OMLASP - Open Machine Learning Application Security Project

Authors
Marcos Rivera Martínez
Francisco José Ramírez Vicente

Attack and mitigation techniques to audit Machine Learning algorithms

Ideas Locas - Telefonica
March 2021

Contents

List of Figures

List of Tables

Chapter 1

What is OMLASP

1.1 Prologue

The Machine Learning field, and more precisely, Deep Learning algorithms, are gaining importance since they can solve classical problems in a much better and more efficient way. That is, they achieve more outstanding results in less time.

Right now, these algorithms are present in everyday life of millions of people. Some examples of these algorithms are Alexa, Aura, Google Assistant, Siri, Cortana, autonomous vehicles, airport security cameras, recommendation algorithms from different platforms such as Netflix and Amazon, pedestrian detection in modern cars, etc.

These algorithms are not error-free. They may contain biases and security flaws. For this reason, they must be audited as well. This guide will describe several attacks, detection, and mitigation techniques to audit the machine learning algorithm's security. This guide will also describe detection methods and, therefore, mitigate as much as possible the biases present in these algorithms. Biased algorithms use unbalanced data that are not representative of reality or that reflect societal biases.

1.2 To whom is it addressed

- Software developers: Artificial Intelligence will soon become a mainstream area today if it is not already. Therefore, software developers will also need to know the weaknesses and biases that impact Machine Learning models.
- Security specialists: they are responsible for testing and improving the software's security.
- Artificial Intelligence Specialists: they are in charge of designing and implementing Artificial Intelligence algorithms. These specialists must know from the very beginning the weaknesses and biases that may compromise them to mitigate them at an early stage. The earlier this is done, the lower the costs and time required.

1.3 Why OMLASP

Owing to the excellent computing capacity and the amount of data available today, Artificial Intelligence algorithms, especially Machine Learning algorithms, are becoming increasingly important. These algorithms are part of the daily lives of millions of people. Until now, no one has been concerned and formally defined a methodology to check the safety of these algorithms and identify and eliminate biases. This guide is intended to become a standard for auditing Machine Learning algorithms. It will not only focus on the security of the algorithms but also on their biases.

Typically, when applications that use Machine Learning or Deep Learning algorithms are deployed, only traditional vulnerabilities are checked from security audits. However, these algorithms are also exposed to other vulnerabilities or deficiencies that could be exploited by attackers and that will be seen throughout this Framework.

This Framework aims to become a standard for Machine Learning algorithm auditing. It will not only focus on the security of the algorithms, but also on the biases of the algorithms. Therefore, it will be divided into the following two sections:

- Security: the attack surface and attack scenarios will be outlined, as well as the capabilities and goals of the attackers. The various existing attack and defense techniques will be described in depth in order to define a methodology that will allow auditing these algorithms.
- Biases: causes, types, and solutions will be explained in detail to establish a methodology to minimize them.

Chapter 2

Introduction to Adversarial Attacks

2.1 What are Adversarial Attacks

The adversarial Machine Learning is a branch of machine learning that seeks to fool a ML model or algorithm providing malicious input to cause an incorrect classification. These algorithms are vulnerable to this class of attacks, since they are designed to be used in particular scenarios with conditions similar to those of training. The rise of these algorithms and their great potential for solving problems has led to them being used for increasingly generic situations (for example, in self-driving vehicles). Most public clouds already have MLaaS services.

Before beginning to explain this type of attack in more depth, it will be shown how Machine Learning algorithms are usually implemented and what operations they usually perform internally once they are in the operational phase.

2.2 Attack surface

It is important to know all the stages and operations that are carried out within a Machine Learning application, both in the training phase and operational phases. Thus, it will be easier to identify where and how an adversary can degrade the system or algorithm.

2.2.1 Training of a Machine Learning algorithm

The next section details the most frequent stages in the implementation of a Machine Learning algorithm:

- Business analysis: before implementing the algorithm, it is necessary to analyze the problem to be solved. You have to identify the business objectives, the input variables of the model, what you want to predict, etc.

- The dataset is obtained: it can come from sensors, such as cameras and motion sensors, or data repositories that can be public or private. When the data comes from sensors, an implicit conversion process is carried out from the physical domain to the digital domain. It is essential to keep that in mind as attacks can occur even before the dataset is obtained.

- Data preprocessing: once you have the data, it must be preprocessed due to the intrinsic features of Machine Learning algorithms. These algorithms require that the input data have a certain dimensionality. For example, if you build a multilayer perceptron that receives input images of the following dimensions 500 x 500 x 3 (number of pixels in width, number of pixels in height, and number of color channels), it will not be able to correctly process images of other dimensions like 600 x 600 x 3. To solve this, what you do is resize the input. Additionally, other techniques can be applied to favor the learning of the algorithms, such as dataset normalization, data augmentation, etc.

- Dataset division: once the data has been preprocessed, it is usually divided into three sets:

  - Training data: as its name suggests, it is used to train the Machine Learning model.
  - Validation data: they are used to carry out tests during the training of the model. It should be pointed out that the model has never seen these data and for this reason they are used to evaluate it.
  - Test data: used to check the success rate of the model once it has been trained. This data set has also not been seen by the model.

When the dataset is small it is usually divided into training, validation and testing data in the following proportions 80%-10%-10%, respectively. If the dataset is larger, the proportions are usually 90%-5%-5%. It is worth mentioning that other different ratios can be chosen and even create more data sets or delete the validation set. All of this depends on the problem to be solved, different design decisions, etc.

- Model training: in this stage, the Machine Learning algorithm that best suits the problem is chosen. Then it is needed to train this algorithm with the training data set. In this phase, the validation data set is also used to check different metrics of the model, such as the success rate, recall, loss of the model, etc.

- Model evaluation: after the model is trained, the test data set is used to evaluate the success rate and whether it meets expectations.

It should be pointed out that some of the stages of this process can be applied iteratively to improve the algorithm and its results. Figure 2.1 summarizes the steps explained.

2.2.2 ML algorithm in operational phase

Once the Machine Learning algorithm has been evaluated and meets expectations, an application can be built that uses the implemented algorithm. When the application receives an input, it has to perform a series of chain operations, which can be seen as a pipe. It is important to emphasize the concept of the pipeline. By this we mean that the output of an operation is the input of the next operation. Each of them is defined below:

- Obtaining the input data: the input data of the application can come from sensors, such as cameras, or data repositories, such as an information system. When the data comes from sensors, an implicit process of conversion to the digital domain is carried out. It is important to keep this in mind, as attacks can occur even before the dataset is obtained.

- Data preprocessing: transformations are performed on the data. These are usually the same transformations that have been made in the training phase of the algorithm.

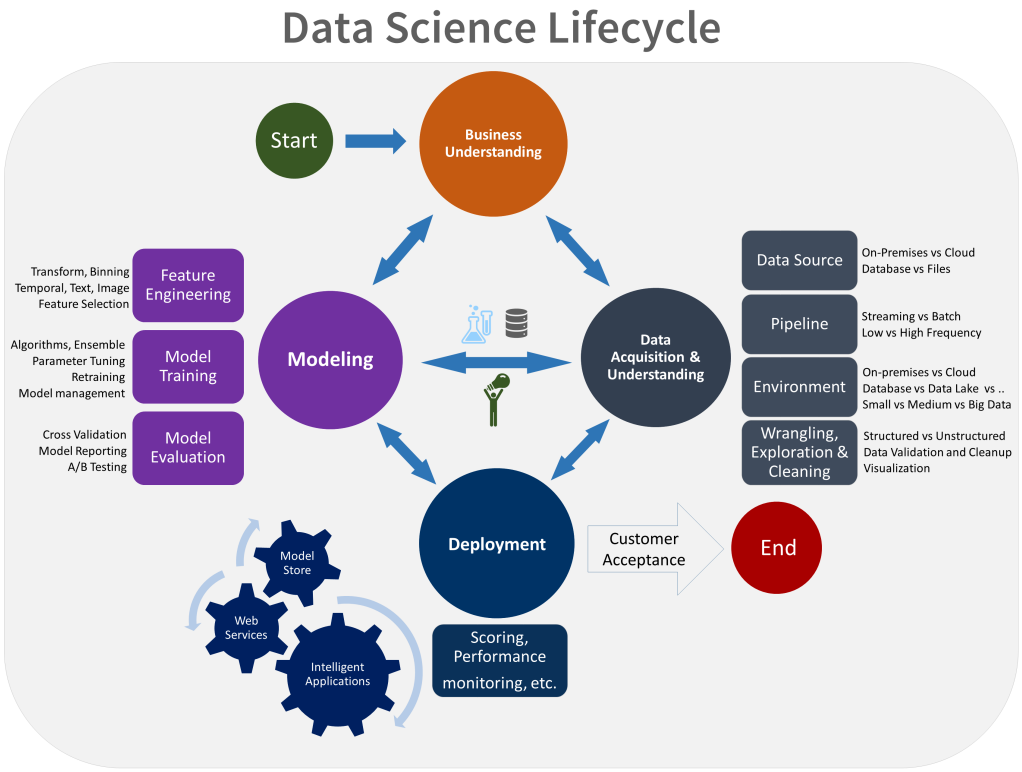

Figure 2.1: Process that is followed when implementing a Machine Learning algorithm. Source: Microsoft

- Model prediction: once the input has been pre-processed, it is sent to the model to produce an output.
- Execution of an action: the output of the model is analyzed and then, based on this, an action is executed. For example, a self-driving car could use a model responsible for detecting traffic signs and then take actions based on the outputs of the model. In case of encountering, for example, a stop sign then the action of braking the car would be taken.

