using the FGSM is given by

$$x' = x + \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(\theta, x, y)) \quad (3.2)$$

where ϵ is, usually, a small constant that is multiplied by the sign of the vector of the gradient. The y value is the label of the generated sample and it allows the adversary to choose if the attack is targeted or non-targeted. The required gradient can be easily computed using the backpropagation algorithm. The figure 3.4 represents an image with a perturbation, generated using FGSM, that while being quasi-imperceptible to the human eye, creates a higher level of confidence in the model of an incorrect label output.

Jacobian-based Saliency Map Attack (JSMA) Previous to the paper written by Papernot et al. [26], most attacks focused on using the L_∞ or the L_2 norm such as the FGSM attack. The goal of the authors was to develop a new attack restricted by the L_0 norm which only allows for a small number of pixels in an image to be changed. The proposed attack uses *adversarial saliency maps* (figure 3.5), which are an extension of saliency maps, to evaluate the effect that each input feature has on the model, in images those features are the pixels. Pixels with the largest saliency are the ones that create the biggest impact on the model meaning that those are the ones to be changed. This method is repeated until the number of pixels to be changed is reached or the adversarial sample was capable of fooling the model.

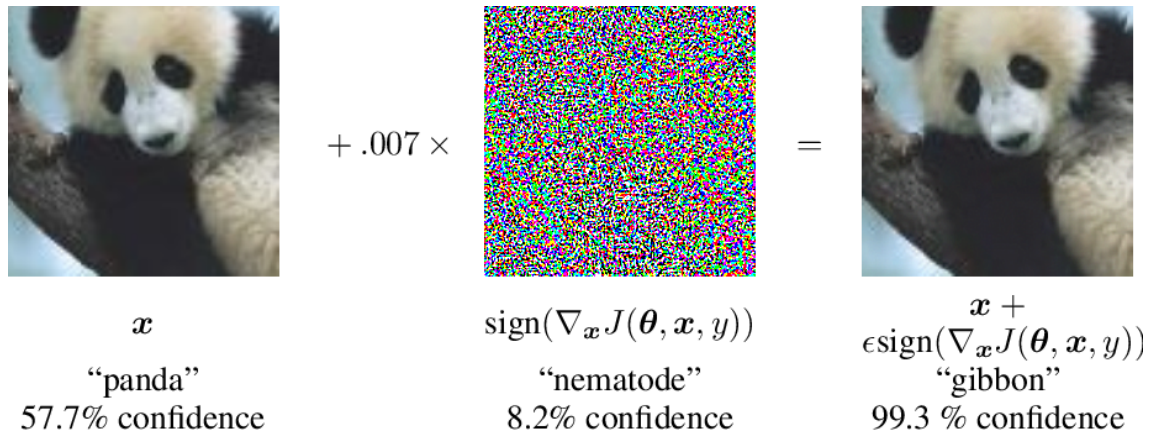


Figure 3.4: An example of an adversarial attack using the Fast Gradient Sign Attack. The image, after being perturbed, was misclassified by the model with a higher confidence level than the original image. (Image credits: Goodfellow et al. [6])

Hot/Cold Attack Rozsa et al. [27] propose an attack based on changing the values of the penultimate layer of a target model (or the logit layer). The logits are modified such that the probability of a class "hot" increases and the probability of the original class, the "cold" class, decreases. The logits of the target model can be represented as:

$$w_{hc} = \begin{cases} |h_j(x)|, & \text{if } j = \bar{y} \\ -h_j(x), & \text{if } j = y \\ 0, & \text{otherwise} \end{cases} \quad (3.3)$$

where h_j represents the j -th element of the logits layer, y is the true label of the image, and \bar{y} is a target class. Then, backpropagation is applied to compute the gradient values of the new logits layer and after obtaining the gradient directions, the adversarial perturbations are found using line-search and binary search.

DeepFool Moosavi-Dezfooli et al. [28] proposed the DeepFool algorithm that starts with an unperturbed image that is inside a region limited by the decision boundaries of the classifier. DeepFool iteratively moves the original input point towards the closest decision boundary using a perturbation vector. In the case of nonlinear classifiers, the decision boundaries are linearized and the distance to the approximated decision boundaries is used to move the original point (seen in figure 3.6). This process is repeated until the original image is misclassified by the model. DeepFool is also capable of measuring the perturbation using various distance metrics such as the L_2 and L_∞ norms. The proposed attack was shown to fool various ML models, notably, GoogLeNet [68], LeNet [69], and CaffeNet [70].

Basic Iterative Method (BIM) A method proposed by [29] that is referred to as Basic Iterative Method or Iterative-Fast Gradient Sign Method. This attack applies the FGSM multiple times with a small step size and uses the max norm to constraint the perturbation added to the sample. Adversarial samples generated by this attack can be formulated as:

$$x'_{i+1} = \text{Clip}_{x,\epsilon} \left\{ x'_i + \alpha \cdot \text{sign}(\nabla_x \mathcal{L}(x'_i, y)) \right\} \text{ where } x'_0 = x \quad (3.4)$$

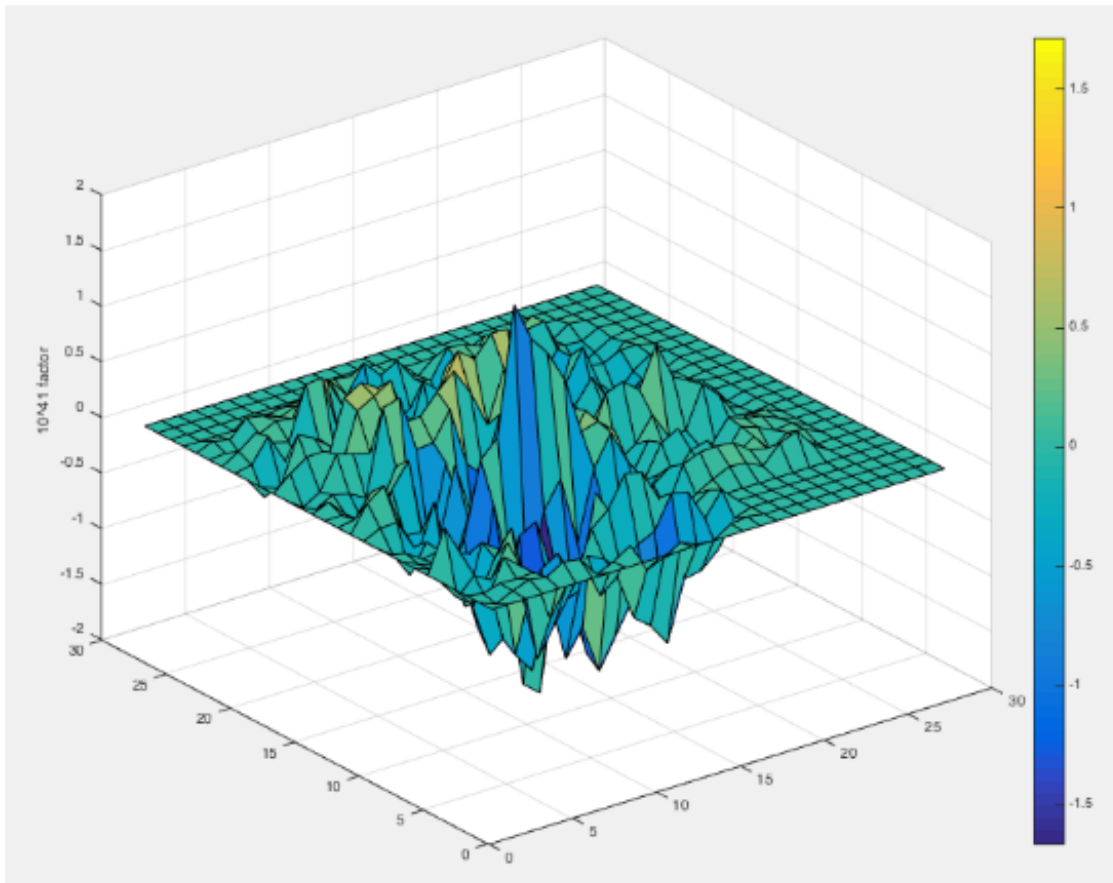


Figure 3.5: Saliency map of a 28x28 image. Large absolute values correspond to the pixels with the greatest impact on the decision of the network (Image credits: [26])

where x is the original benign sample, i is an iteration of the attack, $\mathcal{L}(x'_i, y)$ is the loss function of the target model and $Clip(\cdot)$ restrains the perturbation added to the image.

ILLCM (Iterative Least-Likely Class Method) The same authors of the BIM attack developed a targeted attack method called the Iterative Least-Likely Class Method where the goal is to create an adversarial sample that is classified as the least likely class of the original input.

Substitute Model Attack Papernot et al. [30] proposed one of the first black-box techniques to generate adversarial attacks that consisted in using a substitute model. The attack consists in training a substitute model using a synthetic dataset that is constructed using the output labels of the target model. Then, adversarial examples are constructed using white-box methods on the substitute model which are then used to attack the target model, leveraging the transferability property of classifiers found by Szegedy et al. [5]. The success of this attack is based on the knowledge that the adversary has on the target model and the quality of the constructed synthetic dataset. In order to limit the number of queries that the substitute model makes to the target model, the authors developed the Jacobian-based Dataset Augmentation technique. This technique focuses on approximating the oracle's decision boundaries with few queries and not on maximizing the accuracy of the substitute model. This attack obtained an 84.24% misclassification ratio on a deep neural

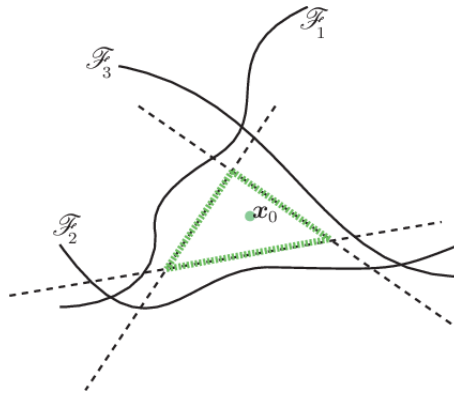


Figure 3.6: Approximation of the decision boundaries to find an adversarial sample. (Image credits: [28])

network of MetaMind, a 96.19% on a classifier hosted by Amazon, and 88.94% on a classifier provided by Google.

Carlini & Wagner Attacks C&W propose a family of white-box attacks in respect to the L_2 , L_0 , and L_∞ bounds. After experimenting with various different and possible loss functions, the one used to tackle the optimization problem proposed by Szegedy et al. can be formulated as:

$$\mathcal{L}_{CW}(x', t) = \max(\max_{i \neq t} \{Z(x')_{(i)}\} - Z(x')_{(t)}, -\kappa) \quad (3.5)$$

where $Z(x')_{(i)}$ is a value of the logits layer of the classifier different, $Z(x')_{(t)}$ is the value of the target label in the logits layer and κ is a parameter that controls the misclassification confidence of the adversarial sample when classified by the model. If t has the highest logit value of all the other logits, the difference will be negative. If the difference between t and the highest value of the logits is negative and that value exceeds κ , the optimization stops since the sample is being misclassified as the target label with high confidence. With a loss function defined, the L_2 attack can be defined as:

$$\min_w \|x'(w) - x\|_2^2 + c \cdot \mathcal{L}_{CW} \quad \text{where } x'(w) = \frac{1}{2}(\tanh(w) + 1) \quad (3.6)$$

where w is a value to be minimized and c is a constant that starts with a low value (e.g., 10^{-4}) and is doubled until an adversarial sample is found or a threshold is exceeded (e.g., 10^{10}). The L_0 and L_∞ attacks are much more complex since the distances are non-differentiable and for that, they both use an iterative approach for finding adversarial examples. The adversarial samples generated using all three attacks were shown to be more powerful than other, at the time, state-of-the-art attacks and also capable of breaking classifiers trained under the defensive distillation technique [71] with almost 100% fooling rate.

UPSET & ANGRI The authors propose two black-box attacks, UPSET and ANGRI that generate targeted adversarial images. The UPSET model tries to generate a single perturbation for each class using only the information of the input image. An adversarial image generated using UPSET can be formalized as:

$$x' = U(x, t) = \max(\min(s \cdot R(t) + x, 1), -1) \quad (3.7)$$

where U is the UPSET network, R is a residual generating network that is responsible for generating the perturbation and t is the target class. The perturbation generated by R is multiplied by a scalar value s that constrains the perturbation to be added to the image. ANGRI, contrary to UPSET, tries to add a perturbation that makes a classifier misclassify an image from class y as being from class t . The formulation of an adversarial sample generated by ANGRI is formalized as:

$$x' = A(x, t) \quad (3.8)$$

where A is the ANGRI network and t is the target label. The loss function used by both UPSET and ANGRI is represented as:

$$\mathcal{L}(x, x', t) = \mathcal{L}_C(x', t) + \mathcal{L}_F(x, x') = - \sum_{i=1}^m \log(C_i(x')[t]) + w \|x' - x\|_k^k \quad (3.9)$$

where m is a number of pre-trained classifiers and $C_i(x')$ is the classification probabilities of the adversarial image, $\mathcal{L}_C(x', t)$ is the classification loss of a classifier and $\mathcal{L}_F(x, x')$ is the fidelity loss which is given by a weight w that ensures balance between the similarity, with the original image, of the adversarial image and the fooling capability of the adversarial image. The constraint of the perturbation on the adversarial image is given by $\|x' - x\|_k^k$ where k is usually 2 and therefore, the constraint is the L_2 norm. Both proposed networks can train on multiple classification systems which allows them to attack multiple target neural networks simultaneously. Since ANGRI focuses on perturbing a single image, it generates stronger adversarial images in terms of similarity with the original image. Since UPSET generates a perturbation for each class, it is then very fast after the training phase, which is during the inference phase.

Universal Adversarial Perturbations Moosavi-Dezfooli et al. [33] propose a method of generating an universal perturbation (figure 3.7), given a training dataset of a model, that when applied to most images of the dataset is capable of fooling the target model. This attack computes perturbations of individual inputs iteratively using a sample-specific attack such as the FGSM. Each input-specific perturbation is added to create the universal perturbation which is then input-agnostic. This method was capable of fooling various ML models such as GoogLeNet with fooling rates close to 94% for the ImageNet dataset. The authors explain this phenomenon as a result of correlations in the decision boundaries of the classifiers.

ZOO Chen et al. [34] proposed a new black-box method for generating adversarial examples. The method is based on the formulation of the C&W attack since it was, at the time, one of the most effective white-box attacks. However, contrary to the C&W attack, this algorithm doesn't assume that it can use backpropagation to obtain the gradient, instead the authors modify the loss function formulation to the following:

$$\text{Targeted: } \mathcal{L}(x, t) = \max_{i \neq t} \{ \max \log(f(x)_{(i)}) - \log(f(x)_{(t)}), -\kappa \} \quad (3.10)$$

$$\text{Untargeted: } \mathcal{L}(x) = \max_{i \neq y} \{ \log(f(x)_{(y)}) - \max \log(f(x)_{(i)}), -\kappa \} \quad (3.11)$$

Since this is a black-box attack, different from the C&W formulation, ZOO is computed using the softmax probabilities $f(x)_{(i)}$. Similar to other formulations, t represents the target class and y represents the true label of the class while i are the probabilities of the softmax layer for the possible classes. κ has the same objective that is had in the

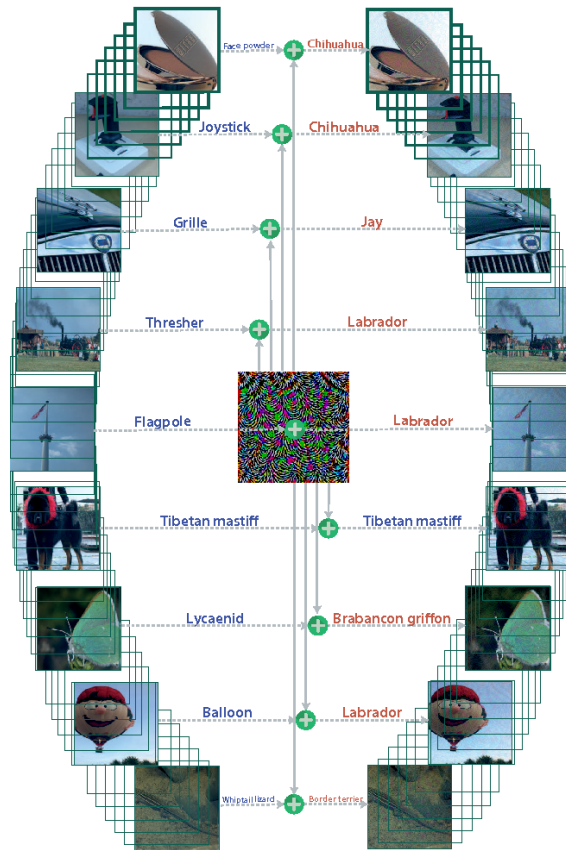


Figure 3.7: A perturbation generated using the UAP method. The perturbation, when applied to various images of a dataset, is capable of fooling the target model. (Image credits: [33])

C&W formulation, to control the minimum desired confidence for the adversarial sample. ZOO computes an approximation of the gradient by using stochastic gradient descent over batches of input dimensions which leads to an increase in performance. The authors propose two variations of the ZOO attack that use coordinate stochastic descent: ZOO-Adam and ZOO-Newton. The results suggested that ZOO-ADAM was slightly faster than ZOO-Newton.

R+FGSM Gradient masking is one of the most common defenses used in order to protect classifiers from gradient-based attacks such as FGSM. Tramèr et al. [35] introduce a variation of the FGSM that consists in adding a small random perturbation to the perturbation generated by the FGSM. This attack was shown capable of attacking an adversarially trained Inception ResNet v2 [72] and Inception v3 [73], obtaining higher fooling rates than the original FGSM.

Spatially Transformed Adversarial Examples (stAdv) One important factor to consider when generating adversarial examples is the similarity between the original sample and the perturbed one. However, when using norms such as the L_2 norm, the adversarial samples created may fail to respect this criterion. Motivated by this, Xiao et al. [36] propose a new category of adversarial attacks that focus on generating realistic adversarial samples by changing the position instead of the values of the pixels in an image. The proposed attack focus on optimizing the *geometrical* similarity between the original benign

sample and the adversarial sample. The authors represent the spatially-transformed image as a flow field $f(\Delta u, \Delta v)$ where each element represents a change in the 2D coordinates of a pixel. The location of a pixel in the original image can be found by computing $(u^{(i)}, v^{(i)}) = (u_{adv}^{(i)} + \Delta u^{(i)}, v_{adv}^{(i)} + \Delta v^{(i)})$. The adversarial image can be obtained by calculating the following equation for each pixel in the image:

$$x_{adv}^{(i)} = \sum_{q \in \mathcal{N}(u^{(i)}, v^{(i)})} x^{(q)} (1 - |u^{(i)} - u^{(q)}|) (1 - |v^{(i)} - v^{(q)}|) \quad (3.12)$$

where $\mathcal{N}(u^{(i)}, v^{(i)})$ is the 4-pixel neighborhood of the i -th pixel with a spatially-transformed coordinate $u^{(i)}, v^{(i)}$. It is also important to note that x_{adv} is differentiable with respect to the flow field f . Given an input image that is not perturbed, adversarial samples are found by minimizing the following function:

$$f^* = \underset{f}{\operatorname{argmin}} \mathcal{L}_{adv}(x, f) + \tau \mathcal{L}_{flow}(f) \quad (3.13)$$

where τ is the balance factor between the two loss functions, \mathcal{L}_{adv} is the classification loss of the adversarial examples in the target model, and \mathcal{L}_{flow} is a metric that preserves the similarity between the original image and the adversarial one (the perturbation distance norm in other attacks). Adversarial samples generated by this method (figure 3.8) were extremely similar to the original samples while at the same obtaining misclassification similar or even higher than other attacks such as FGSM against defended neural networks.

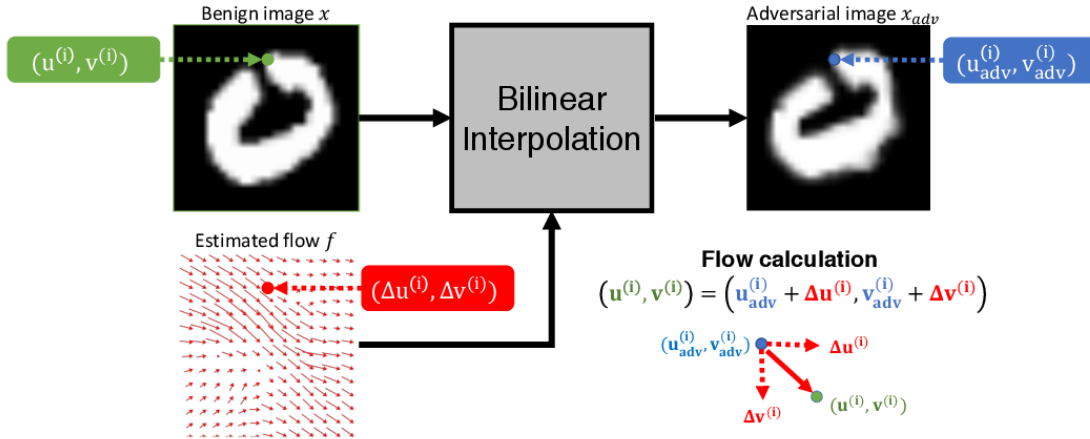


Figure 3.8: The process of generating an adversarial sample using the stAdv attack. The green point is the location of the pixel in the benign sample. The red flow represents the displacement that the pixels were subjected to. The blue dot represents the position of the green dot in the adversarial sample (Image credits: [36])

AdvGAN Similar to a P-ATN, an AdvGAN [37] consists in training a neural network to produce adversarial samples. However, AdvGAN also uses a discriminator, based on the work of Goodfellow et al., in order to produce visually realistic images capable of misleading the target models. The AdvGAN framework mainly consists of a generator G , a discriminator D , and the target neural network f . The objective function of the generator consists of various terms and can be represented as:

$$\mathcal{L} = \mathcal{L}_{adv}^f + \alpha \mathcal{L}_{GAN} + \beta \mathcal{L}_{hinge} \quad (3.14)$$

where \mathcal{L}_{adv}^f is the loss for fooling the target model f (usually the cross-entropy of f), \mathcal{L}_{GAN} is the adversarial loss which is used to ensure that the perturbed data is similar to the benign data, and \mathcal{L}_{hinge} is used to bound the perturbation of the adversarial sample. In the function, α and β are constants that control the importance of the loss functions. Adversarial GAN's can be both used in a white-box and a black-box setting. In the white-box setting, the AdvGAN only needs to have white-box access while the network is being trained, after the training is complete, it doesn't require any more access to the model and can work in a black-box setting. The authors called this a semi-white-box attack. In the black-box setting, the authors use a dynamic distillation technique that consists in constructing a local model to distill the target model. In this scenario, a random subset of instances different from the training dataset is chosen to train the local model. The AdvGAN achieved better performance over the FGSM and C&W attacks in models defended by adversarial training [6], ensemble adversarial training [35], and iterative training [40].

