WHITEPAPER

AN OVERVIEW OF ADVERSARY ATTACKS

S4AllCities
HORIZON 2020
883522

This work was performed within the S4AllCities Project, with the support of the European Commission and the Horizon 2020 Programme, under Grant Agreement No. 883522.

List of Acronyms

Acronym	Meaning
SVM	Support Vector Machine
ANN	Artificial Neural Network
DNN	Deep Neural Network
FC	Fully Connected
CNN	Convolutional Neural Network
GNN	Graph Neural Network
JSMA	Jacobian-based Saliency Map Attack
BIM	Basic Iterative Method
FGSM	Fast Gradient Sign Method
PGD	Projected Gradient Descent
AC-GAN	Auxiliary Classifier Generative Adversarial Network
GAN	Generative Adversarial Network
RNN	Recursive Neural Network
MMD	Maximum Mean Discrepancy
PCA	Principal Component Analysis

Table 1 - List of acronyms.

An Overview of Adversarial Attacks

Jose Maria Miranda Orte, ATOS

1 Introduction

This whitepaper reviews the state of the art regarding adversarial attacks against deep learning algorithms dealing with image-classification. A review of the overall taxonomy of machine learning models and adversarial attacks is presented, followed by examples of attack procedures and potential defense strategies. The whitepaper is based on the S4AllCities project deliverable D3.3 "Visual analytics adversary attacks analysis".

2 Taxonomy of machine learning models and adversarial attacks

The use of inference systems based on neural networks or other architectures is clearly increasing nowadays. These types of algorithms, which classify input data on the basis of prior training, are responsible for more and more important tasks whose integrity would represent a serious risk for the supra-systems in which they are included. The attacks they are subject to are known as "adversarial attacks". In general, they can be considered as manipulated inputs or malicious training data that cause the system to behave in a certain way, different from the one it is designed to behave. This chapter will describe the architectures most susceptible to this type of attack and will define a threat model, in which the different possible scenarios will be explored.

2.1 Victim machine learning models, some definitions

The architectures most susceptible to attack are:

- **Support vector machines**: SVMs are a set of supervised learning algorithms designed to address classification, regression, and outlier identification tasks. Starting from a set of labelled samples, we will train a SVM that will allow us to build a model that predicts to which class a new input belongs. Intuitively, this model represents the sample points in space and searches for hyperplanes that separate the classes to which they belong in an optimal way, achieving a maximum separation, and larger margin between these classes. The set of points that establish the two lines parallel to the hyperplane and between which we choose the greatest margin of separation are called support vectors.
- **Artificial neural networks (ANN) and deep neural networks (DNN)**: Artificial neural networks (ANN) is an algorithm based on a collection of perceptron's, called neurons, distributed in different layers. The objective of each neuron is to map an output to various inputs through an activation function and weights. These weights are defined, in the multi-layer feed forward procedure, during the training phase by means of the back-propagation algorithm. In general, DNN refers to an ANN composed of a certain number of hidden layers. Each layer will be in charge by extracting characteristics from the input data or outputs of the previous layer and by progressively increasing the level of abstraction. It is worth mentioning that the architecture of DNNs makes them particularly vulnerable to adversarial attacks through the generation of malicious inputs. Depending on the defined architecture, they can be classified as follows:
    - **Fully connected neural networks**: (FC) Fully connected neural networks are composed of layers of neurons. In this type of networks, all neurons take the input from the previous layer and process it with the activation function and send the output to the next layer. The first layer is the input layer whose input is x, and the output of the last layer is the score $F(x)$, usually with *softmax*[1] as activation function. Inputs and outputs are related to each other mean by the following equation:

$$z^{(0)} = x; \quad z^{(l+1)} = \sigma(W^l z^l + b^l)$$

      These kind of algorithms use gradient descent and back-propagation for learning proposes; for that, it is necessary to calculate the sensitivity of the score $F(x;\theta)$ with respect to each

[1] The softmax function or normalised exponential function is a generalization of the logistic function normally used in the last layer of deep neural network-based architectures that is used to transform a K-dimensional z-vector of arbitrary real values into a vector of the same dimension whose values are bounded to real values between zero and one.

model parameter θ, as $\frac{\partial(F(x;\theta))}{\partial\theta}$. In adversarial learning, back-propagation is used also for calculating $\frac{\partial(F(x;\theta))}{\partial x}$, that is the output's response to a change in the input. This term is very important in the studies to craft adversarial examples.

- **Convolutional neural networks**: CNN is very popular architecture in computer vision tasks. Using convolutional filters, CNNs extract local features from the images and learn to locate objects within them. From a formal point of view, CNNs are a sparse version of fully connected neural networks with most of weights between layers fixed to zero. As FC networks, also use gradient descent algorithm to training and, therefore, are also susceptible to attack by an adversarial input.
- **Graph neural networks**: GNNs, a type of deep neural network that operates with a graph-based data structure.

2.2 Adversarial threat model

The security of a deep learning model will be assessed by taking into account the attacker's objectives and capabilities to access the model. We start by defining the threat surface, i.e., how and where the attacker will carry out his attack, compromising the integrity of the system.

2.2.1 The machine learning attack surface

The analysis process carried out by a machine learning system can be viewed as a data pipeline. During the testing period this process may consist of several stages: the data acquisition from a repository or from the sensors themselves, the data pre-processing in order to fit with a format compatible with the model (JPEG to matrix for example), the inference made by the model to generate an output and finally, the decision making made on the basis of this model output. For example, if we consider a video surveillance model, these stages would be described as: acquiring the frames in a video stream, reading the pixels and injecting them into the model, extracting the features contained in the image, and generating an output that can be the object recognized in the image. The **attack surface** refers to the location in this pipeline chosen by the attacker to carry out the attack. The attack may be aimed at manipulating the input data set, by which the inference model is extracted, or the model output itself. We can classify attacks according to the chosen attack surface as follows:

- **Evasion attack**: This is the most common adversarial attack, in which an attempt is made to deceive the system by making malicious samples or patterns during the model testing period. No modification of the dataset is made in this type of attack.
- **Poisoning attack**: Known as training data contamination, it takes place during the training period of the deep learning model. The attacker manipulates certain data that conforms the dataset so that the model obtained does not work as expected.
- **Exploratory attack**: This attack aims to find out as much information as possible from the inference model as if it were a black box. Nothing is modified in the dataset.

2.2.2 Knowledge of the system or adversary's capabilities

The capability of an adversary generally refers to the amount of information it has about the system it is trying to attack. Continuing with the example of the video surveillance system used for object detection, we can consider an evasion attack during the testing phase of the system. In this case we could distinguish between two attackers: a stronger one who has access to the model architecture and can use it to analyze different images, or a weaker one who only has access to the images that are inputs to the model during this period. In both cases, the attack surface is the same, but the capabilities of the adversary attackers are clearly different. The capabilities of the attacker can be classified according to whether the attacker aims to attack during the training or inference phase.

2.2.2.1 Training phase capabilities

The attacks on these systems in the training phase are based on the direct modification of part of the samples that are part of the training dataset. Depending on the capabilities of the adversary, they can be classified as follows:

- **Data injection**: The attacker, in this case, does not have access to the training dataset, neither to the model being trained, but has the possibility to augment the training dataset with new data containing malicious information, that will ultimately alter the correct functioning of the trained model.
- **Data modification**: In this case, the attacker still does not have access to the model being trained, but, unlike in the previous case, they have access to the data that conform the training dataset. Through an attack of this type, the adversary will modify part of the data in the dataset before it is used to train the model. Once the model has been trained with this modified data, its functionality and capabilities will be altered.
- **Logic corruption**: In this case, the attacker has access to the learning algorithm used to train the model, and therefore has the ability to alter it at will. This is undoubtedly the most difficult method to defend against as it implies that the attacker has control of the model itself.

2.2.2.2 Testing phase capabilities

Adversarial attacks during the testing phase do not alter in any way the model itself but will force it to give erroneous or less accurate inferences than the model that operates in a nominal situation. The effectiveness of the attack depends on the degree of knowledge the adversary has about the architecture of the model he wants to attack and can be classified as a "**black-box"**, "**white-box"** or "**grey-box**" attack. Before defining in more detail what each type of attack consists of, it is worth defining the training procedure of a deep learning system such as the ones considered. The training process of a deep learning model can be defined by the following expression:

$$\theta \leftarrow train(f, X, y, \gamma)$$

Where:

- f represents the model to be trained
- (X, y) usual input pair in a supervised training procedure from the data distribution μ
- With a randomized training procedure train having randomness γ, for example, random initial weights values.
- θ are the model parameters after the learning procedure.