Figure 2.2 summarizes the aforementioned pipeline operations.

An overview of the entire process that takes place during the implementation and operational phase of Machine Learning algorithms helps to identify which elements are most exposed and therefore most susceptible to attack.

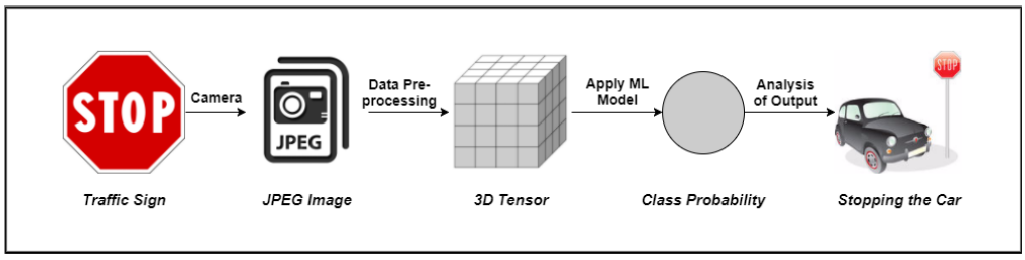

Figure 2.2: Operations that are usually carried out in the operational phase of a Machine Learning algorithm. Source

2.3 Types of attacks

Like traditional cybersecurity attacks, adversarial attacks can be classified into three types, depending on the attacker's knowledge of the Machine Learning algorithm.

- White box attacks: the attacker has access to the dataset, parameters and hyperameters of the model.
- Black box attacks: the attacker only has access to the inputs and outputs of the model. It is important to note that the outputs of a model can be more or less precise. For example, there are models which output is an array of probabilities and other models which output is a label. The more information a model provides, the easier it will be to attack it.
- Gray box attacks: the attacker is in an intermediate point between the previous two attacks.

Attacks can also be classified depending on the actions and objectives of the attacker

- Poisoning attack: it occurs when the attacker has access to the training data and can modify it, usually in an imperceptible way, to later take advantage of it and have control over the model. Figure 2.3 shows a diagram of the attack.
- Evasion attack: it is the most common attack. Once the algorithm is in the operational phase, the attacker modifies input samples so that

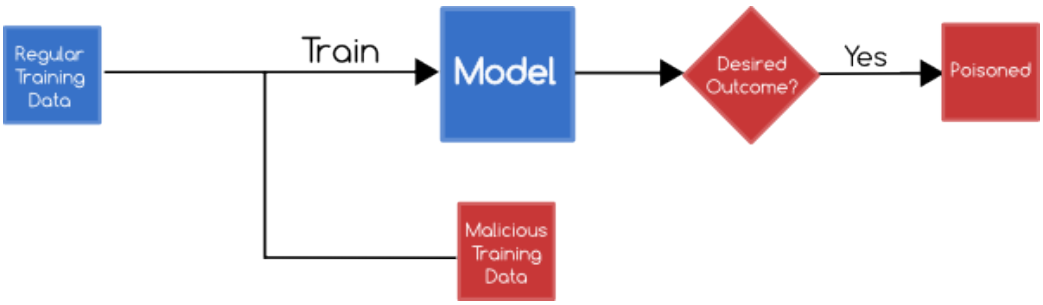

Figure 2.3: Poisoning attacks. Source

the model misclassify them. For example, in the case of a self-dirving car, the input image could be modified (sometimes imperceptibly to the human eye) with the aim that the model makes an incorrect classification and there is a traffic accident. Figure 2.4 shows a diagram of the attack.

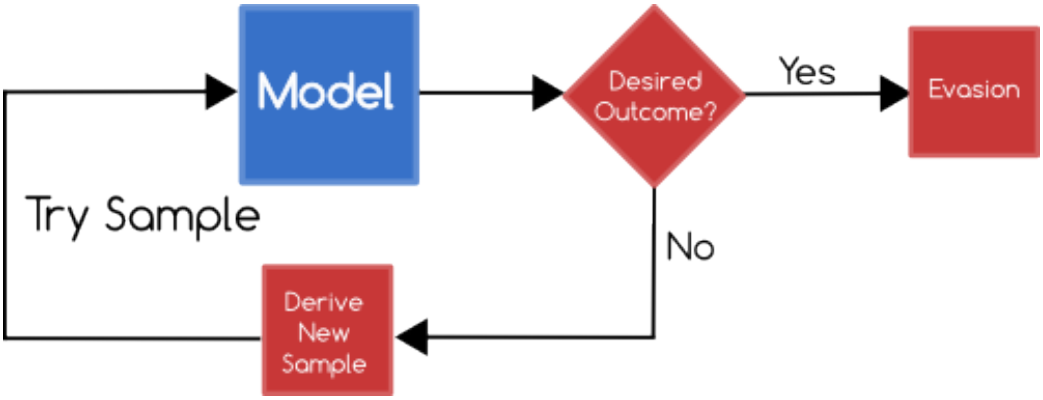

Figure 2.4: Evasion attacks. Source

- Exploratory attack: it is very similar to Cybersecurity reversing attacks. The attacker does not have access to the model and seeks to obtain the maximum information about it. For example, these attacks can be used to determine whether certain data was used to train the algorithm. This attack can be especially relevant when the model has been trained with sensitive data (clinical data, for example). There are also other techniques that allow an attacker to clone a model and go from a black box to a white box attack. It is worth mentioning that

these types of attacks are usually given less importance, but this guide will emphasize them. Figure 2.5 shows a diagram of the attack.

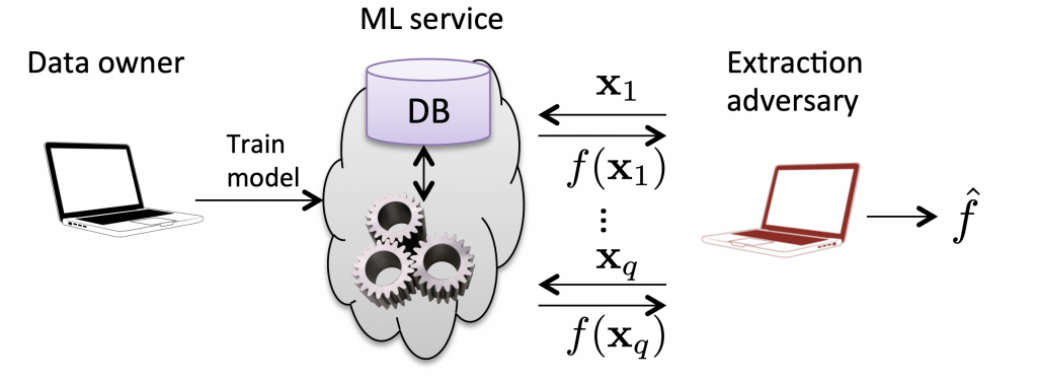

Figure 2.5: Exploratory attacks. Source

Table 2.1 shows how each attack affects the Machine Learning algorithm from the point of view of the three pillars of information security.

Types of attack	Integrity	Confidentiality	Availability
Poisoning	✔	✘	✔
Evasion	✘	✘	✔
Exploration	✘	✔	✘

Table 2.1: How each attack affects the pillars of Information Security

2.4 Attacker objectives

The most common targets of attackers are described below:

- Misclassification: the attacker modifies the input of the model to alter the output classification of the model. It can be used to bypass security measures, content filters, etc.
- Targeted Misclassification: the attacker modifies the model inputs to force the output of the classification model to be a specific target class

- Source / target misclassification: The adversary attempts to force the output of classification for a specific input to be a particular target class. For example, the input image of the 'stop' sign will be predicted as a 'go' sign by the classification model.

- Obtain information about the model: the attacker obtains the maximum information from the model to carry out subsequent attacks or simply to clone it and use it for his own interests. Today there are numerous paid Machine Learning as a Service (MLaaS) services. Attackers could clone these types of services and offer them later at a lower price.

2.5 Actions that can be performed by attackers

The actions that attackers can take on a Machine Learning application to degrade it and meet their objectives are defined below. These are divided into two large groups: actions that attackers can perform in the training phase and actions that they can perform in the operational phase.

2.5.1 Actions in the training phase

Attackers have access to part or the entire training dataset and/or algorithm. They can perform three actions:

- Data injection: the attacker does not have access to the training data set or the algorithm, however, he can inject new training examples, partially modifying the behavior of the algorithm or the application. For instance, on the Internet, some services learn from some of the user's samples. If the user is able to enter many erroneous or modified samples, he could modify the output of the algorithm in his favor. An example of this can be the Google translator service. See the Figure 2.6.

- Data modification: the attacker does not have access to the model, but he has access to the data. Therefore, he can poison or modify the data before it is used to train the Machine Learning algorithm.