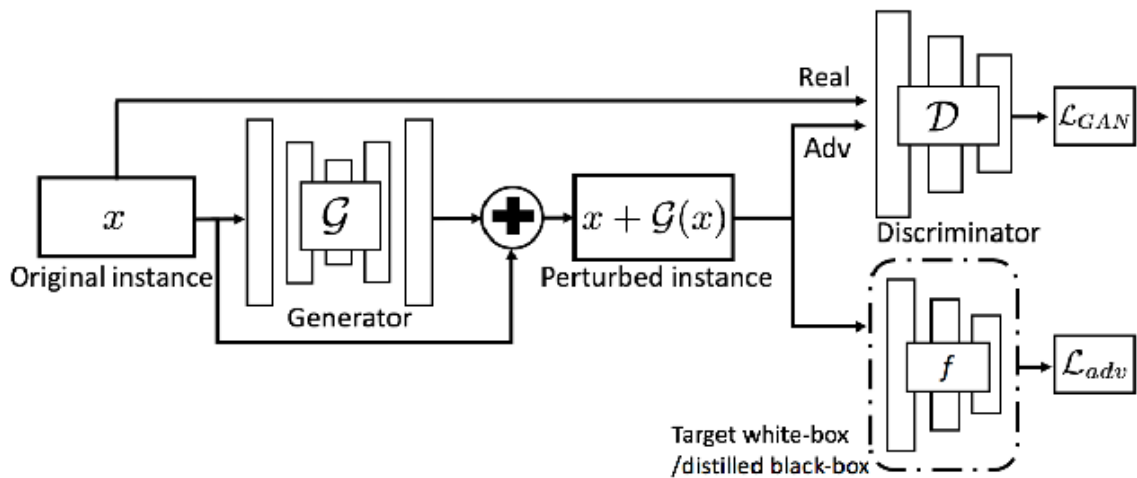


Figure 3.9: The architecture of an AdvGAN. (Image credits: [37])

MI-FGSM The MI-FGSM is a family of attacks that is built upon the I-FGSM and uses the momentum method which is a technique for accelerating gradient descent algorithms. The integration of momentum allows for greater stabilization of update directions for perturbations and allows the gradient sign method to escape local maxima. This allows the adversarial samples to have greater transferability even against stronger models. The family of MI-FGSM attacks isn't limited just to non-targeted attacks under the L_∞ norm restriction, but it is possible to generate attacks under the L_2 norm and targeted attacks.

Boundary Attack Brendel et al. [38] suggest that decision-based attacks are extremely relevant since they are much more probable of happening in the real world since they only require the final decision of the model such as the top output label. Moreover, the authors hypothesize that they can be more robust to standard defenses that are gaining importance in the literature such as gradient masking and adversarial training. At the time of this attack, there was no effective decision-based attack that could scale on natural datasets such as ImageNet. The authors propose the boundary attack that works differently from other methods. The starting point is a clean sample and the algorithm iteratively walks along the boundary between the adversarial label and the clean label. The method needs to guarantee that the image stays adversarial but the distance to the decision boundary

of the clean label is minimized. The process can be targeted or untargeted depending on whether the initial image was chosen or randomly selected from a distribution. An example of a targeted attack can be seen in figure 3.10.

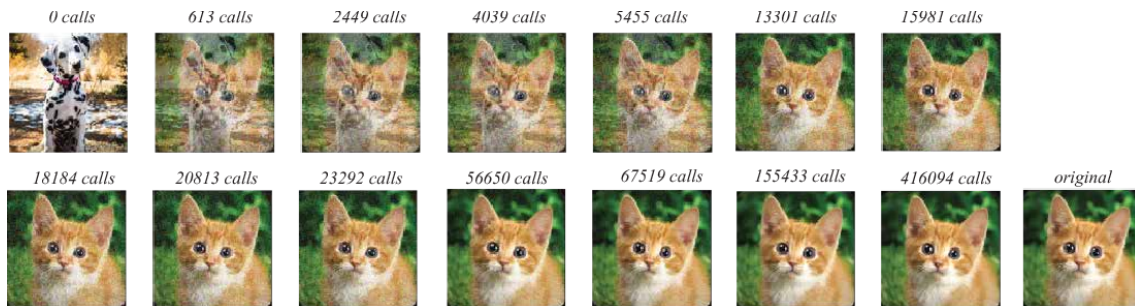


Figure 3.10: The process of generating an adversarial image using the Boundary Attack, one of the first decision-based black-box attacks to be presented. (Image credits: [38])

BPDA Various defenses have been developed in order to protect models from adversarial attacks. One of the most common baseline of defenses in the literature is gradient masking, that focuses on obfuscating gradients, or in other words, gradients that are not useful for white-box attacks. Backward Pass Differentiable Approximation is a technique that can be applied to adversarial attacks in order to bypass gradient masking defenses. This method, essentially, calculates an approximation of the gradients.

Projected Gradient Descent (PGD) In the search for adversarial robustness of networks and of methods capable of training neural networks robust to adversarial attacks, Madry et al. [40] developed a new white-box attack. The attack proposed, Project Gradient Decent, is still one of the most relevant white-box attacks in the literature, serving as the base for other white-box and black-box attacks. PGD is essentially a multi-step variant of the L_∞ FGSM attack and can be formulated as:

$$x^{t+1} = \pi_{x+\mathcal{S}}(x^t + \epsilon \cdot \text{sign}(\nabla_x L(\theta, x, y))) \quad (3.15)$$

While this attack may seem similar to the BIM attack, the PGD attack, contrary to BIM, initializes the example in a random point in the L_∞ -ball which allows for a broader search of samples in the loss function landscape. A very simple visual representation of the PGD algorithm can be seen in figure 3.11.

Elastic Net Attack (EAD) Chen et al. [41] propose a new process of generating adversarial attacks based on elastic-net regularization which is a technique used to solve high-dimensional feature selection problems. The authors' starting point is the loss function used by C&W which is then complemented by introducing the elastic-net regularization. The developed L_1 attack was demonstrated to be capable of generating adversarial samples successful in breaking MNIST, CIFAR10, and ImageNet networks with similar results to state-of-the-art L_2 and L_∞ attacks.

Adversarial Transformation Networks Adversarial Transformation Networks are networks trained to transform an input into an adversarial example. There are two principal approaches to generating adversarial samples using an ATN: Perturbation ATN (P-ATN)

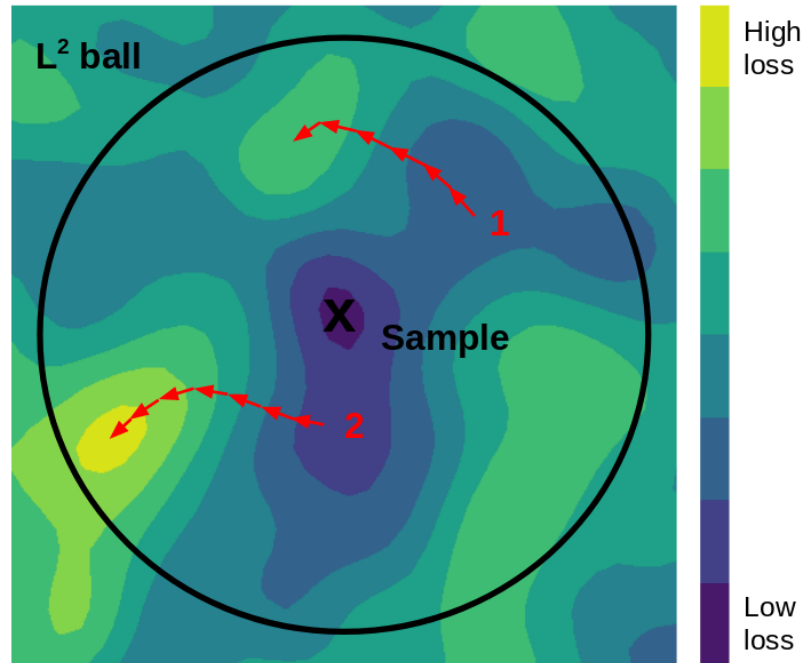


Figure 3.11: The iterative process of finding an adversarial sample using the Projected Gradient Descent Attack. On the first try, the sample found has a low loss which means it will be a weak adversarial sample. In the second try, however, the sample lands in a region with high loss that results in a strong adversarial sample. (Image credits: <https://towardsdatascience.com/know-your-enemy-7f7c5038bdf3>)

and Adversarial Autoencoding (AAE). The P-ATN is an ATN trained to generate a perturbation to a given input while AAE is an ATN that transforms encode-and-reconstruct a benign input into an adversarial sample. After training the networks, adversarial attacks can be generated by simply inputting the benign sample to the ATN resulting in a possible faster generation than other gradient-based approaches such as fast gradient methods. It is important to note that an ATN is only capable of generating adversarial examples of a single class of the target model. In order to generate adversarial attacks of multiple classes, multiple ATN's need to be trained.

Limited Queries and Info Attack Ilyas et al. [44] develop attacks for three different limited black-box limited settings. The first setting limits the number of queries that the attacker can make to the target model. The authors use Natural Evolutionary Strategies (NES) to estimate the gradient from queries and then Projected Gradient Descent is applied to generate the adversarial samples. In the second scenario, the adversary only has partial information about the confidence scores for a given input, meaning that in some cases, only the top-1 class probability is output. In this setting, the starting point is a benign sample of the target class so that it will appear in the top-k classes of the output. The algorithm, over the iterations, needs to make sure the adversarial class stays in the top-k classes and, at the same time, the perturbation added maximizes the probability of the input image getting classified as the target class. The third scenario is an extremely limited one since the attacker only has access to the output labels of the model for a given input. The idea behind this attack focuses on defining a discretized score of an adversarial sample to represent the adversarial nature of that sample at each iteration given only the top-k classes. The tests performed on the Google Cloud Vision API demonstrated that the

attacks were very capable of fooling the ML model even under such limited settings. The results of the partial information attack can be seen in figure 3.12.

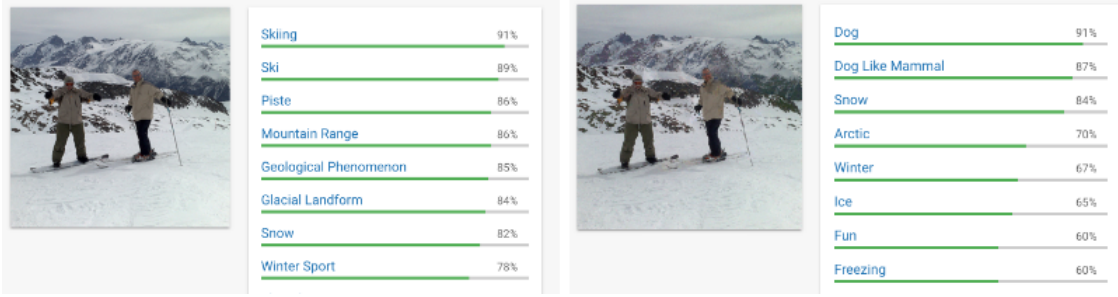


Figure 3.12: The results of the Partial Information Attack on the Google Cloud Vision API. (Image adapted from: [44])

Opt-Attack Cheng et al. tackle the challenge of developing an adversarial attack method in a hard-label setting in which the attacker only has access to the output labels given an input, instead of the probability outputs. The authors reformulate the hard-label black-box settings as an optimization problem. The formulation for both untargeted and targeted attacks is the following:

$$\text{Untargeted attack: } g(\theta) = \underset{\lambda > 0}{\operatorname{argmin}} \left(f\left(x_0 + \lambda \frac{\theta}{\|\theta\|} \right) \neq y_0 \right), \text{ where } y_0 \text{ is the true label} \quad (3.16)$$

$$\text{Targeted attack: } g(\theta) = \underset{\lambda > 0}{\operatorname{argmin}} \left(f\left(x_0 + \lambda \frac{\theta}{\|\theta\|} \right) = t \right), \text{ where } t \text{ is the target label} \quad (3.17)$$

where θ represents the search direction and $g(\theta)$ is the distance from x_0 (the original sample) to the nearest adversarial example in the direction of θ . The algorithm follows the direction θ in search for an adversarial example which allows for a minimization of the distortion $g(\theta)$. The adversarial sample is represented by:

$$x^* = x_0 + g(\theta^*) \frac{\theta^*}{\|\theta^*\|}, \text{ where } \theta^* = \underset{\theta}{\operatorname{argmin}} g(\theta) \quad (3.18)$$

The algorithm works as follows: iteratively, a fine-grained search is performed and then complemented by a binary search to find the decision boundary. After computing the g value, the estimated gradient is calculated using the Randomized Gradient-Free method which is an iterative Zeroth Order Optimization method. θ is then updated using a step size, calculated using a backtracking search-line approach at each step, and the estimated gradient.

Simultaneous Perturbation Stochastic Approximation Attack (SPSA) Similar to other methods that are black-box, the SPSA attack uses the SPSA estimator to compute an approximation of gradients of the target model. The proposed attack estimates the gradients of the loss function of the target model N times and then computes the negative direction estimate of the gradients. Then, it perturbs the original sample along the previously calculated negative gradient direction making sure that the perturbation stays within the allowed L_∞ bounds. The SPSA attack was shown to break various defenses such as PixelDefend [74] and High-level Guided Denoising [75].

High Confidence Low Uncertainty Attack An investigation on the effect of adversarial examples on Bayesian Neural Networks (BNNs) performed by Grosse et al. [47] demonstrated that it is possible to fool BNNs. The authors extend adversarial samples from the output of classifiers to the Bayesian confidence and uncertainty levels where a higher value means more confident and more uncertain, respectively. The optimization problem of high-confidence-low-uncertainty results is the following:

$$\begin{aligned} \min_{\delta} \quad & \|\delta\|_2 \\ \text{s.t confidence} \quad & (f(x + \delta)) > 0.95, \\ \text{and uncertainty} \quad & (f(x + \delta)) \leq \text{uncertainty}(f(x)), \end{aligned} \quad (3.19)$$

where the perturbation is limited using the euclidean norm, the first constraint maximizes the confidence of the classifier, and the second limits the uncertainty. Since the uncertainty estimates of BNNs are non-smooth, the authors use a Gaussian Process Classifier (GPC) as a substitute target model. The constructed adversarial examples were extremely similar to the benign original samples but at the same time, capable of achieving very high levels of confidence and low levels of uncertainty on the GPC. In order to test the transferability of the attack, the HCLU adversarial examples were tested on a DNN, a BNN, and a different GPC. The results show that the examples were very capable of fooling the target classifiers achieving 90% of misclassification ratio in Spam, MNIST19, and FMNIST19 datasets and over 50% in an FMNIST57 and MNIST38 datasets.

AutoZOOM Many of the black-box methods for generating adversarial attacks seen until now, query the target model a vast number of times which can be extremely time inefficient and computationally expensive. Motivated by a lack of methods that are capable of generating strong adversarial samples and are also query-efficient, the authors propose a new black-box framework to generate adversarial samples. AutoZOOM introduces two new mechanisms that diminish the number of queries needed to generate a strong adversarial sample: a random gradient estimation strategy and an autoencoder with two modes. The authors formulate finding an adversarial sample x as the following:

$$\min_{x \in [0,1]^d} \|x - x_0\|_p + \lambda \cdot \mathcal{L}(x, M(F(x)), t) \text{ where } x_0 \text{ is the original image} \quad (3.20)$$

where $\|x - x_0\|_p$ is the perturbation, $\mathcal{L}(\cdot)$ is the training loss of the target model or a modified function such as the one used by C&W. t is the target label and λ is a regularization coefficient. $M(\cdot)$ is a monotonic transformation that is applied to the output of the classifier, represented as $F(x)$. The input image with dimension d and $x \in [0, 1]^d$ constraints the generated adversarial image to the valid image space. Since finding the loss function is impossible in a black-box scenario such as this one, the authors propose approximating the gradients using an averaged random gradient estimation. The averaged random gradient estimator of the loss function of the target model can be defined as:

$$\bar{g} = \frac{1}{q} \sum_{j=1}^q b \cdot \frac{f(x + \beta u_j) - f(x)}{\beta} \cdot u_j \quad (3.21)$$

where β is a smoothing parameter, b is a scalar value that balances the bias and variance trade-off of the error of g and u is a vector that is randomly obtained from an euclidean sphere. q is a number of random directions that dictate the direction of u . Multiple directions reduce the variance of the average random gradient estimation. The authors demonstrate that images with high dimensionality can make gradient estimations very query inefficient. In order to reduce query inefficiency, they perform random gradient

estimation in a lower dimension of the original image. The perturbation added to an image is made using a decoder that translates a perturbation generated in a low dimension space to the original dimension of the input image. AutoZOOM provides an autoencoder (observable in 3.13) and a BiLIN (channel-wise bilinear image resizer) that can serve as the decoder for the previously mentioned step. AutoZOOM obtained more than 93% of query reduction when compared to the ZOO attack [34] while, at the same time, obtaining similar, and even greater, attack success rates.

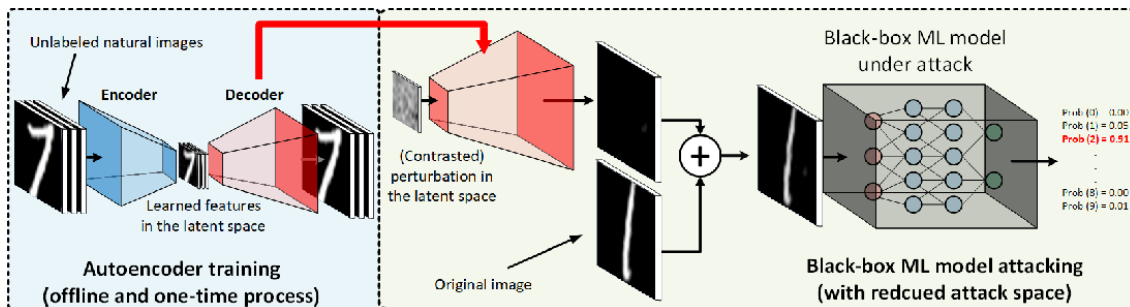


Figure 3.13: The autoencoder used by AutoZOOM to generate adversarial samples. (Image credits: [48])

One Pixel Attack Su et al. [49] suggested a new family of semi-black-box attacks that only modify a very limited number of pixels of an image in order to create an adversarial sample. The requirements for the attack are the predicted classes and the prediction probabilities from the target model. Opposite to other methods, the one-pixel attack doesn't limit the strength of the modification, instead, it focuses on creating the maximum perturbation possible in a single pixel. Instead of using a gradient approach for finding a perturbation, this attack uses the differential evolution algorithm to evolve the candidate solution which, in this case, is the pixel to be changed. The use of differential evolution has a few benefits such as needing less information from the target and having a higher probability of finding the global optima.

Bandits & Prior Attack Ilyas et al. [50] study the existence of prior gradient information based on two observations made: first, the input data point for which gradient is computed is not arbitrary and is reflected in the gradient and, second, when performing iterative gradient attacks, the successive gradients show a heavy correlation between them. Taking this into consideration, the authors propose a new formulation of the gradient estimation problem based on prior information. The authors analyze two classes of prior information extracted from the dataset: time-dependent priors and data-dependent priors. While time-dependent priors are supported by the existence of a similar correlation between successive gradients, the data-dependent priors are supported by a spatially local similarity that exists in images (i.e pixels that are close tend to have similar values). The proposed attack is built on top of the bandit optimization framework, a tool used in online convex optimization. Each t round, the gradient g_t is estimated based on the latent vector of the previous round, v_{t-1} . The loss ℓ_t function is represented by:

$$\ell_t(g) = -\langle \nabla L(x, y), \frac{g}{\|g\|} \rangle \quad (3.22)$$

where g is the gradient estimate and is accessed using the finite differences method and $L(x, y)$ is the classification loss on an image x with true label y . The time-dependent priors

are reflected in the latent vector between the gradient estimations. The data-dependent priors enforce a particular structure of the latent vector.

Sparse L1 Descent (SLIDE) Adversarial training is yet another form of defending classifiers from adversarial attacks. For the defense to have good results and effectively protect the classifier, it needs to use adversarial samples generated by strong and efficient attacks. Tramer et al. [51], motivated by this, propose a new L_1 -based attack, based on the PGD attack, capable of generating stronger and efficient samples. The new attack is also a descent attack but, contrary to the PGD, has finer control over the update step. The proposed attack was shown to outperform the L_1 PGD attack.

NATTACK The approach to generate adversarial attacks by Yandong et al. [52] focus on finding a probability density distribution over a small region (l_p ball) centered around a benign input. A sample drawn from the distribution is likely to be adversarial which allows drawing multiple and different adversarial attacks. The formulation of finding an adversarial sample is given by:

$$x' = \text{proj}_S(g(z)), \quad z \sim \mathcal{N}(z|\mu, \sigma^2) \quad (3.23)$$

where S is the region that contains x and $\mathcal{N}(z|\mu, \sigma^2)$ is a normal distribution whose values of μ and σ^2 are found using NES and grid search, respectively. $g : \mathbb{R}^{\dim(\mu)} \mapsto \mathbb{R}^{\dim(x)}$ maps an instance to the space of the target model input.

SimBA SimBA [53] is a black-box model that is both straightforward and efficient in the number of iterations that it uses to find an adversarial sample. As input, the algorithm takes the input image, a set of orthonormal vectors, and a step size. Each iteration, the attack selects a random q vector from the aforementioned set while making sure that the newly selected vector isn't in the opposite direction of the previous one. This allows for lesser queries to be made and diminishes the time it takes to find a sample. The q vector is then added to the image and, if the probability of misclassification of the image is lower than the previous iteration, the vector is instead subtracted. A variation of the SimBA that uses Discrete Cosine Transformation is also proposed by the authors. Despite the focus of SimBA being simplicity, it was capable of fooling the Google Cloud Vision, achieving a 70% success rate after only 5000 queries.

Wasserstein Attack Wong et al. [25] propose a new category of attacks that uses the Wasserstein distance as the bound of adversarial examples. The proposed attack use Projected Gradient Descent as the way to generate the adversarial examples which are bounded by the Wasserstein distance instead of the regular distances such as the L_2 and the L_∞ norms. However, since computing the Wasserstein distance requires solving an optimization problem that is not efficient in practice, the Sinkhorn-Knopp matrix algorithm is used to compute an approximation.