Based on this definition of training procedure we can find the following classification of attacks according to the capability of the attacker in this phase:

- **White-box attacks**: This is possibly the strongest type of attack that can be carried out due to the amount of information available to the adversary. In this case, the attackers are fully aware of the model used (f), know its architecture, number and type of layers that compose it. They also have information about the algorithm used for training ($train$), for example, gradient-descent optimization, as well as access to the data distribution used for training, (μ). They also know the parameters that form the set of weights once the model has been trained (θ). The attacker uses all this information to analyze the latent space of the model and find the place where it is most vulnerable. The model is then exploited by an adversarial input crafted using all this available information.
- **Black-box attacks**: In this case, by contrast to the previous one, the adversary does not know the model or any details about the training procedure, dataset, or other relevant information. In most cases the attack occurs because the inference model is exposed as a service and the attacker generates a sequence of specially designed inputs in order to extract relevant information by analyzing the outputs of the model. Black box attacks can be classified as follows:
    - **Non-adaptive black-box attack**: In an attack of this type on a model (f), the adversary has knowledge of the data distribution used for training (μ). An example of such a scenario could be where a certain image classification service is exposed, and the owner gives certain details about the dataset used to do the training and also the dataset is public. Under these circumstances, the attacker can train locally a model (f') using a training method ($train'$) and the same data distribution (μ) as the target model. Through this procedure, the attacker would have an approximate version of the target model, that could be used to craft adversarial examples with which to attack the original model as if it were a white box attack.

- **Adaptive black-box attack**: In this adaptive black-box attack, the attacker has no knowledge about the model, the training procedure or the distribution of training data, but can perform queries on the target model as it is exposed as a service, for example. The attacker then chooses a model (f') and a training method $(train')$ and trains a surrogate model using the tuples obtained from querying the target model (x, y). To obtain these tuples that will form the target dataset, the attacker adaptively designs the *x* inputs and obtains the *y* outputs from the inferences made on the target model. The adversary will now use this surrogate model to generate adversarial examples to attack the target model and cause it to malfunction.
- **Strict black-box attack**: In this type of attack, the adversary does not have the possibility to design the input by which to query the target model and can only collect the tuples of inputs and outputs that the model processes. It is therefore a black box model in the strict sense and in order to be effective it may be necessary to collect a large number of these input-output pairs.

2.2.3 Adversarial goals

One of the main objectives of the adversary will be to try to obtain a malfunctioning of the classification model, for example and in general a target inference model, by giving as input x^*. Depending on the adverse effect the attack has on the inference model, we can classify the attacks according to the attacker's goals as follows:

- **Confidence reduction**: The attack attempts to reduce confidence in the identification of a given class. Through such an attack, the value of the confidence in the classification of the input by the inference model is substantially reduced to less than the threshold selected by the user of the model. The input is then no longer identified by the model.
- **Miss-classification**: This attack is intended to achieve that, for a given input image, the classifier assigns it an output class different from the one it should, a priori any class different from the one the input corresponds to.
- **Targeted miss-classification**: In this case the adversary will ensure that for any input image the classifier always outputs a target class chosen by him.
- **Source/target miss-classification**: Now, the attack will try to map a given input image to a non-matching output and will always assign it to this specific target class. For example, any image of a dog will be classified as a cat.

3 Generating Adversarial Examples

In this section we will focus on describing the most relevant methods of adversarial attack, generating adversarial examples on a neural network aiming at image classification or object detection. It will be assumed from now on that the most common architectures will be fully connected neural networks and convolutional neural networks [Krizhevsky et al. 2012]. By their nature we will focus on evasion or poisoning attack methods regardless of the phase where the attack takes place. During the description of the different algorithms, the following general notation will be followed:

- $\boldsymbol{x}$ Data sample to be attacked;
- $\boldsymbol{x'}$ Modified data sample;
- $\boldsymbol{\delta}$ Perturbation;
- $\boldsymbol{B_{\epsilon}(x)}$ lp-distance neighbour ball around x with radius;
- $\boldsymbol{D}$ Natural data distribution;
- $\|\cdot\|_{\boldsymbol{p}}$ lp norm;
- $\boldsymbol{y}$ label corresponding to sample $\boldsymbol{x}$;
- $\boldsymbol{t}$ Target label;
- $\boldsymbol{\mathcal{Y}}$ Group of possible labels, assuming that there are m labels;
- $\boldsymbol{C}$ Classifier whose output is the label y $C(x) = y$;
- $\boldsymbol{F}$ Deep Neural Network score vector output $F(x)\epsilon[0,1]^m$;

- $\boldsymbol{Z}$ Last layer output before SoftMax $F(x) = softmax(Z(x))$;
- $\boldsymbol{\sigma}$ Activation function;
- $\boldsymbol{\theta}$ Model parameters;
- $\boldsymbol{\mathcal{L}}$ Loss function in training phase.

In the following, the methods of attack are presented by generating adversarial examples according to the capabilities of the attacker.

3.1 White-box Attacks

In a white-box attack, as described in previous sections, the attacker has access to the model $\boldsymbol{F}$ of the classifier $\boldsymbol{C}$ and to the tuple (x, y), which represent the input and the label of the assigned class. The goal would be to synthesise a fake image $\boldsymbol{x'}$ sufficiently similar to $\boldsymbol{x}$ so that it can fool the classifier $\boldsymbol{C}$ into inferring an erroneous result. Formally it can be expressed by

$$x' satisfying \ \|x' - x\| \leq \epsilon$$
$$C(x') = t \neq y.$$

We will now present several different approaches in which this methodology will be implemented:

3.1.1 Biggio's attack

This method is described in Biggio et al [2013]. In it, adversarial examples of the MNIST dataset of handwritten digits are crafted by targeting classifiers such as SVMs and fully connected three-layer networks. In this case, a fake image is crafted to deceive the classifier by optimising the discriminant function.

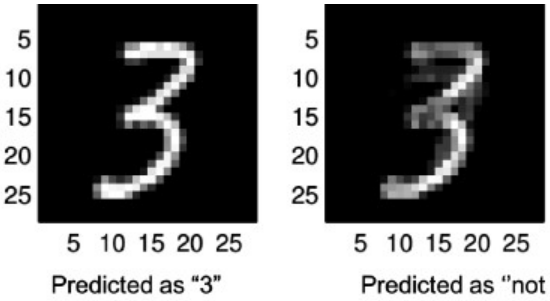

Figure 1: Biggio's attack on SVM (Image credit: (Biggio et al.))

If the model is an SVM, its discriminant function would be $g(x) = \langle x, y \rangle + b$, which will classify the input as a three if the value of the function is greater than zero $g(x) > 0$ and as not a three if its value is less than zero $g(x) \leq 0$. By means of this attack, the adversary will manage to craft a new image x', which by keeping the difference $\|x' - x\|$ small, manages to minimise the value of the discriminant $g(x')$, even transforming it into a negative value. In this way x' will deceive the classifier by maintaining values very close to the original image x, which was perfectly identified by the classifier.

3.1.2 Szegedy's L-BFGS attack

Szegedy et al. [2013], summarize the first attack to an image classifier based on deep neural network. The formulation presented in this paper addresses the problem as an optimisation problem in which it searches for the adversarial image x' most similar to the original image x (minimally distorted) that achieves the following objective:

$$min \|x - x'\|_2^2$$
$$subject\ to\ C(x') = t\ and\ x' \in [0,1]^m$$

The solution of the problem is approached by combining both conditions by means of the loss function as follows:

$$\min c \, \|x - x'\|_2^2 + \boldsymbol{\mathcal{L}}(\theta, x', t)$$
$$subject\ to\ x' \in [0,1]^m$$

In this combined optimisation function, the first term imposes that the new image x'created to deceive the classifier is similar to the original ones x, while the second term forces the algorithm to find an x' that

obtains a very small value of the loss function so that the classifier will assign the target class t to the maliciously crafted input. To solve this optimization problem, they implement the L-BFGS algorithm[2].

3.1.3 Fast gradient sign method

Goodfellow et al. [2014b] introduce one-step method to fast generate an adversarial example. Adversarial images are formed according to the following expressions, depending on whether we want the adversarial example to be classified as belonging to a certain class or not.

$$x' = x + \epsilon sign\big(\nabla_x \mathcal{L}(\theta, x, y)\big)\ non - targeted$$

$$x' = x - \epsilon sign\big(\nabla_x \mathcal{L}(\theta, x, t)\big)\ target\ on\ t$$

In the case where we want the adversarial image to belong to a certain class, the second equation can be interpreted as a one-step of gradient descent to solve the problem:

$$\text{minimize } \mathcal{L}(\theta, x', t)$$

$$\text{subject to } \|x' - x\|_\infty \leq \epsilon \ and\ x' \in [0,1]^m$$

In this case, the objective function searches the adversarial image that minimises the value of the loss function for a target class t in x's ϵ-neighbour ball which is the place where the model $\boldsymbol{F}$ is most likely to infer that the input corresponds to target class t. In this way an image can be crafted that is able to deceive the classifier in one step. This method is faster than the one described in the previous section as it only involves the computation of one back-propagation step. This feature makes it the ideal method for generating datasets of adversarial images due to the speed of image generation.