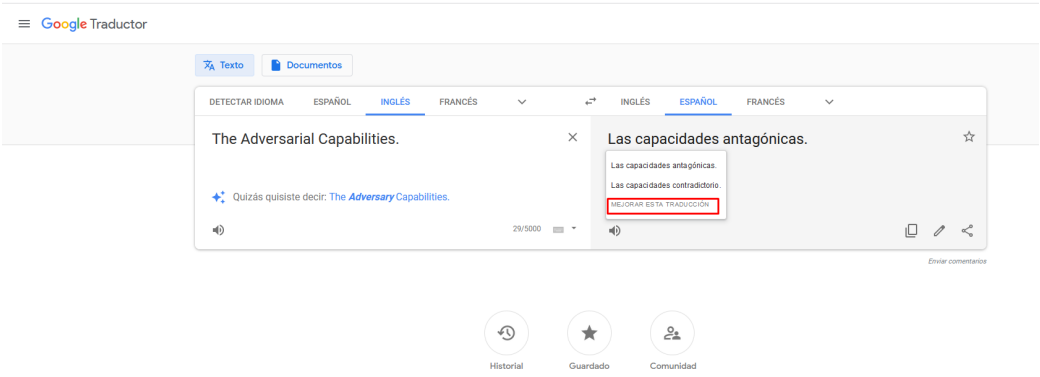

Figure 2.6: Google translator service

- Corrupting the logic of the training algorithm: the attacker only has access to the model. Therefore, he could modify some parameters of the model to make it behave in a certain way. It should be noted that this attack is the most difficult to carry out of those mentioned so far.

2.5.2 Actions in the operational phase

In this phase the attackers are not able to access the training set. Therefore they can only force the model to produce incorrect outputs by modifying the input data. Exploration attacks can accompany these attacks to obtain more information about the model and increase success probability.

Chapter 3

White box attacks

Like traditional cybersecurity attacks, adversarial attacks can be classified into three types, depending on the attacker's knowledge of the Machine Learning algorithm. In this section we will describe white box attacks.

3.1 FGSM attacks (Fast Gradient Sign Method)

In this section, we detail what the FGSM attacks are. The same structure will be followed as in the rest of the techniques explained in this guide: objectives, requirements to perform the attack, how to carry it out, how to detect it and how to protect against this type of attack.

3.1.1 Attacker objectives

FGSM or Fast Gradient Sign Method attacks aim to modify Machine Learning algorithm inputs to produce incorrect algorithm outputs. More concretely:

- Misclassification: the attacker modifies the input of the model to alter the output classification of the model. It can be used to bypass security measures, content filters, etc.
- Targeted Misclassification: the attacker modifies the model inputs to force the output of the classification model to be a specific target class
- Source / target misclassification: the adversary attempts to force the output of classification for a specific input to be a particular target

class. For example, the input image of the 'stop' sign will be predicted as a 'go' sign by the classification model.

These attacks can be used to bypass other security products, content filters, cause incorrect actions (keep in mind that applications take actions based on the output of the Machine Learning model), etc.

3.1.2 Requirements

- Access to the model: this is a white box attack. It is important to have access to the model to be able to calculate its cost function.

3.1.3 Affected Machine Learning Models

- Linear models.
- Nonlinear models that perform linear operations.
- Neural networks.

3.1.4 Attack

To explain this attack, neural networks will be used as an example. This does not mean that these attacks cannot affect other Machine Learning algorithms or models.

Before starting, it is important to know some of the operations performed by neural networks, in this case the FCNN or Fully Connected Neural Networks. Neural networks can be seen as mathematical functions capable of mapping an input X to an output Y. The simplest ones are governed by the following equation:

$$\hat{Y} = \sigma(X * W + b)$$

Where:

- $\hat{Y}$ is the output generated by the network
- σ is the activation function. It is usually a non-linear function

- X is the input data
- W, b are the parameters of the neural network

As can be seen, a neural network is a multiplication of matrices. It is important to emphasize that the activation functions of a neural network are usually non-linear, therefore, the output is also non-linear. Although it is true that the $X * W + b$ operation is linear. It is important to take all of this into account, as FGSM attacks take advantage of the linear operations of these models.

FGSM attacks seek to generate adversarial examples, almost indistinguishable by the human eye, that are capable of confusing a Machine Learning model. For this reason, these attacks are usually applied to images or computer vision applications. See the Figure 3.1. It should be pointed out that in this case the modifications on the image have been exaggerated so that they are perceptible by the human eye.

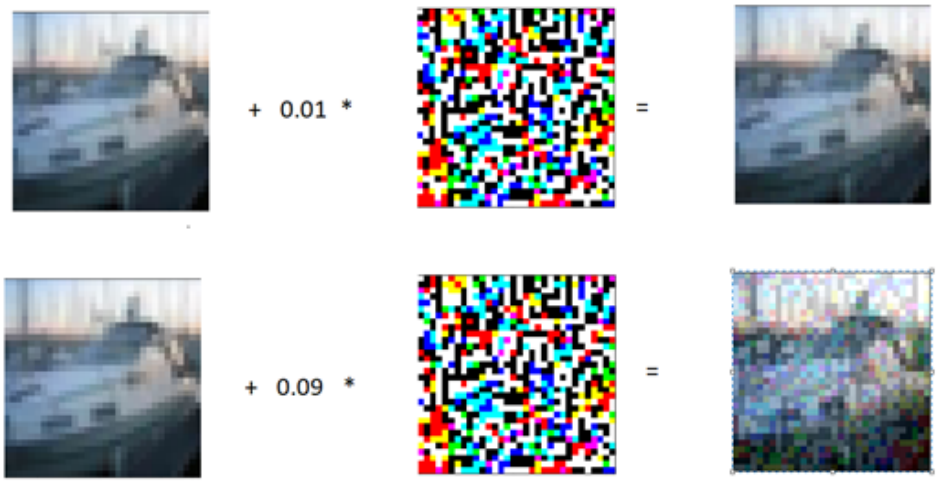

Figure 3.1: Images modified with a factor of 0.01 and 0.09

Before continuing it is important to know how images are represented in the computer domain. A grayscale image is an array that takes values between 0 and 255. See the Figure 3.2.

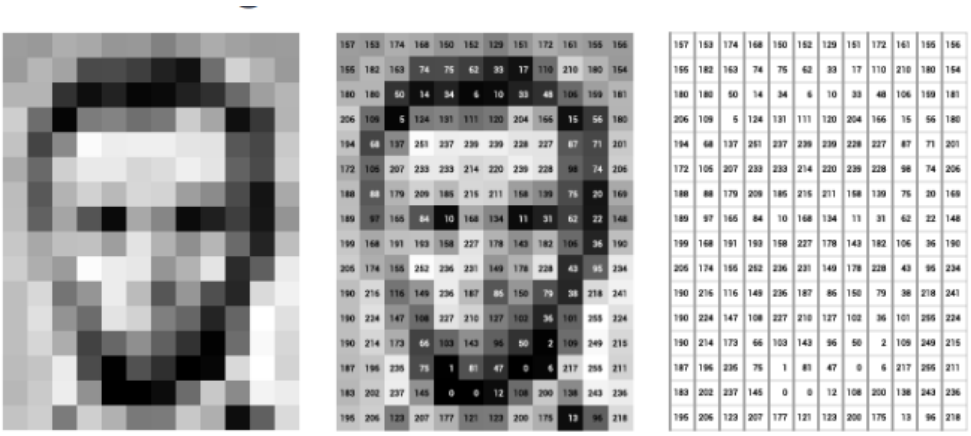

Figure 3.2: Grayscale image. Source

Color images typically have 3 channels, one for red, one for green, and one for blue (RGB). Each channel is an array that takes values between 0 and 255. The dimensions of the arrays depend on the width and height of the image. For example, when you have a color image with a width and height of 300 pixels, then you have 3 matrices (one for each color) of dimensions 300 x 300. See the Figure 3.3

Since intensity of the pixels of the images take values between 0 and 255, it is expected that the Machine Learning model will behave in the same way given the image $X = [[21, 34], [128, 11]]$ and $X' = [[21.001, 34.001], [128.001, 11.001]]$. The reality is that this is not the case, very small variations in certain pixels can cause very large variations in the output of the Machine Learning model. The more parameters the model has and the larger the input image, the more vulnerable it will be to this type of attack. Now we will see why.

To facilitate explanation and understanding, some terms have been removed from the neural network formula, leaving it as follows: $\hat{Y} = X * W$. The modified entry X' can be seen as the result of adding $X + S$, where $X = [[21, 34], [128, 11]]$ and $S = [[0.001, 0.001], [0.001, 0.001]]$. Therefore, now the output of the network is determined by the following formula $\hat{Y} = X * W + S * W$. With this formula it can be seen that if W and S coincide in sign and also W is a high-dimensional matrix (in Deep Learning models there are millions of parameters) the output can be greatly affected.