Threshold & Few-pixel Attacks The authors hypothesize that not all distance metrics are needed to evaluate the robustness of a classifier and that those metrics can lead to adversarial examples that can be easily detected by humans. The authors propose a new formulation for adversarial ML that has into consideration the perturbation of the sample:

$$\min g(x + \epsilon_x)_c \text{ subject to } \|\epsilon_x\| \leq th \quad (3.24)$$

where th is a pre-defined threshold value. This value has the objective of limiting the perturbation that an image can have in order to stop it from being unrecognizable. Using this new formula that takes into consideration the quality of the samples, the authors propose new two black-box attacks that aren't gradient-based and instead, use the canonical version of the optimization algorithm Covariance Matrix Adaptation Evolution Strategy (CMA-ES). The first attack is the threshold attack which uses the L_∞ norm and the algorithm search space is given by $m \times n \times c$ where $m \times n$ is the size of the image and c is the number of channels. The perturbation to be added to the image only needs to respect the threshold value defined. The second attack is a variation of the One-pixel attack [49] and it uses the L_0 norm. The search space for this algorithm is smaller than the threshold attack since it is a combination of the pixel values and position for all the pixels. The results obtained showed that both attacks were capable of fooling various ML models such as ResNet, DenseNet, WideResNet, AllConv, and CapsNet, however, the threshold value had, as expected, a great impact on the results. While the threshold attack with $th = 1$ only achieved 30% of misclassification error, with $th = 10$ achieved 98% error in the WideResNet.

Decoupled Direction and Norm (DDN) Attack The authors [55] propose an alternative method for generating white-box gradient-based attacks. Various white-box attacks such as the Carlini&Wagner attack and the L-BFGS attack, convert the constrained problem of generating adversarial examples into an unconstrained problem. However, choosing the constant multiplier is performed in an *ad hoc* way which can result in a very large number of iterations. The authors propose a method that doesn't impose a penalty on the L_2 norm during the optimization step of the method and instead, they project the perturbation on a ϵ -sphere around the original sample. The norm is constrained and increases if the sample isn't adversarial and decreases otherwise (figure 3.14). The proposed attack performs similarly to other attacks such as DeepFool and the C&W L_2 attack but generates adversarial samples much faster. Under a 100 iterations max restriction, the DDN attack performed always better than the C&W attack.

POBA-GA Chen et al. [56] propose a novel Perturbation Optimization Black-box Attack based on Genetic Algorithm capable of generating adversarial samples comparable to samples created by white-box attacks. The algorithm of POBA-GA starts by generating random perturbations, the quality and the diversity of the initial perturbations are crucial to obtaining an optimal solution. A low-quality initial solution may need more iterations to converge to the optimal solution. The initial types of perturbation are generated based on different noise point pixel thresholds, the number of noise points, and noise point size. Each perturbation is then added to an original sample in order to create the corresponding adversarial samples. The adversarial samples are then evaluated using a fitness function. The fitness function evaluates the similarity of the adversarial sample to the original image and the misclassification confidence of the model on the adversarial sample. The fitness function is defined by:

$$\phi(AS_i^t) = P(AS_i^t) - \frac{\alpha}{\max Z(A^0)} Z(A_i^t) \quad (3.25)$$

where $\phi(AS_i^t)$ is the fitness function of the adversarial sample $\phi(AS_i^t)$, $P(AS_i^t)$ is the calculated attack performance of the sample, $Z(A_i^t)$ is the perturbation metric of the attack proposed by the authors and $\frac{\alpha}{\max Z(A^0)}$ is used to control the proportion of attack performance and perturbation. t is an iteration step of the algorithm. To reduce the number of queries needed to obtain a capable sample, the authors updated the fitness

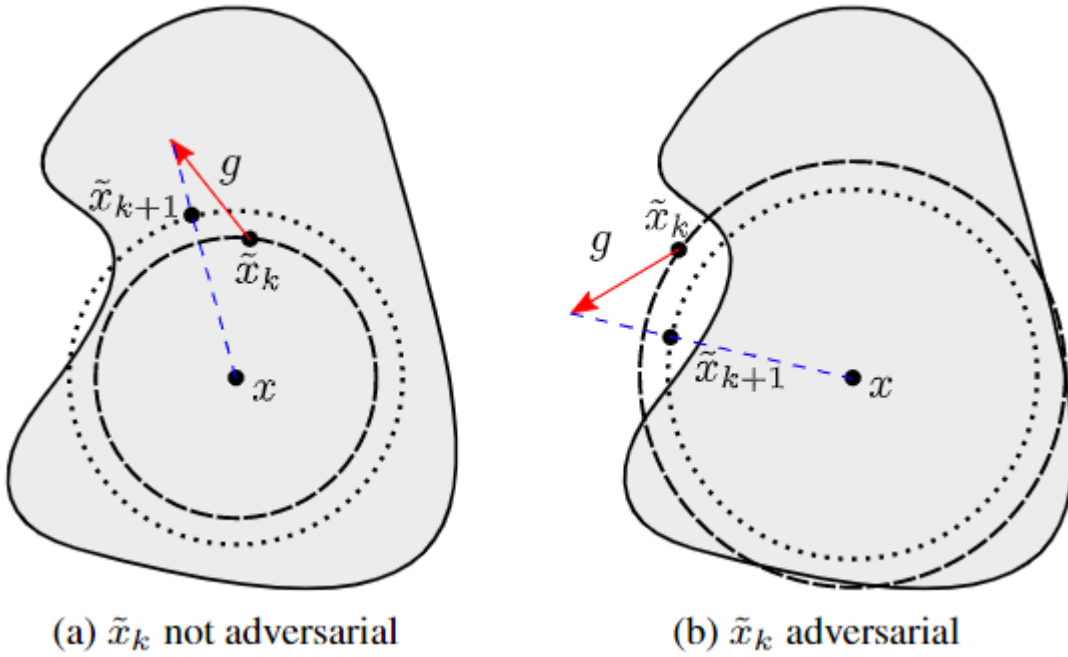


Figure 3.14: The process of finding an adversarial image used by the Decoupled Direction and Norm attack. The adversarial sample is found when the projection of x leaves the original class boundary. (Image credits: [55])

function to take into consideration if the attack is successful or not. If the attack is not successful, there isn't a need to consider the attack perturbation and instead only focus on the attack performance. The updated fitness function is the following:

$$\phi(AS_i^t) = \begin{cases} p(y_1|AS_i^t) - p(y_0|AS_i^t) - \frac{\alpha}{\max Z(A^{t_0})} Z(A_i^t) & \text{if } y_1 \neq y_0 \\ p(y_2|AS_i^t) - p(y_0|AS_i^t), & \text{if } y_1 = y_0 \end{cases} \quad (3.26)$$

where t_0 is the number of iterations when the initial attack is successful and $\frac{\alpha}{\max Z(A^{t_0})} Z(A_i^t)$ controls the perturbation. After evaluating the adversarial sample, if the termination condition is reached the algorithm stops because it has found a suitable adversarial sample. In case the stopping condition was not met, genetic algorithm operators are applied such as selection, crossover and mutation are responsible for evolving the generation of the population. A visual representation of the attack can be seen in figure 3.15.

CornerSearch Attack Adversarial samples while capable of fooling neural network classifiers many are also easily detected to the human eye. Croce et al. [57] propose a new technique capable of generating sparse and imperceivable adversarial samples (observable in figure 3.16). The authors present two methods capable of generating L_0 , $L_0 + L_\infty$, and $L_0 + \sigma$ attacks. The first method is a black-box attack that only needs the logits of the target classifier and the second method is a generalization of the PGD attack. The black-box attack scheme starts by creating one-pixel modifications on the input image. The modifications created depend on the perturbation bound chosen. Despite having different conditions for generating the one-pixel modifications, at the end of this first step there are $(z^{(j)})_{j=1}^M$ images where $M = 8d$ if the image is RGB or $M = 2d$ if the image is grayscale. The generated images are then sorted to identify the pixels that push the decision of the

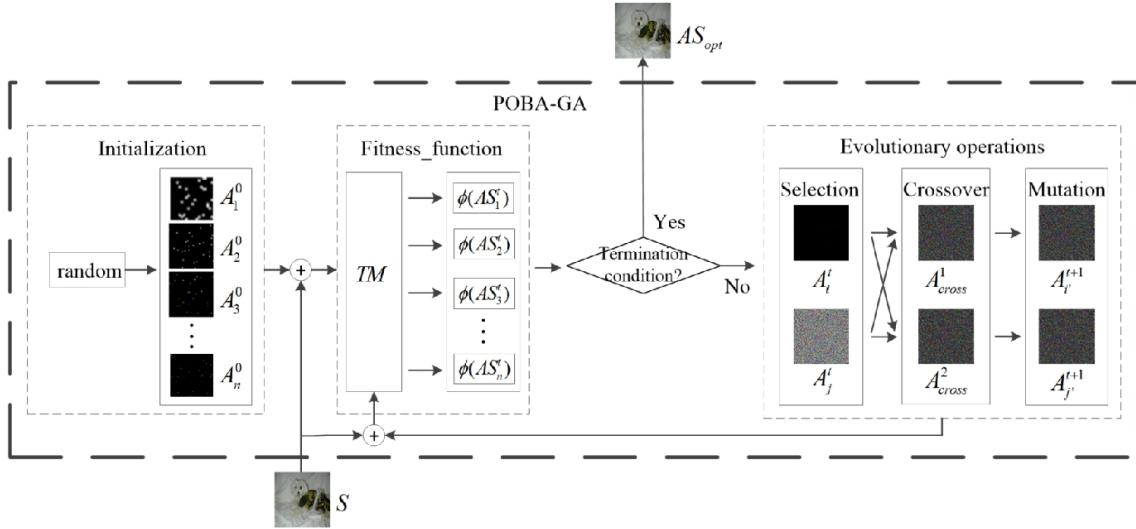


Figure 3.15: The process of generating an adversarial sample using the POBA-GA attack. (Image credits: [56])

classifier towards a specific class or an unspecific one. If any of the changed images alone change the decision of the target classifier the algorithm stops here. The next step of the algorithm focus on combining multiple one-pixel modifications in order to generate an adversarial sample. Since the images generated in the previous step were sorted based on the class that they pushed the classifier, the one-pixel modifications responsible for pushing the image towards a specific class are also sorted. To generate a candidate adversarial image to be classified as a class r , the top N one-pixel modifications that pushed the images towards the class r are chosen. The candidate image y_r is generated by applying all the top N one-pixel modifications. This method is much faster compared to an iterative method because allows feeding all these images in batches to the classifier. Moreover, since it isn't iterative, it doesn't depend on steps that prevent the algorithm from being stuck in local maxima. This method, despite being black-box, obtained similar misclassification rates to white-box attacks and better than black-box attacks while at the same time requiring fewer pixels to be changed.

LogBarrier Attack Most of the white-box attacks presented until now are gradient-based attacks that utilize the loss function of the target model to generate adversarial samples. Finlay et al. [58] propose a new method of generating adversarial examples that, while being gradient-based, doesn't utilize the training loss function of the model and instead, utilizes the logarithmic barrier optimization method. The problem of generating untargeted adversarial samples is written as:

$$\begin{aligned} \min_{\delta} \quad & m(\delta) \\ \text{s.t.} \quad & \arg \max f(x + \delta) \neq c \end{aligned} \quad (3.27)$$

where c is the correct label of an image. In order for an image to be misclassified, the following needs to occur:

$$\max_{\delta} \max_{i \neq c} f_i(x) - f_c(x) > 0 \quad (3.28)$$

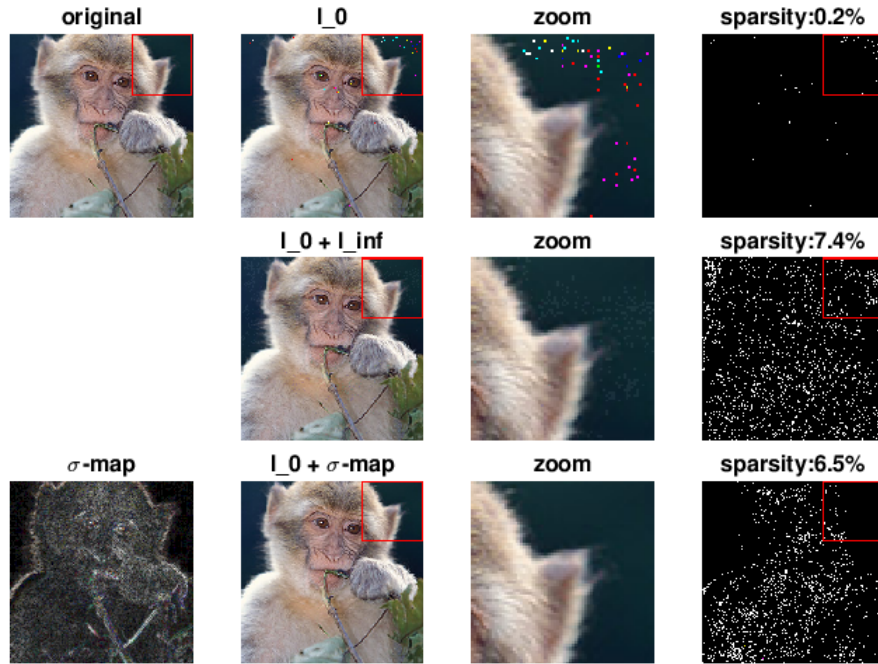


Figure 3.16: Adversarial images generated by the CornerSearch attack. The three variations of the attack produce samples with extremely low sparsity that are almost imperceptible. (Image credits: [57])

where i is an index of the target model’s prediction. The authors use the logarithmic barrier optimization method to find adversarial samples:

$$\min_{\delta} m(\delta) - \lambda \log(f_{max} - f_c) \quad (3.29)$$

where f_{max} is defined as $\max_i f_i(x + \delta)$, f_c is defined as $f_c(x + \delta)$ and λ is a penalty term that enforces the optimization algorithm to search for optimal solutions. The LogBarrier attack algorithm (figure 3.17) needs to start with an image that is misclassified by the target model. A misclassified image can be simply an image of a different class or an image of the original class perturbed using random noise. The following step consists in solving the previously mentioned equation using a fixed λ and gradient descent. If the gradient descent step moves the adversarial image over the bound resulting in a correct classification by the model, a backtracking method is used to revert the image to a previous iteration. Despite having many hyperparameters, the authors claim that the tuning process is relatively quick.

Brendel & Bethge Attack The Brendel & Bethge attack [59] combines the boundary attack with a gradient-based estimation of the boundaries. The attack starts from a point in an adversarial input that is generally far away from a clean image. After finding the adversarial boundary, the point performs an iterative descent in order to minimize the distance to the clean input. Each step is computed by solving a quadratic trust-region optimization problem. Calculating the optimization problem guarantees that: the perturbation has a minimal L_p ($p = 0, 1, 2$, or ∞) distance to the clean input, is placed on the adversarial boundary, the step size is always within a given trust-region radius, and the valid input range. Since the attack follows the adversary boundary, it is less likely to be stuck due to obfuscated gradients. The L_2 attack was shown to be more query

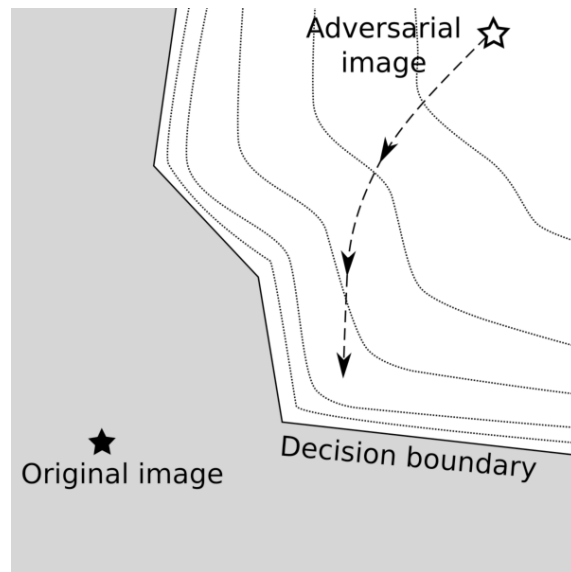


Figure 3.17: The process of finding an adversarial image using the LogBarrier attack. The starting sample is already adversarial and iteratively approaches the decision boundary of the original image class. (Image credits: [58])

efficient than the C&W attack and on par with the DDN attack while at the same time demonstrating to be less sensitive to wrong hyperparameter tuning.

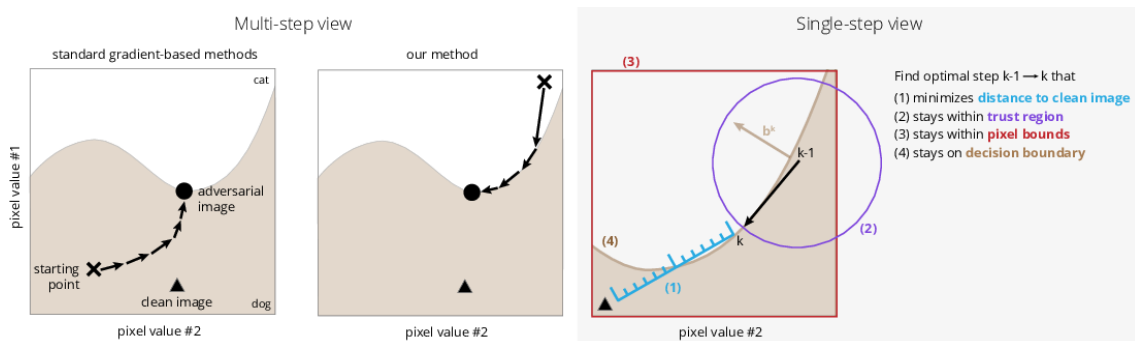


Figure 3.18: Schematic of the Brendel & Bethge approach. Considering an input which a model either interprets as a dog (shaded region) or as a cat (white region). Given a clean dog image (solid triangle), the algorithm searches for the closest image classified as a cat. The attack starts from an adversarial image far away from the clean image and walks along the boundary towards the closest adversarial (middle). In each step, an optimization problem is solved to find the optimal descent direction along the boundary that stays within the valid pixel bounds and the trust region (right). (Image credits: [59])

Shadow Attack Ghiasi et al. [60] propose a new white-box attack capable of fooling certified classifiers which are classifiers that produce a certificate that guarantees that an image is not an adversarial sample. The generated samples (figure 3.19) fool those same classifiers causing them to generate spoofed certificates. The shadow attack solves a different optimization problem than the one used by various other attacks such as the

PGD attack and instead, solves the following problem:

$$\max_{\bar{y} \neq y, \delta} -L(\theta, x + \delta || \bar{y}) - \lambda_c C(\delta) - \lambda_{tv} TV(\delta) - \lambda_s Dissim(\delta) \quad (3.30)$$

where \bar{y} is any possible class different than the true label y , x is the original sample, L is the spoof loss function (the authors use the average cross-entropy loss function of a batch of images perturbed with random noise), λ_c , λ_{tv} , λ_s are scalar penalty weights. $TV(\delta)$, $C(\delta)$ and $Dissim(\delta)$ are penalties that force the perturbations to be small, smooth and without big color changes. This constraints allow big p -norm values while still maintaining the visual changes relatively small. The attack was proven capable of breaking the Randomized Smoothing [76], a defense against L_2 attacks, and CROWN-IBP [77] which is an Interval Bound Propagation method defense against L_∞ attacks.

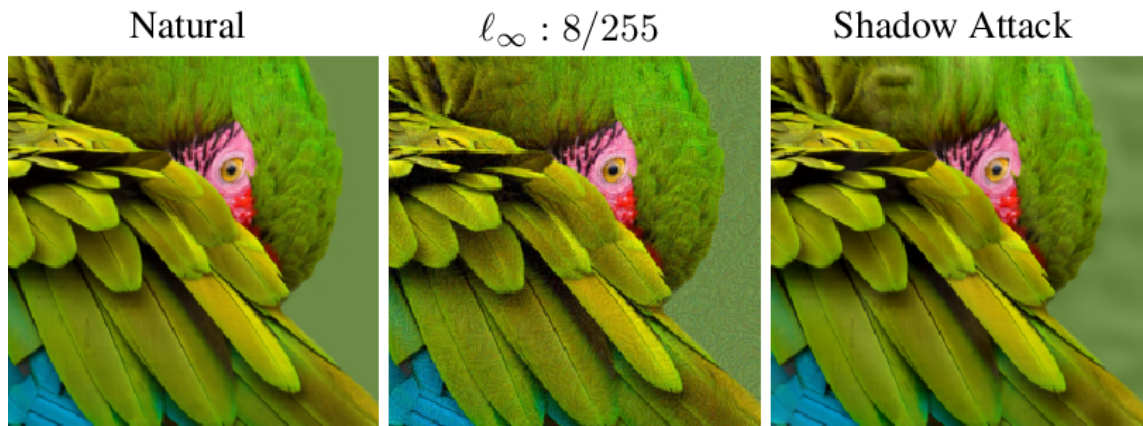


Figure 3.19: A natural image (left), an adversarial image generated by an L_∞ attack (middle), and an adversarial sample generated by the shadow attack (right). The perturbation generated by the shadow attack is large while blending in the image. (Image adapted from: [60])

HopSkipJump Attack The HopSkipJump attack is a decision-based black-box attack that only requires the output labels returned by the target model in order to produce adversarial samples. The authors formulate the decision-based attack as an optimization problem. The attack proposed first starts with an adversarial sample and then iterates the following steps until an adversarial sample has been found. The first step of the algorithm approaches the boundary that separates the original class from the adversarial class, via a modified binary search. After being at the boundary, an estimate of the direction of the gradient is calculated via the Monte Carlo method. The step size is updated along the gradient direction and is decreased until the perturbation is successful. This method was also shown to be more query efficient than the Limited Queries and Info Attack [44] and the Opt-Attack [45] and be more time-efficient than the Boundary attack [38].

GeoDA Rahmati et al. [62] propose a new decision-based black-box attack that takes advantage of the linearization of the decision boundary in deep neural networks in the vicinity of samples. In the proposed scenario, the adversary only has access to the top 1 label that is output by the target classifier within a limited number of queries that can be performed. This attack is based on the property that it is possible to approximately compute a data point x that is close to the decision boundary using a hyperplane that

passes through a boundary point x_b that is close to x with a normal vector w (figure 3.20). The optimization problem of finding an adversarial sample can be formulated as follows:

$$\begin{aligned} \min_v \quad & \mathcal{D}(x, x + v) \\ \text{s.t.} \quad & w^T(x + v) - w^T x_B = 0 \end{aligned} \quad (3.31)$$

where x_b is a point close to the boundary that can be calculated using binary search and w is the estimator. The estimator is computed using geometric priors that are obtained from performing queries to the target model. The experimentation performed on the ImageNet dataset demonstrated that GeoDA outperforms several state-of-the-art black-box attacks such as the HopSkipJump attack and the Boundary attack, requiring fewer queries and iterations than the other methods.