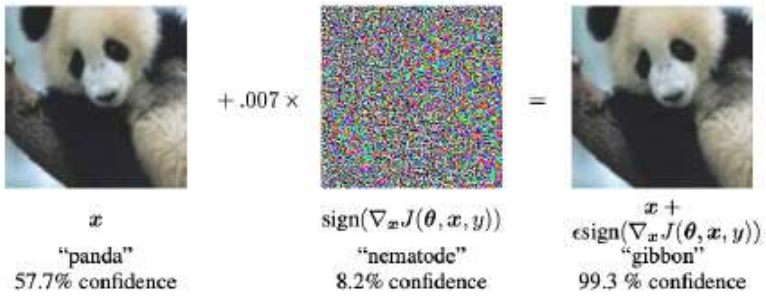

Figure 2: Fast gradient sign attack example (Image credit: (Goodfellow et al.))

3.1.4 Deep fool

This methodology presented for the first time in Moosavi-Dezfooli et al. [2016] introduces the concept of the decision boundary of a model $\boldsymbol{F}$ around a point x, that represent an image. By this procedure, an attempt is made to find a path through which the input x varies until it crosses the decision boundary.

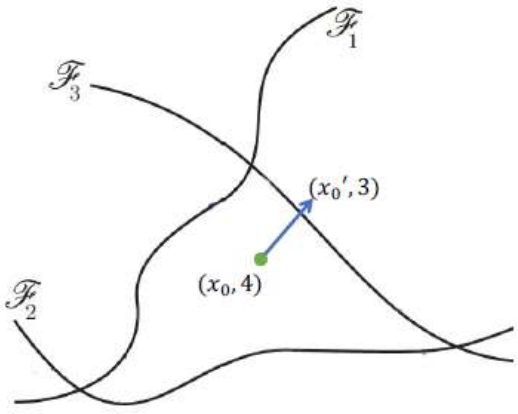

Figure 3: Decision boundaries representation (Image credit: [Moosavi-Dezfooli et al. 2016])

As described in the previous figure, let's imagine that the attack is initiated on the image x_0, which belongs to class 4 and we want to modify it so that we obtain x'_0 which will now be classified as 3, in this case the decision boundary is could be described as:

$$\mathcal{F}_3 = \{x: F(x)_4 - F(x)_3 = 0\}$$

To carry out the attack, the expression of the hyperplane defining the boundary is linearised by means of Taylor expansion, resulting in an expression of this type:

$$\mathcal{F}'_3 = \{x: f(x) \approx f(x_0) + \langle \nabla_x f(x_0)(x - x_0) \rangle = 0\}$$

Now calculating the orthogonal vector w from x_0 to the plane $\mathcal{F}'_3$. By moving in the direction of the orthogonal vector the algorithm will be able to find the necessary perturbations which added to x_0 from the location at x'_0 will be classified as 3 by the system. Moosavi-Dezfooli [2016] work showed that in the common DNN architecture for image classification, most of the test cases were very close to the boundaries. This is why a large majority of test cases can be attacked by small perturbations of norm less than 0.1. This fact shows how sensitive these algorithms are to small perturbations.

3.1.5 Jacobian-based saliency map attack

Papernot et al. [2016a] introduced Jacobian-based Saliency Map Attack (JSMA) this method, based on the calculation of the Jacobian matrix of the $\mathcal{F}$ score function, consists of deceiving the image classifier by altering the value of certain pixels to their saturation values. In this case the Jacobian of the function will be used iteratively, by changing the value of a pixel, at the end the input x' will be classified as belonging to the target class t.

$$J_F(x) = \frac{\partial F(x)}{\partial x} = \left\{\frac{\partial F_j(x)}{\partial x_i}\right\}_{ixj}$$

Manipulating the pixel x_i will increase or decrease the value of $\mathcal{F}$ causing the model to assign a higher score to the target class t for the modified input x'.

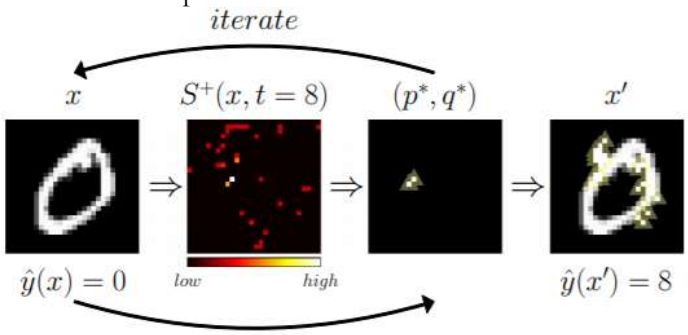

Figure 4: Jacobian-based Saliency Map Attack

3.1.6 Basic Iterative Method (BIM) / Projected Gradient Descent (PGD) attack

Basic Iterative Method (BIM) is the iterative version of the Fast Gradient Sign Method (FGSM), it was first presented by Kurakin et al. [2016b], it is also known as Projected Gradient Descent (PGD) in Madry et al. [2017]. This is a method for crafting the adversarial image x' iteratively without setting the target class t. Following the scheme below:

$$x_0 = x$$
$$x^{t+1} = Funct_{x,\epsilon}(x^t + \alpha sign(\nabla_x \mathcal{L}(\theta, x^t, y)))$$

Where $Funct_{x,\epsilon}$ is the function to project its argument to the surface of ϵ-neighbour ball $B_\epsilon(x)$: $\{x': \|x' - x\|_\infty \leq \epsilon\}$, and α the step size.

By this process, the points x' that maximise the value of the loss function in the ball neighborhood of the original image x are found. When the intensity of perturbation needed to find the adversarial image is limited, this procedure will find the most aggressive adversarial examples with the highest probability of success, known as the "most adversarial" examples.

3.1.7 Carlini & Wagner's attack

Carlini and Wagner [2017b] is the first attack that appears as a response to a defence strategy presented in Papernot et al. [2016b] and that it was effective against the FGSM and L-BFGS attacks seen above. This method involves a modification to the approach made in section 3.1.2 by solving the following equations instead:

$$minimize \ \|x - x'\|_2^2 + \mathrm{c} \cdot \mathrm{f}(\mathrm{x}', \mathrm{t})$$
$$subject\ to\ x' \in [0,1]^m$$

Where f is defined as $f(x', t) = (max_{i \neq t} Z(x')_i - Z(x')_t)^+$. By minimising $f(x', t)$ the algorithm finds an x' that reaches the highest value for the target class t, so the classifier will be deceived into inferring that x' belongs to t. Moreover, c can be chosen in such a way that the differences between x and x' are minimised.

As in the L-BFGS method, here we can consider that $f(x, y)$ is a loss function, but in this case it penalises situations in which the model assigns a larger value to class i for input x than to the corresponding class y to which it belongs, for this reason we can say that this function behaves as a margin loss function. The difference with the L-BFGS attack is therefore that Carlini and Wagner [2017b] use margin loss $f(x, t)$ instead of cross entropy loss $\mathcal{L}(\mathrm{x}, \mathrm{t})$. The advantage of this method over L-BFGS is that when the adversary image satisfies $C(x') = t$ the value of margin loss is zero, $f(x', t)$, so that the image x' is the least distorted image with respect to the original x.

3.1.8 Ground truth attack

The adversarial attacks and the associated defence methodologies were constantly improving and trying to outdo each other. The interest in checking where the limit was, if it existed, was more than evident. For this reason, work began to emerge, such as that presented by Carlini et al. [2017a] in which the search for the adversary image that theoretically presents the least distortion with respect to the target input is proposed.

This attack is based on algorithms for verifying properties of neural networks such as the one presented by Katz et al. [2017], based on Reluplex algorithm.

In this case the parameters of the model F and the (x, y) pairs are encoded as subjects of a linear programming system in order to later solve the system by checking if there is an x' in a ϵ-neighbor ball of x that is able to deceive the system. If the answer is yes, we can continue to reduce ϵ until we reach a point where there is no such adversarial image. The last one we have found capable of deceiving the system is called "ground truth adversarial example" and it is demonstrated that this will be the image with the least differences from the original that exists.

This attack and the work on which it is based were the first grounded development that addressed the calculation of the robustness (minimum perturbation) of classification systems.