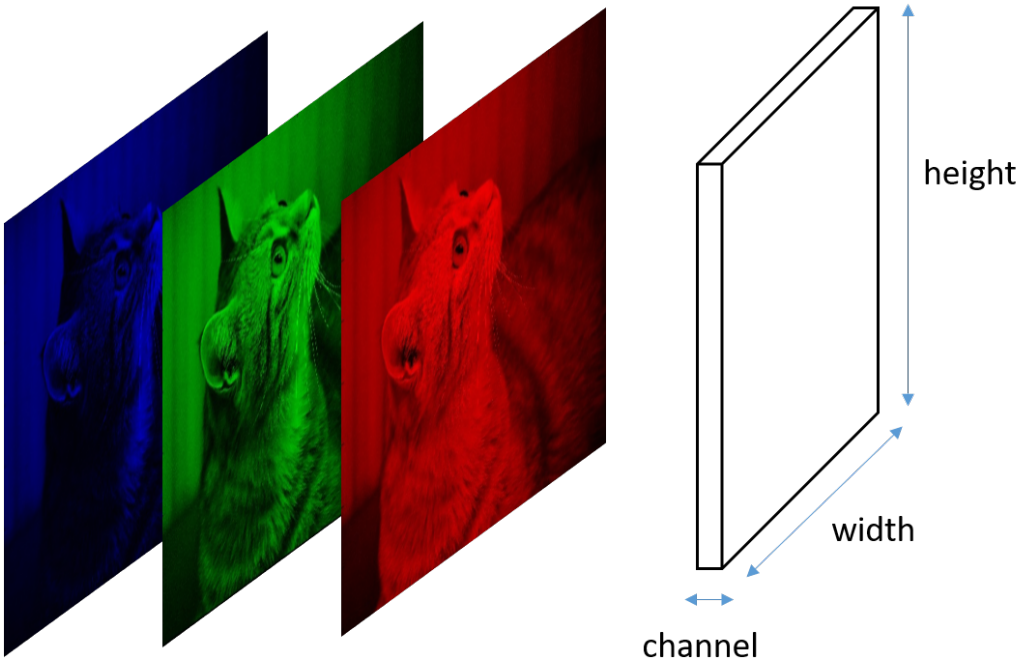

Figure 3.3: Color images. Source

An example is provided below. Given:

- $X = [[15, 22, 128, 90], [15, 22, 128, 90], [111, 45, 52, 76], [11, 12, 134, 255]]$
- $W = [[0.5, 0.5, 0.5, 0.5], [0.5, 0.5, 0.5, 0.5], [0.5, 0.5, 0.5, 0.5], [0.5, 0.5, 0.5, 0.5]]$
- $$S = [[0.001, 0.001, 0.001, 0.001], [0.001, 0.001, 0.001, 0.001], \\ [0.001, 0.001, 0.001, 0.001], [0.001, 0.001, 0.001, 0.001]]$$
- $X' = X + S$

The following results are obtained

- Results with the unmodified image: $\hat{Y} = X * W = 558$

- Results with modified image: $\hat{Y} = X * W + S * W = 603.008$

As can be seen, with only 16 parameters the output has been modified by about 10%. Therefore, small modifications to the input can cause large modifications to the output. The amplification of the output is given by the following formula:

$$Amplification = E * N * M$$

Where:

- $E = ||S||$
- N is the number of dimensions of W
- M is the mean value of W

FGSM attacks modify images using the following formula: $X' = X + S$, where $S = \epsilon * sign(\nabla x J(\Theta, X, Y))$
where:

- J is the initial cost function of the model
- Θ are the network parameters
- X are the input images
- Y are the input image labels
- ϵ is the parameter that regulates the change in the input image
- α is a parameter to regulate the importance of the initial cost function and the cost function given the modified inputs.

The formula for FGSM attacks is very similar to what has been seen so far. The only thing that changes is the way the images are modified. As can be seen, we are calculating the sign of the gradients of the cost function with respect to the input image and then multiplying by a value ϵ. The sign of the gradients allows to know the approximate direction and sense to maximize the cost function of the model. With ϵ the amount of loss is regulated. The larger ϵ the greater the variations on the image and the more noticeable it will be to the human eye. Therefore, the image is being modified in such

a way that the probability of failure of the neural network or the Machine Learning model is maximized.

The Figure 3.4 shows the attack more graphically. The black arrow with a yellow border is the gradient vector that has been calculated. If we look closely, this arrow points towards the direction of maximum growth of the cost function. If the sign function is applied to the vector then its approximate direction and sense are calculated. With the variable ϵ we are calculating the length of the vector, that is, the length of the arrow.

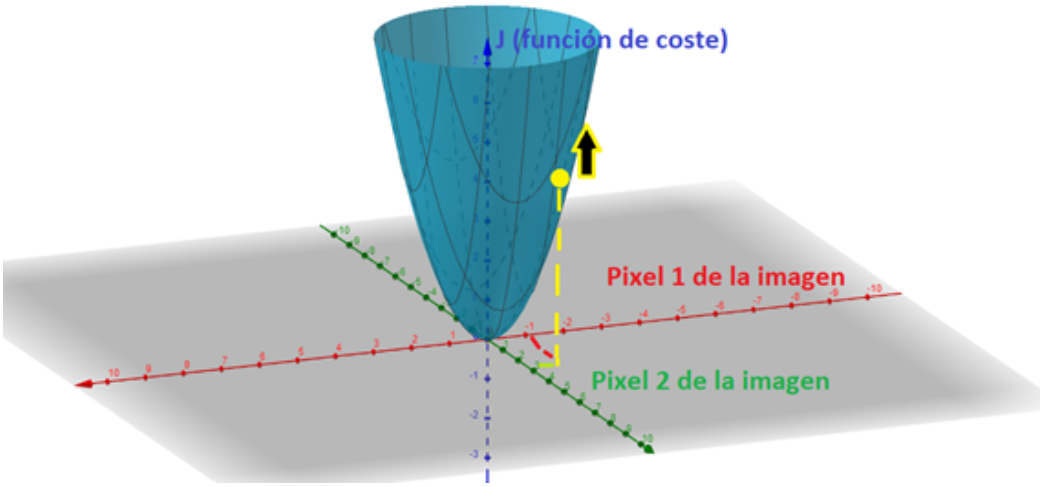

Figure 3.4: Gradient of the loss function with respect to the input image

3.1.5 Mitigation

Before starting with the mitigation of FGSM attacks, it is important to provide some data on the failure rate that they can cause on Machine Learning models and, therefore, the consequences of not taking these attacks into account when building and deploy the models. According to the work of Explaining and Harnessing Adversarial Examples a model trained for the MNIST data set with an $\epsilon = 0.25$ can misclassify the 99.9% of samples. These researchers have also tested it with the CIFAR-10 data set and claim that they have obtained an error rate of 87% for a $\epsilon = 0.1$

As can be seen, the error rates are really high, although it is also true that values of ϵ are very high too. These error rates also depend on the Machine Learning model that has been used.

To effectively protect against FGSM attacks, it is necessary to retrain the algorithm with the following cost function:

$$J'(\Theta, X, Y) = \alpha * J(\Theta, X, Y) + (1 - \alpha) * J(\Theta, x + \epsilon * sign(\nabla x J(\Theta, X, Y))$$

where:

- J is the initial cost function of the model
- Θ are the network parameters
- X are the input images
- Y are the input image labels
- ϵ is the parameter that regulates the change in the input image
- α is a parameter to regulate the importance of the initial cost function and the cost function given the modified inputs.

As can be seen, the cost function is made up of two terms. The first one (this one here $\alpha * J(\Theta, X, Y)$) is the initial cost function of the model, that is, it calculates the error that the algorithm makes when it receives an unmodified image. The second term (this one here $(1 - \alpha) * J(\Theta, x + \epsilon * sign(\nabla x J(\Theta, X, Y)))$ is the cost function given the modified image, that is, it calculates the error that the algorithm makes when it receives a modified image with a value of ϵ. Therefore, this new cost function will allow the Machine Learning algorithm to behave in a robust way when it receives images modified with the FGSM attack formula.

Additionally, certain modifications can be made to the network architecture, which favor the robustness of the model. Some of these modifications are the depth of the network, the number of neurons per layer, regularization parameters of the network, etc. These changes have to be introduced empirically, that is, try with different values and find the optimal one, the

one that makes the network behave better for this type of attack.

With all the changes discussed so far, the error rates can be reduced to approximately 15%. Not bad at all, it has dropped from 90% to 15%.

It should be noted that the new cost function that has been discussed not only makes the model behave more robustly against FGSM attacks, but also prevents overfitting and, therefore, better results are obtained with unmodified images. This is because you are implicitly training the algorithm with a larger training set, as you are using the initial (unmodified) images and the modified images. Therefore, a double objective has been achieved: to improve the model and defend it against FGSM attacks.

3.1.6 Detection

To detect FGSM attacks, a classification algorithm (for example, a neural network) can be trained, which is capable of classifying images as "modified" or "unmodified". It should be noted that it is not possible to train a model that is capable of detecting modifications in all existing algorithms. As you have seen, FGSM attacks maximize the loss of a specific model. For this reason, the FGSM attack detection and classification algorithm cannot (or should not be) reused for other models.

It is also important to note that this new model should be protected against FGSM attacks (following the instructions in the mitigation section), since it could also be attacked and degraded.

3.2 Scaling Image Attacks

This section details what Image Scaling Attacks consist of. The same structure will be followed as in the rest of the techniques explained in this guide: objectives, requirements to perform the attack, how to carry it out, how to detect it and how to protect against this type of attack.

3.2.1 Attacker objectives

This attack aims to modify the input image of the neural network in such a way that when a scaling operation is performed on the image, a totally different image is obtained. In the Figure 3.5 you can see an example, where the original image is a cat and after reducing the size of the image an image of a dog is obtained.