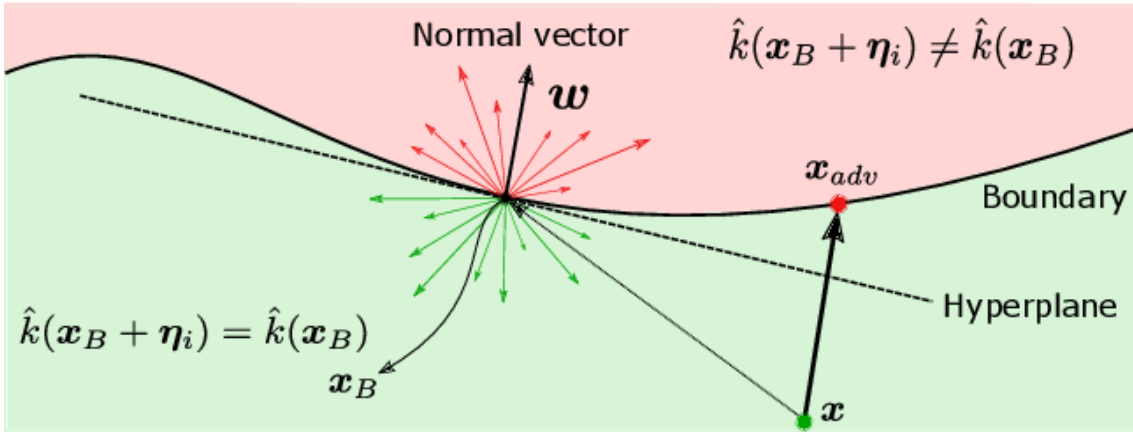


Figure 3.20: Visual representation of GeoDA. (Image credits: [62])

AdvFlow Dolatabadi et al. [63] propose a new method of generating black-box adversarial attacks using flow-based methods and NES to estimate the gradient. Normalizing flows are usually used to create more data samples that follow the data distribution of an original dataset. The authors speculate that adversarial data follows the data distribution of the original clean data and with that, they start by training a flow-based model on clean data distribution. The data that the flow-based model uses doesn't necessarily need to be the training data of the target model. The authors use unseen test data and obtained extremely similar results in both scenarios. Since it is assumed that the distribution of clean and adversarial data is the same, the next step is to change the base distribution from $\mathcal{N}(z, 0|I)$ to $\mathcal{N}(z|\mu, \sigma^2 I)$. Here, μ is a value that is found using NES, σ is a hyperparameter and an adversarial sample is given by:

$$x' = \text{proj}_S(f(z)), z \sim \mathcal{N}(z|\mu, \sigma^2 I) \quad (3.32)$$

where $\text{proj}_S(\cdot)$ projects the adversaries to the set $S(x)$ and $f(\cdot)$ represents the flow-based model. The adversarial samples generated are tested to see if they are misclassified and the μ is updated using NES. A visual representation of the AdvFlow algorithm can be seen in Figure 3.21.

Square Attack The Square attack is a score-based black-box attack that doesn't require any information about the target model such as the gradient. The authors propose one L_∞ -Square attack and a L_2 -Square attack but the algorithm for both attacks is similar.

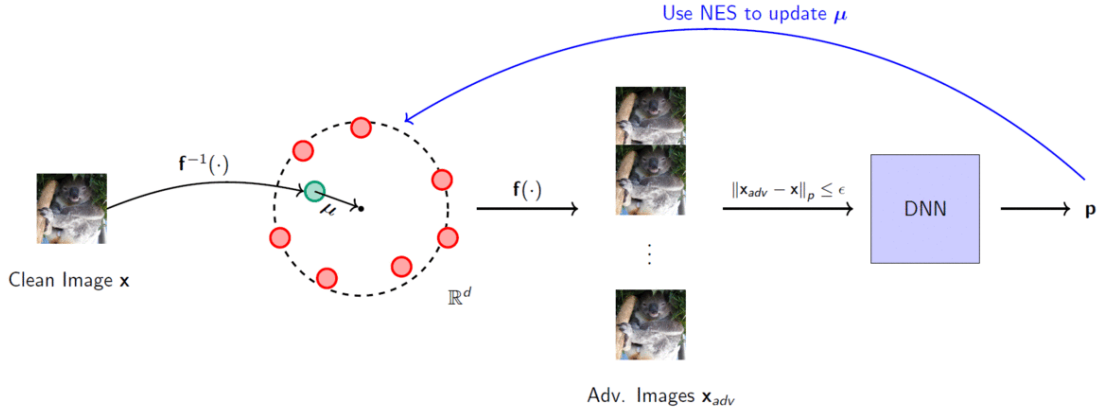


Figure 3.21: Visual representation of AdvFlow. (Image credits: [63])

This method is based on a variation of the random search optimization algorithm. First, the algorithm splits an input image with size $w \cdot w$ into squares with length h where h is the size of the square to be changed and is given by the closest positive integer to $\sqrt{p * w^2}$. The parameter p corresponds to the percentage of elements of the input image to be modified and is the only value that the adversary can choose ($p \in [0, 1]$) which results in less hyperparameter tuning than other algorithms. The algorithm iteratively picks a square of the image and calculates its h to be modified and then generates a perturbation to be applied if it creates a bigger loss than the previous perturbation, otherwise is discarded. The algorithm stops when the sample is adversarial and is capable of fooling the target model.

GreedyFool Attack The authors propose a new method of generating sparse adversarial samples in a white-box scenario. The GreedyFool algorithm consists of two stages. In the first stage, iteratively, is performed a forward-backward pass with modifications made to the pixels of the image. The gradient value of that image is then calculated in order to understand which pixels create a bigger adversary sample (higher gradients mean higher contribution in the resulting adversarial image). The loss function and the gradient value of the adversarial image in iteration t are represented by the following equations:

$$g_t = \nabla_{x_t^{adv}} \mathcal{L}(x_t^{adv}, y, \mathcal{H}) \quad (3.33)$$

$$\mathcal{L}(x, y, \mathcal{H}) = \max(\mathcal{H}(x)_y - \max_{i \neq y} \{\mathcal{H}(x)_i\}, -\kappa) \quad (3.34)$$

where $\mathcal{L}(x_t^{adv}, y, \mathcal{H})$ is the loss function of the adversarial image generated in iteration t , $\nabla_{x_t^{adv}} \mathcal{L}(x_t^{adv}, y, \mathcal{H})$ is the gradient value of the adversarial image generated in iteration t , $\mathcal{L}(x, y, \mathcal{H})$ is the loss function proposed by C&W and κ controls the attack strength (default is 0 but higher values allow for a better transferability of the generated samples). In order to guarantee the similarity between the adversarial sample and the original sample, the authors use a distortion map that stores the distortion level of a pixel. A higher distortion level means that a change in that pixel created a more visible change. In the second stage

of the attack, the pixels that have a higher distortion level and create a small adversarial change in the image are changed to their original values. This stage consists in reducing the number of changed pixels to maintain sparsity. To create the distortion map that is used in the second stage of the algorithm, the authors propose the use of a GAN. The results show that GreedyFool obtains fooling rates higher than 85%, in the CIFAR10 and ImageNet datasets. In comparison to other methods such as JSMA, PGD (with norm L_0), and SparseFool, GreedyFool obtains higher fooling rates while needing to change fewer pixels.

Analysis of the presented adversarial attacks From the attacks presented in this section (see table 3.1), it is clear that during the emergence of adversarial attacks authors focused on white-box attacks, however, recently, more black-box attacks have been proposed. More so, the most recent black-box attacks have a similar attack success rate to white-box attacks. This is extremely concerning because adversaries, even with almost no information about the target model, are capable of fooling, with high success rates, various state-of-the-art classifiers. It is even more worrying because even targeted attacks are extremely successful. Additionally, current adversarial defenses, thought to be powerful against adversarial attacks, still aren't capable of fully defending the models. This presses, even more, the need for tools capable of helping researchers make relevant progress in the adversarial attacks and defenses landscape.

3.1.4 Real-World Applications

ML interest is growing and its use is also helping to accelerate innovations in various fields such as cancer diagnosis and self-driving cars. The broad use of ML systems pushes the need of understanding if ML is secure or if it is vulnerable to attacks. In this section, multiple real-world applications of ML are addressed alongside adversarial samples that were developed to fool those same systems.

Eykholt et al. [78] developed a general white-box attack algorithm Robust Physical Perturbations (RP2) to generate adversarial perturbations that could be used in a real-world case scenario. The focus of the work was on road sign classification. This is extremely relevant for different reasons. Self-driving cars have incorporated an ML algorithm that needs to compute all different objects that are present in the road, including traffic signs. More so, road signs are extremely simple in their design, so it is hard to hide perturbations. Despite being generally simple, an environment such as a road is extremely unconstrained of noise such as the distance and angle from the car to the traffic sign. If an attacker tries to attack a vehicle and isn't able to directly affect the ML system, one way to produce major damage to the vehicle and its occupants could be to modify objects in the real world, such as traffic signs. The algorithm developed by the authors was shown to be capable of generating perturbations on physical objects such as the aforementioned traffic signs. The authors tested the samples on two classifiers: LISA-CNN, a CNN trained on the LISA traffic sign dataset, and a GTSRB-CNN, a CNN trained on the German Traffic Sign Recognition Benchmark. The perturbations created were in the form of stickers and posters. The perturbations and the results can be seen in figure 3.22. In the case of the stop signs, the models classified the sign as being a 45 mph speed limit sign.

Sharif et al. [79] explored the vulnerabilities of facial biometric systems in order to understand if attackers could cause damage in systems used for sensitive purposes like surveillance and access control. In biometric systems, attackers usually have little control over the input of the target system. The only control over the inputs that adversaries have is to












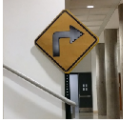



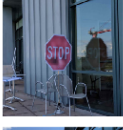

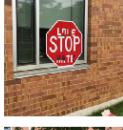



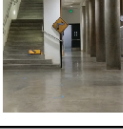

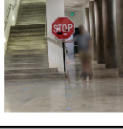
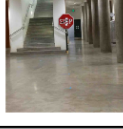
Distance/Angle	Subtle Poster	Subtle Poster Right Turn	Camouflage Graffiti	Camouflage Art (LISA-CNN)	Camouflage Art (GTSRB-CNN)
5' 0°					
5' 15°					
10' 0°					
10' 30°					
40' 0°					
Targeted-Attack Success	100%	73.33%	66.67%	100%	80%

Figure 3.22: Physical adversarial perturbations for traffic signs against LISA-CNN and GTSRB-CNN classifiers (Image credits: [78])

change their physical appearance. For an attacker, changing his physical appearance and still not drawing attention is something that may prove to be difficult.

The authors propose a new class of white-box attacks that consider difficulties that an attacker may face: attacks that are physically realizable and inconspicuous. More so, the attacks were developed to fool the recognition systems in two different ways: impersonation and dodging. The attacks generated were in the form of eyeglasses frames. The attacks were proven to be extremely capable of fooling facial recognition systems, and they can be observed in figure 3.23.

Thys et al. [80] explored the possibility of fooling person detection systems, mainly, surveillance cameras by using adversarial attacks. The authors propose an attack based on adversarial patches. The goal was to create a system that was capable of generating adversarial patches that could be then used to fool the object detector classifier, YOLOv2. The proposed adversarial patch attack was proved to be very effective in fooling the aforementioned classifier. The attack can be observed in figure 3.24. Once again, this proves that ML systems are under threat of adversarial examples and that a security system using ML models might be vulnerable to adversarial patches.

3.2 State-of-the-Art Frameworks

The following section presents the current state-of-the-art libraries for robustness evaluation. They allow users to implement one or various evasion adversarial attacks to evaluate the robustness of computer vision models. In order to implement an attack, the user needs



Figure 3.23: Impersonation and dodging attacks. The top row represents the impersonators using the adversarial glasses and the bottom row represents the attributed classification by the target classifier. (Image credits: [79])

to provide a classifier and a group of benign samples, in this case, images, to be used as input to said attacks. The libraries provide the misclassification rate of the classifier on the adversarial samples generated. In each of the following sections, a more detailed view of each of the current frameworks is performed. The evasion attacks currently supported in each framework are presented in table 3.3. It is important to consider that various attacks implemented by the authors of the frameworks are an adaption of the original attack.

The Adversarial Robustness Toolbox [81] is currently the most used tool for robustness evaluation. ART was designed to work on any version of Python 3 and, at the time, supports TensorFlow 1 and 2, PyTorch, Keras, MXNet, Scikit-learn, XGBoost, LightGBM, CatBoost, and GPy. It is the toolbox with the most diverse number of attacks available for users to experiment as demonstrated in 3.3. It is currently the only library supporting exploratory and poisoning adversarial attacks. This framework also implements 28 state-of-the-art adversarial, resulting in the framework with the most available adversarial attacks, and defenses. More so, there are available metrics of robustness, other than misclassification rate, to evaluate a classifier such as CLEVER and Loss sensitivity of a model. It is possible to view the original sample, the perturbation generated by the attack, and the adversarial sample. As support for users, various examples can be used to get familiarized with the tool and with adversarial attacks in the form of PY files or Jupyter Notebooks. ART is currently in version 1.9.0, released on 18 of December 2021.

Foolbox [82] requires Python 3.6 or newer and natively supports PyTorch, TensorFlow, and JAX. Foolbox currently only has one example of how the library works in the form of a Jupyter Notebook. Evaluating the benign samples' accuracy and the accuracy of the adversarial samples obtained by the model is also possible to compute. Furthermore, users can calculate the plot of robust accuracy over the epsilon (where epsilon is the perturbation of the images) and can visualize the original image, the generated perturbation, and the perturbed image. Foolbox is currently in version 3.3.1 having its latest update on 5 of June 2021.

CleverHans [83] version 4.0.0, currently supports JAX, PyTorch, and TensorFlow 2. The current iteration of CleverHans only supports a few evasion attacks (as seen in 3.3) and doesn't support adversarial defenses. However, in version 3.1.0 more attacks are supported

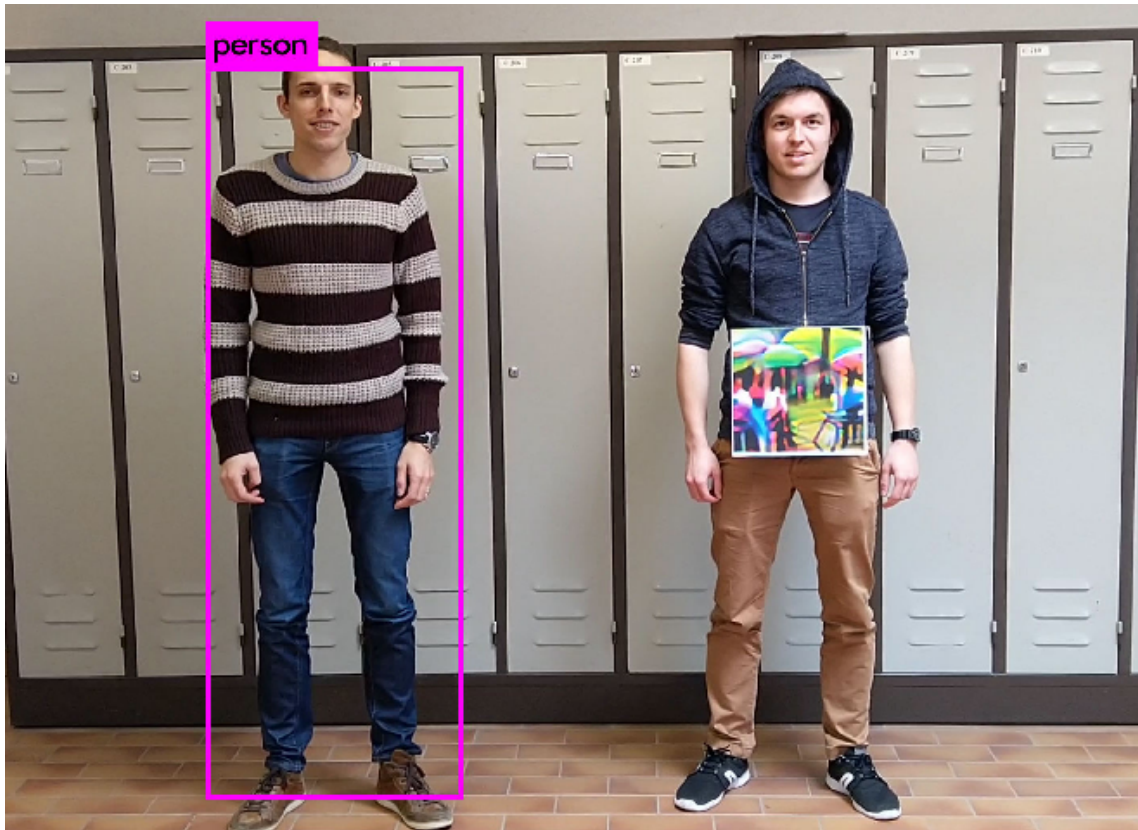


Figure 3.24: The left person is detected by the classifier, however, the right person wearing the patch is ignored. (Image credits: [80])

and examples are available for users. As mentioned before, CleverHans version 4.0.0 is in development and was last updated on 23 September 2021.

AdverTorch [84] currently supports Python 3.6 and PyTorch 1.0.0 & 0.4.1. Just like in other toolboxes, there are some available examples for users to better understand the framework. AdverTorch is currently in version 0.2.4 and was last updated on 30 of July 2021.

AdversarialBox is a Python toolbox that belongs to the AdvBox family developed by Goodman et al. [85] to generate adversarial attacks in TensorFlow, PyTorch, Caffe2, PaddlePaddle, MxNet, and Keras. It currently supports various releases of Python 3. This library supports evasion adversarial attacks and adversarial defenses. It also supports three life real-world attack scenarios: Face Recognition Attack, Stealth T-shirt, and DeepFake Face Detect. The AdvBox family also provides a perceptron to evaluate the robustness of a classifier using other metrics such as the perturbation needed for an image to be misclassified. Various implementations of attacks are also available in Jupyter Notebooks. AdversarialBox was last updated on 26 of August 2020.

DeepRobust [86], at this time, works with Python 3.6 or newer and PyTorch 1.2.0 or newer. This framework supports evasion adversarial attacks and defense methods in the image domain. For the attacks currently implemented in the library, only the MNIST and the CIFAR10 datasets and the CNN, ResNet-18/34, DenseNet, VGG-11/13/16/19 neural networks are supported. This toolbox also supports adversarial attacks and defenses of Graph networks. At present, only the GCN network and Cora, Cora-ml, Citeseer, Polblogs, and Pubmed datasets are supported. A few examples of the implementation of the toolbox

are also available. DeepRobust is on version 0.2.4 and was last updated on 23 of December 2021.

DEEPSEC [87], at this moment, supports PyTorch 0.4. This library employs both adversarial evasion attacks and adversarial defenses. This library also makes available utility metrics of both attacks and defenses. Similar to other frameworks, there are examples ready to use for users who may want to delve deeper into adversarial learning. There are a few metrics to analyze the robustness of a model that is also available.

Carlini [88] critiqued the DEEPSEC library and pointed out several flaws with the implementation attacks and defenses and the experimentation results shown in [87]. From the various problems identified, the two most relevant are: *The permitted distortion (ϵ) is too large to be meaningful.* and *Multiple attacks are implemented incorrectly.* This occurrence results in possible incorrect evaluations of the robustness of a model. DEEPSEC was last updated on 21 of May 2019.

The main features supported by each framework are exhibited in table 3.2 and the supported attacks in table 3.3. From the analysis of each framework, it is evident that the most supported machine learning frameworks are TensorFlow and PyTorch. ART and Foolbox are the frameworks that contain the most extensive number of adversarial attacks, which could explain their popularity. DEEPSEC was demonstrated to have incorrectly implemented attacks and hasn't received support in the latest years. Alongside DEEPSEC, AdversarialBox, CleverHans, and AdverTorch have received a small scale of updates in the last months. In addition, many of the presented toolboxes don't have the most recent and powerful adversarial attacks that are accessible in the literature. Finally, the information that researchers can obtain from benchmark tests is insufficient; most of the toolboxes only present the misclassification rate of the adversarial samples. This is one of the main objectives with the framework developed in this dissertation, to provide a robustness metric to allow users to compare their models. A deep dive on the framework developed and its features is made on Chapter 5.

