Master's Degree in Computer Engineering

Master's Degree Thesis

Transferability of Adversarial Attacks: Main Influencing Factors

Supervisors
prof. Cataldo Basile
prof. Massimiliano Todisco

Candidate
Edoardo Giordano

Company Supervisor
Dott. Ric. PhD Jose Victor del Razo Sarmina

Academic Year 2020-2021

To my grandparents

Summary

The growth in computation power and storage strongly reduced the time required for a machine learning model to be trained and allowed the development of more complex models that can address more sophisticated tasks. These improvements led in the last years to the introduction of AI systems into a growing number of applications. Some of them are also used in contexts that could lead to safety issue, where a wrong behaviour, in some cases, could even endanger the life of a person.

In the thesis we tried to deepen the understanding of malicious attacks that can be carried out against machine learning based systems. The most common type of this attack is known as adversarial examples. It allows to fool a network into a wrong behaviour by providing slightly ad hoc modified inputs. The relevance of this attack is due to the fact a human observer cannot notice the difference from a normal input. In detail, we analyzed the property of transferability of an attack by focusing on the knowledge the attacker has with respect to the target ML system. In this context, we analyse how knowing the training set and the network model affects the transferability of an adversarial example. The research analyzes both image and face recognition challenges, with the application of more complex netowrk models to address the second problem.

The results highlight that the knowledge of the training set is relevant only in the case where also the network model is known. Otherwise, knowing the dataset only affects the result of few percentage points. What happen to be significant in the generation of the adversarial example is the quantity of noise we introduce. Moreover, comparing the scenario of image recognition with the one of face recognition, we notice a remarkable drop in the number of transferable case and in the influence of the quantity of noise. We attribute this difference to the higher complexity of the network used in the second challenge.

Acknowledgements

I would like to acknowledge Spike Reply for the opportunity of researching on a such interesting and current topic. In particular, I would like to thank my supervisor del Razo Sarmina Jose Victor and my colleague Hazboun Elias who helped me through discussions and suggestions, for the duration of the whole project. I would also like to acknowledge my professors Cataldo Basile and Massimiliano Todisco.

Then, I would like to acknowledge my family that gave me the possibility to study without any additional concern and also for the support and the confrontation they offered me during the important decisions. I also want to also ackwnoledge my granparents that were always cheerful for every goal achieved.

Finally, I would like to thank all my friends that made the journey easier.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The development of Intelligence systems has been a focus of research since the early 1956. Nevertheless, only in the last years the researches increased remarkably as is possible to see in Figure 1.1. This increase mainly comes from the Machine Learning area, which study systems that use big amount of data to learn how to solve tasks. This interest in the field is also reflected by the number of application that are used in everyday life, which is continuously increasing. These systems are applied to improve the current state of common programs, such as advertisement detectors, or solves new challenges such as the voice recognition system. Some of them also play a role in safety related application, where the life of the user can be endangered. For example, in autonomous vehicles the recognition of road signs and the decision on the action to perform are all taken through a Machine Learning algorithms. These new algorithms are also used in cybersecurity related application for both attack and defense purposes. Indeed, they are used such as intrusion detection systems, but they also allow an attacker to carry on an offensive without needing the knowledge previously required by the user. For these and other reason this new technology will lead to threat in digital, physical and political domain [5].

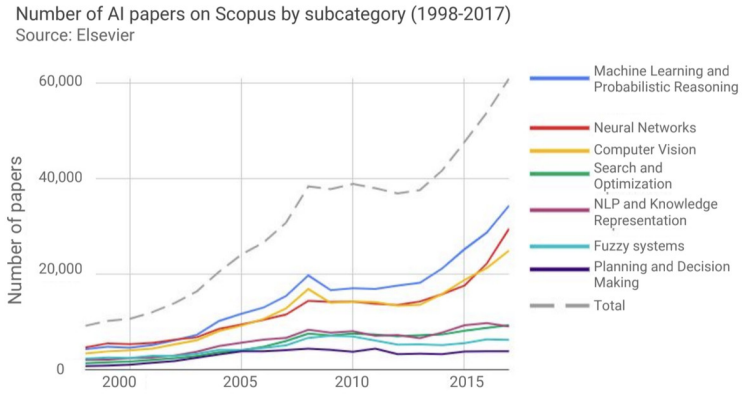


Figure 1.1: Research on AI technologies over the years [1]

As happened for every important IT technology, as soon as the number of application increased, the researchers started to study the security of these new algorithms. As often happens, also Machine Learning models introduced a new kind of vulnerability. Indeed, it was proven that is possible to fool these algorithms through an adversarial example. This is an ad hoc crafted input (i.e. an ordinary input with some added noise) that will lead the ML model into a miss-classification, that is usually translated in a wrong response to a given input. The danger of those inputs is the fact that a human observer will easily correctly classifying them, without being able to notice the added perturbation. Moreover, understanding the cause of this behaviour is hard and there are no defensive mechanisms that work in every situation. Therefore is not possible to determine if an algorithm is secure or not [6], so this vulnerability is an open problem.

During the years, researchers also demonstrated that an adversarial example generated for one network can fool also another network that does not share any element with the first one. This property is called transferability of an adversarial example. Due to the use of ML systems in security application and their vulnerability to adversarial attacks, the transferability property leads to big security problems. Since the attack does not require access to the internal structure of network, the possibility to submit data to the system allows an attacker to modify the behaviour of the system.

Many researches [7], [8], [9] focused on the possibility of fooling these systems more than studying what affect the transferability of an adversarial example. Therefore, the research question this work focuses on is: how and to what extent does the attacker's knowledge on a given target network affect the transferability of an adversarial attack. More in detail, the goal of the thesis is to understand the correlation between the knowledge of the target system, in terms of training set (the data that the network used when learning how to address the problem) and network model (which is the internal topology of the machine learning algorithm), and the transferability of an adversarial example that we can generate through a different network. We will approach the problem by dividing it in three scenarios:

- White Box: the attacker knows the dataset and the network model of the target system;
- Gray Box: only the dataset used in the training phase is known;
- Black Box: neither the data nor the network model are known.

The contribution of this thesis aims to support the understanding on how to protect modern AI algorithms and how to make them more robust against adversarial attacks. For example we should help answering question such as: is it necessary to protect the training data? Or, the model of a given algorithm should be kept secret? What are the consequences of doing so?

As follow a summary of the organization of the thesis with a small presentation of the topic discussed in each chapter.

- *Chapter* 2: presents the basic of Machine Learning and Neural Network, then it analyzes the generation of an adversarial example and concludes by presenting the tools and techniques that were used during the developing of the algorithm;

- *Chapter* 3: summarizes the most relevant research in the field of adversarial examples, to allow the reader to understand the state of the art of those attacks;
- *Chapter* 4: explains the research question and presents the different scenarios used during the analyses, explaining why those situations are considered relevant to reply to the problem;
- *Chapter* 5: introduces the algorithm used during the analyses, explaining the key elements and their functionality;
- *Chapter* 6: shows the results of the tests, by comparing the different scenarios, and explains the different behaviours;
- *Chapter* 7: reports the conclusion regarding the analysis and add some other consideration extracted from the results, by comparing the image and face recognition analysis.

Chapter 2

Background

This chapter provides the background information necessary to better understand the scope of the thesis. Moreover, it presents a general introduction to Machine Learning and then describe some tools and techniques that were vastly used during the development of the algorithm.

2.1 Machine Learning

Machine learning (ML) is a branch of Artificial Intelligence (AI) that concern the study of systems that use experience to improve performance [10]. In detail, it focuses on designing and developing prediction algorithm that are able to train themselves over the inputs, which can be examples, direct experience or instructions. These data are usually electronic inputs prepared for analysis and collected into training set. The model will use this collection of data to extract the most relevant feature for the given task. The success of this process is strongly dependent on the quality and the size of the data.

As in other area of computer science, some relevant measure of the quality of ML algorithm are time and memory complexity. However, when evaluating these new models, is necessary to introduce the notion of sample complexity, which represent the required amount of data for a network to learn how to classify the given classes.

2.1.1 Machine Learning Tasks

Machine learning tasks can be classified into:

- *Classification*: this challenge consists of assigning a category to each input. Usually, this task presents less than few hundred classes, but it can remarkably increase in some difficult application such as speech recognition. A common application of this problem is, for example, image classification in which the goal is to understand the object inside a picture such as a cat or a plane.