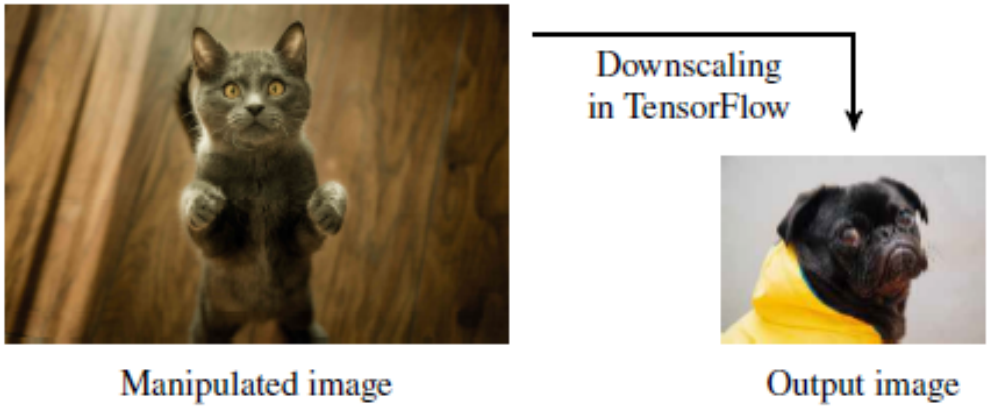

Figure 3.5: Example of scaling attack. When the image size is reduced, a totally different image is obtained. Source

Image scaling is a widely used procedure in Computer Vision and a very common preprocessing algorithm in Machine Learning, since most of them only accept input of a certain size. For example, the InceptionV3 neural network accepts only 224 x 224-dimensional input images. Therefore, if you have a larger or smaller image, it is necessary to rescale it before sending it to the model.

This attack will only focus on reducing the size of the images, as this is the most common operation.

3.2.2 Requirements

This attack is very simple to carry out if the attacker knows the preprocessing operations that are performed on the received image before being sent to the neural network or the Machine Learning algorithm. If the attacker does

not know the preprocessing operations, he could try different sizes of images and scaling algorithms until he finds those that produce the greatest loss on the model.

Therefore, to perform this attack, the attacker must know the target size of the image and the scaling algorithm that is used in the preprocessing stage.

3.2.3 Affected Machine Learning Models

Different kind of models

3.2.4 Attack

Images can be viewed as signals, in the same way as audio and video. The difference is that audio, for example, is a continuous signal and is one-dimensional, while images are discrete signals and have two dimensions. Therefore, if you want to reduce the size of an image or signal, you only have to sample it every so often, as can be seen in the Figure 3.6.

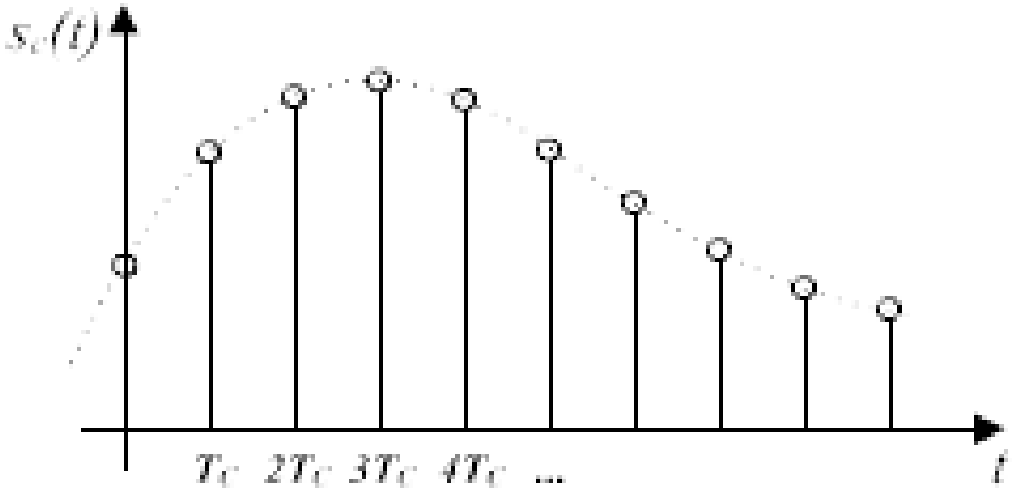

Figure 3.6: Sampling a signal. Source

After performing the sampling, seen in the Figure 3.6, it would be possible to recover a very similar signal simply by joining the points with a straight

line. But what happens if it is sampled at a lower frequency, as can be seen in the Figure 3.7.

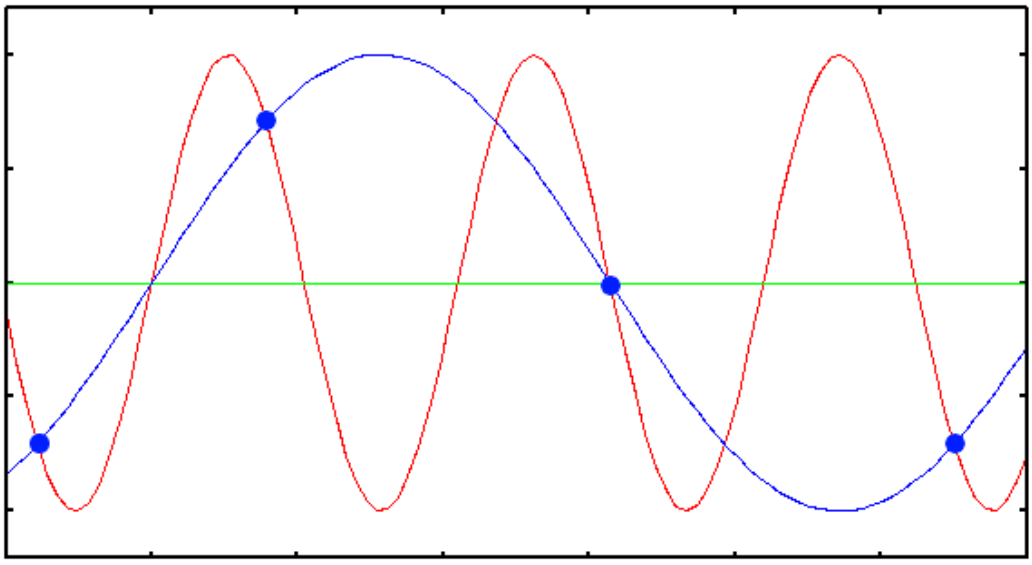

Figure 3.7: Aliasing problem. Source

As can be seen, when the sampling frequency is small it is not possible to recover the signal. According to the Nyquist-Shannon theorem, the sampling frequency must be at least twice the frequency of the signal in order to correctly recover or approximate the original signal. This can be formalized mathematically as:

$$F_s = 2 * F_m$$

F_s being the sampling frequency and F_m the maximum frequency of the original signal.

If it is known at which points the signal is sampled, then the image scaling attack could be performed, modifying just those points, as can be seen in the Figure 3.8 (the image on the left is the signal without modify and the image on the right is the modified image right at the sampling points).

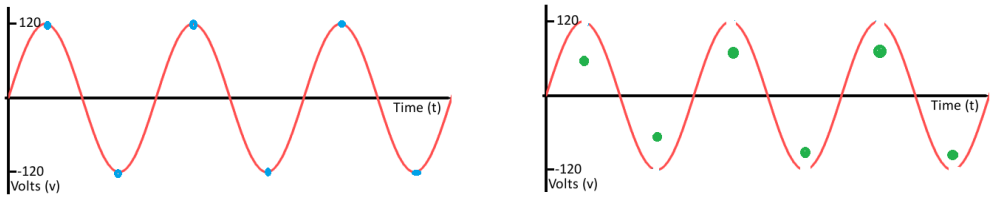

Figure 3.8: Modifying a signal right at the sample points. On the right is the original signal and on the left is the modified signal.

Scaling operations on images use sampling, but most of them also apply a series of filters or convolutions to reduce the frequency of the signal and reduce the effect of aliasing. Therefore, the attack becomes somewhat more complicated. Aliasing usually occurs when the Nyquist-Shannon theorem already explained is not fulfilled.

The convolution operation can be understood as a moving average (applying a function with a sliding window). See the Figure 3.9.

If the convolution operations that are usually performed to scale the images are analyzed, all of them perform an average of the surrounding pixels. The size of the convolution filter makes the output image larger or smaller. In the Figure 3.10 are the most common convolution filters used by the most popular Computer Vision libraries, such as OpenCV, Pillow and TensorFlow. For example, the Nearest function performs an average of the surrounding pixels and the Bilinear function performs a weighted average of the neighboring pixels based on how close they are to the sample pixel.

Therefore, if the sampling frequency and convolution filter are known, the pixels in the image to be modified can be found. The rest of the pixels should be kept unchanged. It is worth mentioning that the larger the input image and the smaller the target image (reduced image), the less noticeable the attack will be.

The scaling attack can be summarized in the Figure 3.11.