Table 3.2: Functionalities supported in the current state-of-the-art frameworks

Feature	ART	AdverTorch	Foolbox	CleverHans	AdversarialBox	DeepRobust	DEEPSEC
Evasion Adversarial Attacks	✓	✓	✓	✓	✓	✓	✓
Poisoning Adversarial Attacks	✓						
Exploratory Adversarial Attacks	✓						
Adversarial Defenses	✓				✓	✓	✓
TensorFlow	✓		✓	✓	✓		
PyTorch	✓	✓	✓	✓	✓	✓	✓
Keras	✓						
Scikit-learn	✓						
XGBoost	✓						
MXNet	✓				✓		
LightGBM	✓						
CatBoost	✓						
GPy	✓						
Continued on next page							

Table 3.2 – continued from previous page

Feature	ART	AdverTorch	Foolbox	CleverHans	AdversarialBox	DeepRobust	DEEPSEC
JAX			✓	✓			
Caffe2					✓		
PaddlePaddle					✓		
Examples Ready-to-Use	✓	✓	✓		✓	✓	
Graph Adversarial Attacks						✓	
Graph Adversarial Defenses						✓	
Robustness metrics	✓		✓		✓		✓
End of Table 3.2							

Table 3.3: Evasion Adversarial Attacks supported in the current state-of-the-art frameworks

Attack	ART	AdverTorch	Foolbox	CleverHans	AdversarialBox	DeepRobust	DEEPSEC
Additive Gaussian Noise			✓				
Additive Uniform Noise			✓				
AdversarialPatch [89]	✓						
Auto Attack [90]	✓						
Auto-PGD [90]	✓						
BPDA [39]		✓				✓	
BIM [29]	✓	✓	✓	✓	✓		✓
Binarization Refinement			✓				
Binary Search Contrast Reduction			✓				
Boundary [38]	✓		✓				
Brendel & Bethge [59]	✓		✓				
C&W [31]	✓	✓	✓	✓	✓	✓	✓
Clipping Aware Additive Gaussian Noise [91]			✓				
Clipping Aware Additive Uniform Noise [91]			✓				
Clipping Aware Repeated Additive Gaussian Noise [91]			✓				
Clipping Aware Repeated Additive Uniform Noise [91]			✓				
Contrast Reduction Attack			✓				
DDN [55]		✓	✓				
Decision Tree Attack [92]	✓						
DeepFool [28]	✓	✓	✓		✓	✓	✓
Dpatch [93]	✓						
EAD [41]	✓	✓	✓				✓
Fast Adaptive Boundary [94]		✓					
Fast Gradient Method [6]	✓	✓	✓	✓	✓	✓	✓
Feature Adversaries [95]	✓	✓					
Continued on next page							

Table 3.3 – continued from previous page

Attack	ART	AdverTorch	Foolbox	CleverHans	AdversarialBox	DeepRobust	DEEPSEC
Gaussian Blur			✓				
GeoDA [62]	✓						
HCLU [47]	✓						
HopSkipJump [61]	✓			✓			
Inversion Attack [96]			✓				
ILLCM [29]					✓		✓
JSMA [26]	✓	✓			✓		✓
L-BFGS [5]		✓			✓	✓	✓
LLCM [29]							✓
Linear Search Blended Uniform Noise			✓				
Linear Search Contrast Reduction			✓				
Local Search Attack [97]					✓		
MI-FGSM [43]				✓	✓		✓
NATTACK [52]						✓	
NewtonFool [98]	✓		✓				
One-Pixel/Few-Pixel	✓				✓	✓	
Opt-Margin [99]							✓
PGD [40]	✓	✓	✓	✓		✓	✓
R+FGSM [35]							✓
R+LLC [35]							✓
Repeated Additive Gaussian Noise			✓				
Repeated Additive Uniform Noise			✓				
RobustDPatch [100]	✓						
Salt And Pepper Noise			✓				
Shadow [60]	✓						
ShapeShifter [101]	✓						
SimBA [53]	✓						
Single Pixel Attack [97]		✓					
Sparse L1 Descent (SLIDE) [51]		✓		✓			
Spatial Transformations Attack [102]	✓						
Spatially Transformed Attack [36]		✓					
SPSA [46]		✓		✓			
Square [64]	✓						
Threshold [54]	✓						
UAP [33]	✓					✓	✓
Virtual Adversarial Method [103]	✓		✓				
Wasserstein [25]	✓						
YOPO Attack [104]						✓	
ZOO [34]	✓						

End of Table 3.3

This page is intentionally left blank.

Chapter 4

Development Plan

This chapter presents the planning of the work for the first and the second semester of this dissertation. The estimated time-interval for every task is represented in a Gantt chart. The methodology proposed in the first semester that was to guide the development during the second semester is also presented.

4.1 Work Plan

The work plan followed in the first semester, as well as the plan for the second semester, are represented in figure 4.1. The proposed time frame for each task is a prevision and therefore, some small shifts in some of the tasks may occur without risking the final delivery.

Selection of attacks to implement Supported by the work done in the first semester, complemented with additional research to be performed in the second semester, the most relevant adversarial attacks will be chosen. The work performed in this step is crucial to make sure that the framework has a complete and robust set of attacks.

Development of the framework The process concerned with the initial architecture and implementation of the framework. It encompasses the integration of the attacks attacks chosen and the functionalities that allow the users to perform a benchmark and obtain relevant robustness and statistical information about the model and the attacks performed.

Development of a robustness metric This task is concerned with the development of a metric that allows researchers to better understand the level of robustness of a model. For instance, given two models that are benchmarked, exclusively comparing the misclassification value of each model isn't sufficient to understand the robustness of the models. This task aims to solve that problem.

Testing the framework The process of evaluating the implemented framework. This step encompasses evaluating the reliability and stability of the functionalities implemented.

Comparison analysis with current state-of-the-art frameworks Comparing the developed framework with current frameworks and evaluating the differences and the bene-

fits that the developed framework in addition to the support that it may give to researchers.

Scientific dissemination In order to present the framework to the researchers working in the adversarial attacks panorama, it is expected to write a scientific paper with the findings and the advances made.

Write the dissertation This process is performed over the second semester and reports the discoveries and the evolving process of the dissertation.

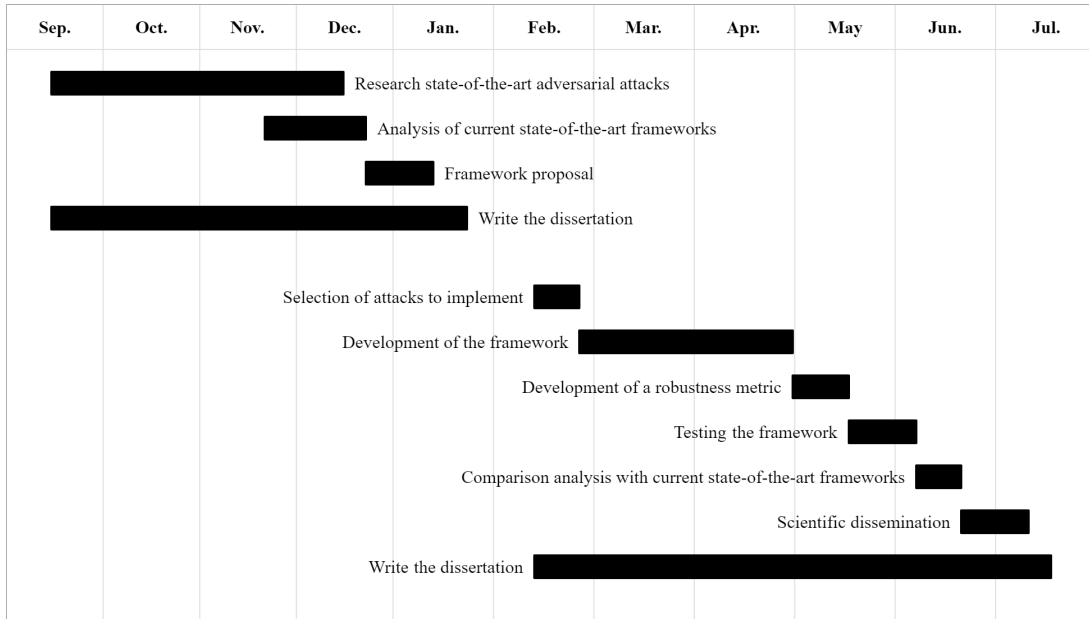


Figure 4.1: Gantt chart of the development plan

4.2 Methodology

In the first semester, the proposed method of developing the framework was the Waterfall model. This model assumes that the requirements were well defined which was the case, observable in the next section, As mentioned in section 5.1, small changes to the requirements may need to be made over the next semester. However, these changes will be appropriately considered and evaluated to make sure that they don't put at risk the success of the final product.

Since the requirements of the framework were well defined in the first semester, the Waterfall model was the chosen to represent the development phase during the second semester. The Waterfall model represents the development as a group of separate phase in which a new phase can begin once the previous one has been completed. Following the requirements specification, the next phase is the design of the tool, which is followed by the implementation, then verification and finally the deployment of the developed tool.

This page is intentionally left blank.

Chapter 5

The Framework

This chapter presents the proposed and developed framework. Firstly, it puts forward the software requirements specification that was made as well as the risks associated with the framework development. Next, it presents the executed work plan of the second semester and the architecture and implementation of the proposed framework and its main features, the pipeline mechanism and the model robustness metric.

5.1 Software Requirements Specification

As mentioned in previous chapters, there is a need to develop better adversarial attacks and defenses that push forward the security of Machine Learning (ML) models. In order to help developers test their defenses and attacks, the current state-of-the-art frameworks, mentioned in the previous chapter were developed. However, some of those frameworks suffer from a lack of adversarial attacks, metrics to evaluate the models' robustness, and metrics to evaluate the attack benchmark.

This dissertation aimed to develop a framework that provides the most relevant attacks for researchers to test their models. In the first phase of this dissertation, a preemptive analysis of the software requirements was performed which can be seen in a MoSCoW diagram, observable in figure 5.1. It is important to refer that some changes in the requirements were expected to change throughout the framework development.

The software requirements projected in the first semester were the main guide for the development that was performed in the second semester. In order to claim that the final project was a success, the Must Have requirements of the MoSCoW diagram, shown in section 5.1, need to be satisfied in the current iteration of the framework. Every Must Have requirement was achieved; the current framework supports, however, only one machine learning framework for Python 3, PyTorch. The first initial weeks of the semester were used to evaluate current adversarial attacks in the literature as well as their availability and implementation. From the research that was performed, most of the attacks were implemented in either TensorFlow 1, TensorFlow 2, or PyTorch. Since the implementation of adversarial attacks isn't the main objective of this dissertation, and it is of most importance to guarantee that the attacks are correctly implemented, the only attacks that were going to be taken into consideration were the ones with code provided by the original authors. Incorporating and working with attacks from two very different frameworks would increase exponentially the scope of this dissertation therefore, after analyzing the attacks implemented in TensorFlow, it was decided to focus on attacks implemented in PyTorch. Despite



Figure 5.1: MoSCoW diagram representing the software requirements of the framework

various recognized attacks being implemented in TensorFlow, various changes would need to be made to each implementation which is out of scope for this dissertation. With this in consideration, the current framework supports a single machine learning framework for Python 3, and PyTorch and supports 8 different adversarial attacks implemented in the same language. All of the other software requirements were also achieved, however, the implementation specifications will be addressed in further sections which will detail the implementation of main feature of the framework - the pipeline mechanism as well as the newly proposed metric to evaluate the robustness of a model.

5.2 Risk Analysis

Since the implementation of the adversarial attacks wasn't the main focus of this dissertation, the main code needed to be sourced the authors of those same attacks which needed to be peer-reviewed and accepted. There are multiple risks associated with this course of action. Motivated by this, a careful risk assessment was performed in order to identify and establish the actions of all the possible risks that could happen during the development. The figure 5.2 represents the main risks identified as well as the corresponding likelihood, impact and appropriate course of action to deal with them.

RISK CONDITION	RISK CONSEQUENCE	LIKELIHOOD	IMPACT	ACTION	RESPONSE
The code of the adversarial attack could be unavailable	The attack could be impossible to implement	IMPROBABLE	EXTREME	Avoid	The selection of the attacks needs to consider the availability of the source code
The code of the adversarial attacks could be implemented in a different machine learning framework	The code needs to be converted into the languages supported by the framework	PROBABLE	EXTREME	Mitigate	Convert the code into the appropriate languages, but in some cases the attack may be discarded
The execution time of a benchmark are longer than current state-of-the-art frameworks	Users may prefer other frameworks	POSSIBLE	HIGH	Mitigate	Guarantee a correct implementation of the attacks
Incorrect implementation of the attacks	The adversarial robustness benchmarks results are incorrect	POSSIBLE	HIGH	Mitigate	Compare results of the attack with current state-of-the-arte frameworks to guarantee the implementation is correct
Insufficient computational resources to process an user model	The benchmark will not be possible	IMPROBABLE	EXTREME	Accept	This risk is accepted because there is no mitigation possible
The user chooses all the attacks available	The benchmark can be extremely slow	POSSIBLE	EXTREME	Accept	The user will need to wait or select fewer attacks.

Figure 5.2: Risk assessment table

The risk assessment table developed in the first semester, seen in 5.2, proved to be a valuable tool since many of the risks were encountered during the development phase of the framework. As mentioned before, the current framework supports only PyTorch which conveys that only attacks developed using this framework are supported. This indicates that strong and reliable attacks in the literature that are implemented in other machine learning languages are not currently implemented in the current state of the framework. The risks associated with unavailable code or code in different machine learning languages had an extreme impact and in both cases, these were avoided. The risk associated with the execution time of the framework being higher than other frameworks was also avoided. The current iteration of the framework offers a pipeline mechanism that is unique in the current literature as well as a model robustness metric that is also unique which allows for discarding this risk. The pipeline mechanism allows users to choose the models and the adversarial attacks at their leisure which could result in a failed benchmark if the users' computational resources are insufficient. The associated risks are accepted since each user needs to execute a benchmark that considers the available computational resources.

5.3 Executed Work Plan

The development work plan deviated from the original work plan. The initial selection of attacks to implement took more time than expected for the reason that the attacks were implemented in different machine learning frameworks and many authors don't provide the original code for their attacks which lead to a deeper analysis of which attacks were possible to implement. Contrary to the original plan made in the first semester, the development of the framework didn't follow a strict Waterfall methodology since it was necessary to test the features along the implementation leading to a more *ad-hoc* methodology. This was due to the various changes that each adversarial attack needed to have since the implementation of each attack was very different from the other. The development of the robustness metric also took more time than expected since the metric formula changed multiple times until reaching its current state. The various changes during the framework development as well as the various changes to the robustness metric delayed the scientific paper writing and the writing of this dissertation. The Gantt chart of the executed work plan can be seen in figure 5.3.

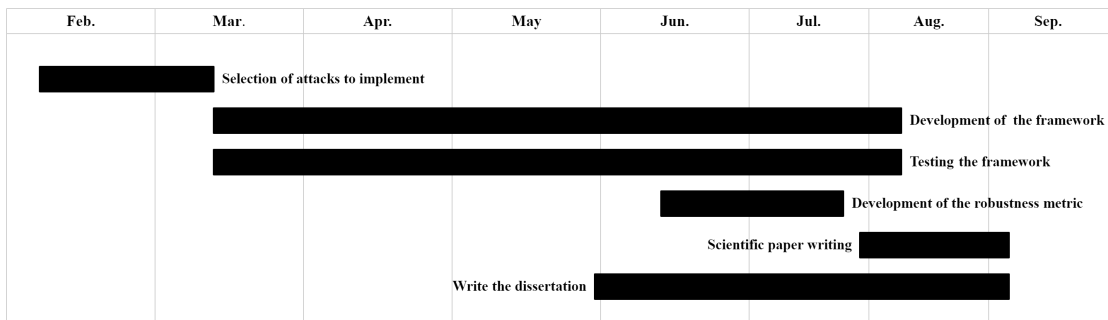


Figure 5.3: 2nd Semester Work Plan

5.4 Architecture and Implementation

The first step in the development phase was planning the architecture of the framework. The file structure of the framework must be set up in such a way that allows users to easily understand and access the necessary files for their needs. The framework file structure is divided into two main folders, the *engine* folder, and the *examples* folder. The latter includes various examples so that users can get familiar with the workings of the framework. This includes examples for each of the attacks currently implemented, as well as examples using the pipeline mechanism. The engine folder contains the crucial files of the framework. This folder contains two main folders, the *attacks* folder, and the *core* folder. The *attacks* directory has additional folders that correspond to attacks that are fully supported or that are still in development. As mentioned previously, the adversarial attacks' original code was obtained from repositories that were made available by its' authors. However, the original implementation of several attacks was incorporated in a very specific example, using a specific model and a specific dataset or images that were used by the authors. It is imperative that users of this framework can use various models trained using different datasets and images, which resulted in the need to change the attack's original implementation to generalize its possible use cases. The first step was to create an abstract attack class that serves as the blueprint for all the other attacks. With this class created, various changes were made to various attacks currently in the framework. This resulted in each *attack* folder having a folder *lib* and various additional files, the primary one being the file with the *proxy* prefix followed by the name of the attack. The *core* folder contains various files that are needed to execute functions of the framework or complementary functions. This is where the pipeline mechanism function is also located. The figure 5.4 represents the folder structure of the current iteration of the framework.

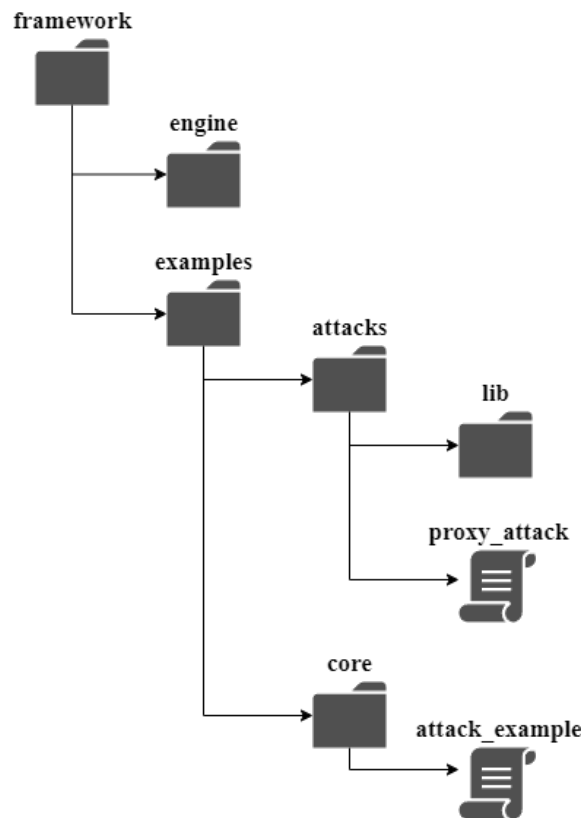


Figure 5.4: Framework file structure

As mentioned previously, the main focus of this dissertation is on the pipeline mechanism and the robustness metric, however, for the framework to work it needs to contain various adversarial attacks as these are a required factor in evaluating the robustness of a model. During the phase of selection of attacks to implement, the papers previously mentioned in the section 3.1.3 were reviewed in order to gather information about the original code implementation of the authors. A table with the collected information can be seen in table 5.1. It is important to reiterate that attacks implemented by the original author were put front to implement in the framework, this occurs because it is extremely important to guarantee that the framework only supports attacks that are not peer-reviewed because otherwise, incorrect attacks could lead to erroneous or incomplete evaluations of the robustness of evaluated models. From the table 5.1, it is clear that there are a very large number of attacks in the literature that don't have their original implementation currently available, and those that do, have it under old versions of TensorFlow 1, PyTorch, or NumPy. There are 17 attacks implemented in TensorFlow 1 with only 12 being implemented by the original author, 0 implemented in TensorFlow 2, 23 implemented in PyTorch with only 12 made available by the original author, and 6 in NumPy with 3 implemented by the original author. Choosing PyTorch as the machine learning tool to use was also supported by these numbers; having the same number of original implementations of both TensorFlow 1 and PyTorch pushes the latter tool as the preferred option for various reasons. Mainly, implementations in older versions, which is the case for most of the attacks, are easily updated due to a lower complexity when compared to TensorFlow. PyTorch is also gaining a lot of popularity which pushes forward the number of possible attacks to be implemented in the future under the same language which is the opposite of TensorFlow 1. Since TensorFlow has updated into TensorFlow 2, TensorFlow 1 has lost popularity and since there are no original implementations in TF2, supporting this tool would not be feasible. From the table, the attacks supported in PyTorch made available by the original authors are AdvFlow, Bandits & Prior, DDN, DeepFool, GeoDA, GreedyFool, LogBarrier, Opt-Attack, Shadow, SimBA, Square, and Wasserstein. The attacks implemented using NumPy are also possible to adapt to PyTorch which results in three possible attacks to support, CornerSearch, Threshold & Few-pixel, and UAP. From these attacks, the framework currently supports Bandits & Prior, DDN, DeepFool, GeoDA, LogBarrier, SimBA, Square, and UAP.

Table 5.1: List of attacks analyzed during the selection phase of the framework development

Attack	Original author?	TF1	TF2	PyTorch	NumPy
AdvFlow	Yes			X	
AdvGAN	No	X		X	
ATNs	No	X		X	
AutoZOOM	Yes	X			
Bandits & Prior	Yes			X	
BIM (I-FGSM)	No			X	
Boundary Attack	No				X
BPDA	Yes	X			
Brendel & Bethge	Yes				
C&W	Yes	X			
CornerSearch	Yes				X
DDN	Yes			X	
DeepFool	Yes			X	
EAD	Yes	X			
FGSM	No			X	

Continued on next page

Table 5.1 – continued from previous page

Attack	Original author?	TF1	TF2	PyTorch	NumPy
GeoDA	Yes			X	
GreedyFool	Yes			X	
HCLU	No				
HopSkipJump	Yes				
Hot/Cold	No				
ILLCM	No			X	
JSMA	No				X
L-BFGS	No	X		X	
Limited Queries and Info	Yes	X			
LogBarrier	Yes			X	
MI-FGSM	Yes	X			
NATTACK	Yes	X			
One-Pixel	No			X	X
Opt-Attack	Yes			X	
PGD	Yes	X			
POBA-GA	No			X	
R+FGSM	No			X	
Shadow	Yes			X	
SimBA	Yes			X	
Sparse L1 Descent (SLIDE)	Yes	X			
SPSA	Yes	X			
Square	Yes	X		X	
stAdv	No	X			
Substitute Model	No			X	
Threshold & Few-pixel	Yes				X
UAP	Yes				X
UPSET & ANGRI	No				
Wasserstein	Yes			X	
ZOO	Yes	X			
End of Table 5.1					

Subsequent to implementing the adversarial attacks, the next phase was the development of the pipeline mechanism, the main feature of the toolbox. It was important to develop a pipeline that could receive various models, images, and attacks and could output relevant statistical information about the procedure. The pipeline needs to receive several arguments that are crucial to its operation, as seen in figure ???. The selection of attacks to execute is performed using a JSON config file that users can customize by selecting the preferred attacks as well as the additional hyperparameters that an attacker needs to receive to work, such as the one represented in figure 5.6. This allows users to test in the same execution various attacks with different parameters which can then be shared with other users resulting in much-improved portability. An example of the formatting of this file is given under the examples folder of the framework. The pipeline needs other input arguments such as a Python dictionary with the PyTorch models to test, a list with the images to use and their corresponding ground truth labels, the batch size to use, the pixel size of each image, the mean value, and the standard deviation of the images or the dataset, the number of classes that an image can belong to, the number of channels of the images (for example, RGB would be 3 channels) and a PyTorch dataset loader with the training images of the models (this is necessary because various white-box attacks may need this information to generate adversarial images).

```

pipeline = Pipeline()
pipeline.execute(config, models, images.to(device), labels.to(device),
                image_size, batch_size, trainloader, device=device,
                dataset_name=dataset_name, mean=CIFAR_MEAN, std=CIFAR_STD,
                num_classes=10)

```

Figure 5.5: A call to the pipeline class. The execute function has several inputs that are needed for the execution to start.