- *Regression*: this challenge concerns the prediction of the exact value (usually a real number). During the training phase, the penalty of an incorrect result is strongly

dependent on the distance between the correct and the predicted values. An example of this problem is the prediction of stock values.

- *Ranking*: this challenge consists of ordering items according to certain criteria. A common application is to order the results of a web search.
- *Clustering*: this challenge consists of partitioning a set of data into homogeneous subsets. This solution is usually applied when working over very large dataset, such as social Networks, where you try to identify natural communities.
- *Dimensionality reduction*: this challenge consists of transforming the input into a lower dimensional representation of it, while maintaining the most relevant properties of the initial input. A common example is the pre-processing performed during computer vision tasks.

2.1.2 Learning Scenarios

Another classification derives by the learning techniques that the model implements. In particular, these different techniques are strongly dependent by the type of data, how these data are provided to the network and the test set used to evaluate the algorithm.

Supervised Machine Learning

The goal of supervised ML is to develop algorithms that can classify the submitted data over some features [11]. Therefore, the model will be able to assign class labels to test inputs where the value of those features is known, but the label is unknown.

The generation of the dataset requires some fixed step. The first is collecting the data. This task should be assigned to an expert in the field, which will highlight the most relevant features required in the data for the classification. In case this approach is not suitable, a brute-force method is suggested, but this approach usually introduces more noise and therefore requires a more substantial pre-processing phase. The second step is the data pre-processing. The objective of this task is to decide how to handle missing data and outliers. The last step is to perform a feature subset selection, which aims to remove irrelevant and redundant features.

Unsupervised Machine Learning

Unsupervised ML is a technique that aims to develop an algorithm which is able to extract hidden structures from the data, as in the clustering challenge. A major difference between this technique and supervised ML is the absence of a training set. This implicates that we cannot perform any more cross-validation [12]. Another important difference regards the performance of this algorithm. Indeed, although most of them are developed with the goal of optimality, is not possible to guarantee that the globally optimal solution will be found. To achieve this goal, the algorithm should try all possible partition, which is not feasible even for a moderate sample size dataset. Therefore, these methods implement heuristic algorithms, which are strongly dependent by the starting parameters. When performing

unsupervised machine learning, the most relevant tasks are the selection of samples, the selection of features, the choice of similarity and the choice of the algorithm.

Semi-Supervised Machine Learning

Semi-supervised ML is a paradigm that tries to merge supervised and unsupervised ML techniques. Therefore, the idea is to develop a system that can learn in the presence of both labeled and unlabeled data [13]. The main goal of this new technique is to analyze how joining labeled and unlabeled data can affect the learning process, and to achieve better results taking advantage of it. The main application of this technique regards the situation where the labeled dataset is insufficient or labelling data is expensive. In this scenario the use of unlabeled data, which is easier to obtain, can improve the final results. A promising application for this technique is represented by the challenge of understanding the human category learning, where most of the inputs are unlabeled. The success of Semi-supervised learning strongly depends by the underlying assumptions, such as the chosen algorithm.

Reinforcement Learning

Reinforcement learning is a technique that aims to train the models by reword and punishment, without the need of describing how to achieve the task [14]. Therefore, the system has to map the response between the current situation and all possible actions in order to chose the one that yield to higher reward. This is obtain by simulating and evaluating the outcome all possible cases. Usually, the taken action does not only affect the immediate result, but also the next decision. In order words, these systems present a status and based on it they take decisions. These two features of trial and error and delayed reward are the key element of reinforcement learning [15].

2.1.3 Artificial Neural Networks

Nowadays, most of machine learning algorithms are based on an Artificial Neural Network (ANN), also known as Neural Network (NN). The goal of NN is to implement a simplified version of the human brain [16].

As is possible to see from Figure 2.1, a NN is composed by a group of connected units called neurons. These units are organized in layers: the input layer acquires the external data, then a variable number of hidden layers elaborate the data before transmitting it to the final layer, the output one, where the network returns the classification of the given input. Each neuron receives inputs from the previous layer, performs a weighted sum of these values and then passes the result through activation function, which usually is non-linear. Then, each neuron transmits its value to the neurons of the next layers, where the same operation is repeated. This process is repeated until the output layer is reached. This final layer is the one responsible to convert the elaborated data into a vector of probability that will allow the user to obtain the classification of the network for the given the input.

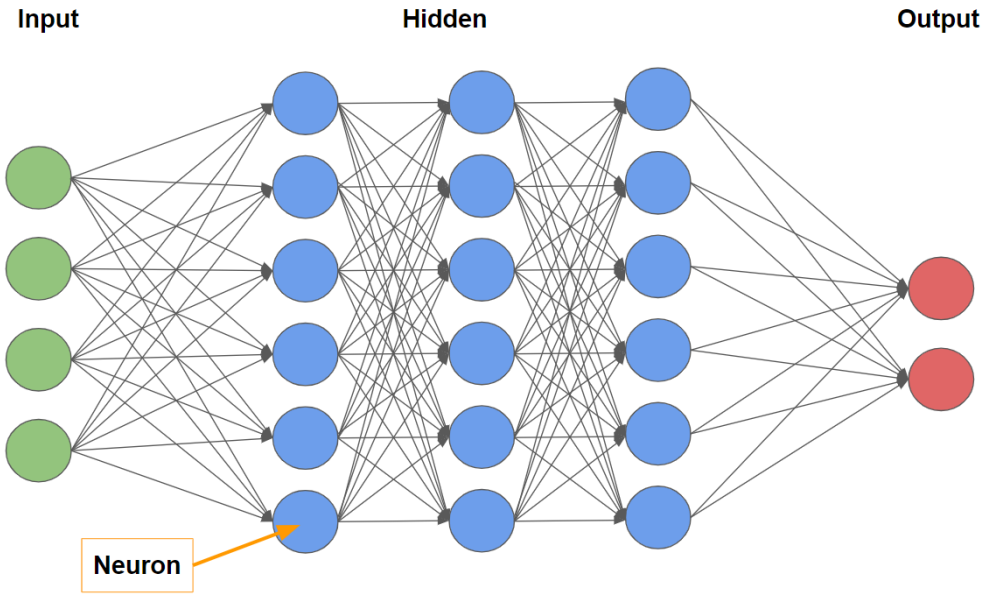


Figure 2.1: Graphical representation of a Neural Network with three hidden layer

All the parameter such as the number of neurons per level and their interconnection, the activation function and the number of hidden layers vary between different implementation of NNs.

A network that presents more than one hidden layer is usually addressed as Deep Neural Network (DNN). On the other hand, if the matrix operation is handled by the network with a convolutional calculus, the model will be identified as a Convolutional Neural Network.

When a training set is present, the network, in order to improve its classification, adjusts the weight used by each neuron within the weighted sum through the backward propagation. This is achieved by executing a loss function, which evaluate the distance between the prediction of the network and the correct value, and the result is back propagated to update the weights of the network. This update can be performed in both a synchronous and an asynchronous way.

2.2 Adversarial Example

Due to the increasing use of AI application, the research world started to investigate the security of those new algorithms. The Brundage article [5] well analyses the threat landscape of AI security. The authors states that AI system are programs that influence our everyday life, and that in the future this control will even increase. Moreover, the paper focuses on the changes that the expansion of AI adoption will bring to the landscape of security threats, while considering the duality of this new technology (which will helps both attacker and defender). Indeed, the authors state that the use of AI will reduce the

cost and the time required to perform cyber attacks that previously required human labor, which will also increase the set of actors that can carry out these attacks. In addition, they consider the fact that use of AI could carry out attacks that were previously impossible for humans, introducing so new vulnerabilities. The last aspect that they consider is the fact that new AI application will not always affect utility and security in the same way, since the malicious use of this application is simpler than implementing the counter measure to thwart them. This could lead to relevant threat in digital, physical and political domain.

The most interesting aspect, from a research perspective, is the one concerning the new vulnerabilities characteristic of AI systems. Indeed, during the years was proven that is possible to lead a network to a miss-classification. This result can be achieved by submitting to the network an ad hoc crafted input, which is called adversarial example, that is forged by adding a specific perturbation to a determinate input. The relevance of this attack is related to the fact that it will be almost imperceptible to a human observer [17].