To perform the attack it is needed a source image (S), of dimensions $m\ x\ n$,

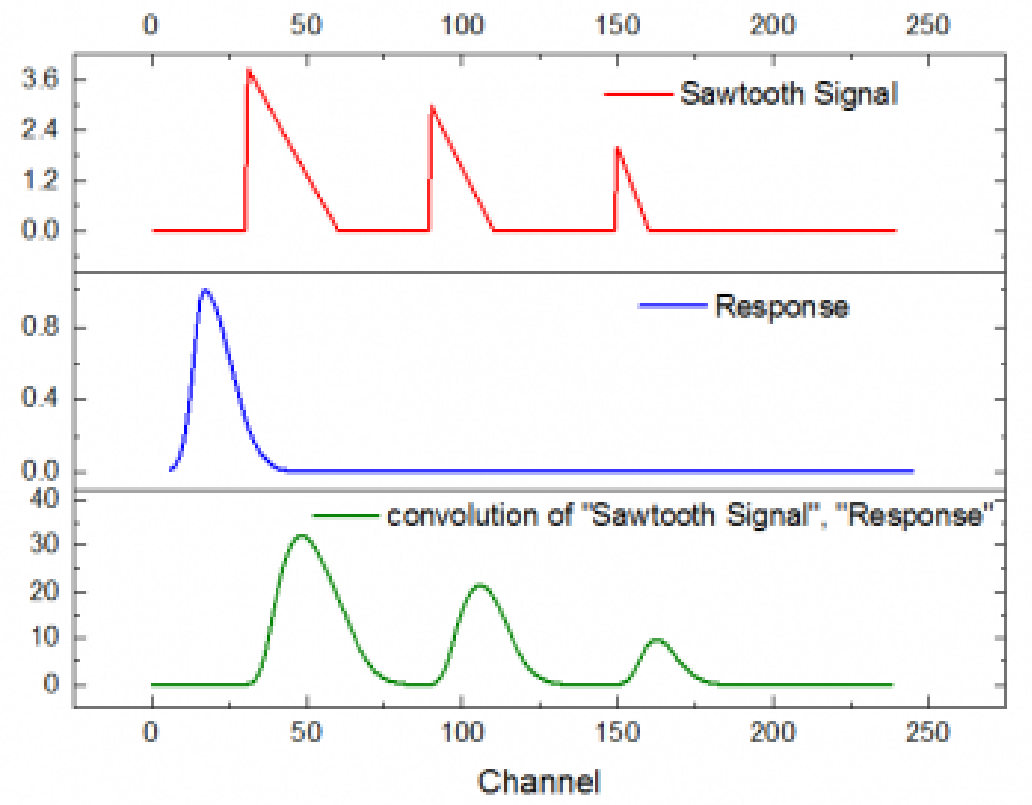

Figure 3.9: Convolution operation. Source

and a target image (T), of dimensions $m'\ x\ n'$ where $m'\ x\ n' < m\ x\ n$. Subsequently, the attack is performed producing image A, of dimensions $m\ x\ n$. When this image is scaled, the image D will be obtained, with dimensions $m'\ x\ n'$, which will be totally different from the source image (S).

The attack can be solved as an optimization problem, where there are two objectives:

- $scale(A) \sim T$: the result of applying the scaling (reduction) operation must be similar to the T image.
- $A \sim S$: the image resulting from the attack has to be as little imperceptible as possible.

Figure 3.12 shows the mathematical formalization of this problem.

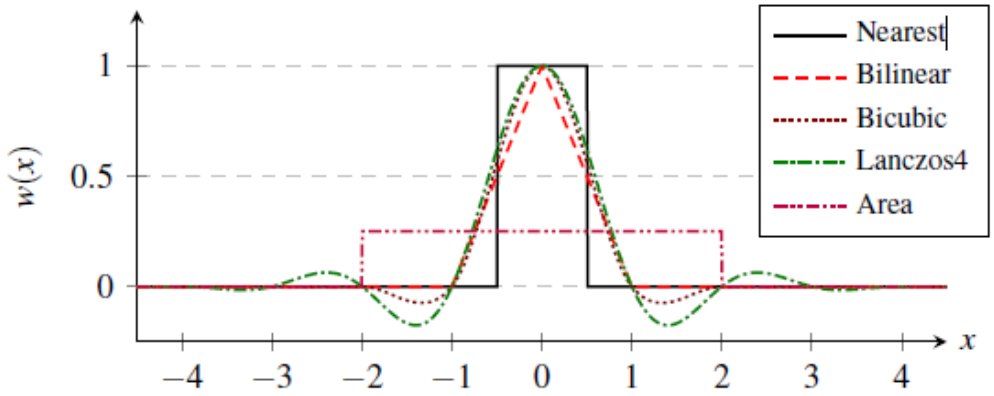

Figure 3.10: Most common convolution filters used by the most popular Computer Vision libraries. Source

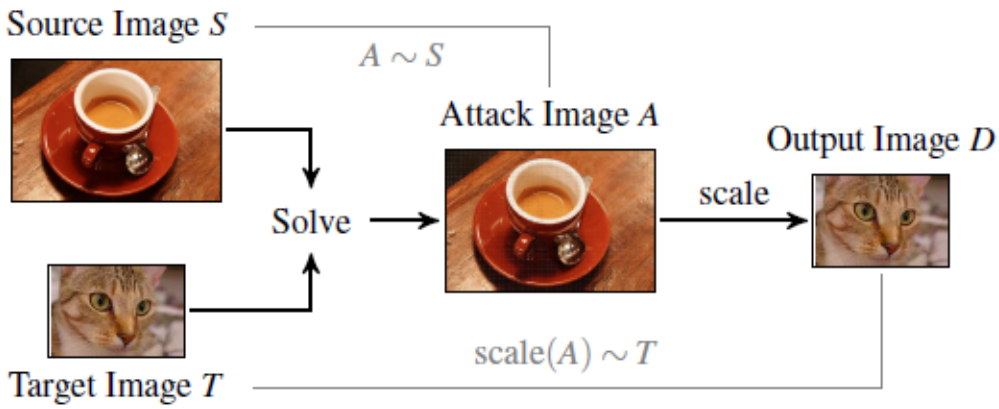

Figure 3.11: Scaling attack summary. Source

where:

- Δ is a disturbance of S
- $\Delta + S = A$

$$\min(\|\Delta\|_2^2)$$
$$\text{s.t.} \quad \|\text{scale}(S+\Delta) - T\|_\infty \leqslant \varepsilon\,.$$

Figure 3.12: Optimization formula to perform a scaling attack

3.2.5 Mitigation

The fewer pixels that are taken into account when scaling, the easier it is to achieve the two objectives, already mentioned, that must be met in the attack. There are two important concepts in scaling attacks:

- Scale ratio: it can be calculated as $scaleRatio = size(originalImg)/size(reducedImg)$. The larger this number, the fewer pixels are taken into account when scaling (reducing) the image. To increase this value it is only necessary to increase the size of the source image (S).
- Kernel width: the smaller this number, the fewer pixels are taken into account when performing the scaling operation.

In the Figure 3.13 it can be seen that as β (scale ratio) is increased and σ (kernel width) is decreased, the scaling attack succeeds. Succeeding means that the two objectives of the attack are met.

If the most used image frameworks are analyzed, such as OpenCV, TensorFlow and Pillow, it can be seen that they use a fixed-size convolution kernel ($\sigma = 1$, $\sigma = 2$ o $\sigma = 4$).

Since for a scaling attack to be successful, β must be large and σ must be small, one of these factors must be avoided. You have no control over the β parameter, as the size of the source image (S) can vary. However, you have control over the σ parameter. Therefore, this parameter has to be adapted based on β. The bigger β the bigger σ will have to be to avoid the scaling attack. What it does this intuitively is that the larger the image, the more pixels are taken into account for the scaling operation and, therefore, the more pixels the attacker will have to modify and the less imperceptible the

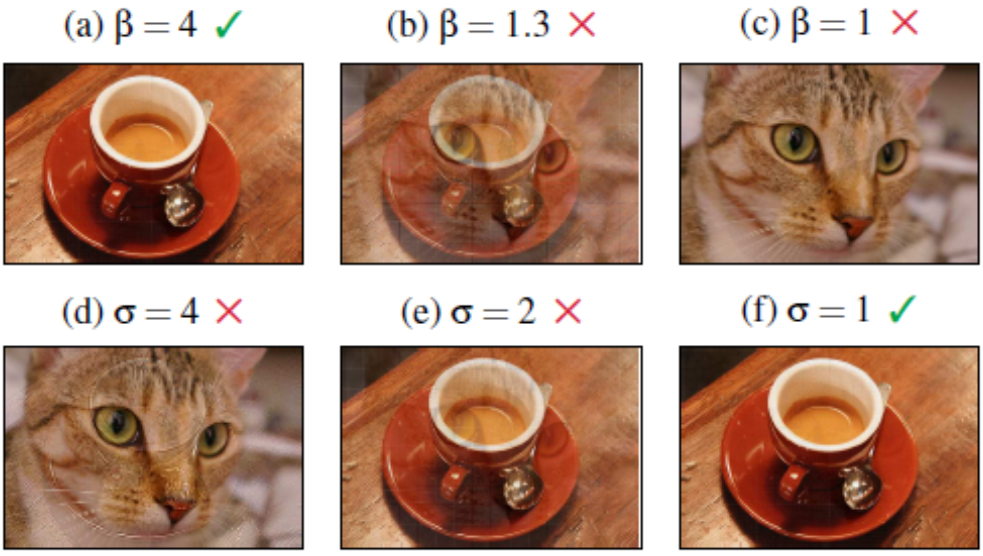

Figure 3.13: Influece of the scaling ratio and kernel size. Source

image will be.

The problem with this method of mitigation against scaling attacks is that the larger σ, the more pixels are taken into account (the attack is less likely to be successful), however the scaling operation is slower. Therefore, it is necessary to reach a compromise between safety and speed.

Additionally, it is recommended not to use the algorithms of the libraries listed in Figure 3.14.

The algorithm that is not vulnerable to scaling attacks is the one that uses the image area. It is not vulnerable, because it does not use fixed σ values, it varies them depending on the image area.

Library	Algorithm	Median		Random	
		Attacks	Unmod.	Attacks	Unmod.
OpenCV	Nearest	99.6%	99.0%	89.3%	89.1%
	Bilinear	100.0%	99.4%	97.7%	98.0%
	Bicubic	100.0%	99.2%	91.4%	93.4%
TF	Nearest	99.6%	99.0%	88.9%	89.1%
	Bilinear	100.0%	98.9%	97.7%	97.7%
	Bicubic	100.0%	99.4%	91.7%	92.0%
Pillow	Nearest	100.0%	99.6%	88.1%	90.4%

Figure 3.14: Algorithms and libraries vulnerable to scaling attacks. Source

Chapter 4

Black box attacks

Like traditional cybersecurity attacks, adversarial attacks can be classified into three types, depending on the attacker's knowledge of the Machine Learning algorithm. In this section we will describe black box attacks.