After receiving the inputs, the pipeline is now ready to be executed. The execution iterates throughout all the attacks by each model; at the beginning of the iteration, each model evaluates all of the input images without any perturbations to obtain a list of the classified labels by the model. This is an important step to evaluate the initial accuracy of the model without any perturbation being added to the images. After that, each attack chosen in the *config* file will be executed and the output information is stored in a Python dictionary, this includes the generated images by the attack and its corresponding labels obtained by the model, the number of total queries that the attack used, the L2 norms of each image, and the success of the attack (this corresponds to the percentage of generated images that were misclassified by the model which is essentially the number of adversarial images created). All of the information associated with every attack executed and every model is saved in another Python dictionary which is then saved as a PyTorch file (with *.pt* extension) so that users can access all of the information relative to the execution. This file is saved under a folder *outputs* with the name of the folder referring to the date of the execution and the dataset that was used. Under this folder, all of the original images used are also stored. Every model input to the pipeline also has a folder with the images generated from every attack that was also chosen by the user. A *CSV* file with information relative to each image is also generated which includes the ground truth label of the image, the label attributed by the classifier of the original image, the label of the adversarial image attributed by the classifier, the L2 value of the image and the number of queries that took to generate that image. The number of adversarial images correctly classified by the model and the number of original images without perturbation correctly classified by the model are also stored under the same *data.csv* file. If the user chooses the option to generate plots, at the end of the execution, plots with the images generated by each attack will also be available to further analyze the experimentation performed, observable in figure 5.7. At the final step, the pipeline calculates the Model Robustness Score of each model and the maximum possible score achievable. This is the model robustness metric that was one of the focal points of this dissertation and will be explained in the next section.

In addition to the developed functions of the framework such as the adversarial attacks and the pipeline mechanism, the architecture of the current iteration can be represented by the figure 5.8. The *Attack* class is the abstract class that all other attacks implemented use as a blueprint, such as the Bandits & Prior attack that is represented as the class *BanditsPrior*. The *Pipeline* class is also represented and incorporates all the adversarial attacks currently supported, however, in order to abbreviate the UML diagram, only the Bandits & Prior attack class is shown. In addition to this, the framework depends, not only, on the implementation of the original attacks but also on external libraries such as

```
{
  "attacks": [
    {
      "name": "BanditsPrior",
      "run": "True",
      "max_queries": 1000,
      "fd_eta": 0.01,
      "image_lr": 0.5,
      "online_lr": 0.1,
      "mode": "l2",
      "exploration": 0.01,
      "tile_size": 50,
      "epsilon": 5.0,
      "log_progress": "False",
      "nes": "False",
      "tiling": "False",
      "gradient_iters": 1
    },
    {
      "name": "DDN",
      "run": "True",
      "steps": 1000,
      "targeted": "False",
      "gamma": 0.05,
      "init_norm": 1.0,
      "quantize": "True",
      "levels": 256
    }
  ]
}
```

Figure 5.6: An example of a config file. In this example, the Bandits & Prior and the DDN attacks are going to be executed in the pipeline.

PyTorch, Torchvision, NumPy, and Pillow.

5.5 Model Robustness Score

Developing a metric to evaluate classifiers was one of the main objectives of this dissertation, which meant that the final formula needed to be thought out with extreme care to be able to give relevant information about the robustness of classifiers. This phase of the development took longer than expected due to changes that were made to the model robustness score formula. Before developing the formula, the factors related to the adversarial robustness were evaluated such as the number of correctly classified adversarial images by a model, the perturbation in each adversarial image, that corresponds to the L2 of the adversarial image, and the number of non-perturbed images that were correctly classified by the model, essentially this corresponds to the accuracy of the model on the original images. To evaluate the robustness of a model, all of the previous factors have an important role. First of all, if a model, trained under an image classification dataset, such as the CIFAR-10, incorrectly classifies non-perturbed images that belong to that

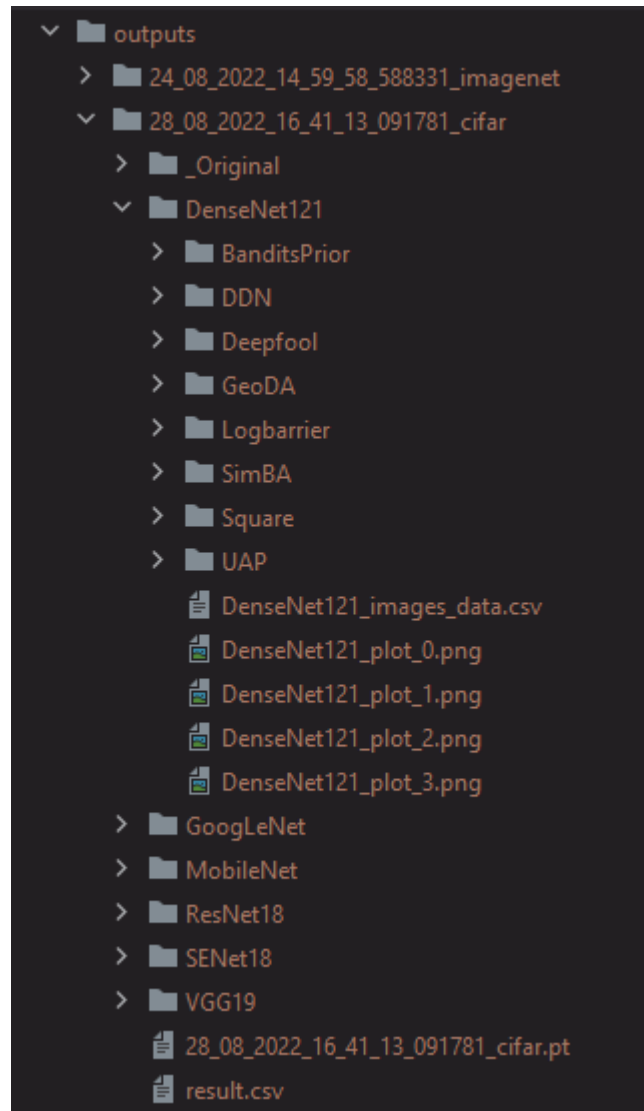


Figure 5.7: The *outputs* folder with examples of executions using the pipeline mechanism of the framework

dataset, the robustness score of that model needs to be lower than a model that classifies the same images correctly. Secondly, each attack generates adversarial images with different amounts of perturbation depending on the model. This symbolizes that between adversarial images generated for two models, the more robust model is the one that its adversarial images generated with the most amount of perturbation, resulting in higher L2 scores. On the other hand, if the adversarial images generated for a model have lower L2 scores, with almost non-visible perturbations, the model is less robust because images with small amounts of perturbation easily change their accuracy. Finally, if even a high amount of perturbation in an image isn't sufficient to change the classification attributed by the model to that image, that implies that the model is unaffected by the perturbation which needs to be represented in the model robustness score.

With the previous factors in mind, the robustness score of a model is given by the following formula which is a maximization formula, meaning that the objective of a model is to obtain the maximum value possible:

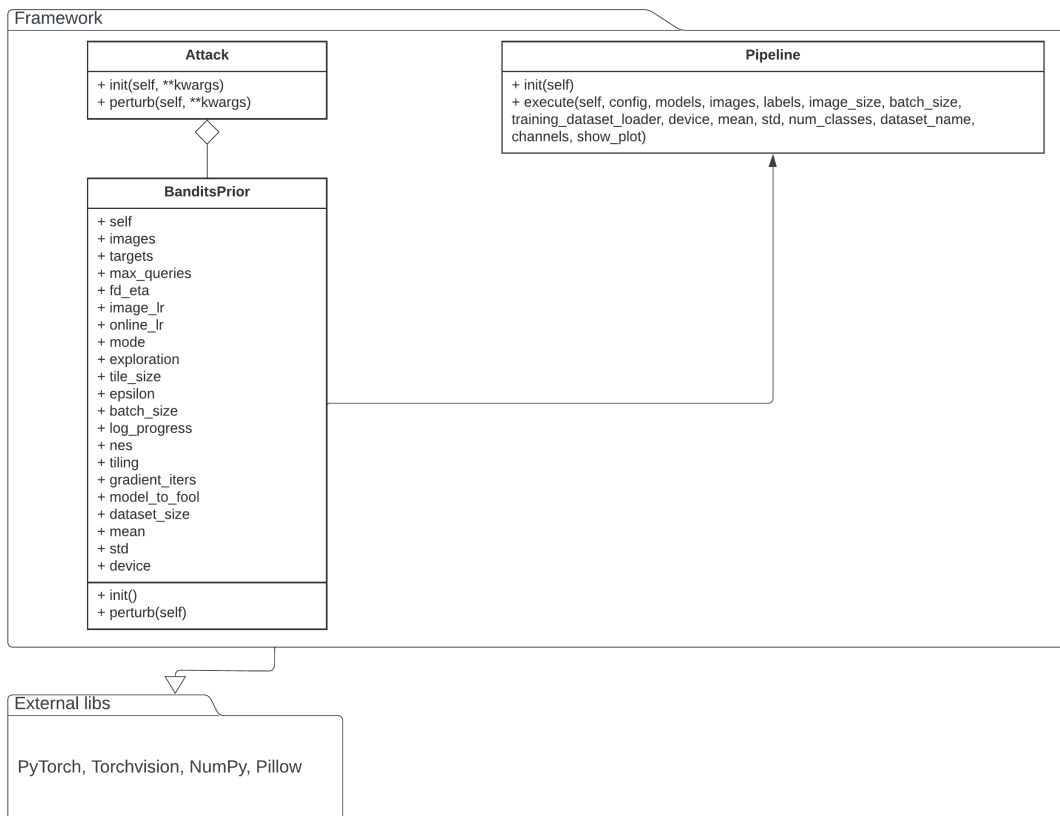


Figure 5.8: UML diagram representing the main classes of the framework and the external libraries

$$MRS = \sum_{i=1}^N \sum_{j=1}^M label * (1 - (\frac{1}{1+L2} * 0.5 + 0.5 * misclassified)) \quad (5.1)$$

The calculation of the robustness of a model takes into consideration all the attacks (represented by the first sum) that were used and the adversarial images that were generated by them (represented by the second sum of the formula). The factor $\frac{1}{1+L2}$ represents the strength of the perturbed image, since the L2 value represents the amount of perturbation added, a robust adversarial image has a very low L2 value (close to 0) and is misclassified by the model, which is represented by the factor *misclassified*. If the L2 value is, hypothetically, 0 and the perturbed image is misclassified by the model then the sum of these two factors is 1, which is then subtracted to the constant 1 that is represented in the formula. This would mean that for this sample of a specific attack the model behaved very poorly. However, if the L2 was large and still the model classified the sample correctly, that would mean that the value would be very close to 0 which after the subtraction factor would result in a value very close to 1 which leads to the belief the model was capable of correctly classifying the perturbed image. The initial *label* factor is assigned to 1 if the classified label of the original image by the model is the same as the ground truth label of the original image, otherwise, the value is assigned to 0.

The developed formula evaluates the robustness of a model based on its performance against several adversarial images generated by a set of selected adversarial attacks. This

score allows users to understand more clearly the robustness of their models based on the number of correctly classified adversarial images, the perturbation that each image needed to fool the model, and the number of correctly classified non-perturbed images. The aim is that users, not only use the accuracy of the model on non-perturbed samples but also use this score to evaluate its capabilities and to push forward the development of models more robust to adversarial attacks.

This page is intentionally left blank.

Chapter 6

Experimentation

Over the development of the framework, various tests were conducted to assess its capabilities. To evaluate its final state and in order to get relevant and meaningful conclusions that can be taken from robustness evaluations, a final test was conducted with all the supported adversarial attacks and various machine learning models trained with an image classification dataset. The following sections will present the complete settings that were used and the results and conclusions of the same experiment.

6.1 Settings

As mentioned before, in this experiment all of the supported attacks by the framework were used in the test which includes the following attacks: Bandits & Prior, DDN, Deepfool, GeoDA, Logbarrier, SimBA, Square Attack, and UAP. The models used, DenseNet121, GoogLeNet, MobileNet, ResNet18, SENet18, and VGG19, belong to a GitHub repository by kuangliu¹. The original accuracy of the models can be seen in the table 6.1. The settings used for each attack can be viewed in table 6.2. The first 20 images of the image classification dataset CIFAR-10², seen in figure 6.1, were used in the experiment as the input images to generate adversarial images using the aforementioned models and adversarial attacks.

Table 6.1: Test accuracy of models used in the experimentation

Model	Accuracy
DenseNet121	95.04%
GoogLeNet	95.65%
MobileNet	91.20%
ResNet18	93.02%
SENet18	95.06%
VGG19	93.83%
End of Table 6.1	

¹<https://github.com/kuangliu/pytorch-cifar>

²<https://www.cs.toronto.edu/~kriz/cifar.html>

Figure 6.1: CIFAR-10 samples used in the experimentation



Table 6.2: Settings of the attacks used in the experimentation

Attack	Hyperparameters
Bandits & Prior	max_queries: 1000 fd_eta: 0.01 image_lr: 0.5 online_lr: 0.1 mode: "l2" exploration: 0.01 tile_size: 50 epsilon: 5.0 log_progress: "False" nes: "False" tiling: "False" gradient_iters: 1
DDN	steps: 1000 targeted: "False" gamma": 0.05 init_norm: 1.0 quantize: "True" levels: 256
DeepFool	overshoot: 0.02 max_iter: 50
GeoDA	dist: "l1" tol: 0.0001 sigma: 0.0002 mu: 0.6 delta: 255 search_space: "sub" sub_dim: 75 Q_max: 5000
LogBarrier	norm: 2 lower_bound: 0 upper_bound: 1 dt: 0.01 alpha: 0.1 beta: 0.75 gamma: 0.5 max_outer: 15 tol: 1e-6 max_inner: 100 T: 500
SimBA	freq_dims: 14 stride: 7 epsilon: 0.2 l1_bound: 0.0 order: "rand" targeted: "False" log_every: 10 num_iters: 0
Continued on next page	

Table 6.2 – continued from previous page

Attack	Hyperparameters
	pixel_attack: "True"
UAP	overshoot: 0.02 num_classes: 10 xi: 10 delta: 0.2 max_iter_df: 10 max_iter_uni: -1 norm: "inf"
SquareAttack	eps: 5 n_iters: 1000 p_init: 0.05 targeted: "False" constraint: "linf"
End of Table 6.2	

6.2 Results and Discussion

From the experimentation that was performed, it is important to observe the images that were generated, which of them were capable of fooling the models, and the model robustness score attributed to each model. As mentioned previously, all of the attacks currently available in the framework were used during the test and 20 images from the CIFAR-10 dataset were used as the test images. This results in a maximum possible score of 160 for each model which would stipulate that no attack was capable of generating an image that fooled the model. From the scores obtained, visible in table 6.3, the most robust model was the ResNet18, which obtained a score of 46.91, and was capable of correctly classifying 24 adversarial images generated. The least robust classifier was the MobileNet with a score of 42.40 with 27 adversarial images correctly classified. In order to understand why a model that was capable of classifying correctly more adversarial images had an inferior robustness score than a model that classified correctly less adversarial images, in this case, 3 images, it is important to analyze the L2 scores of the adversarial images generated as well as the images themselves.

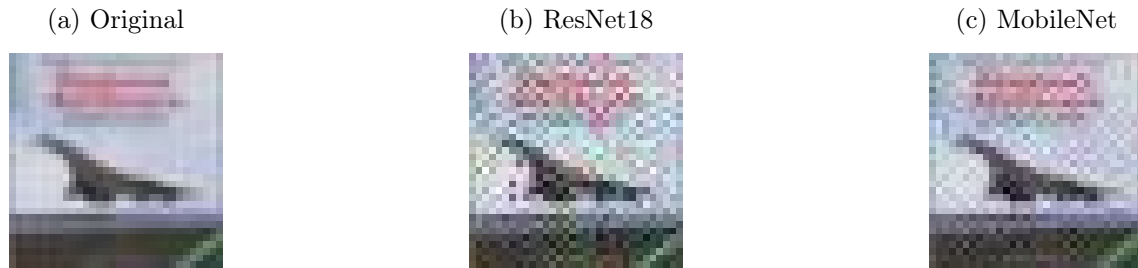
It is important to allude, however, that a classifier with higher robustness score classified correctly a less number of adversarial images than a classifier with lower robustness score since, the L2 value of each adversarial image is a crucial factor in the calculation of the robustness score of a model. Since the L2 value of each adversarial image has a very big weight on the score of the model, there are two possible scenarios that could explain this event. On one hand, there were images with small L2 values, which are images with very small perturbations, capable of fooling the MobileNet classifier. On another hand, the adversarial images generated for the ResNet18, despite having higher values of L2 which would indicate larger and more visible perturbations, were not capable of fooling the classifier or the adversarial images that fooled the classifier had very high L2 values. The combination of these two factors is a possible explanation for why a model with a higher Model Robustness Score classified correctly a lesser number of adversarial images than a model with a lower Model Robustness Score. In order to understand if this is the case and what is the cause of this result, it is important to understand the images that were generated.

Model	Model Robustness Score	Number of adversarial images correctly classified
ResNet18	46.91	24
GoogLeNet	44.50	24
DenseNet121	46.22	24
MobileNet	42.40	27
SENet18	44.80	18
VGG19	45.06	21

Table 6.3: Model Robustness Values

Initially, it is important to look at the generated images from each attack for both models. The starting point will be the adversarial images generated using samples 1 to 5 represented in the figures 6.3 and 6.4 for the ResNet18 and MobileNet respectively, and the corresponding L2 values. The first image (top row) generated from each attack has always a higher value of L2, for the ResNet18 model than the images generated for the MobileNet. This indicates that the perturbation needed to fool the ResNet18 was higher than the perturbation needed to fool the MobileNet. Even though the number of queries used for each attack isn't utilized in the formula to calculate the robustness of a model, the results also indicate that for this image, the attacks Bandits & Prior, GeoDA, SimBA, and Square needed more queries in order to generate an adversarial image. Similar to both models, the attacks DDN and UAP were incapable of generating an adversarial sample. The second image demonstrates interesting results since the DDN attack wasn't capable of generating an adversarial sample capable of fooling the ResNet18 but every attack generated an adversarial image capable of generating a sample for the MobileNet. However, the generated images, for the MobileNet, from the Bandits Prior, DeepFool, GeoDA, and LogBarrier have higher values than the ones generated for the ResNet18. The Bandits Prior attack was also incapable of generating an adversarial image capable of fooling both of the networks. Similar to what happened for the first image, the L2 values for all of the images, generated from the attacks were higher for the ResNet18 than for the MobileNet except for the Square attack that generated images with equal L2 values for both networks. For the fourth image, the DDN attack didn't generate an image that was capable of fooling the ResNet18; the same happened for the Bandits Prior and the MobileNet. The other attacks generated once again higher L2 adversarial images for the ResNet18 with the exception being the Square Attack that similar to what had happened previously generated images with equal perturbations for both networks. The Bandits Prior didn't generate adversarial images from the fifth image for both of the networks, however, the UAP attack also wasn't capable of generating for the MobileNet. Once again, the rest of the attacks generated higher perturbation images for the ResNet18. From these images, it is clear that the images generated from the attacks for the ResNet18 have higher levels of perturbation (L2 is higher) than the ones generated for the MobileNet. This is clear when comparing the generated adversarial images for both models, observable in figure 6.2. In the figure, the original sample and the generated samples by the attack UAP for the ResNet18 and MobileNet can be seen, and it is clear that there is a more significant perturbation in the image generated for the ResNet18, with an L2 value of 4.34, than for the MobileNet, with an L2 value of 2.60. The remaining generated images need to follow this pattern, in other words, many of the generated adversarial images for the ResNet18 have higher amounts of perturbation when compared to the adversarial images generated for the MobileNet. The L2 values of all the images can be seen in the table 6.4 and the generated images from each attack for those two models can be seen in the appendix . The results indicate that, as expected, the L2 values of the adversarial images have an extremely important weight in evaluating the robustness of the models using the developed metric.

Figure 6.2: Comparison between the original sample 4 (a), the adversarial image generated by the UAP attack for the ResNet18 (b), and the adversarial image generated by UAP for the MobileNet (c)



This is extremely relevant because it shifts the focus from only considering the accuracy of models on adversarial samples and focusing on the quantity of perturbation that is needed to fool a model. This work pushes forward the need to evaluate models using various factors such as the L2 value of the samples allowing for a more multifaceted evaluation of those same models.