3.1.9 Other l_p attacks

If we look at the methods discussed so far, they were all based on the search for adversarial examples on the basis of satisfying a certain constraint on the value of the norm of the differences between x' and x. In all of them the l_2 or $l_\infty$was used, now we will focus on attack methods based on minimising another type of norm, in general l_p:

1. **One-pixel attack**: Su et al. [2019] develop their work in a similar way to that set out in Szegedy's L-BFGS attack but based on l_0 norm. When limiting the value of the norm l_0 on the difference between the target image and the adversary image, the number of pixels that can be modified is also limited. This paper presents remarkable results showing that on the CIFAR10 dataset and a well-trained model (85.5% accuracy) based on the VGG16 architecture, more than 63% of the test data can be attacked by changing the value of a single pixel in each image. This fact highlights the vulnerability of such systems.
2. **Elastic-net attack (EAD):** Chen et al. [2018] takes a similar approach to Szegedy's L-BFGS attack but based on modifications limited to a combination of l_1 and l_2 norm. Later we will look several works demonstrating the effectiveness of various defences against l_2 and l_∞-based attacks that are not effective against l_1.

3.1.10 Universal attack

All attacks described so far refer to a specific target image x and in general, to a target class t. Moosavi-Dezfooli et al. [2017a] presented for the first time an attack that fooled the classifier model in almost all images of the test group. They successfully crafting a δ perturbation that satisfied the following expressions:

$$\|\delta\|_p \leq \epsilon$$

$$\mathbb{P}_{x\sim D(x)}\big(C(x+\delta) \neq C(x)\big) \leq 1-\sigma$$

Formulation that evidences that it is pretended to find a perturbation that gives misclassifications in most of the samples. The experiments shown in this paper demonstrate that they were able to find a delta perturbation capable of attacking 85.4% of the test samples using the ILSVRC 2012 dataset and the ResNet-152 model as classifier. The existence, demonstrated from this work, of such universal attacks shows that the weakness of these systems is inherent in their architecture and does not depend on the input data.

3.1.11 Spatially transformed attack

Traditionally, adversarial attacks aim to deceive classifier systems by crafting non-obvious patterns that, when added to the original image, accomplish their mission. The more similar the altered image (adversarial image) is to the original, the better and stronger the attack is considered to be, and therefore the more sophisticated the method of defence will have to be. The work of Xiao et al. [2018b] changed the paradigm, in which they proposed a new method of crafting adversarial images in which the modifications are not made at the level of varying the colour intensity of each pixel. Here it is proposed the possibility of achieving the same objective by performing transformations such as translations, rotations or deformations of part of the image and its local characteristics. As can be seen in the image, minimal modifications are made to the target image, just enough to deceive the classifier and go unnoticed by the human eye.

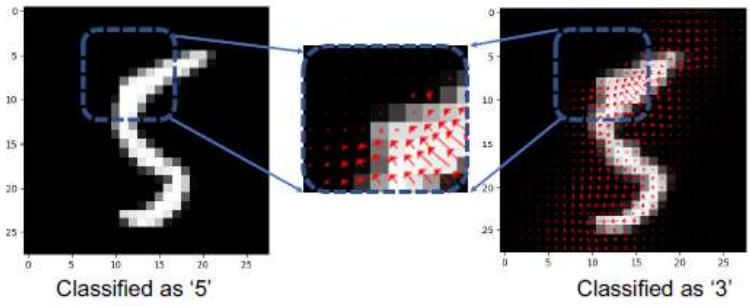

Figure 5: Spatially transformed attack example over MNIST example

3.1.12 Unrestricted adversarial examples

After giving a review of the most relevant methodologies for generating adversarial attacks centred on the domain of image classifier systems, it is necessary to highlight the common denominator of all the attacks that have been presented. All of them base their action on adding imperceptible patterns on existing images, which work well without being modified, or modifying them minimally so that they are able to deceive the classifier system. Song et al. [2018] develop a method to generate unrestricted adversarial examples. The examples generated by this method need not look like any victim image, but it is still a legitimate image to human eyes and can mislead the classifier. The most important advance of this methodology is that it was able to successfully attack all defence systems proposed for existing methods based on the generation of adversary images from perturbations of target images.

The method consists of training an Auxiliary Classifier Generative Adversarial Network (AC-GAN), this model can generate a legitimate image x from a noise vector z_0 and the target classifier C classifies it as belonging to class y. To craft an adversarial image, we now start from a noise vector z close to z_0, the attack will succeed if the output of the AC-GAN is able to fool the model C. As z is close to z_0 in its latent space, the output of the AC-GAN should belong to the same class y but it will be different from the starting image x, being able to deceive C.

3.2 Physical world Attack

The attacks described so far take place in the digital sphere. Modifications crafted with the aim of deceiving the classifier model must be added to the digital image that will be the input to the system. However, there is another type of attack, which can be considered as physical attacks, in which the crafted patterns are now printed in the physical world and are acquired by the system through the usual sensors, video cameras, photo cameras, microphones, etc.

Several studies began to explore this possibility, such as Eykholt et al. [2017] in which remarkable results were obtained by sticking stickers on stop signs. These physical modifications drastically reduced the recognition accuracy of these signs by the signal recognition systems of autonomous cars. This type of attack soon became one of the most dangerous because it could be exploited in the real world and on applications based on this type of technology, such as facial recognition, video surveillance systems, autonomous vehicles and so on.

During the development of S4AllCities, this type of attack will be explored in two ways. We will design a physical pattern that when shown to the system by a person will be invisible to the video surveillance system and we will also tackle the challenge of trying to devise a pattern to trick the system to identify hand weapons, specifically knives.

3.2.1 Exploring adversarial examples in physical world

The first experiments in this field were done by Kurakin et al. [2016b] trying to test the feasibility of adversarial patterns generated by techniques such as FGSM or BIM when translated to the real world. The aim was to evaluate the robustness of these attacks to transformations such as illumination, change of position and size with respect to the camera as well as the transformations suffered by the adversarial pattern in the printing process.

The tests carried out by Kurakin et al. [2016b] showed that after transformations such as those described, most of the adversarial examples still retained their ability to fool the classifier, especially those that had been generated using FGSM. This evidence demonstrated the viability of this type of attack in any environment.

3.2.2 Eykholt's attack on road signs

As mentioned in the introduction to this section, Eykholt et al. [2017] were able to fool a traffic signal classification system by placing certain stickers at strategic locations on a stop sign. As can be seen in Figure 6, these patterns consisted of color stickers on the sign itself.

Figure 6: Stop sign attack (Image credit: [Eykholt et al. 2017])

The methodology developed consisted of:

1. First, a classical attack is implemented, in the digital domain, based on the $l1$ norm ($\|x' - x\|_1$) on stop signal images in order to find the areas of application of the patterns. As mentioned before, $l1$-based attacks generate sparse disturbances, which helps to find the location of the attack.
2. The second step consists of using an attack based on the $l2$ norm on the areas delimited by the first step to generate the colour patterns.
3. The last step is to print the generated adverse patterns and place them on the signal at the positions determined by the process.

According to the authors, the attack can deceive the navigation system of an autonomous car from any distance and orientation. This case of a physical attack on a system that operates autonomously and relies

on sensors whose signal is processed by convolutional neural networks and must maintain high levels of security is a clear example of the risk associated with attacks of this type.

3.2.3 Athalye's 3D adversarial object

There are numerous works and methodologies for crafting adversarial examples that, when translated to the real world, still have validity and the ability to deceive classifier systems. The big challenge they face in general is twofold:

- When it is printed, the adversarial image must maintain its ability to deceive on the physical plane although it was designed in the digital domain. It is therefore necessary to consider the transformation of colour intensities that the pattern will undergo in the printing process.
- They have to remain effective attacks against variations in position, orientation and illumination with respect to the camera used as a sensor.

Taking these aspects into account, it is important to highlight the work of Athalye et al. [2017] which reported for the first time the construction of a 3D printed adversary example capable of fooling the classifier regardless of position, orientation and distance to the camera. As can be seen in the image, a 3D printed turtle figure was created that was classified as a rifle.

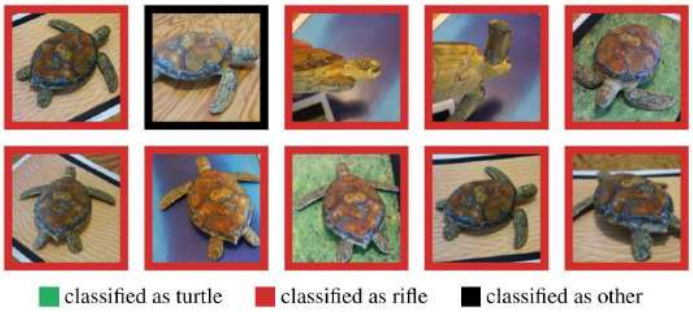

Figure 7: Adversarial attack in 3D domain (Image credit: [Athaiye et al. 2017])

The paradigm shift in the attack on computer vision systems based on neural network architectures is overwhelming. Attacks on computerised systems in general can no longer only originate in the digital domain, as was always considered when talking about cyber security. The physical domain also needs to be controlled as it can be a source of attacks through sensors such as cameras.