4.1 Reversing the weights of a model

Today there are many models that are exposed on the Internet in the form of APIs, either for public use, such as Machine Learning as a Service (MLaaS) models, or for private use. If the results that these algorithms return are very detailed, parameters and hyperparameters of the model can be inferred. Sometimes roughly and sometimes exactly (thus cloning the model).

Once the target model has been cloned, different types of white box attack can be performed, such as inferring the training set used to train that model, building a dataset to evade it (for example, using the FGSM technique), etc.

This section details how the weights of a model can be obtained. The same structure will be followed as in the rest of the techniques explained in this guide: objectives, requirements to perform the attack, how to carry it out, how to detect it and how to protect against this type of attack.

4.1.1 Attacker objectives

The attackers want to clone the model with the following objectives:

- Avoid or save economic costs: in some cases of Machine Learning as a Service (MLaaS) it can be cheaper to clone the algorithm and use it locally instead of paying for the use of the service. As an example, one million requests to the Azure Computer Vision API, which is capable of detecting adult content, OCR, celebrities, etc., costs approximately 1500$ (see image below). If a company plans to use this API very intensively, it may be more profitable to clone the model and run it within its own infrastructure. It is worth mentioning that with a million requests it is possible to create a dataset that occupies approximately 3 TB of disk space.

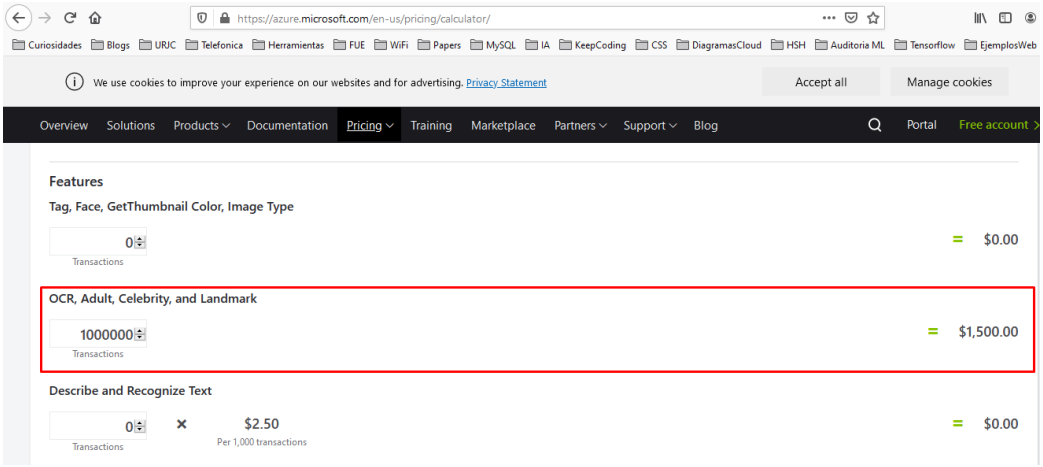

Figure 4.1: Azure pricing calculator

- Evade the model: the attacker wants to create the same model (or a similar one) to be able to analyze it locally and thus be able to perform white box attacks on the target model.

- Obtain information about the training data: there are techniques that, given a model, allow conclusions to be drawn about the training set used. This can be of interest to an attacker when sensitive data has been used to train the model.

4.1.2 Requirements

It is a black box attack, therefore, there are no requirements, although if some parameters of the model are known, it would be easier to carry out this attack. It is worth mentioning that some of the models exposed on the Internet such as Machine Learning as a Service (MLaaS) may have some type of associated documentation where it is possible to find more information about the algorithm they use.

4.1.3 Machine Learning models affected

All algorithms are affected by this attack, however, some are more susceptible than others. The models that can be fully cloned are listed below:

- Linear regression
- Logistic regression
- Softmax

Other models that cannot be fully cloned are listed below, but can be approximated:

- Neural networks
- Decision trees

4.1.4 Attack

There are different attack techniques depending on the model. The attack techniques for the following models are explained below: linear regression, logistic regression, and neural networks.

Linear regression

Linear regression is a mathematical model, often used in Machine Learning applications, to approximate the dependency relationship between a dependent variable, which we will call $\hat{Y}$, and several independent variables, which we will call X_i. This model can be expressed by the following formula:

$$\hat{Y} = W_0 * X_0 + W_1 * X_1 + W_2 * X_2 + ... + W_i * X_i + \beta$$

To simplify the explanation, the formula will be transformed into the following equivalent expression. It is worth mentioning that the objective of this transformation is to eliminate (or transform) the β parameter

$$\hat{Y} = \theta_0 * X_0 + \theta_1 * X_1 + ... + \theta_i * X_i$$

where:

- $X_0 = 1$
- θ_i are the old parameters W_i and β

As its name suggests, linear regression is only made up of linear operations. Therefore, if the input and output of the model X_i y $\hat{Y}$ are known then a linear system of equations can be constructed to find the parameters of the model. The equations of this system would have the following form:

$$\begin{cases} \hat{Y_1} = \theta_0 * X_0 + \theta_1 * X_1 + ... + \theta_i * X_i \\ \hat{Y_2} = \theta_0 * X_0 + \theta_1 * X_1 + ... + \theta_i * X_i \\ ... \\ \hat{Y_i} = \theta_0 * X_0 + \theta_1 * X_1 + ... + \theta_i * X_i \end{cases}$$

For a better understanding of this attack, the following example is presented. Suppose that some service provider, such as Amazon, provides a service that allows predicting or estimating the price of a house (in hundreds of thousands of dollars) from 4 features: number of bedrooms, number of square meters, distance to the supermarket in meters and distance to the health center in meters. See the Figure 4.2.

In this case, it is not necessary to know if the Amazon service uses a linear regression model to estimate the price of houses. You would have to try different models until you find the one that is closest to the results of the Amazon service. In this case, it will be tested with the linear regression model. As the service accepts 4 input variables or a 4-dimensional array (number of bedrooms, number of square meters, distance to the supermarket and distance to the health center) then it can be deduced that the model has 5 parameters and that it is governed by the following formula:

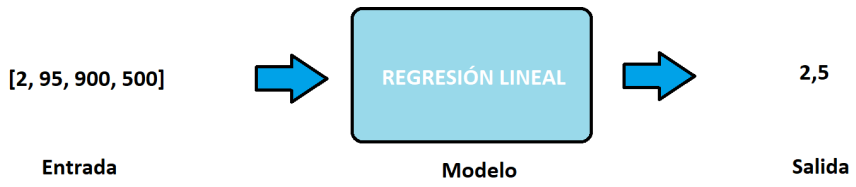

Figure 4.2: Example of linear regression

$$PrecioCasa = \theta_0 * 1 + \theta_1 * NDormitorios + \theta_2 * Nm2 + \theta_3 * DistSupermercado + \theta_4 * DistMedico$$

Given that there are 5 parameters, 5 systems of equations should be proposed and, therefore, it is necessary to execute 5 requests to the Amazon service in order to collect 5 responses. If we send the following data to the Amazon service:

- Input: [2, 95, 500, 200]. Output returned by Amazon: 2.5
- Input: [1. 80, 200, 600]. Output returned by Amazon: 1.9
- Input: [3, 120, 600, 600]. Output returned by Amazon: 2.9
- Input: [4, 180, 1200, 100]. Output returned by Amazon: 4.5
- Input: [2, 120, 150, 150]. Output returned by Amazon: 3.1

We could propose the following system of equations:

$$\begin{cases} 2.5 = \theta_0 * 1 + \theta_1 * 2 + \theta_2 * 95 + \theta_3 * 500 + \theta_4 * 200 \\ 1.9 = \theta_0 * 1 + \theta_1 * 1 + \theta_2 * 80 + \theta_3 * 200 + \theta_4 * 600 \\ 2.9 = \theta_0 * 1 + \theta_1 * 3 + \theta_2 * 120 + \theta_3 * 600 + \theta_4 * 600 \\ 4.5 = \theta_0 * 1 + \theta_1 * 4 + \theta_2 * 180 + \theta_3 * 1200 + \theta_4 * 100 \\ 3.1 = \theta_0 * 1 + \theta_1 * 2 + \theta_2 * 120 + \theta_3 * 150 + \theta_4 * 150 \end{cases}$$

When solving this system of equations, the θ parameters of the Machine Learning algorithm used by the Amazon service would have already been

found.

After solving this example, it is important to highlight the following conclusions about the extraction of parameters from the linear regression model:

- If the input vector has N dimensions, the number of parameters will be N + 1.
- If the number of parameters is N + 1 then a system of equations of N + 1 equations and N + 1 variables must be proposed.
- Solving the system of equations provides the weights of the model, thus cloning it identically.