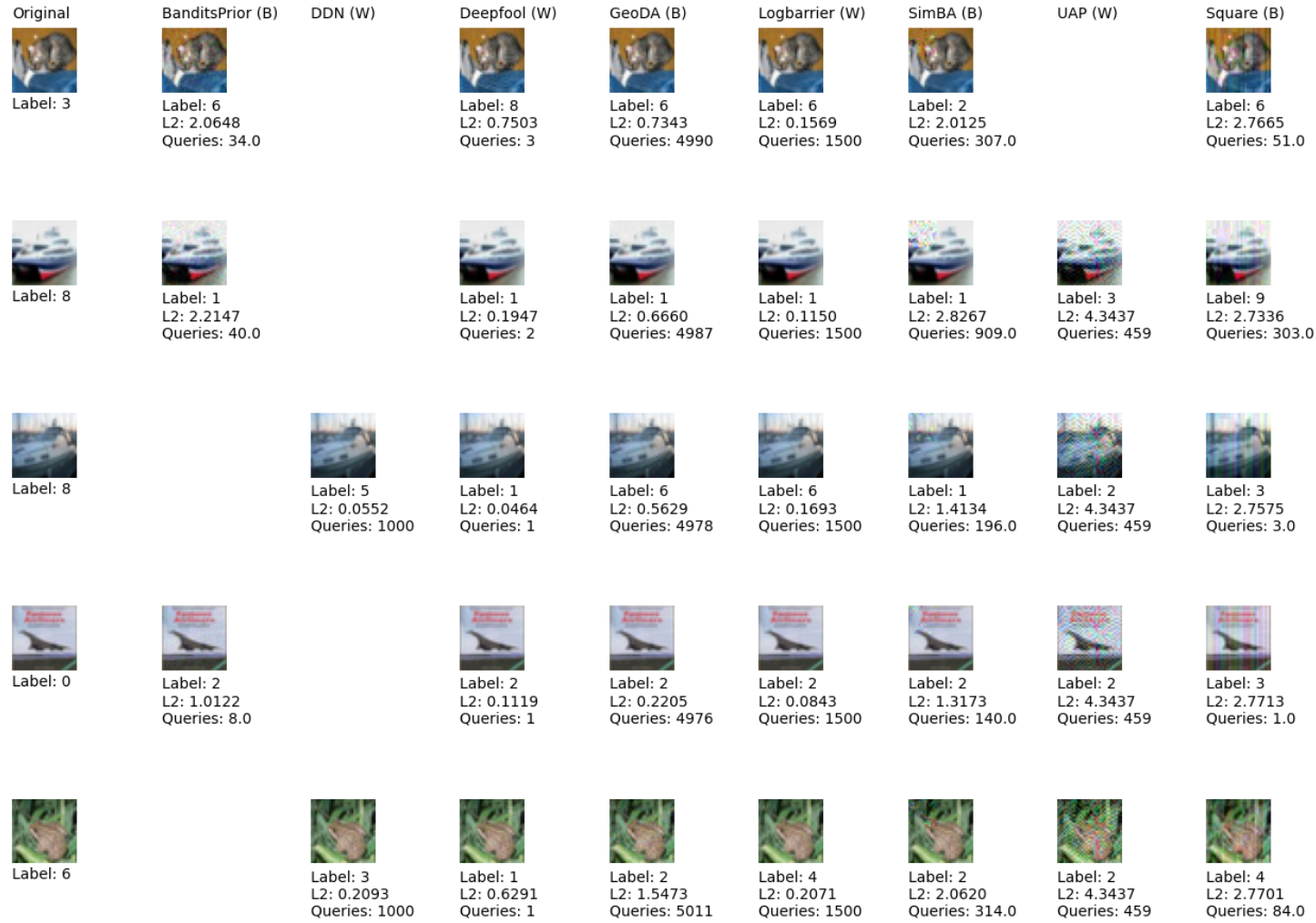


Figure 6.3: ResNet18 generated images (1-5)

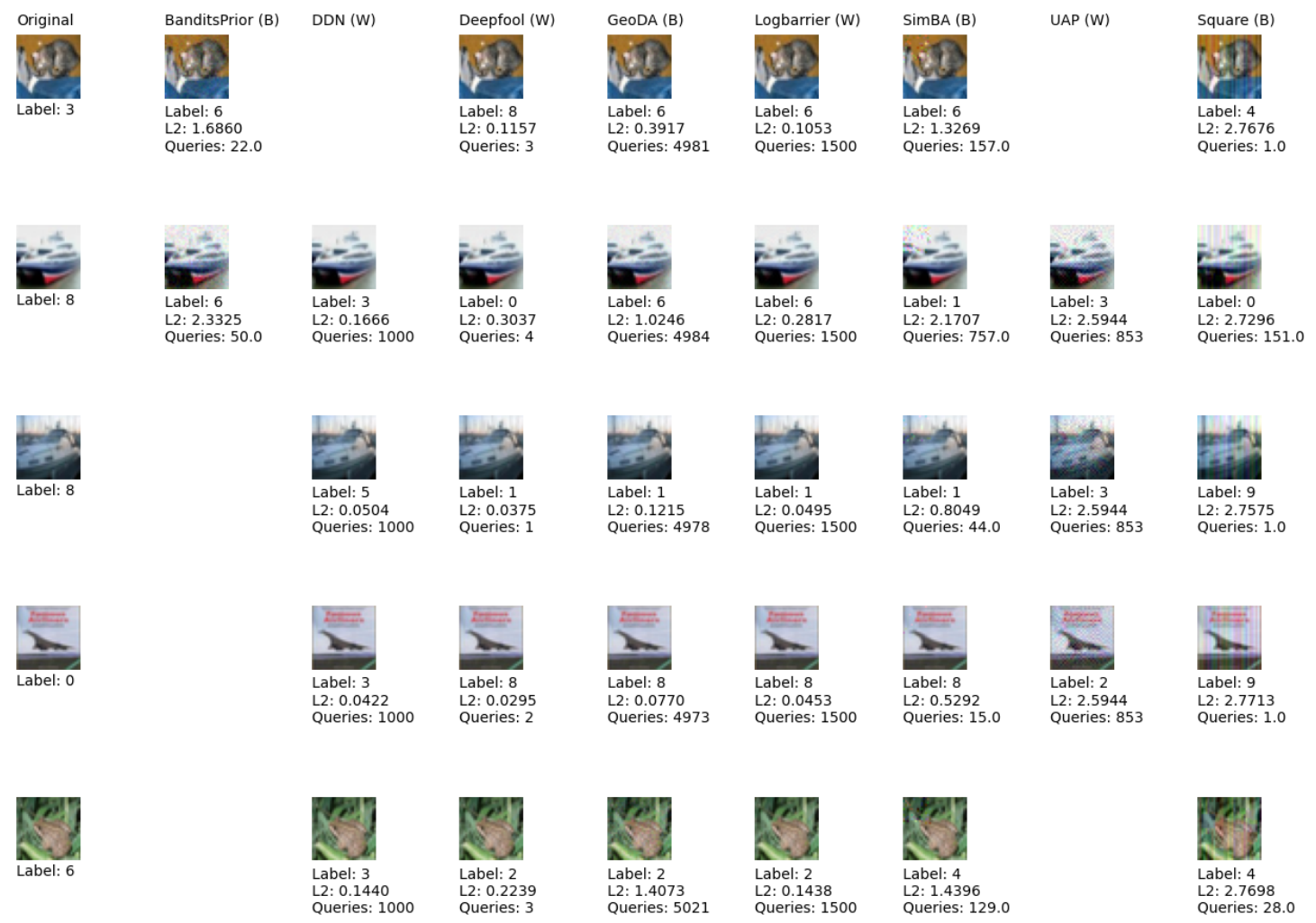


Figure 6.4: MobileNet generated images (1-5)

Table 6.4: L2 scores of the adversarial images generated for the ResNet18 and the MobileNet

Sample	BP		DDN		DF		GeoDA		LB		SimBA		UAP		Square	
	R	M	R	M	R	M	R	M	R	M	R	M	R	M	R	M
1	2.06	1.69	x	x	0.75	0.12	0.73	0.39	0.16	0.11	2.01	1.33	x	x	2.77	2.77
2	2.21	2.33	x	0.17	0.19	0.30	0.67	1.02	0.12	0.28	2.83	2.17	4.34	2.59	2.73	2.73
3	x	x	0.06	0.05	0.05	0.04	0.56	0.12	0.17	0.05	1.41	0.80	4.34	2.59	2.76	2.76
4	1.01	x	x	0.04	0.12	0.03	0.22	0.08	0.08	0.05	1.32	0.53	4.34	2.59	2.77	2.77
5	x	x	0.21	0.14	0.63	0.22	1.55	1.41	0.21	0.14	2.06	1.44	4.34	x	2.77	2.77
6	x	x	0.15	0.06	0.46	0.06	1.41	1.07	0.18	0.07	2.62	1.10	4.34	2.59	2.77	2.77
7	2.14	x	x	0.09	0.31	0.08	0.65	0.26	0.16	0.09	1.96	x	4.34	x	2.73	2.72
8	x	x	0.05	0.08	0.05	0.09	0.35	0.48	0.08	0.07	0.72	1.00	4.34	2.59	2.76	2.76
9	x	1.23	0.08	x	0.39	0.11	0.52	0.36	0.15	0.11	1.97	1.07	x	x	2.77	2.77
10	x	x	0.08	0.09	0.12	0.10	0.21	0.55	0.11	0.10	1.21	1.05	4.34	2.59	2.76	2.76
11	x	x	0.12	0.08	0.38	0.07	0.47	0.20	0.12	0.08	2.40	1.73	4.34	2.59	2.77	2.77
12	3.77	x	x	0.15	1.44	0.37	1.91	1.00	0.39	0.27	x	2.53	4.34	x	2.72	2.73
13	1.01	1.66	x	x	0.09	0.09	0.19	0.34	0.07	0.12	0.10	1.25	4.34	2.59	2.76	2.76
14	3.39	1.98	x	0.19	0.96	0.32	1.23	0.75	0.33	0.19	x	2.19	4.34	2.59	2.68	2.69
15	x	x	0.28	0.13	0.80	0.16	1.19	0.51	0.33	0.26	3.61	1.56	4.34	x	2.75	2.75
16	x	x	0.04	0.07	0.03	0.06	0.08	0.19	0.05	0.10	0.80	1.20	4.34	x	2.77	2.77
17	2.51	1.47	x	x	0.47	0.13	0.73	0.33	0.23	0.01	1.59	0.94	4.34	2.59	2.70	2.71
18	x	x	0.13	0.01	0.38	0.12	0.79	0.35	0.18	0.10	2.02	1.40	4.34	2.59	2.77	2.77
19	2.38	x	x	0.16	0.72	0.23	1.09	0.63	0.26	0.17	1.76	2.81	4.34	2.59	2.66	2.67
20	5.00	x	x	0.19	1.25	0.37	2.27	2.65	0.27	0.24	2.53	2.30	4.34	x	2.77	2.77
End of Table 6.4																

BP: Bandits & Prior

DF: DeepFool

LB: Logbarrier

R: ResNet18

M: MobileNet

x: Attack was incapable of generating an adversarial image capable of fooling the model

This page is intentionally left blank.

Chapter 7

Conclusion and Future Work

Current machine learning models are not secure; adversaries have a wide range of tools that can use to put at risk classifiers that are being used in everyday lives. Adversarial samples are small perturbations added to inputs that are capable of fooling even state-of-the-art deep learning models.

The first stage of this dissertation focused on researching adversarial attacks, which are methods for generating adversarial samples, specifically, in the image classification landscape. In recent years, the adversarial attacks panorama has expanded which resulted in authors developing various types of adversarial attacks such as poisoning attacks, exploratory attacks, and evasion attacks. Since many adversarial attacks have surged and the importance of developing safe and robust models has been growing, various authors have developed frameworks to group various attacks to help researchers make advancements in the adversarial learning panorama. Despite some frameworks being equipped with many adversarial attacks, some of them have also been abandoned but most important, none of them offer a way of comparing the robustness levels of different machine learning models.

Motivated by this, a new framework was developed in the second iteration of this dissertation. In the first semester, a work plan was put forward in addition to the software requirements in a form of a MoSCoW diagram. Despite the original work plan not being followed due to various drawbacks, the Must Have software requirements were fulfilled resulting in a successful final product. Currently supporting eight different adversarial attacks, the framework, developed in Python and PyTorch, presents a pipeline mechanism that allows users to easily test their models against the supported attacks. From the execution of the pipeline, the users have access to statistical information relative to the models and to the adversarial images generated from each attack. In order to evaluate the robustness level of the models, a new metric was proposed, that takes into consideration all of the images generated from each attack and attributes a model robustness score based on the capabilities of the model to correctly classify the perturbed images.

With the intent of testing the developed framework and its capabilities, experimentation was performed. By utilizing the new pipeline mechanism, various PyTorch models were trained using an image classification dataset, the CIFAR-10 dataset, and the eight supported attacks the experimentation was executed. From the results, the most robust model was the ResNet18, achieving a Model Robustness Score of 46.91, with the least being the MobileNet only achieving a score of 42.40. Further analysis of the results, the ResNet18 only correctly classified 24 adversarial images while the lesser robust MobileNet classified correctly 27 adversarial images. This leads to an interesting conclusion, that evaluating the robustness of a model shouldn't be based only on the accuracy of the model but should

also take into consideration the perturbation of each adversarial sample generated. The work performed allowed us to understand more clearly the impact that adversarial images have on the models and allows users to easily perform robustness benchmarks on their models.

The next steps would focus on incorporating more attacks and supporting more machine learning frameworks such as TensorFlow 1 and TensorFlow 2 and developing the pipeline mechanism to support those same machine learning tools. The intent is also to grow a community around this framework which would allow for authors with peer-reviewed adversarial attacks to incorporate them into the framework. Relative to the robustness metric developed, there are other factors that should be looked into in the future such as the number of queries that an attack took in order to generate an adversarial sample. Giving weight to the number of queries that an attack took, in the model robustness score formula, is something that should be considered in the future.

References

- [1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [2] Alireza Nooraiepour and Tolga M. Duman. Randomized convolutional codes for the wiretap channel. *IEEE Transactions on Communications*, 65(8):3442–3452, 2017.
- [3] M.A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.
- [4] Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way, 2018.
- [5] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.
- [6] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.
- [7] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S. Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres, 2018.
- [8] Thomas Tanay and Lewis D. Griffin. A boundary tilting perspective on the phenomenon of adversarial examples. *ArXiv*, abs/1608.07690, 2016.
- [9] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS’18*, page 5019–5031, Red Hook, NY, USA, 2018. Curran Associates Inc.
- [10] Sebastien Bubeck, Yin Tat Lee, Eric Price, and Ilya Razenshteyn. Adversarial examples from computational constraints. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 831–840. PMLR, 09–15 Jun 2019.
- [11] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. *Adversarial Examples Are Not Bugs, They Are Features*. Curran Associates Inc., Red Hook, NY, USA, 2019.
- [12] Robust ml. <https://www.robust-ml.org>.
- [13] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P. Wellman. Sok: Security and privacy in machine learning. In *2018 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 399–414, 2018.

- [14] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the science of security and privacy in machine learning, 2016.
- [15] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. *ArXiv*, abs/1801.10578, 2018.
- [16] Yinpeng Dong, Qi-An Fu, Xiao Yang, Tianyu Pang, Hang Su, Zihao Xiao, and Jun Zhu. Benchmarking adversarial robustness on image classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 318–328, 2020.
- [17] Battista Biggio, Blaine Nelson, and Pavel Laskov. Support vector machines under adversarial label noise. In Chun-Nan Hsu and Wee Sun Lee, editors, *Proceedings of the Asian Conference on Machine Learning*, volume 20 of *Proceedings of Machine Learning Research*, pages 97–112, South Garden Hotels and Resorts, Taoyuan, Taiwan, 14–15 Nov 2011. PMLR.
- [18] Marius Kloft and Pavel Laskov. Online anomaly detection under adversarial impact. In Yee Whye Teh and Mike Titterton, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 405–412, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.
- [19] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Conference on Machine Learning*, ICML’12, page 1467–1474, Madison, WI, USA, 2012. Omnipress.
- [20] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning, 2017.
- [21] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdoor attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.
- [22] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models, 2017.
- [23] Matt Fredrikson, Eric Lantz, Somesh Jha, Simon M Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. *Proceedings of the ... USENIX Security Symposium. UNIX Security Symposium*, 2014:17–32, 2014.
- [24] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS ’15, page 1322–1333, New York, NY, USA, 2015. Association for Computing Machinery.
- [25] Eric Wong, Frank Schmidt, and Zico Kolter. Wasserstein adversarial examples via projected Sinkhorn iterations. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6808–6817. PMLR, 09–15 Jun 2019.

-
- [26] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 372–387, 2016.
- [27] Andras Rozsa, Ethan M. Rudd, and Terrance E. Boult. Adversarial diversity and hard positive generation, 2016.
- [28] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, 2016.
- [29] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *ICLR Workshop*, 2017.
- [30] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, ASIA CCS '17*, page 506–519, New York, NY, USA, 2017. Association for Computing Machinery.
- [31] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, 2017.
- [32] Sayantan Sarkar, Ankan Bansal, Upal Mahbub, and Rama Chellappa. Upset and angri : Breaking high performance image classifiers, 2017.
- [33] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 86–94, 2017.
- [34] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. *ZOO: Zeroth Order Optimization Based Black-Box Attacks to Deep Neural Networks without Training Substitute Models*, page 15–26. Association for Computing Machinery, New York, NY, USA, 2017.
- [35] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *ArXiv*, abs/1705.07204, 2018.
- [36] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples, 2018.
- [37] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI'18*, page 3905–3911. AAAI Press, 2018.
- [38] Wieland Brendel *, Jonas Rauber *, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations*, 2018.
- [39] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, 2018.

- [40] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.
- [41] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: Elastic-net attacks to deep neural networks via adversarial examples, 2018.
- [42] Shumeet Baluja and Ian Fischer. Learning to attack: Adversarial transformation networks. In *Proceedings of AAAI-2018*, 2018.
- [43] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li. Boosting adversarial attacks with momentum. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9185–9193, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society.
- [44] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information, 2018.
- [45] Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: an optimization-based approach, 2018.
- [46] Jonathan Uesato, Brendan O’Donoghue, Aäron van den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. *ArXiv*, abs/1802.05666, 2018.
- [47] Kathrin Grosse, David Pfaff, Michael Thomas Smith, and Michael Backes. The limitations of model uncertainty in adversarial settings, 2019.
- [48] Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks, 2020.
- [49] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.
- [50] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors, 2019.
- [51] Florian Tramer and Dan Boneh. Adversarial training and robustness for multiple perturbations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [52] Yandong Li, Lijun Li, Liqiang Wang, Tong Zhang, and Boqing Gong. NATTACK: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3866–3876. PMLR, 09–15 Jun 2019.
- [53] Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. Simple black-box adversarial attacks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine*

-
- Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2484–2493. PMLR, 09–15 Jun 2019.
- [54] Shashank Kotyan and Danilo Vasconcellos Vargas. Adversarial robustness assessment: Why both l_0 and l_∞ attacks are necessary, 2020.
- [55] Jérôme Rony, Luiz G. Hafemann, Luiz S. Oliveira, Ismail Ben Ayed, Robert Sabourin, and Eric Granger. Decoupling direction and norm for efficient gradient-based l_2 adversarial attacks and defenses, 2019.
- [56] Jinyin Chen, Mengmeng Su, Shijing Shen, Hui Xiong, and Haibin Zheng. Poba-ga: Perturbation optimized black-box adversarial attacks via genetic algorithm. *Computers & Security*, 85:89–106, 2019.
- [57] Francesco Croce and Matthias Hein. Sparse and imperceivable adversarial attacks. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4723–4731, 2019.
- [58] Chris Finlay, Aram-Alexandre Pooladian, and Adam Oberman. The logbarrier adversarial attack: Making effective use of decision boundary information. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4861–4869, 2019.
- [59] Wieland Brendel, Jonas Rauber, Matthias Kümmerer, Ivan Ustyuzhaninov, and Matthias Bethge. Accurate, reliable and fast robustness evaluation. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [60] Amin Ghiasi, Ali Shafahi, and Tom Goldstein. Breaking certified defenses: Semantic adversarial examples with spoofed robustness certificates. In *International Conference on Learning Representations*, 2020.
- [61] Jianbo Chen, Michael I. Jordan, and Martin J. Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1277–1294, 2020.
- [62] Ali Rahmati, Seyed-Mohsen Moosavi-Dezfooli, Pascal Frossard, and Huaiyu Dai. Geoda: A geometric framework for black-box adversarial attacks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8443–8452, 2020.
- [63] Hadi M. Dolatabadi, Sarah Erfani, and Christopher Leckie. Advflow: Inconspicuous black-box adversarial attacks using normalizing flows, 2020.
- [64] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search, 2020.
- [65] Xiaoyi Dong, Dongdong Chen, Jianmin Bao, Chuan Qin, Lu Yuan, Weiming Zhang, Nenghai Yu, and Dong Chen. Greedyfool: Distortion-aware sparse adversarial attack. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 11226–11236. Curran Associates, Inc., 2020.
- [66] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International*

- Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.
- [67] Quoc V. Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeff Dean, and Andrew Y. Ng. Building high-level features using large scale unsupervised learning. In *Proceedings of the 29th International Conference on International Conference on Machine Learning*, ICML'12, page 507–514, Madison, WI, USA, 2012. Omnipress.
- [68] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.
- [69] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [70] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia*, MM '14, page 675–678, New York, NY, USA, 2014. Association for Computing Machinery.
- [71] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.
- [72] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4278–4284. AAAI Press, 2017.
- [73] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.
- [74] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *International Conference on Learning Representations*, 2018.
- [75] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1778–1787, 2018.
- [76] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1310–1320. PMLR, 09–15 Jun 2019.
- [77] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. In *International Conference on Learning Representations*, 2020.

-
- [78] Kevin Eykholt, Ivan Evtimov, Earlene Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018.
- [79] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16*, page 1528–1540, New York, NY, USA, 2016. Association for Computing Machinery.
- [80] Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: adversarial patches to attack person detection, 2019.
- [81] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Amrith 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.
- [82] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017.
- [83] 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.
- [84] Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. AdverTorch v0.1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019.
- [85] Dou Goodman, Hao Xin, Wang Yang, Wu Yuesheng, Xiong Junfeng, and Zhang Huan. Advbox: a toolbox to generate adversarial examples that fool neural networks, 2020.
- [86] Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. Deeprobust: A pytorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149*, 2020.
- [87] Xiang Ling, Shouling Ji, Jiayu Zou, Jiannan Wang, Chunming Wu, Bo Li, and Ting Wang. Deepsec: A uniform platform for security analysis of deep learning model. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 673–690, 2019.
- [88] Nicholas Carlini. A critique of the deepsec platform for security analysis of deep learning models. *ArXiv*, abs/1905.07112, 2019.
- [89] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch, 2018.
- [90] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2206–2216. PMLR, 13–18 Jul 2020.

- [91] Jonas Rauber and Matthias Bethge. Fast differentiable clipping-aware normalization and rescaling. *arXiv preprint arXiv:2007.07677*, 2020.
- [92] Nicolas Papernot, Patrick McDaniel, and Ian J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *ArXiv*, abs/1605.07277, 2016.
- [93] Xin Liu, Huanrui Yang, Ziwei Liu, Linghao Song, Hai Li, and Yiran Chen. Dpatch: An adversarial patch attack on object detectors. *arXiv preprint arXiv:1806.02299*, 2018.
- [94] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *ICML*, 2020.
- [95] Sara Sabour, Yanshuai Cao, Fartash Faghri, and David J. Fleet. Adversarial manipulation of deep representations. *CoRR*, abs/1511.05122, 2016.
- [96] Hossein Hosseini, Baicen Xiao, Mayoore S. Jaiswal, and Radha Poovendran. On the limitation of convolutional neural networks in recognizing negative images. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 352–358, 2017.
- [97] Nina Narodytska and Shiva Kasiviswanathan. Simple black-box adversarial attacks on deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1310–1318, 2017.
- [98] Uyeong Jang, Xi Wu, and Somesh Jha. Objective metrics and gradient descent algorithms for adversarial examples in machine learning. In *Proceedings of the 33rd Annual Computer Security Applications Conference, ACSAC 2017*, page 262–277, New York, NY, USA, 2017. Association for Computing Machinery.
- [99] Warren He, Bo Li, and Dawn Xiaodong Song. Decision boundary analysis of adversarial examples. In *ICLR*, 2018.
- [100] Mark Lee and Zico Kolter. On physical adversarial patches for object detection, 2019.
- [101] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng Chau. ShapeShifter: Robust physical adversarial attack on faster r-CNN object detector. In *Machine Learning and Knowledge Discovery in Databases*, pages 52–68. Springer International Publishing, 2019.
- [102] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. In *ICML*, 2019.
- [103] Takeru Miyato, Shin ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing with virtual adversarial training. *arXiv: Machine Learning*, 2015.
- [104] Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Accelerating adversarial training via maximal principle, 2019.

Appendices

This page is intentionally left blank.