3.3 Black-box Attacks

The following black-box attacks are worth mentioning because of their importance, as they are possible even with the limited information the attacker has about the inference system.

3.3.1 Substitute model

Papernot et al. [2017] presented for the first time an algorithm capable of attacking a convolution neural network under very restrictive conditions, the attacker did not know the parameters of the target model and had no access to the training dataset. This was the first black box attack ever recorded and published. The attacker in this case could only give an input x to the system and get the output of the classifier y. Furthermore it could be assumed that he also had information regarding the data domain that the classifier was able to process, handwritten digits, photographs, faces, etc. and knowledge of the classifier architecture at a very high level, e.g. CNN, RNN.

This type of attack exploits a property that adversarial examples have which is the "**transferability**". The theory is very simple, an adversarial example x' that can attack a model $F1$, can also attack a system $F2$ as long as its structure is similar to $F1$. Starting from this premise the methodology is clear, the attacker starts from the training of a system F' that behaves in the same way as the target system, F. Once they manage to obtain the system F', an adversary attack is crafted on it. In this way, the adversary image that is able to fool the equivalent system, F', will also be a valid attack on the target system, F. The attack will be composed of the following steps:

1. Generate an alternative training dataset, using examples from other datasets or generating one of your own.

2. Train the surrogate model on the alternative dataset, this model will be F'.
3. Increase the alternative dataset and retrain the subrogated model iteratively to improve the accuracy of the model.
4. Attack the subrogated model F' using the procedures described so far. It should be noted that the most effective types of attacks will be the ones that have high transferability such as FGSM and PGD.

3.3.2 Zoo: zeroth order optimization based black-box attack

This attack describes a very similar situation to the previous one, but the attacker has a little more information. In the previous case the attacker only knew the input and the label returned by the classifier system; now in addition to y also has access to the value of the inference confidence. As shown by Chen et al. [2017] in this case it will not be necessary to build a surrogate model equivalent to the target model in order to craft an attack. Chen et al. propose a methodology to infer the gradient around the target sample x by observing changes in the confidence of the prediction $F(x)$ as the pixel values of x are tuned. As can be seen from the equation detailing the method, the approximation consists of the application of the derivative in the discrete domain of the target image, x.

$$\frac{\partial F(x)}{\partial x_i} \approx \frac{F(x + he_i) - F(x - he_i)}{2h}$$

Making h small enough, the above expression is a good enough approximation to the gradient to apply any attack described above.

3.3.3 Query-efficient black-box attack

In general, the two previous methods require many queries to the classifier to build the surrogate model or to approximate the value of the gradient around the target image. This fact can make it very expensive and even impossible to materialize an effective attack of this type in reasonable time. For these reasons new alternatives began to be explored for the estimation of the gradient in the surroundings of the image x such as those presented in Ilyas et al. [2018] in which, by using natural evolution strategies, the points around the target x, where to make the following queries, are selected in a more efficient way.

3.4 Semi-white (Grey) box Attack

Xiao et al. [2018] introduced an attack methodology that can be considered as a semi-white box attack. In this case, a Generative Adversarial Network (GAN) is trained to behave in the same way as the target model. Once the attacker has trained the GAN, he can directly generate adversarial examples from the generative network. GAN-based methods have the advantage of speeding up the process of producing adversarial examples and generating more natural and undetectable examples. Deb et al. [2018a] propose the use of GAN to generate examples of synthetic faces, absolutely realistic examples capable of deceiving face recognition systems.

3.5 Poisoning attacks

When we defined the attack surfaces in previous sections, we saw that, in addition to the attacks described so far, all of which correspond to evasion attacks, there are also attacks that can be performed before the classifier model has been trained. These attacks, known as poisoning attacks, consist of crafting adversary images that are inserted into the training dataset as if they were a legitimate part of it before the classifier was trained. This attack involves knowing the architecture of the target model and that will subsequently be trained using this dataset. It should be noted that graph-based neural network (GNN) architectures are susceptible to this type of attack. The most common poisoning methods are described below.

3.5.1 Biggio's poisoning attack on SVM

This method introduced by Biggio et al. [2012] managed to reduce the accuracy of the SVM model effectively by poisoning the training dataset. The method consists in creating an adversarial sample x_c that once introduced in the training dataset causes the SVM model, F_x, to have a large total loss in the entire validation dataset. This adversarial example is crafted by applying incremental training techniques

[Cauwenberghs and Poggio, 2001] that can model the influence that each training example has on the final model (SVM).
This technique of deceiving by dataset poisoning is not easily transferable to neural network models. The relationship between the samples used for training and the model is not explicit in the case of these architectures, although it is successful for support vector machine systems.

3.5.2 Koh's model explanation

Following the philosophy of the previous method, Koh & Liang [2017] present a work that aims to translate the previous attack on SVM models to neural network architectures. To do so, they developed a methodology for interpreting these architectures that sought to answer the following question: How would the prediction of a DNN model change if the training dataset changes? Their model is able to answer this question by being able to give an explicit measure of the change in the final loss function of the neural network when a single piece of data in the dataset has changed. Moreover, this can be performed without the need to re-train the model. Thanks to this work, it was possible to adopt this type of attack in neural network architectures with success, as the training examples that had the greatest influence on the prediction of the model could now be found.

3.5.3 Poison frogs

The method presented by Shafahi et al. [2018a] implies to insert in the dataset a true sample with a true label in order to achieve the classifier make a mistake when labelling an image from the test group. In more detail, the attacker selects a target image from the test group x_t with a true label y_t. The attack is initiated using as a basis another sample x_b of class y_b, also true. Then solving:

$$x' = \arg min_x \|Z(x) - Z(t)\|_2^2 + \beta\|x - x_b\|_2^2$$

The adversary image x' is found and inserted into the dataset. Once the model has been trained with this altered dataset, the $Xtrain + \{x\}'$ model will classify x' as belonging to the class y_b since there is very little distance between x' and y_b. If we now use the trained model to infer the class to which x_t belongs, as the objective of x' forced the vector x_t and x' to be close, then x' and x_t will give the same output and both inputs will correspond to class y_b.

4 Defense strategies

This chapter will describe the main groups of defenses put forward to deal with known adversary attacks. It is very important to emphasize this point, the defense and adversary attack process seem to go hand in hand and are destined to improve and reinforce each other. The main defensive countermeasures against adversary attacks can fall into one of three main groups:

- **Gradient masking or gradient obfuscation**: As there are numerous types of attacks based on the knowledge of the gradient of the target model, attempts to hide this information may be beneficial to avoid the attack.
- **Robust optimization**: Increasing the robustness of the classifier by relearning its parameters again is another aspect that will be developed in more detail later. These techniques will allow the classifier to be able to correctly classify adversarial examples in the future.
- **Adversarial examples detection**: By studying the distribution of the examples that compose the original dataset, adverse examples can be identified and prevented from entering the classifier.

4.1 Gradient Masking/Obfuscation

Since there are a significant number of attacks that rely on knowledge of the gradient of the target model, this defensive method is based on trying to hide or mask this information relative to the classifier model. Within this group of algorithms, we can highlight the following classes.

4.1.1 Defensive distillation

The concept "Distillation" was first introduced in the work of Hinton et al. [2015] where it is defined as a training technique to reduce the size of the deep neural network (DNN), applicable to any type of

architecture. This goal is achieved by training a small DNN that has as inputs the output of the last layer of the original DNN just before the softmax layer. Later publications such as Papernot et al. [2016b] redefine the distillation procedure to train a network that is able to defend against attacks based on FGSM and Szegedy's L-BFGS. The training process described is as follows:

- The network F in the dataset (X, Y) is trained by setting the temperature value of the softmax activation function to T. (Where temperature T mean: $softmax\,(x,T)_i = \frac{e^{\frac{x_i}{T}}}{\sum_j e^{\frac{x_j}{T}}}$ $i = 0,2,\ldots.K-1$).
- Then the values of F(X) are calculated after the softmax and the temperature T scores are evaluated.
- Now we will train the F'_T model using the temperature value T for the softmax on the dataset with soft labels $(X, F(X))$. The F'_T model is known as distilled model.
- The distilled net F'_T with a temperature value equal to 1, F'_1 will be used to predict test data or adversarial examples.