When working in the image recognition area, this process consist in adding noise to the picture to slightly modify the value of some pixel and to lead the network into a miss-classification, as is possible to see in Figure 2.2. This is generally achieved by evaluating the loss function for the given input and then, instead of backtracking the results to train the network, use it to modify the submitted image.

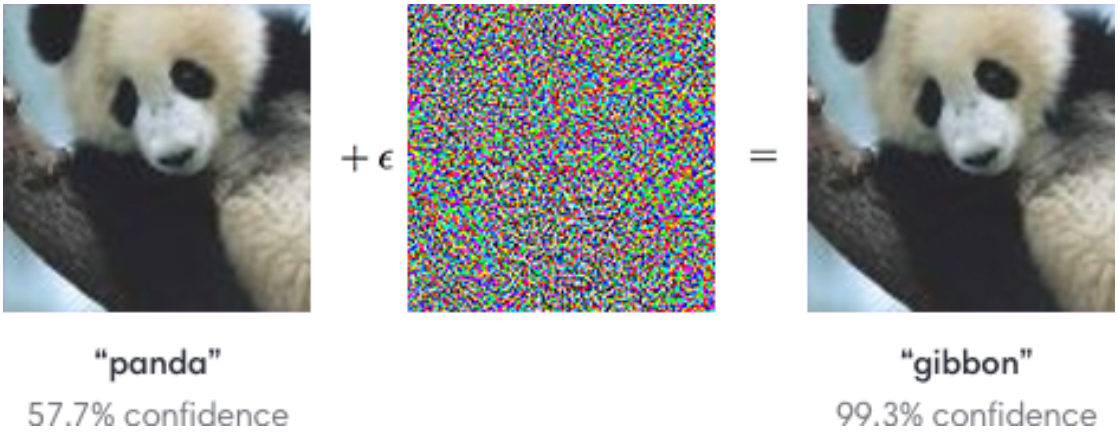


Figure 2.2: An adversarial image that present on the left a clean image of a panda, in the middle the added noise and on the right the miss-classification as a gibbon [2].

Regarding the generation of adversarial example, is possible to differentiate the categorization with respect to the final goal of the attack. Particularly, when working in recognition area, either for images or faces, there are two main type of attacks: untargeted and targeted.

Untargeted Attack

The most general attack, within the image recognition field, is the untargeted attack. The goal of this approach is to lead the network into miss-classifying the input as belonging

to a different class from its correct one. Generally, the best way of achieving this is to increase the difference from the correct class, without focusing on the taken direction. Indeed, since the approach aims to any possible misclassification, it has a wider solution space available and therefore it leads to the best results.

Targeted Attack

A similar implementation, which aims to a more precise result is the targeted attack. As deductible by the name, its goal is not only to lead the network into a miss-classification, but also to be able to select the class with which the input will be classified. In other words, the idea is to be able to lead the network into miss-classifying an object belonging to class A as belonging to a certain class B. In this context, an attack is considered successful when without the perturbation the network classifies the input correctly, but after adding the noise it classifies the new input as an object belonging to the chosen class. When working within face recognition, this approach is also known as impersonation.

Single-step and Iterative attacks

Another important distinction that we have to keep in mind when talking about adversarial example, is if we are either generating a single-step or an iterative attack. Even if both approaches generate the noise with the same method (evaluating the loss and adding or removing it to perform an untargeted or targeted attack respectively) they differ in the number of time that the process is repeated.

The single-step approach aims to generate all the noise over the original image in one iteration. The iterative method, on the other hand, generate the noise each time on result of previous iteration. This approach creates a perturbation that achieves better result over the local network. On the downside, it reduces the transferability of the generated example since the noise in such way generated is more dependent of the weights of the network.

2.3 Tools and Techniques

In this section are presented some existing tools and techniques that were used to develop the algorithm.

2.3.1 Pytorch

When approaching the machine learning world the first objective is to identify the approach that will be used to solve the problem, in order to decide the framework to use. The two main characters in the field are TensorFlow and Pytorch. As explained in [18], TensorFlow was developed by Google and is more oriented toward the production-grade deep learning solutions. On the other hand, Pytorch was developed by Facebook in 2016 with the idea to simplify the usage and the models. It is based on torch, a framework for fast computation, and it presents similarity with pre-existing python library such as Numpy. Due to its nature, Pytorch is more suitable for research application.

Due to the nature and goal of my studies and to follow the implementation strategies adopted by previous research we decided to use Pytorch as framework for my experiments and my work.

Pytorch implements all the data structure and all the function required to train networks, such as models, datasets, optimizer and loss functions. All the existing models present also a pre-trained version, which was trained over the imagenet-1000 dataset. The majority of Pytorch's functions work with tensors, multi-dimensional matrices.

Regarding the computer vision challenge, Pytorch handles a colour image by transforming it into a three-dimensional matrix where the first dimension represents the number colours, therefore it will have a value of three in a colourful image that represents red, green and blue.

2.3.2 The Models and the Datasets

As previously explained, the python library offers the implementation of various models that are based on research papers, which investigate different internal NN structures. Therefore, instead of implementing my own version of these proposed models we used the one offered by Pytorch. Regarding the dataset we considered as main requirement the classes to be well represented. This is mainly due to fact that the thesis requires to train some network on small percentage, such as 12.5% or 25%, of the whole dataset. Indeed, in those cases if each class is not represented by enough images, the final accuracy of the network substantially decreases and this strongly affect the results of the research. This problem arose when working within the face recognition challenge since it is harder to find a good number of pictures of the same person.

For the Image Recognition challenge, we used a ResNet18 [19], as main local DNN, a GoogleNet [20], as second local network, and a AlexNet [21] as target network. All the three networks were trained on the CIFAR10 dataset [22], which present 60000 colour images in 10 classes. Due to the low complexity of the dataset is possible to achieve a good accuracy even with basic model such as the ones previously presented.

When working on the Face Recognition challenge, we decided to use a dataset derived by vggface2 [23]. Moreover, a small subset composed by 10 classes and on average 350 image for each of them was extracted. We also decided to use more complex networks that were able to achieve a good result also on a more complex dataset, to allow continuity with future experiments. Therefore we used a Densnet161 [24], as main network, a more complex version of [19] called ResNet152 and as target network an InceptionResnetV1, pretrained over the complete vggface2 dataset as explained in [25].

2.3.3 Transfer Learning

In order to reduce the resources required to train the network and also increase its final accuracy a technique called transfer learning was applied. When working with small database such as MNIST is possible to achieve a good accuracy, over 90%, in few iterations using a relatively simple ML model. Instead, working with more complex data sets requires to increase the number of iterations and to use more complex models to achieve good accuracy. Therefore, is possible to use the transfer learning technique to obtain better

result in term of accuracy of the network, reducing the constraint over the data set and the resources required.

This technique aims to use the knowledge acquired by a source task to perform a new targeted task. This solution is different by multitask learning since it does not require the network to be able to perform both source and target tasks at the same time [26]. Figure 2.3 compares the traditional training with transfer learning.

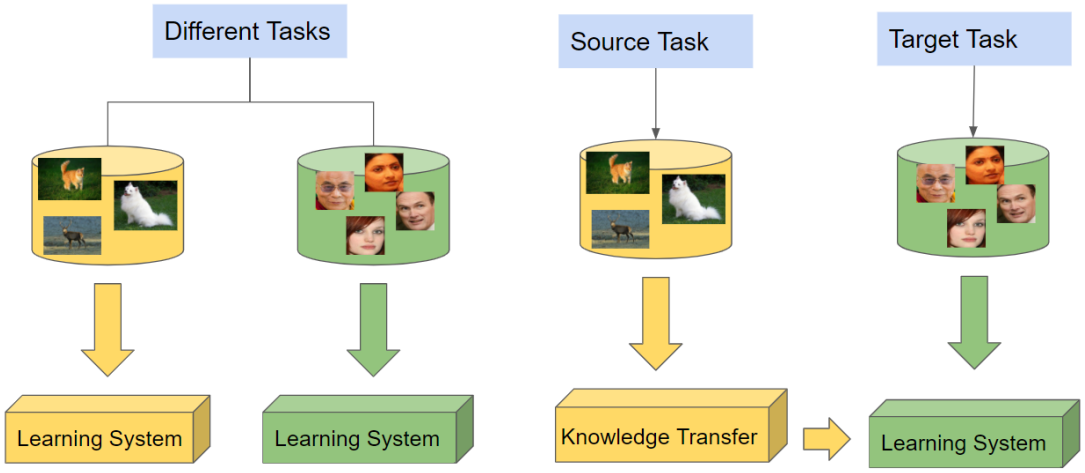


Figure 2.3: A visual comparison between a tradition machine learning approach, on the left, and transfer learning technique, on the right.