Appendix A

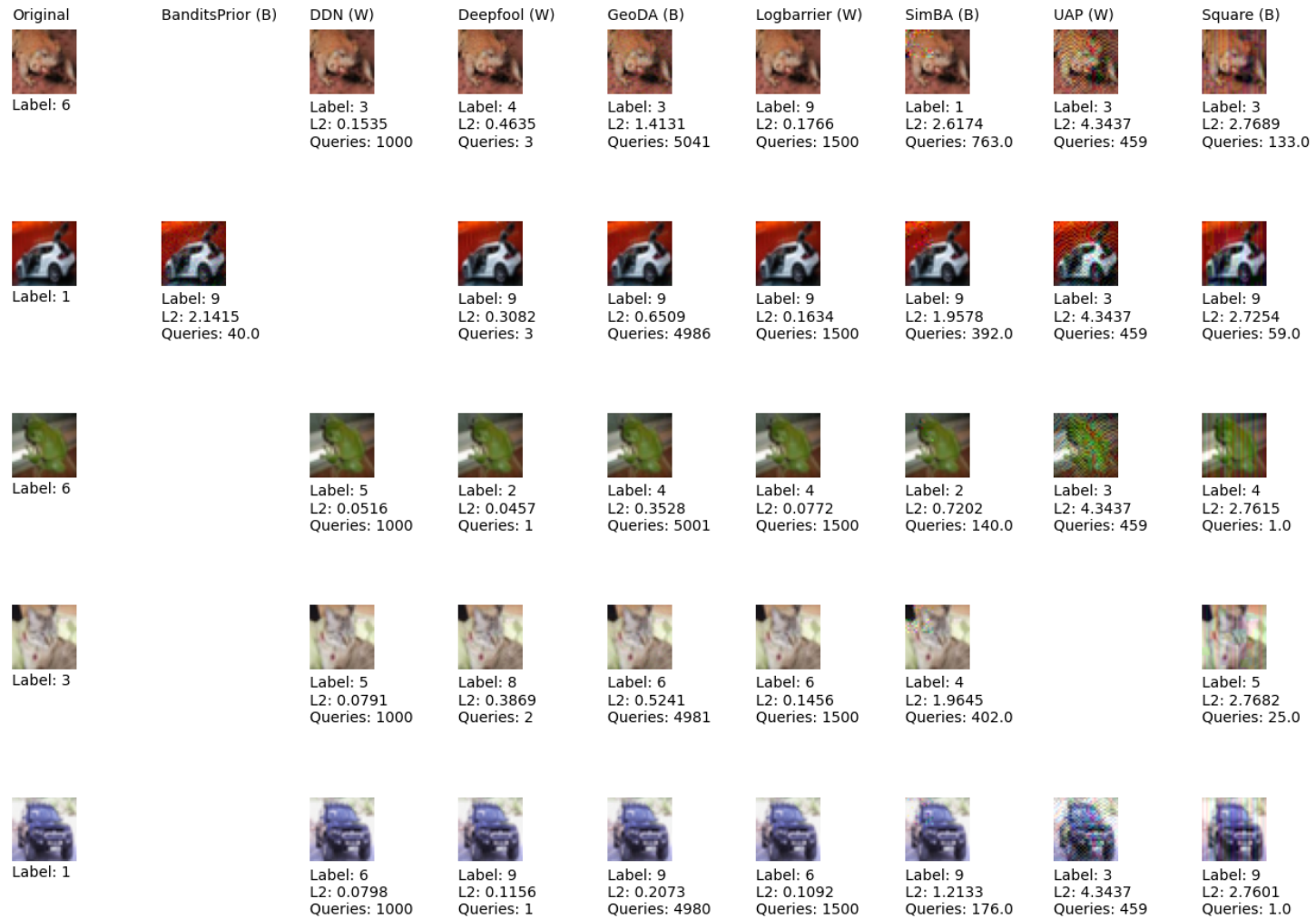


Figure 1: ResNet18 generated images (6-10)

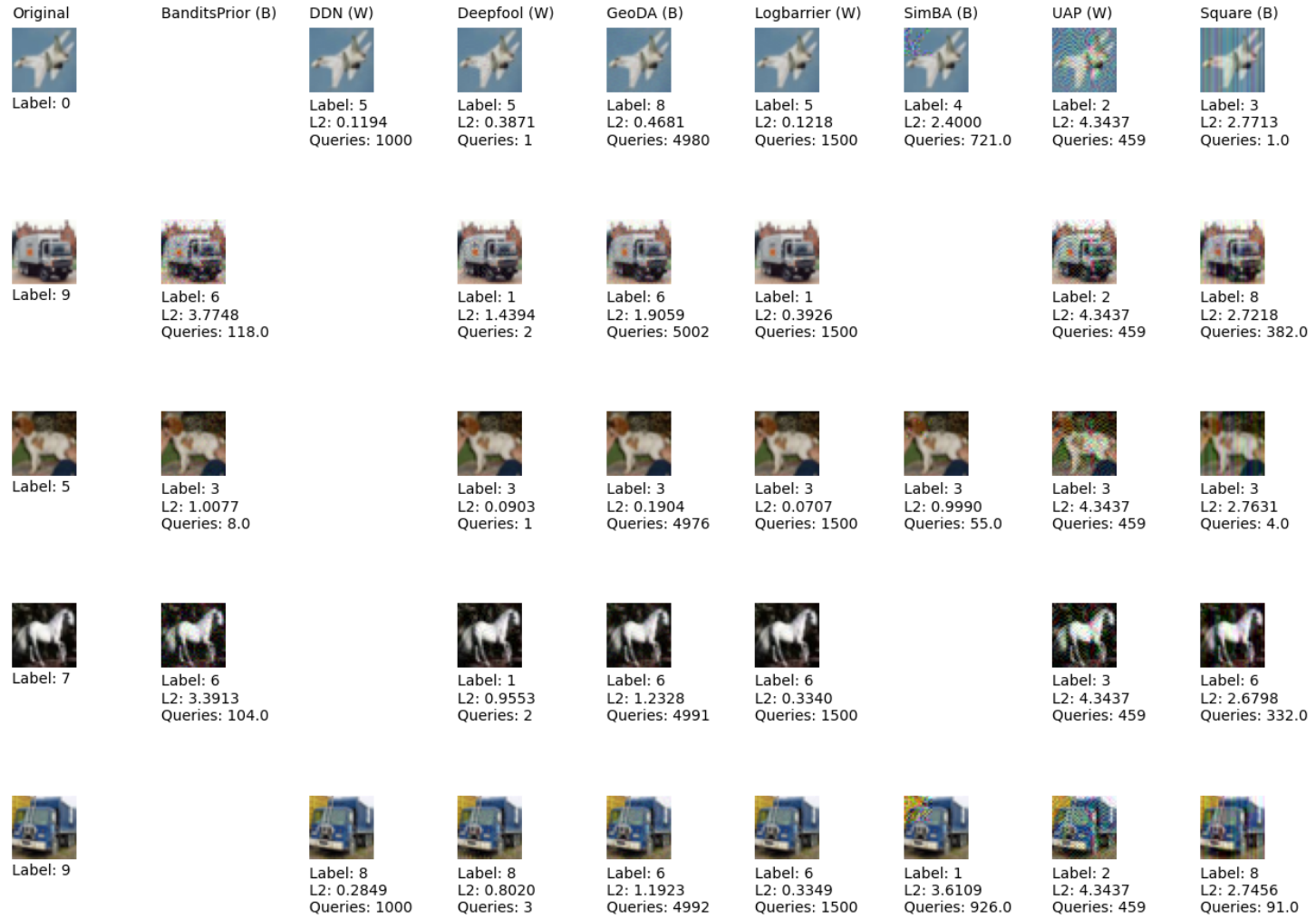


Figure 2: ResNet18 generated images (11-15)

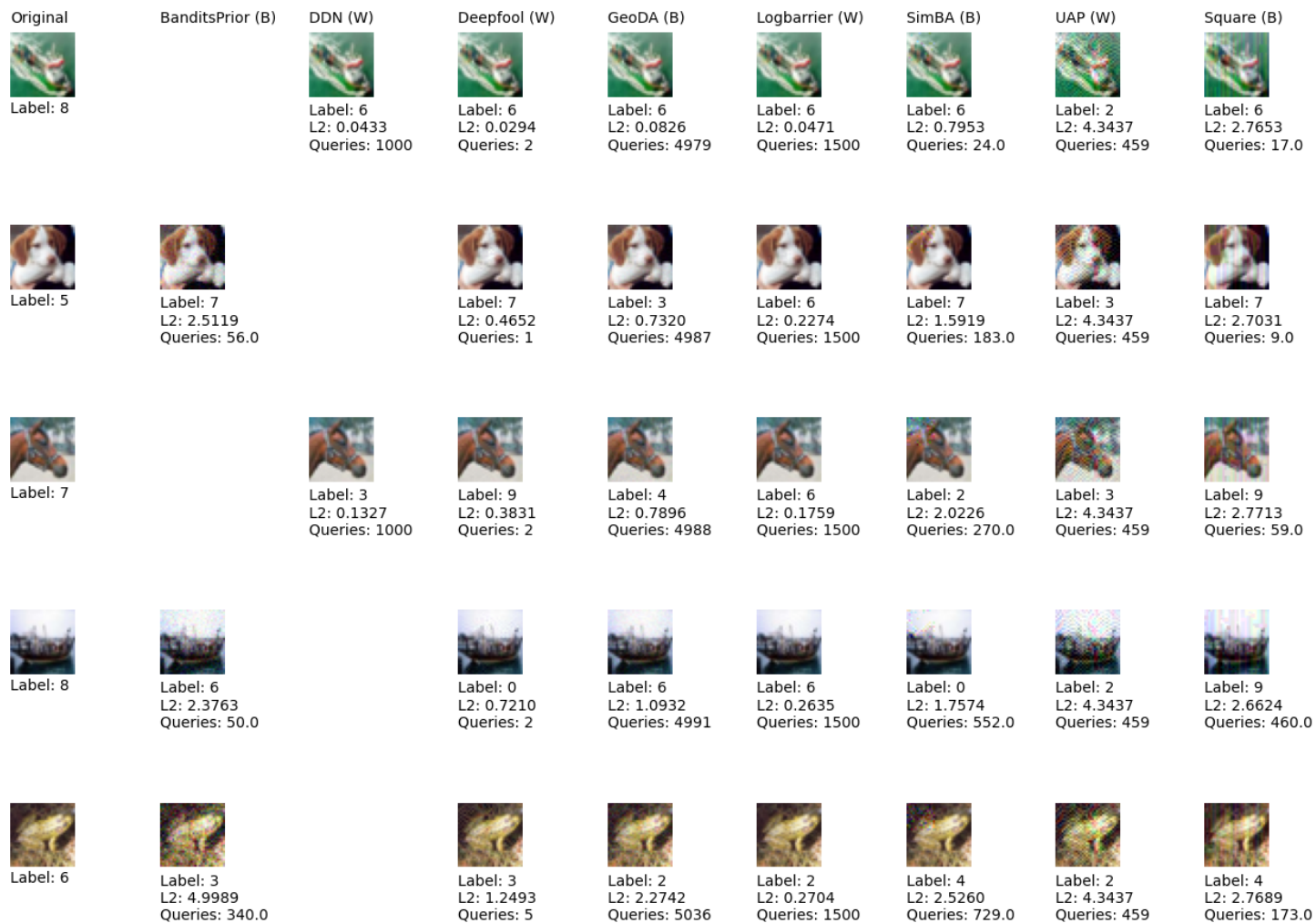


Figure 3: ResNet18 generated images (16-20)

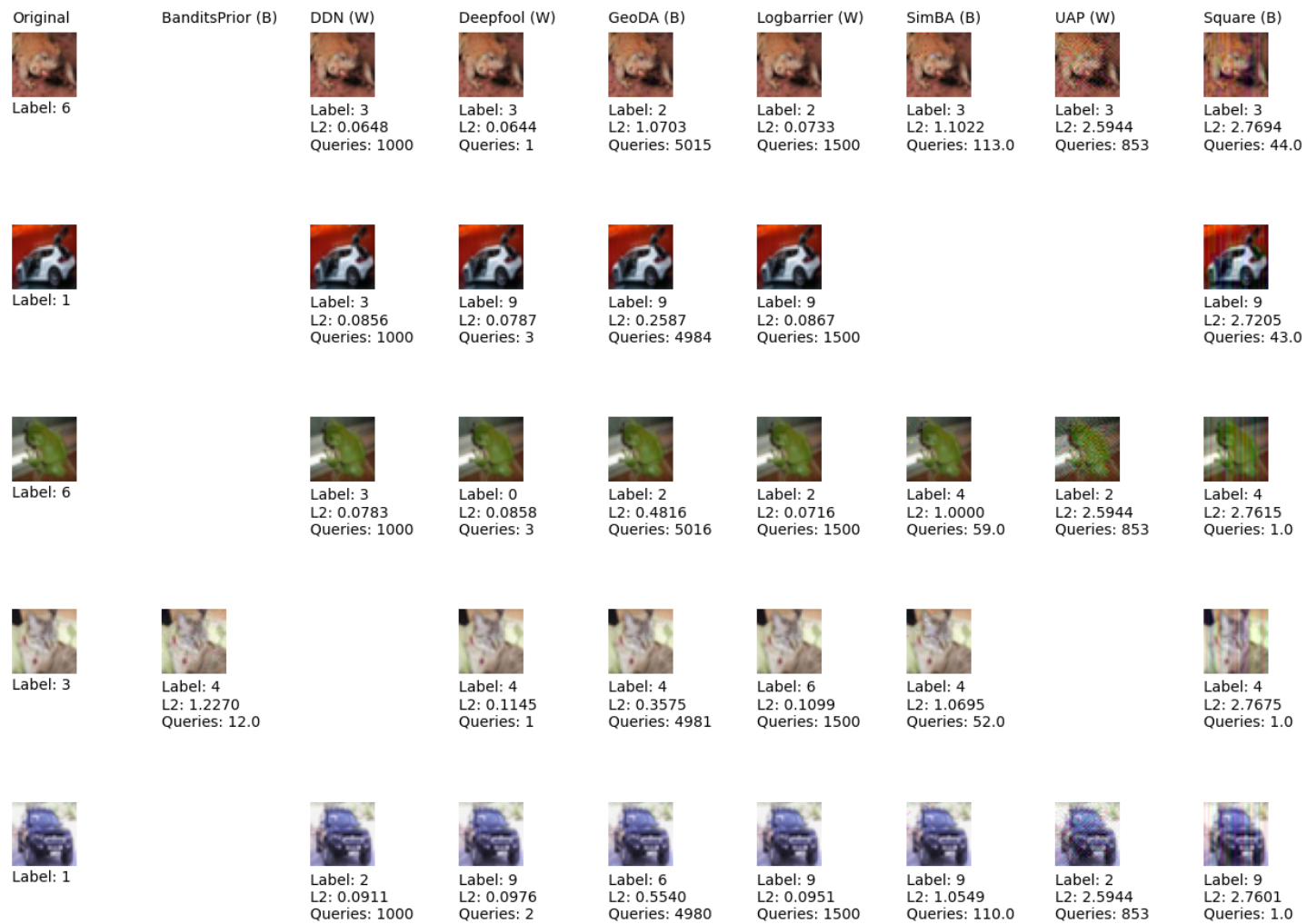


Figure 4: MobileNet generated images (6-10)

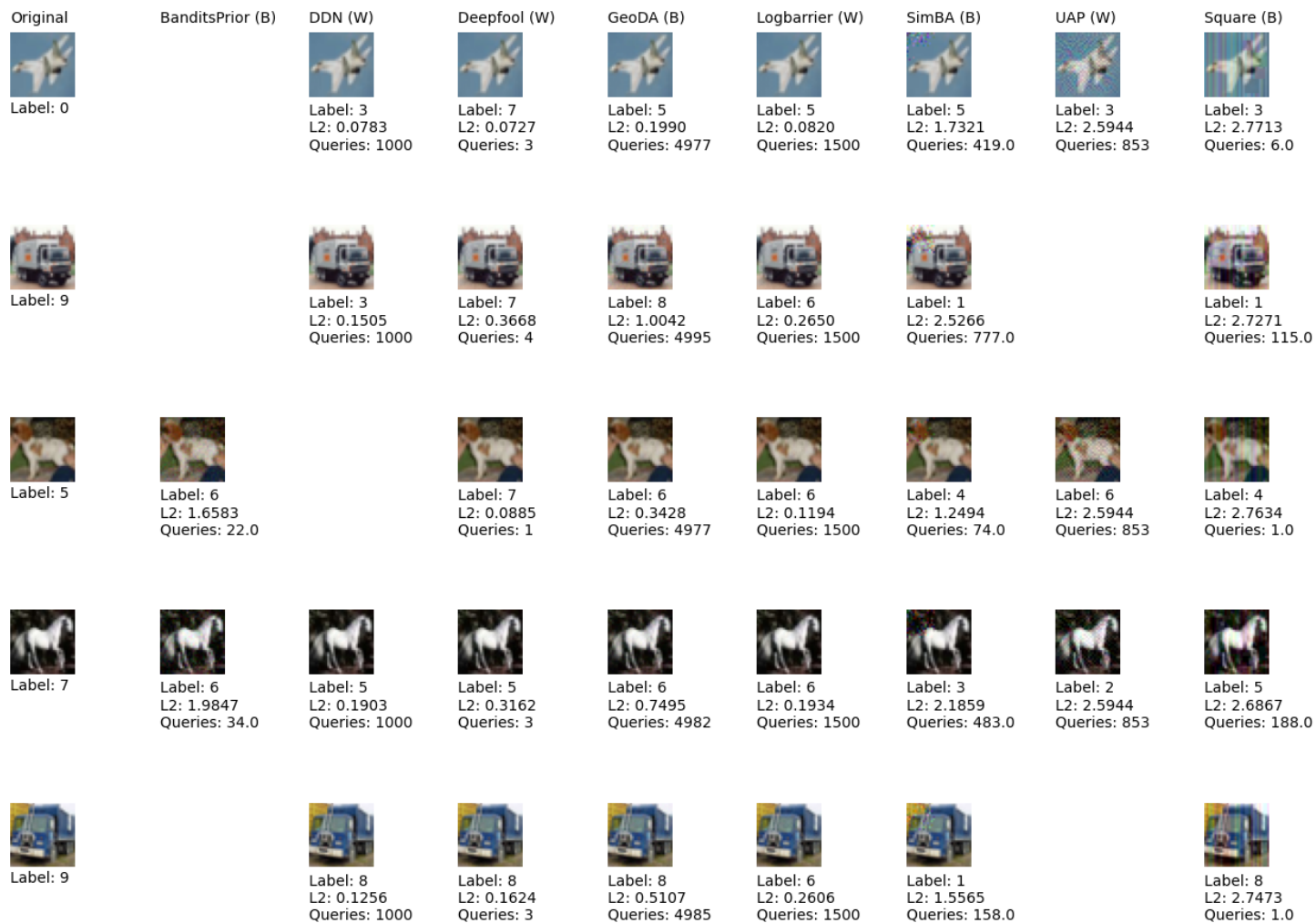


Figure 5: MobileNet generated images (11-15)

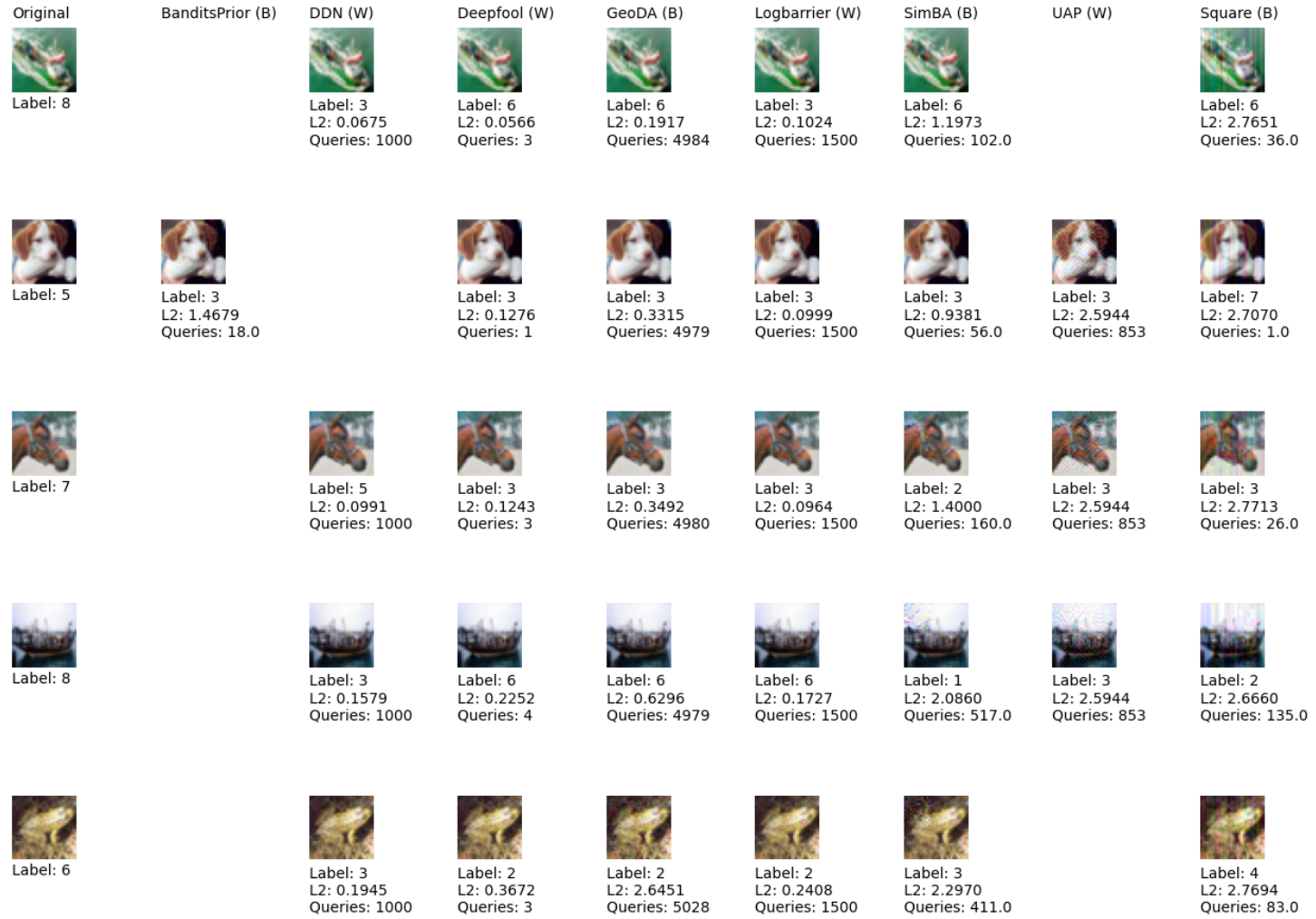


Figure 6: MobileNet generated images (16-20)

This page is intentionally left blank.

Appendix B