When we train a distilled network with temperature value T and evaluate it with $T = 1$, we are forcing the inputs to the softmax to be larger, namely a factor T. That is, the output of the original network just before the softmax, for example x will be $x * T$ as well as a neighboring sample x', will be $x' * T$. When we evaluate the output of the softmax with $T = 1$, the output vector will be: $(\epsilon, \epsilon, \epsilon, \ldots, 1-(m-1)\epsilon, \epsilon, \epsilon \ldots \epsilon)$ where the output value of the target class is very close to 1 and the rest very close to 0. In practice the value of epsilon is so small that in 32-bit floating-point machine precision rounds it to zero making it impossible to find the value of the gradient of F'_1 and thus protect the system from attacks that are based precisely on trying to infer this gradient.

4.1.2 Shattered gradients

Another possibility is to process the input data as discussed in Buckman et al. [2018] or Guo et al. [2017] The idea is basically to process the dataset and in general the input data by means of a non-differentiable and non-smooth function $g(.)$ and now train the DNN model f on $g(X)$ dataset. The trained classifier $f(g(X))$ is not differentiable in terms of x, protecting the system from attack based on knowledge of the gradient.

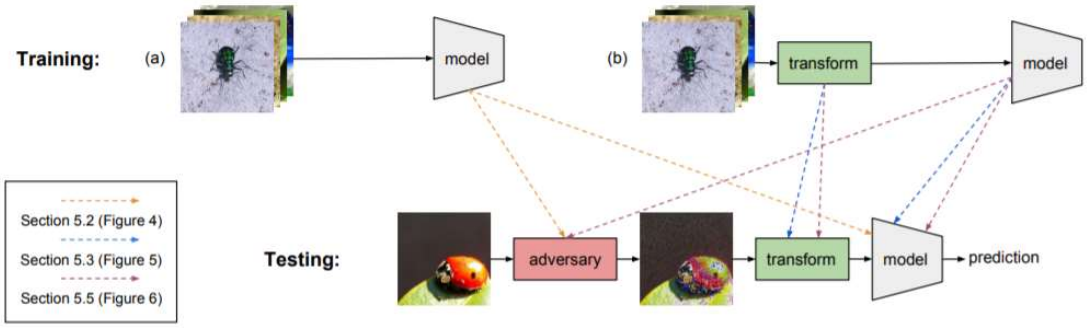

Figure 8: Non-differentiable transformation (Image credit: [Buckman et al. 2018])

4.1.3 Stochastic/randomized gradients

The several techniques described below are intended to introduce randomness into the classification system so that the adversary attack has a lower success rate. One case consists of training a set of classifiers $s = \{Ft: t = 1,2,3,\ldots k\}$ using the same dataset. Once trained, at evaluation time, when the user makes requests to the classifier the system will randomly use one of them to perform the inference. In this way the probabilities of a successful attack are reduced. Other strategies such as Dhillon et al. [2018] randomly remove neurons from certain layers or, as shown by Xie et al. [2017a], the input image is also randomly resized and filled with zeros until it returns to the original size.

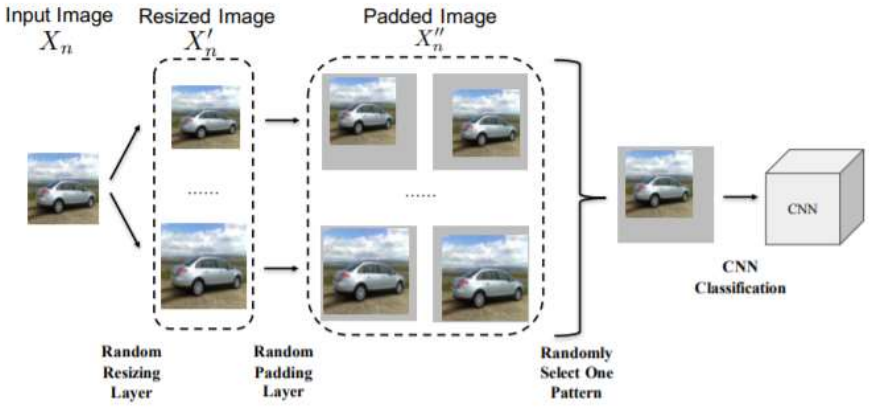

Figure 9: Randomly resize example (Image credit: [Xie et al. 2017a])

4.1.4 Exploding & vanishing gradients

This defence methodology involves putting a generative network before the classifier. This configuration will result in a very deep structure of the generative network plus classifier. The goal of this configuration is that the cumulative product of the partial derivatives of each layer, which is ultimately the gradient, is either too small or too large, and thus trying to prevent the attacker from being able to accurately calculate the gradient of the classifier.

This type of techniques can be seen as data purifiers, models that receive the adversarial example and purify it before passing it to the classifier. PixelDefend (Song et al. [2017]) and Defense-GAN (Samangouei et al. [2018]) are two of the architectures that implement such solutions.

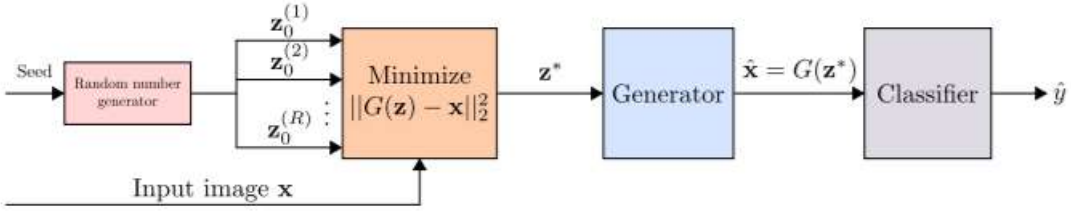

Figure 10: Defense-GAN algorithm overview (Image credit: [Samangouei et al. 2018])

4.1.5 Gradient masking/obfuscation methods are not safe

After the publication of the gradient obfuscation methods presented in the various sections of the section, it became clear that all of these strategies were not effective in defending against adversary attacks with an acceptable level of assurance. Work such as Carlini and Wagner [2017b] and Athalye et al. [2018] highlighted the weaknesses of these defences and thus the existence of attacks that were still effective against them.

4.2 Robust Optimization

Robust optimization algorithms will improve the robustness of DNN-based classification models by changing the way the model learns. The aim is to study how the model parameters can give correct predictions in the presence of potentially adversarial examples. This objective can be achieved by focusing developments on two principles:

1. Discovering the parameters of the model θ^* that minimize the average adversarial loss.
$$\theta^* = \arg min_{\theta \in \Theta} \mathbb{E}_{x \sim D} \, max_{\|x' - x\| \leq \epsilon} \mathcal{L}(\theta, x', t)$$
2. Discovering the parameters of the model θ^* that maximize the average minimum perturbation distance:
$$\theta^* = \arg min_{\theta \in \Theta} \mathbb{E}_{x \sim D} \, min_{C(x') \neq y} \|x' - x\|$$

A robust optimization algorithm must have knowledge of the characteristics of the model that can be attacked as well as of the attacks it has to defend against, in particular it must know the adversarial domain D. Many works in this field are based on developing defenses against adversarial examples generated from small perturbations of the l_p norm. Although these defenses would not protect the system against other

attacks, it is essential to do this exercise as it is the basis for developing defenses against other types of attacks. For this reason, defense methodologies developed based on the l_p norm are presented below.

4.2.1 Regularization methods

At the beginning, the works done on the defence of deep learning algorithms were based on the characteristics that such algorithms must have to be robust, i.e., to resist adversarial attacks. In other words, they suggested that robust models should be stable to perturbations in the inputs. In Szegedy et al. [2013] the use of the Lipschitz constant L between consecutive layers as a boundary condition for training is proposed. In general, training the models by imposing these regularization conditions would help the model to be more robust to adversarial attacks. Two regularization methods are presented below:

- **Penalization by Lipschitz constant between layers:** The Lipschitz constant L_k is the maximum value of the variation of the output values of a function with respect to the variation of its input values. It is therefore a measure of the sensitivity of a function to perturbations of its inputs. n the work developed by Szegedy et al. the use of this method of regularisation between layers during the training of the classification model is suggested. In this way the following expression should be fulfilled between two layers:

$$\forall x, \delta \; \|h_k(x; W_k) - \; h_k(x + \delta; W_k)\| \leq L_k \|\delta\|$$

  In this way the output of the classifier would not be affected by small disturbances at the input. Later work such as Cisse et al. [2017] formalised this idea by associating the Lipschitz constant with the adversarial risk of a model. In this paper the Lipschitz constant of the loss function is defined. Making use of this penalty for each hidden layer of the model during training would increase the overall robustness of the system and therefore better resist adversarial attacks.
- **Penalise the partial derivatives of the layers**: In Gu and Rigazio [2014] a regularisation algorithm called Deep Contrastive Network was introduced. This method suggests introducing a penalty in the value of the partial derivatives of each layer when back-propagation is performed, so variations in the input to the layer do not imply large variations in the output.