Logistic regression

Logistic regression is a mathematical model, often used in Machine Learning applications, to predict the result of a categorical variable, which we will call $\hat{Y}$, from independent variables, which we will call X_i. It is governed by the following formula:

$$\hat{Y} = \sigma(W_0 * X_0 + W_1 * X_1 + W_2 * X_2 + ... + W_i * X_i + \beta)$$

where:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Next, the logistic regression formula is simplified. It is intended to remove or transform the β parameter.

$$\hat{Y} = \sigma(\theta_0 * X_0 + \theta_1 * X_1 + ... + \theta_i * X_i)$$

where:

- $X_0 = 1$
- θ_i are the old parameters W_i y β
- $\sigma(z) = \frac{1}{1+e^{-z}}$

As can be seen, the formula is very similar to that of linear regression. Logistic regression uses linear operations except for the σ function. Therefore, if it is capable of eliminating, clearing or substituting this non-linear function, then a system of linear equations can be proposed, as was done in linear regression. To eliminate it, simply apply the inverse of σ to the logistic regression formula, as can be seen below.

$$\sigma^{-1}(\hat{Y}) = \sigma^{-1}(\sigma(\theta_0 * X_0 + \theta_1 * X_1 + ... + \theta_i * X_i))$$

where $\sigma^{-1}(x) = ln(\frac{x}{1-x})$

For a better understanding of this attack, the following example is presented below. Suppose that some financial service provider, such as Banco Santander, provides a service that allows predicting the probability with which a person will be granted a mortgage based on 4 features: mortgage amount in tens of thousands of euros, salary of the person in thousands of euros, age and number of members in the family. An example can be seen in the Figure 4.3.

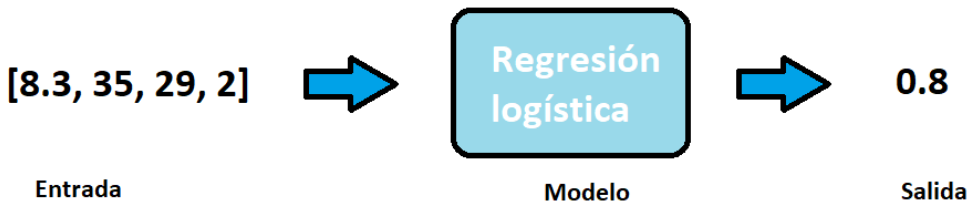

Figure 4.3: Example of logistic regression

In this case, it is not necessary to know whether the service uses a linear regression model to estimate the probability of granting a loan to a person. You would have to try different models until you find the one that is closest to the results of the service. In this case, it will be tested with the logistic regression model. As the service accepts 4 input variables or a 4-dimensional array (mortgage amount, salary, age and number of family members) then it can be deduced that the model has 5 parameters and that it is governed by the following formula:

$$\hat{Y} = \sigma(\theta_0 * 1 + \theta_1 * CuantiaHipoteca + \theta_2 * Sueldo + \theta_3 * Edad\theta_4 * Miembros)$$

Since there are 5 parameters, 5 systems of equations would have to be proposed and, therefore, it is necessary to launch 5 requests to the Banco Santander service in order to collect 5 responses. If the following data is sent to the service:

- Input: [12, 20, 23, 1]. Output returned by the service: 0.2
- Input: [8.2. 29, 29, 2]. Output returned by the service: 0.5
- Input: [15.3, 25, 27, 1]. Output returned by the service: 0.1
- Input: [6.5, 40, 38, 4]. Output returned by the service: 0.83
- Input: [5.4, 50, 36, 4]. Output returned by the service: 0.91

The following system of equations could be proposed:

$$\begin{cases} \sigma^{-1}(0.2) = \theta_0 * 1 + \theta_1 * 12 + \theta_2 * 20 + \theta_3 * 23 + \theta_4 * 1 \\ \sigma^{-1}(0.5) = \theta_0 * 1 + \theta_1 * 8.2 + \theta_2 * 29 + \theta_3 * 29 + \theta_4 * 2 \\ \sigma^{-1}(0.1) = \theta_0 * 1 + \theta_1 * 15.3 + \theta_2 * 25 + \theta_3 * 27 + \theta_4 * 1 \\ \sigma^{-1}(0.83) = \theta_0 * 1 + \theta_1 * 6.5 + \theta_2 * 40 + \theta_3 * 38 + \theta_4 * 4 \\ \sigma^{-1}(0.91) = \theta_0 * 1 + \theta_1 * 5.4 + \theta_2 * 50 + \theta_3 * 36 + \theta_4 * 4 \end{cases}$$

When solving this system of equations, the parameters θ of the Machine Learning algorithm used by the Banco Santander service would have already been found.

After solving this example, it is important to highlight the following conclusions about the extraction of parameters from the logistic regression model:

- If the input vector has N dimensions, the number of parameters will be N + 1.
- If the number of parameters is N + 1 then a system of equations of N + 1 equations and N + 1 variables must be proposed.
- Solving the system of equations provides the weights of the model, thus cloning it identically.

Neural networks

In the case of deep neural networks, it is computationally and analytically complicated to obtain, exactly, the model weights, since most of them contain non-linear equations and cannot be eliminated before proposing the system of equations, as we did in logistic regression. For example, a 3-layer neural network is governed by the following equation. Note that a layer can be made up of several neurons.

$$\hat{Y} = \sigma(W_3 * \sigma(W_2 * \sigma(W_1 * X + \beta_1) + \beta_2) + \beta_3)$$

where:

- X is the matrix of inputs to the model
- W_i is the weight matrix of layer i
- β_i are the biases of neurons of layer i
- σ is the activation function. It doesn't have to be a sigmoid.

Even if the inverse of the activation function were applied, as it was done in the logistic regression, it would not be possible to propose a system of linear equations. Next, an attempt is made to clear the parameters of the model to show that it is not possible to propose a system of linear equations:

1. The equation is simplified.

$$\hat{Y} = \sigma(\theta_3 * \sigma(\theta_2 * \sigma(\theta_1 * X)))$$

2. The inverse applies to the activation function.

$$\sigma^{-1}(\hat{Y}) = \theta_3 * \sigma(\theta_2 * \sigma(\theta_1 * X))$$

3. The inverse is applied again on the activation function

$$\sigma^{-1}(\sigma^{-1}(\hat{Y})) = \sigma^{-1}(\theta_3) * \theta_2 * \sigma(\theta_1 * X)$$

4. The inverse is applied again on the activation function

$$\sigma^{-1}(\sigma^{-1}(\sigma^{-1}(\hat{Y}))) = \sigma^{-1}(\sigma^{-1}(\theta_3)) * \sigma^{-1}(\theta_2) * \theta_1 * X$$

As can be seen, the final resulting equation is a non-linear equation. To solve it, it would be necessary to know the activation functions, the number of neurons per layer and the depth of the neural network. Deep neural networks tend to be made up of millions of parameters, so proposing a system of equations of millions of equations with millions of variables is computationally expensive, although in some cases it may be feasible.

Due to the variability of the hyperparameters and the computational cost, it is more appropriate to approximate a model similar to the objective model (victim) by gradient descent. To do this, it is necessary to obtain N samples (the more the better) from the victim model, that is, N pseudo-random samples have to be generated and sent to the victim model. The output of the model must be stored. Subsequently, a neural network will be trained with these inputs and outputs.

When proposing this attack, it is convenient to take into account the following aspects:

- Number of model inputs: if the victim model receives color images with a maximum size of 1200 x 1200 it can be assumed that the model contains 1200 x 1200 x 3 input neurons. It should also be noted that the input data is very likely to be pre-processed before providing it to the model (e.g. data cleaning, image resizing, etc).
- Number of outputs of the model: if the victim model produces a 10-dimensional output, it is very likely that the model has 10 output neurons. Although it must also be taken into account that the output of the model can be preprocessed before being returned.
- Model composition: in complex tasks, such as image recognition, the Machine Learning application can be composed of several models, which makes it even more difficult to approximate.
- Obtaining an approximate model can facilitate white box attacks on the victim or target model.

4.1.5 Mitigation

The most effective way to mitigate this attack is to reveal as little information about the model as possible.

For example, in the attack against the logistic regression model it was possible to find the identical weights of the model because it returned the exact probability. The most effective solution for this example would be to return an answer of "YES" or "NO" depending on the output probability of the model. An example could be:

$$\begin{cases} \text{YES} \quad if \quad \hat{Y} >= 0.5 \\ \text{NO} \quad if \quad \hat{Y} < 0.5 \end{cases}$$

4.1.6 Detection

There are no ways to detect this attack.

Chapter 5

Conclution

As has been proven so far, algorithms and, therefore, Machine Learning applications are also exposed to security vulnerabilities. Developers and Artificial Intelligence specialists should be aware of these attacks to mitigate them in the early stages of application design and development. Security specialists should also know them to audit and improve their security. In this framework, several white box attacks have been seen, in which images are modified to fool a Machine Learning model. Black box techniques have also been seen, in which parameters of a model are cloned or approximated using the inputs and outputs of the model. There are also other attack techniques that allow transferring FGSM attacks from one model to another, facilitating black box attacks. The main idea of this Framework is to collect, in a collaborative way, the largest amount of possible attack and mitigation techniques to create a knowledge base and, therefore, facilitate protection and audit of Machine Learning algorithms. It must be remembered that these algorithms are very present in the lives of millions of people and every day they will be more.

It is worth mentioning that we are working on adding more attack techniques to the Framework. Some of them are: variations of FGSM attacks (iterative FGSM and Carlini and Wagner attack), transfer FGSM attacks from one model to another one, determining if a data set has been used to train the algorithm, etc.

Bibliography

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

[2] Erwin Quiring, David Klein, Daniel Arp, Martin Johns, and Konrad Rieck. Adversarial preprocessing: Understanding and preventing image-scaling attacks in machine learning. 2020.

[3] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. 2016.