The relation between source and target tasks and domains allows the categorization of transfer learning in three cases:

- *Inductive transfer learning*: this solution is applied when the source and the target tasks are different, without keeping in consideration the domains;
- *Transduce transfer learning*: this solution analyzes the case in which source and target tasks are the same, but the two domains are different;
- *Unsupervised transfer learning*: this solution aims to solve the same problem of the transduce transfer learning, but it mainly focuses on the clustering problem.

Practically, Transfer Learning consists of using the weights of a pre-existing network as starting value when training a new one. Based on the relation between the two data sets and the size of the second one, is possible to identify two possible way of applying transfer learning [27].

Feature extractor

The first solution is applied when a pre-existing network is used as feature extractor and then a final layer is added to the network as classifier. In other words, the original network is used as a black box machine, without modifying its weights, that extracts the features of the input, while the final layer will be then trained to convert the output of the original network into the correct classification. This solution is usually suggested when working with a data set where class are not well represented inside training data, but it still presents a good similarity with the original training set.

Fine tuning approach

The second solution consist in using the network only as starting point. Indeed, the idea is to use the weights of the pre-existing network as initial weights and therefore retrain the whole network on the new data. Even if the values are modified during this second training phase, this solution still allows to achieve better result than using random values to initialize the weights. This approach is more suitable when a big data set is available since the model can fine tune the extraction process over the important features of the new training set. It is also suggested when the two data sets are not alike, even if the new training set present a small number of elements per class, since the new data will require a different feature extraction for a better classification.

Chapter 3

Related Work

This chapter presents the state of the art with respect to adversarial examples. We first discuss adversarial attacks beyond the computer vision area. We then concentrate on computer vision and present the most relevant research for our work.

3.1 Diffusion of AI

This section provides a brief overview of possible adversarial attack in different machine learning application. The inclusion of these papers in my work wants to show the interest of the researchers in adversarial attacks, not limited to the computer vision challenge. Indeed, as is possible to see from Figure 3.1, in the last year the number of publication concerning the adversarial attacks remarkably increased.

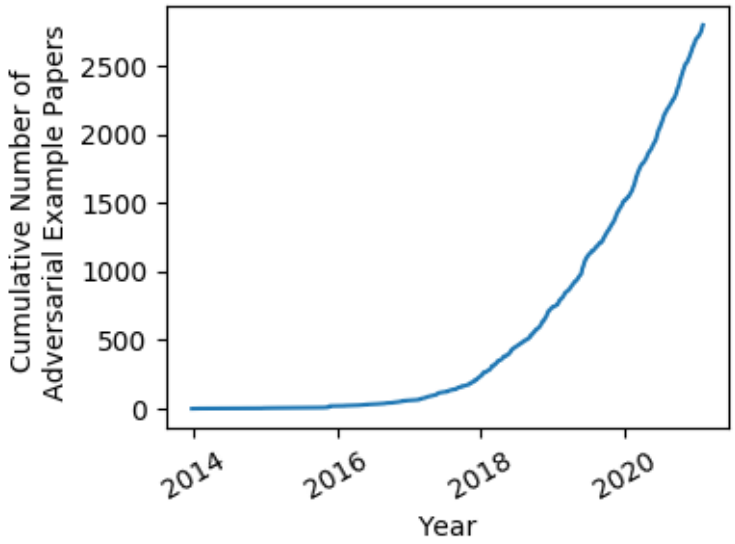


Figure 3.1: Research on AI adversarial example over the years [3]

The first adversarial example was proposed by Dalvi in 2004 when with his collaborator proved the possibility to fool a spam filter by adding few good words to the email [28]. Moreover, with the increasing number of AI systems, the focus lead into studying the possibility to carry those attack on everyday application such as AdBlock [8] or a copyright detection system [29]. The researchers also proved the possibility to fool speech recognition system in performing commands, without the user being able to recognize them [9] or even without being able to hear them [30].

3.2 Transferability of Adversarial Examples

A great contribution to the research was given by Papernot with his papers, which show how an attacker can fool a ML model without having access to it. At first, as explained in [31], he presented the possibility to transfer adversarial example between two different models. Moreover, present the result of the intra-class and inter-class transferability, demonstrating that in not only possible transfer adversarial example between different implementation of the same ML model, such as DNN, but also between different classification algorithms, such as Decision Tree or Linear Regression.

Our work differs from [31] in few aspects. First, the research was conducted only in the image recognition area with a simpler dataset called MNIST. A second difference is the fact that those experiments only focus on a black box attack and therefore they do not perform an analysis on the percentage of dataset used. Moreover, they use a basic Gradient Sign Descendent with a σ value of 0.3, which because it is not better specified, we assume to be a single-step approach. This allow a higher transferability, since the perturbation is less dependent of the local DNN, but it is also easily prevented by some defensive mechanism [32]. Indeed, in this paper the authors show the possibility to increase the robustness of a network by adding adversarial examples to the training set. Even if this solution proves to remarkably increase the accuracy over the targeted attack, it only affects the ones generated with a single-step approach and not with an iterative one.

Papernot demonstrates that the knowledge of either the internal model or the training data are not required in order to carry on an attack; the new technique is called black box attack and it presented in [33]. The process is based on observing the label assigned by a locally trained DNN to chosen inputs. Thanks to the local DNN, adversarial inputs are generated and then tested on the targeted Network. Those attack are proven to be also capable of evading defense strategies previously found to be a good countermeasure to adversarial example.

The relevance of this paper is obvious with the goal of attacking an unknown system, but unfortunately the authors do not deepen into the transferability of this new attack.

3.3 Attacks against Image Recognition

Most of the researchers focus on the possibility to attack a Machine Learning models, more than the possibility to transfer those attacks. Regarding the generation of an adversarial attack with respect to image recognition in the digital domain, the authors mainly focus on the idea of minimizing the perturbation of the adversarial attack.

An example of this approach is presented in [34], in which they show how, by adding a small adversarial scratch that cover less than 5% of the image, they obtain a success rate of more than 90%.

Another approach, which reduces even more the perturbation, is presented in [7], where the authors present the possibility of creating a one-pixel perturbation based on more complex mathematical evaluations of the image, such as differential evolution.

Both approaches do not analyze the transferability of their results, but they allowed us to understand the importance of introducing fewer noise as possible when working in the digital domain.

3.4 Adversarial Attack in the Physical World

When performing the same attack in the physical domain, we have to understand that is not possible to apply the same techniques that we used in the digital domain. For example, if we try to reproduce the one-bit perturbation, it would be very hard to reproduce the exact same attack with the pixel in exact same position (not compared with the other objects in the picture, but as its absolute position). Moreover, even if we could achieve this objective the difference in lighting could make it ineffective.

Therefore, when researchers want to create an attack that is reproducible in the physical world, more than focusing on reducing the perturbation introduced, they limit the area that can be modified. In other words, the idea is to create a gadget, usable in the real world, and add all the noise onto it, without limiting the quantity on noise introduced.

An early example of this solution is presented in Figure 3.2 [4], where the authors developed a pair of glasses where the colours of the frame are created with a DNN to be adversarial. The main limitation of this proposal is the fact that they use only frontal pictures with a fixed light, that allows the authors to avoid some problem such as translation and rotation of the object or the change in the illumination.

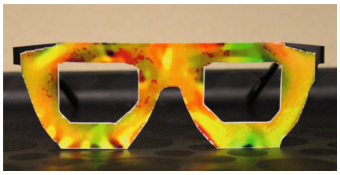


Figure 3.2: Example of a gadget that is able to fool a real world face recognition system [4].