4.2.2 Adversarial (RE)training

The main defence methods based on the extension of the training datasets with adversarial samples will be described below. The aim of these techniques is to make the network able to discern between real images and adversarial attacks:

1. **Adversarial training with FGSM:** The idea, originally proposed by Goodfellow et al. [2014b] is relatively intuitive and consists of adding correctly labelled adversarial samples to the training dataset so that the classifier is able to classify them correctly. In this work Goodfellow used the non-targeted FGSM attack to generate the adverse samples x'. The system is then trained with these well-labelled samples, (x', y). The resulting system trained with this extended dataset improves its robustness to FGSM-generated samples. The problem with this method is that it performs well against attacks based on FGSM-generated examples but is vulnerable to iterative attacks of other types, including one-step attacks as demonstrated by Tramer et al. [2017].
2. **Adversarial training with PGD:** Due to the proven limitations of the previous solution, Mandry et al. [2017] proposed to use the Projected Gradient Descent (PGD) attack to generate a new dataset with adverse examples (only) as this method theoretically generates the "most adverse" example in the l_∞ ball around x: $B_\epsilon(x)$. In this way the most adversarial example, x_{adv} is located where the F model is most vulnerable. Training the model with these more adverse examples allows to identify the parameters θ of the model that minimise the adversarial loss. If the model trained in this way has smaller loss than the most adversarial examples, then it will be protected for all x' in the environment $B_\epsilon(x)$. This defence performs well against iterative and one-step attacks but implies that an iterative attack has to be generated for each sample in the dataset, which prolongs training times by a lot.
3. **Ensemble adversarial training:** Tramer et al. presented a training method that could protect models based on CNN architectures against single-step attacks and could be applied to large datasets such as ImageNet. The method consists of expanding the training dataset with adverse samples crafted by other classifiers retrained on this dataset. For example, to improve the

robustness of model F, then they train models F_1, F_2, and F_3 on the same dataset. These models will have different hyperparameters than the F model. Once trained, they use these models to craft adversarial examples with the single step FGSM method. Because of the transferability property of single-step attacks, these adversarial examples will also be able to deceive the F-model. That is, these examples will be good approximations to the most adversarial examples for the F-model at x. Now training the F-model on these examples will minimise the adversarial loss of F. This defence performs better than the previous ones, decoupling the models that generate the adverse examples from the model to be defended, and in general they are able to defend against single-step and black box attacks.

4.2.3 Certified defences

Adversarial training has established itself as an effective method of defence although it still does not guarantee absolute safety of models trained using this paradigm. It is always possible that the attack could be generated using more powerful techniques that manage to fool the protected systems in this way. It would, therefore, be irresponsible to apply these adversarial training algorithms on safety-critical systems.
As we have seen in previous sections, the Reluplex algorithm introduced by Carlini et al. is the first rigorous approach to verify the robustness of a DNN model. This method is able to calculate the exact value of the minimum disturbance distance $r(x;F)$ for a model F. In other words, it gives us precise information on the limit beyond which the model is safe. Perturbations whose norm is less than $r(x;F)$ will not pose safety problems. Reluplex thus gives us the exact value of $r(x;F)$ that verifies the robustness of the model F at x. From this point on, there were several other works that tried to develop methodologies to verify the robustness of the models, Raughunathan et al. [2018a], Wong and Kolter [2017], Hein and Andriushchenko [2017] tried to find trainable certificates $C(x,F)$ calculated in x for the model F that were lower bound of minimal perturbation distance for F in x. $C(x,F) \leq r(x;F)$.
As shown in the figure below, any perturbation bounded by the $C(x,F)$ norm must be safe for the model. The training process that optimises these certificates will therefore improve the robustness of the model.

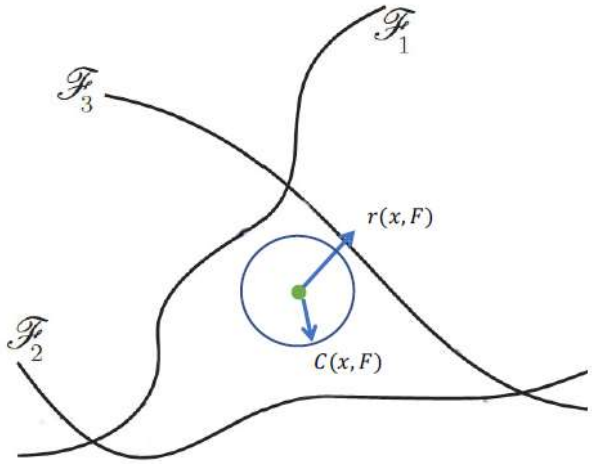

Figure 11: Model is safe in $C(x,F)$ ball

4.3 Adversarial Example Detection

Detecting when an image is benign or on the contrary hides an adversarial attack before it is analyzed by the classifier model is another defense philosophy called adversarial example detection. In this situation, if the input image is detected as adversarial, the DNN system rejects it and therefore no label is assigned to it either. Carlini and Wagner [2017a] classify these systems with such adversarial example detectors in three different types depending on the attacker's knowledge of the system.

- A **Zero-Knowledge Adversary**: When the adversary only has access to the F's model parameters.
- A **Perfect-Knowledge Adversary**: The model F and detection scheme D is known.
- A **Limited-Knowledge Adversary**: The model F and the detection scheme D are known but there is no access to the D parameters.

We will now look at some of the most relevant methods of detecting adversarial examples.

4.3.1 An auxiliary model to classify adversarial examples

This definition generally encompasses all auxiliary models that have been trained to detect adversarial examples and to distinguish them from benign examples. For example, Grosse et al. [2017] trained a DNN model with $k + 1$ classes, the first k were the classes of objects that the system could classify in the normal way and the extra class was assigned to the adversarial examples. Gong et al. [2017] designed a similar system but trained it to binary classify whether or not an example was adversarial. The work of Metzen et al. [2017a] is based on a system that is able to detect whether an input is adversarial or non-adversarial based on classifying information extracted from a given layer of the classifier it is intended to defend.

4.3.2 Using statistics to distinguish adversarial examples

There are also methods capable of discerning when an image is benign or is an adversarial example by making a comparative statistical study of its properties. When performing a PCA (Principal Component Analysis) of an image, work such as Hendrycks and Gimpel [2016] shows that genuine images assign higher weights to the first values of the PCA while adversarial examples tend to assign higher values to the last values. In this way one could distinguish whether an image is adversarial or not by analysing its PCA. It is worth mentioning other approaches in this area such as the use of the Maximum Mean Discrepancy (MMD) test.

4.3.3 Checking the prediction consistency

Other studies look at the consistency of the classifier results against changes in the classifier parameters or gradual changes in the input images. They are based on the hypothesis that when an input is benign, these small changes should not change the results of the classifier. The work of Feinman et al. [2017] introduces randomness into the model through Dropout. If the classifier gives very different predictions for input x, before and after its application, the example is likely to be adversarial. In Xu et al. [2017] the input itself is manipulated by checking the consistency of the classifier prediction. For each input image the colour depth in the image is varied. This method assumes the hypothesis that in benign images, the reduction of the colour depth does not alter the classification result.

5 Application in S4AllCities

Within the context of S4AllCities, it is important to be cognizant that smart cities may deploy common visual analytics algorithms, which are, therefore, susceptible to the type of attacks described herein. To highlight this potential vulnerability, an image designed to attack the YOLO2 algorithm was designed, preventing video streams processed using this algorithm to identify specific objects. Specifically, the image designed (shown in Figure 12) was intended to cheat the algorithm from identifying the presence of humans.