A solution to this problem was presented by Brown [35], where the authors create a patch that, when added to a scene, prevents the classifier to identify any other object in the picture. This new technique increases the attack surface since the patch can be added anywhere, and it is robust to a wide range of transformation.

Another relevant study is presented in [36]; in the paper the authors demonstrate the possibility to 3D-print an object that is adversarial even when subject to physical transformation, such as rotation and translation, or different illumination.

A further improvement is shown by the idea presented in [37], where the authors create an adversarial t-shirt the allow avoiding detection. This is relevant considering the fact that a t-shirt is not a rigid object and therefore it introduces by itself a new level of complexity.

Even if these papers do no present any study regarding the transferability of their examples, they offer a wide number of solution on how is possible to move an attack into the physical world, which is relevant for further analysis.

Chapter 4

Problem Statement

This chapter shows the goal of the thesis and the different scenarios we analyzed during the research.

Previous work focuses mostly on generating the adversarial inputs more than their transferability. Therefore, we decided to analyze how the dataset and the network model affect this process. This is relevant for understanding the success chances of an attacker fooling a second network while training on a known network.

There is a certain pattern in a large share of the related work. Generally authors start by approaching the problem through a small a simple dataset, which is generally belonging to the image recognition challenge, and continued by investigating how to attack real system world. Some of them also replicated the same approach in the face recognition area [4],[38].

Following the same approach, we chose to develop the algorithm and to test it on a simple and small dataset within the image recognition field. Then, we reproduced the analysis over the identification of a face recognition systems and the final step of performing the attack against a real-world system is left for future work.

We look into how the knowledge of an attacker can increase the number of perturbed images that can be transferred from one network to another. In this context, the idea is to offer a comparison between a white-box and a black-box attack. The first one represents the case in which an attacker has full knowledge of target implementation, whereas in the latter the attacker does not have access to the internal implementation.

The focus of the research is to understand how the training set and the network model affect the succes of transferring the adversial example. In order to achieve this goal, the idea is to compare the results of different simulation of the attacker's knowledge, which affect the training of the local networks, to understand their relevance in transferring the attack. This process is summurized in Figure 4.1. These analyses should allow forecasting the probability for an attacker to success into fooling the ML model based on its knowledge.

Another potential use concerns the case where the attacker has limited resources, and the studies can be used to find the best compromise between the elapsed time to train the network, the available computer resources and the expected results. Indeed, the number of elements of the dataset strongly affects the time needed to train the network. Anyway, this motivation should not play a major role since the network is trained una tantum,

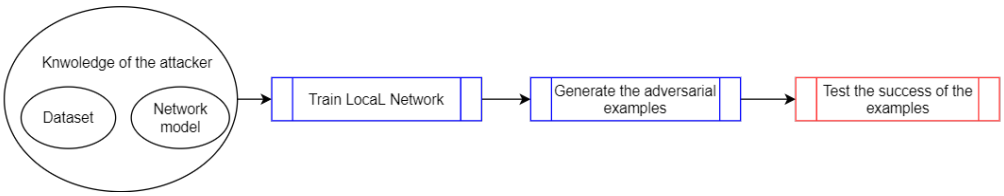


Figure 4.1: High level representation of the attacker approach.

and the training set does not affect the generation of the adversarial example, in terms of time.

To pursue this goal, we organize the research in various scenarios which identify the knowledge that an attacker can have with respect to the target network.

Regarding the relevance of the dataset, we analyzed two cases: we either know the dataset or we do not. Note that in the case we know the dataset, the possibility of knowing only part of it is also considered. In this context, each test has a different local DNN, which is the used for generating the adversarial images and is trained over a subset of the whole training set, while the target DNN does not change and it is trained on the whole dataset.

Regarding the knowledge of the model, the situation is more straight forward. The attacker either knows or does not know the model used to build the application. Note that during the experiments we always used DNN networks, since those models seems to better perform in the to the computer vision domain.

As is possible to see from Table 4.1, the combination allows four possible scenarios, but we concentrated only three of them.

	Same Dataset	Different Dataset
Same Network model	White Box scenario	Not analysed
Different Network model	Gray Box scenario	Black Box scenario

Table 4.1: Graphical representation of the different scenarios

The decision of pursuing only three of them is mainly based on the consideration that is very unlikely that the attacker knows the network but not the dataset. Indeed, nowadays the number of DNN models is bigger than the number of well-known datasets. Moreover, it is possible to know only part of a dataset but would be irrelevant and much more unlikely than knowing part of the internal model of the network. In this context, knowing only a part of the network would be probably useless, since most of the networks share anyway the same basic element, but the interconnection between them is what characterize a model. In addition, for the sake of the experiment, is also required at least to know the classes that the target network can correctly classify, while there is no requirement of notion over the model structure.

4.1 White Box Scenario

It this the attacker has a full knowledge of the target application. Therefore, he knows both the training set and the machine learning model used to develop the system. This is an unlikely situation to happen, since usually the company should try to hide solution settings. The relevance of this case is mainly to generate a term of comparison when analyzing the other results.

It is reasonable to expect that this scenario will lead to the highest transferability. In this context, we can consider these results as maximum achievable values for the given settings (algorithm, network and dataset), and analyze other result with respect to these ones. Moreover, from this scenario we expect a straight dependency between the percentage of dataset used during the training of the local network and transferability of the results. Since we know the model, having more dataset allow us to better emulate the target network and therefore achieve higher transferability.

Even if these constraints are tight and unusual, there are few cases in which an attacker could be interest into studying this scenario:

- A team starts an open-source project, this will lead an attacker to know everything about it. Even if the team will not use this work in a cyber security topic, the attacker could still fool the network to reduce the credibility and the fame of the company.
- A team uses an open-source project as starting point to develop their own, which allows to reduce the cost of the development but introduce cybersecurity vulnerabilities.
- An attacker is able to get the information regarding the network and the training set through a social engineering attack, or by using other existing vulnerability in the company systems.

In all those cases the attacker would not have directly access to the target network and therefore to perform an attack he should use the principle of transferability. To do so, he would use the same network and the same dataset to train the local DNN and then he would be able to generate the adversarial attacks.

4.2 Gray Box Scenario

The second scenario analyzed is the case where the attacker knows only the dataset or part of it, but not the network model and will be referred as Gray Box scenario. The main contribution of this experiments is regarding the influence of the training set over the generation of the adversarial examples. In other words, since the only knowledge of the attacker is the dataset or part of it, a confrontation between the result obtained within this scenario and the other ones, mainly the black-box attack, will give a reference in the relevance of knowing the dataset.

From this scenario we expect a smaller contribution of the dataset compared with the one of the white-box attack, but still significant. Even if we do not know how the image is

classified, knowing the dataset should leads the two models to create a similar distribution of the solution space. However, since we generate the noise specifically on how the image is classified and we do not know the target classifier, the transferability should decrease.

A possible situation of this scenario could happen during the presentation of a project. Indeed, the company, to better explain to the final user, could present examples of the training data that belongs to a well-known dataset. Moreover, even if the company used that dataset only as base over which build their whole new dataset, this could lead an attacker to determinate its probability of success.

4.3 Black Box Scenario

The last case is a full black-box scenario. In this case the attacker neither knows the machine learning models or the data set used during the training phase. The only knowledge that the attacker has, for the sake of the experiment, is the fact that the target network is able to correctly classify elements belonging to a class. The results of this scenario enlighten two different aspects. The first one, as already anticipated, is the comparison with previous scenarios. Indeed, we can compare the results to predict the probability of success based on how the algorithm works within the white-box scenario. The second aspect, on the other hand, gives us an estimate of success without the need to first query the target network.

From this scenario we expect to have a lower success rate than the Gray Box case. Indeed, since we do not share any data, also the solution space could present a different distribution.

This scenario is the most likely to happen when we want to perform an attack against a target network that we do not control. Since it does not require any previous knowledge it is feasible in every situation in which we apply it. Also, the constraint regarding the target class can be easily addressed. If we can interact once with the target network, we could simply query it and use the result of this question as our target class. This assumption withstand also when working within the face recognition world. Once the target is decided, through an online research, is possible to get information regarding the employee of the company that uses the access control system. Then, through the social media we could get the picture required to train the local DNN.

Chapter 5

The Design

In this chapter is presented the algorithms used to both train the network and generate the adversarial attacks.

All the algorithms were developed and executed on cloud machines. In detail, we used two different web applications, Kaggle and Google Colaboratory, that allow the execution of Jupyter notebook environment on GPUs for free. Both these frameworks are supported by Google, and allow the users to access free GPUs for limited amount of time. For the purpose of performance analysis, we used Kaggle platform since Google Colaboratory does not offer always the same GPU. Kaggle offers a Tesla P100 Nvidia GPU with 16GB of dedicated memory.

When creating the algorithm, we divided it into two independent parts, the first one is responsible to handle the training of the network while the second one is in charge of the generation of the adversarial attack. This division is driven by the fact that the executions of the two phases are not correlated to each other. Indeed, the code used to train the network is a standard solution and is not affected by the final use of the network. In addiction, the training phase is responsible to select the required subset of data, to instruct the network over it and then save the final states of the model.

In order to better emulate a real situation, the same target network was used over all the experiments of the same scenario. On the other hand, each scenario requires a different target network, since it either presents a different model or is trained on a different dataset, as shown in 4.1. Regarding the local DNNs, they are different during each test since they need to be trained on a different dataset each time. Moreover, due to the nature of the experiment, which requires to use only part of the dataset, we could not use pre-trained version of the model, since generally those models are trained on the whole available dataset.

The second part of the experiment focused on creating the adversarial attacks. As previously introduced, there are multiple way to generate an adversarial attack. Since there are strong defensive mechanisms against a single-step attack, we decided to perform an iterative method, even if this was proven to be less transferable.

When deciding between targeted or untargeted attack, we considered more relevant the possibility to impersonate a certain subject than being just able to force a miss-classification. Moreover, when a targeted attack is performed, one can also analyze the

possibility to be untargeted. As long as we consider only the miss-classification per se and not the final target of the miss-classification, we obtain the result of an untargeted attack. Furthermore, if we try to perform an untargeted attack, during each iteration the algorithm could target a different class, and therefore we could not analyze its success in a targeted scenario since we do not have a reference class.

Due to these reasons, we implemented a targeted version of an iterative Fast Gradient Sign Method. The details of this algorithm are presented in Section 5.2. After choosing the target class, the noise is generated for the given input, then, after merging the input with the perturbation, the result is submitted as new input to the algorithm. The process is then repeated for a fixed number of iteration before checking the transferability of the final image.

The generation of adversarial images is repeated two times, during first run the target class is selected randomly (but different from the correct one) for each image. These experiments allow collecting enough sample to perform a statistical analysis. The idea is group the test based on the pair original - target label and then extract the best target to attack for a given class. In details, for each label is selected the target that presents the highest percentage of transferability. It is selected by dividing, for each pair of original - target label, the number of images that are adversarial for the target DNN over the number of pictures able to fool the local DNNs. The results of this analysis are then stored in order to use them during the second phase.

During the second iteration, the target is not chosen randomly any more. The same operation as before is now performed over all the images belonging to the validation set. This time, using as target the one extracted from previous analysis. With the same formula of previous iteration, we evaluate the success of the whole attack, without separating cases as done in the first iteration.

5.1 Training the Network

The training process, as previously explained, is used to fine tune the classification function toward the correct outcome. In order to reduce the required time, we applied the technique of transfer learning by using the weights of the pre-trained networks, offered by the python library, as initialization values for my new networks. This allows us to achieve a good accuracy, around 93% over the CIFAR10 dataset, with few iterations.

The values of the key elements used during the training phase are shown below:

- $NumberofEpochs$: all the networks were trained for 20 epochs;
- $Optimizationfunction$: as optimization function was used a Stochastic Gradient Descent;
- $LinearRegression$: a value of 0.01 was used for the linear regression parameter;
- $Momentum$: a value of 0.9 was used for the momentum parameter;
- $BatchSize$: the data were organized in group of 8 images, mainly to have continuity between image and face recognition;

- *LossFunction*: as loss function a Cross Entropy algorithm was applied.

Each model was trained over different portion of the dataset, randomly chosen to keep a good distribution of the class. Four different percentage were selected to carry on the experiments: 12.5%, 25%, 50% and 100%. Note that using too few images of the dataset does not allow the network to reach a good accuracy and this leads to a non-transferability of the adversarial examples. On the other hand, we considered irrelevant using a percentage higher than 100% (i.e. having more image for the training of the local DNN than for the target one) for two unrelated reasons. The first is that for an attacker would be unlikely to have access to a bigger dataset that the one used by a owner of the target DNN. The second, instead, is the idea that better fine tuning our local network with respect to the target one should not lead to better results, since the algorithm would try to exploit some peculiarities that are not shared by the target network. Moreover, those percentages of images were extracted only regarding the training set, since the validation set do not affect the training phase of the networks.

The time taken by the training phase is strongly dependent by the dataset size and the complexity of the model involved. This behaviour is noticeable in the image recognition case where the networks present different training times. For example, comparing the training time of the Googlenet with the one of the Alexnet, we can see a difference of approximately 30 minutes. Indeed, the first took 56 minutes to train while the second one only 28 minutes. On the other hand, when training the face recognition models, since the new dataset present a much smaller number of images, the training time is reduced to 17 minutes even if the complexity of the models increased. Moreover, in this case all the models take a similar time to train, probably due to a more similarity in the complexity of the networks.

Usually when training a network some transformation, such as rotation and normalization of the colour, are applied to the inputs. This helps the network to better generalize over the training data and should avoid over-fitting. Note that applying a normalization over the colours is not suggested when the final goal is to generate adversarial example. Indeed, the generation of adversarial examples perform worse when working over the normalized images. Moreover, it requires to normalize every value used during the process and it achieve a smaller transferability by 5% on average in the tested cases.

5.2 Generating the adversarial example

The process of generating an adversarial example is schematized in Figure 6.4.

The process is composed by two main steps, creation and verification. The first part (blue components) is responsible to generate the adversarial example, and, after a successful attack, the perturbed image is transferred to the target network (red component) to test its transferability.

The first step is to load the data; an important parameter of this process is the batch size. Indeed, since we need to work on the single image when creating the perturbation, we will use a batch size of 1. This approach is not mandatory, but it allows the code to be more understandable.

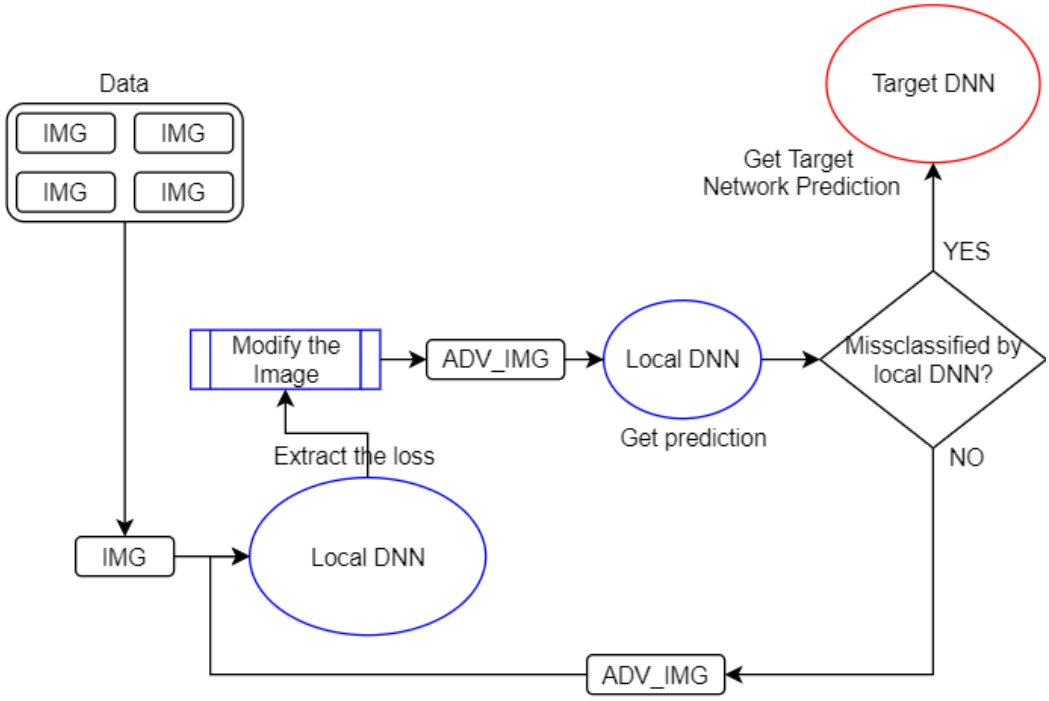


Figure 5.1: Graph representing the process of generating and testing an adversarial image

After loading the data, we submit them to the local DNN and verify that the classification is the correct one. This process is useful since in this way we can eliminate the cases that are already adversarial, which would affect the analysis.