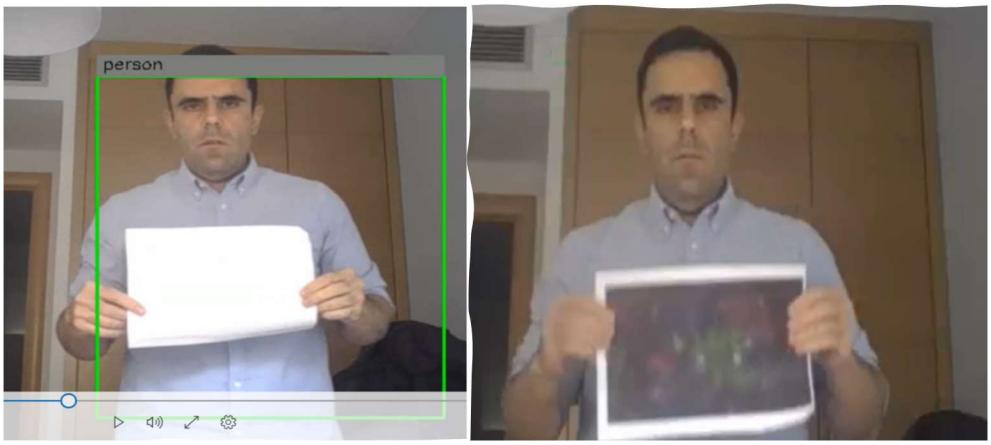

Figure 12 Image used in S4AllCities to attack YOLO2 person recognition algorithm

References

[Athalye et al. 2017] Athalye, A., Engstrom, L., Ilyas, A., and Kwok, K. Synthesizing robust adversarial examples. arXiv preprint arXiv:1707.07397, 2017.
[Athalye et al. 2018] Athalye, A., Carlini, N., andWagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. arXiv preprint arXiv:1802.00420, 2018.
[Biggio et al. 2012] Biggio, B., Nelson, B., and Laskov, P. Poisoning attacks against support vector machines. arXiv preprint arXiv:1206.6389, 2012
[Biggio et al. 2013] Biggio, B., Corona, I., Maiorca, D., Nelson, B., ˇ Srndi´c, N., Laskov, P., Giacinto, G., and Roli, F. Evasion attacks against machine learning at test time. In Joint European conference on machine learning and knowledge discovery in databases, pp. 387–402. Springer, 2013.
[Buckman et al. 2018] Buckman, J., Roy, A., Raffel, C., and Goodfellow, I. Thermometer encoding: One hot way to resist adversarial examples. In International Conference on Learning Representations, 2018.
[Carlini et al. 2017a] Carlini, N., Katz, G., Barrett, C., and Dill, D. L. Provably minimally-distorted adversarial examples. arXiv preprint arXiv:1709.10207, 2017a
[Carlini and Wagner 2017a] Carlini, N. and Wagner, D. Adversarial examples are not easily detected: Bypassing ten detection methods. In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, pp. 3–14. ACM, 2017a.
[Carlini and Wagner 2017b] Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In 2017 IEEE Symposium on Security and Privacy (SP), pp. 39–57. IEEE, 2017b.
[Cauwenberghs and Poggio 2001] Cauwenberghs, G. and Poggio, T. Incremental and decremental support vector machine learning. In Advances in neural information processing systems, pp. 409–415, 2001.
[Chen et al. 2017] Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, pp. 15–26. ACM, 2017.
[Chen et al. 2018] Chen, P.-Y., Sharma, Y., Zhang, H., Yi, J., and Hsieh, C.-J. Ead: elastic-net attacks to deep neural networks via adversarial examples. In Thirty-second AAAI conference on artificial intelligence, 2018.
[Cisse et al. 2017] Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. Parseval networks: Improving robustness to adversarial examples. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 854–863. JMLR. org, 2017.
[Deb et al. 2018a] Deb, D., Zhang, J., and Jain, A. K. Advfaces: Adversarial fase synthesis. arXiv preprint arXiv:1908.05008, 2019., 2018a.
[Dhillon et al. 2018] Dhillon, G. S., Azizzadenesheli, K., Lipton, Z. C., Bernstein, J., Kossaifi, J., Khanna, A., and Anandkumar, A. Stochastic activation pruning for robust adversarial defense. arXiv preprint arXiv:1803.01442, 2018.
[Eykholt et al. 2017] Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D. Robust physical-world attacks on deep learning models. arXiv preprint arXiv:1707.08945, 2017.
[Feinman et al. 2017] Feinman, R., Curtin, R. R., Shintre, S., and Gardner, A. B. Detecting adversarial samples from artifacts. arXiv preprint arXiv:1703.00410, 2017.
[Gong et al. 2017] Gong, Z., Wang, W., and Ku, W.-S. Adversarial and clean data are not twins. arXiv preprint arXiv:1704.04960, 2017.
[Goodfellow et al. 2014b] Goodfellow, I.J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572, 2014b.
[Grosse et al. 2017] Grosse, K., Manoharan, P., Papernot, N., Backes, M., and Mc-Daniel, P. On the (statistical) detection of adversarial examples. arXiv preprint arXiv:1702.06280, 2017.
[Gu and Rigazio 2014] Gu, S. and Rigazio, L. Towards deep neural network architectures robust to adversarial examples. arXiv preprint arXiv:1412.5068, 2014.
[Guo et al. 2017] Guo, C., Rana, M., Cisse, M., and Van Der Maaten, L. Countering adversarial images using input transformations. arXiv preprint arXiv:1711.00117, 2017.
[Hein and Andriushchenko 2017] Hein, M. and Andriushchenko, M. Formal guarantees on the robustness of a classifier against adversarial manipulation. In Advances in Neural Information Processing Systems, pp. 2266– 2276, 2017.

[Hendrycks and Gimpel 2016] Hendrycks, D. and Gimpel, K. Early methods for detecting adversarial images. arXiv preprint arXiv:1608.00530, 2016.
[Hinton et al. 2015] Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015.
[Ilyas et al. 2018] Ilyas, A., Engstrom, L., Athalye, A., and Lin, J. Black-box adversarial attacks with limited queries and information. arXiv preprint arXiv:1804.08598, 2018.
[Katz et al. 2017] Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient smt solver for verifying deep neural networks. In International Conference on Computer Aided Verification, pp. 97–117. Springer, 2017.
[Koh and Liang 2017] Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 1885–1894. JMLR. org, 2017.
[Krizhevsky et al. 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pp. 1097–1105, 2012.
[Kurakin et al. 2016b] Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533, 2016b.
[Madry et al. 2017] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083, 2017.
[Metzen et al. 2017a] Metzen, J. H., Genewein, T., Fischer, V., and Bischoff, B. On detecting adversarial perturbations. arXiv preprint arXiv:1702.04267, 2017a.
[Moosavi-Desfooli et al. 2016] Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2574–2582, 2016.
[Moosavi-Dezfooli et al. 2017a] Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. Universal adversarial perturbations. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1765–1773, 2017a.
[Papernot et al. 2016a] Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. In 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 372–387. IEEE, 2016a.
[Papernot et al. 2016b] Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In 2016 IEEE Symposium on Security and Privacy (SP), pp. 582–597. IEEE, 2016b.
[Papernot et al. 2017] Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. Practical black-box attacks against machine learning. In Proceedings of the 2017 ACM on Asia conference on computer and communications security, pp. 506–519. ACM, 2017.
[Raghunathan et al. 2018a] Raghunathan, A., Steinhardt, J., and Liang, P. Certified defenses against adversarial examples. arXiv preprint arXiv:1801.09344, 2018a.
[Samangouei et al. 2018] Samangouei, P., Kabkab, M., and Chellappa, R. Defense-gan: Protecting classifiers against adversarial attacks using generative models. arXiv preprint arXiv:1805.06605, 2018.
[Shafahi et al. 2018a] Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. Poison frogs! targeted cleanlabel poisoning attacks on neural networks. In Advances in Neural Information Processing Systems, pp. 6103–6113, 2018a.
[Song et al. 2017] Song, Y., Kim, T., Nowozin, S., Ermon, S., and Kushman, N. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. arXiv preprint arXiv:1710.10766, 2017.
[Song et al. 2018] Song, Y., Shu, R., Kushman, N., and Ermon, S. Constructing unrestricted adversarial examples with generative models. In Advances in Neural Information Processing Systems, pp. 8312–8323, 2018.
[Su et al. 2019] Su, J., Vargas, D. V., and Sakurai, K. One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation, 2019.
[Szegedy et al. 2013] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199, 2013.
[Tramer et al. 2017] Tramer, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. Ensemble adversarial training: Attacks and defenses. arXiv preprint arXiv:1705.07204, 2017.

[Wong et al. 2017] Wong, E. and Kolter, J. Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. arXiv preprint arXiv:1711.00851, 2017.
[Xiao et al. 2018a] Xiao, C., Li, B., Zhu, J.-Y., He, W., Liu, M., and Song, D. Generating adversarial examples with adversarial networks. arXiv preprint arXiv:1801.02610, 2018a.
[Xiao et al. 2018b] Xiao, C., Zhu, J.-Y., Li, B., He, W., Liu, M., and Song, D. Spatially transformed adversarial examples. arXiv preprint arXiv:1801.02612, 2018b.
[Xie et al. 2017a] Xie, C., Wang, J., Zhang, Z., Ren, Z., and Yuille, A. Mitigating adversarial effects through randomization. arXiv preprint arXiv:1711.01991, 2017a.
[Xu et al. 2017] Xu, W., Evans, D., and Qi, Y. Feature squeezing: Detecting adversarial examples in deep neural networks. arXiv preprint arXiv:1704.01155, 2017.