After this check, the image it is submitted to the local DNN and the loss is extracted. As loss function, we used the Pytorch implementation of the Negative Log Likelihood solution. Since we want to perform a targeted attack, in order to generate the noise, we will not evaluate the loss function between the image and its original class, but between the image and the chosen target class.

The next step is to apply a modified implementation of an Iterative Fast Gradient Signed Descent (I-FGSD) [2]. The basic algorithm consists of using the sign of the loss to perturb the image. In other words, the idea is to use rounded value of the loss over each pixel to the closest value between -1, 0 or 1. Then this value is multiplied to the maximum allowed noise, equivalent to the value of σ and it is merged with the image. In the context of performing a targeted attack, we will have to subtract the noise from the original picture. After it a boundary check of all the noise introduced is performed by subtracting the original image to the new one.

Then, the new image is submitted to the local network to see if it is adversarial or not. In case of success, the loop is early terminated, and the image is submitted to the target network, otherwise the result became the input of the next iteration. A maximum number of iterations is set to avoid too many repetitions. The maximum number of iterations is set to 50, even when using the optimized algorithm.

When transferring the adversarial image to the target network we can expect three different result: a successful targeted attack, a successful untargeted attack or a failure. Each experiment is repeated four different times, with four different values of σ, which allow us to understand how the quantity of noise affects the result. Moreover, the four cases allow respectively a perturbation of 0.01, 0.05, 0.15 and 0.3 on the value of the pixel. This value has tu be considered as the difference between the difference between the original value of the pixel, and the modified one.

The code responsible for this phase is reported in Section A.2.

5.2.1 Optimizations

An important contribution to the results was given by the solution presented by the winners of the NIPS 2017 competition on adversarial attack [39]. They propose a slight modification of the previously presented algorithm that strongly increase the transferability of the attack. In particular, the first suggestion affects the calculation of the loss. Indeed, instead of using a single loss, they suggest merging together more losses. In this context, they present a solution in which the loss extracted from each model is merged together with others through a weighted sum. Note that the addiction of the coefficient has to add up to one, in order to avoid introducing too much noise. The idea behind this approach, is to create a perturbation that is less dependent from the single network and more related to the image. Indeed, merging together two or more losses, arising from different networks, should reduce the influence proper of the single network over the final perturbation. Moreover, every network calculates the loss based on its weights and on its model structure, therefore using different models help to create a noise that is more related to the image itself than how this input is classified by the single network.

Regarding the choice of the weights, if the attacker has some knowledge regarding the target DNN he should modify the weights to increase the influence of the most similar network. In this context, during the first case scenario only the local DNN with the same model of the target one was keep in consideration for the generation of the loss. When working in the other two scenarios of my experiment, we use a coefficient of 2/3 for the main network and 1/3 for the auxiliary one.

Another improvement is related to the creation of the noise itself. Indeed, instead of using the sign function, the loss value is rounded to the closest integer and then is clamped between -2 and 2. Moreover, they suggest introducing a coefficient when adding the perturbation to the image. This coefficient is easily evaluated as

$$\alpha = \sigma / i$$

where σ represent the total noise that is allowed in the process and i the total number of iterations. The introduction of this momentum helps to escape local optima and stabilize update direction. Due to this modification, we removed the early termination condition in order to always introduce all the allowed noise, since during the test more noise was a sign of more transferability. Therefore, the success of attack is tested only after the last iteration.

In Figure 5.2 is possible to see a visual representation of the optimization that highlights the main differences (green components) with respect to the basic solution.

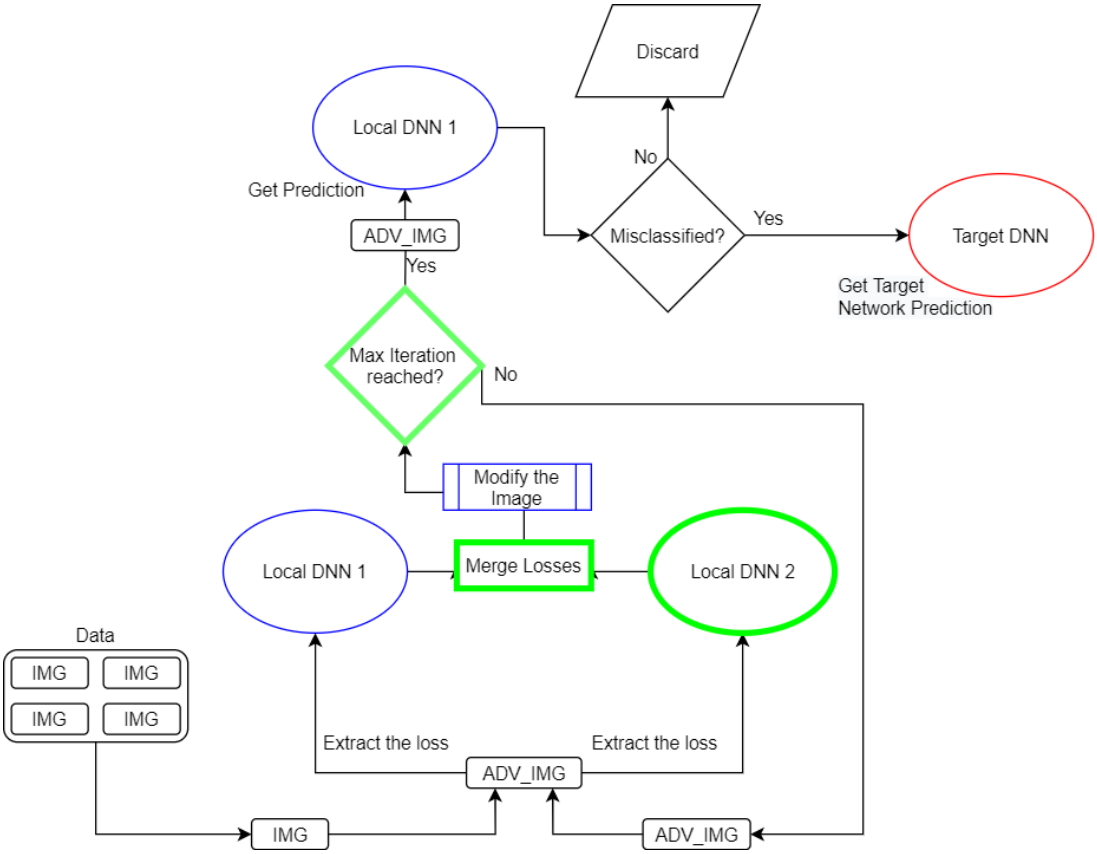


Figure 5.2: Graph representing the optimized process of generating and testing an adversarial image

During the experiments we tested both approaches (tradition and optimized) and, due to the better results of this last solution, all the experiment have then uses this methodology when creating the adversarial examples.

Regarding the elapsed time needed to generate the adversarial attack, with the machine setting previously presented, this solution takes an average of 6 and half seconds for each image for the whole process. For first scenario, were the loss is evaluated only from one model, the time reduces under 3 seconds, this solution correspond also to the basic implementation of the algorithm.

5.2.2 Extracting the best targets

As previously mentioned, the process that creates adversarial examples is repeated twice, the first time, the target class is chosen randomly for each image while the second one uses the analysis of the first one to chose the target. Since the final goal is to generate a targeted attack finding the most similar class plays an important role.

In this context, we performed the first iteration to collect enough data and to extract

the best target for each class. Note that these analyses are always performed on the validation set to make the classification of the image more independent from the network. In order to achieve this, for each pair original - target class, we evaluated the number of successful attacks in the target network, over the number of successful attack in the local network. Then, for each class, the the pair with the highest ratio is chosen, and the corresponding class became the best target. The operation is repeated for all the classes of the dataset. This led to the creation of an array, which for each class allows to retrieve the best target and to use it during the creation of the adversarial attacks. This approach should identify the target class that is most similar to the starting label. Indeed, as explained by Papernot [33], the less the noise introduced the higher the attack transferability.

During the experiment we repeated the generation of the adversarial attack only twice, once for the image recognition dataset and one for the face recognition dataset. Even if the process could be theoretically repeated for each iteration of the experiment, we choose to create the best target by analyzing only one experiment (with all values of σ) due to the long time required to generate a sufficient quantity of data.

The results of my analyses are represented in the following table where on the first line are listed all the classes, while on the second raw the best target used during the experiment. Table 5.1 correspond to image recognition challenge, while Table 5.2 report the result of face recognition analyses. In Appendix B is possible to find a visual representation of the different pairs of original - target class.

Original Label	0	1	2	3	4	5	6	7	8	9
Target Label	2	9	6	2	2	3	4	4	0	3

Table 5.1: Correspondance between Original label and target one for image recognition executions

Original Label	0	1	2	3	4	5	6	7	8	9
Target Label	4	2	5	4	0	2	5	3	3	0

Table 5.2: Correspondance between Original label and target one for face recognition executions

Chapter 6

Transferring an Adversarial Attack

This chapter presents and analyses the results of the tests. It is organized into two sections, that focuses on image and face recognition challenges respectively. Moreover, for each section the different scenarios are presented independently but with a comparison between each other in order to highlight similarities and differences between them.

To evaluate the success of the attack, for each run, we collected the number of images that were able to carry on a targeted attack on the local DNN. Moreover, those perturbed images were then transferred to the target network. This could lead to three different results, two of them are considered successful while the last one is a failure.

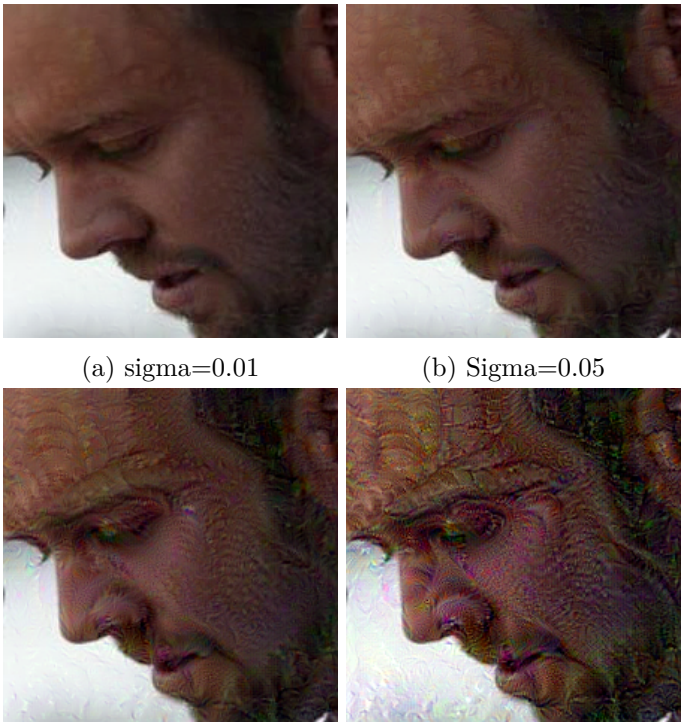
- the perturbed image is considered a successful untargeted attack when the adversarial network miss-classify the input with no limitation regarding the class;
- the perturbed image is considered a successful targeted attack when the adversarial network miss-classify the input with the desired class;
- the perturbed image is correctly classified and the attack failed.

As is possible to understand by previous definition, the untargeted version will always present better result than the targeted one, since it also include the results of the second case.

For each experiment two graphs are reported: one for the targeted attack and another for the un-targeted version. All the graphs presented, with the exception of the Black Box scenario, have the same structure. On the x axis there is the percentage of dataset used during the training phase of the local DNN, while on the y axis is the percentage of example that are correctly miss-classified. The colour lines, on the other hand, represent the different value of σ[1] during each experiment (0.01, 0.05, 0.15, 0.3 which are represented by the blue, orange, green and red line respectively). The idea result is the one that achieve

[1] the maximum perturbation allowed for each pixel

the best accuracy with using as less dataset as possible and introducing less noise. Figure 6.1 shows the influence of different value of σ over the same image.



(a) sigma=0.01 (b) Sigma=0.05

(c) Sigma=0.15 (d) Sigma=0.3

Figure 6.1: Example of the different quantity of noise introduced in the experiments.

6.1 Image Recognition

The image recognition analysis were performed over the CIFAR10 dataset with three different network: ResNet18, GoogleNet and AlexNet, as implemented by Pytorch. Note that the first two were used as local DNN while the third one as target network.

As it is possible to see from the following images, even when the noise introduced is at its maximum value, is still possible to recognize the original class of the image. Note that the low quality of the images is due to pre-process. Indeed, each image was originally of 32x32 pixel, and is scaled to 224x224 pixel before generating the noise. This led to a decrease in quality of the images. In Figure 6.2 is possible to see some example of adversarial cases with different level of noise.

6.1.1 White Box Scenario

As expected this experiment is the one that lead to the best results. As is possible to see from Figure 6.3a we achieve a 100% of transferred case when introducing the maximum

(a) Sigma=0.01 (b) Sigma=0.05 (c) Sigma=0.3

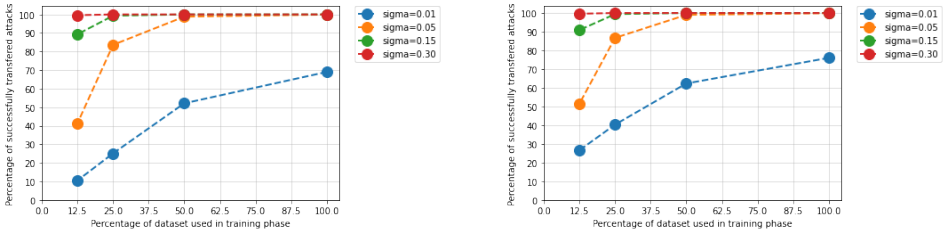
Figure 6.2: Example of the result when applying the algorithm to the image recognition dataset.

noise. On the other hand, if we analyse the least successful case we can notice that we still achieve a transferability of 10% in the worst case and 70% when using the whole dataset. Those result are really significant since they are achieved with a perturbation of only 0.01 which is almost undetectable by the human eye.

Analysing the blue line is possible to see that, when we introduce a small quantity of noise, the dataset strongly affect the result. This is behaviour is also present with a σ value of 0.05 (orange line) when trained with less than 50% of dataset. From this two cases we can deduce that, when the noise is not enough, the transferability is directly proportional to per percentage of dataset. For example, if we look at the blue line at the 12.5% case it shows a transferability equal to only 14% of the results we had at the 100% case. This is probably due to the fact the transferability of a perturbed image, with few noise, is strongly dependent on the difference between the local and the target classifier. In detail, since we use the same model and the same dataset, both networks will learn how to classify the images in a similar way, but using a smaller part of the dataset the local DNN will develop a less tuned classifier that will poorly reproduce the target network. A higher degree of noise, compensate this miss-match giving to the local DNN a higher degree of freedom in the solution space. This allows the network to reach the core of the class and not just rely upon some outliers, that are probably not present in the target DNN due to the randomization in the training process.

Analysing the graph presented in Figure 6.3b is possible to notice that the results of the untargeted attack follow the same pattern of the targeted ones, but with higher success rate. This behaviour, as previously explained, is due to the fact that the two attacks are performed in the same way, but this second version consider a wider range of results as successful. Moreover, due to the good results of the targeted scenario, the untargeted successful cases are a smaller add-on percentage on the targeted one. For example, if we consider the best improvement (blue line at 12.5%) it increase the success rate of 160%, which corresponds to 16 percentage points, but as soon as we use the whole dataset used, still on the blue line, we obtain an increase of only 10%, which correspond to only 7 percentage points. In conclusion, the improvement brought by the untargeted example is

more relevant with smaller dataset and fewer noise.



(a) targeted (b) untargeted

Figure 6.3: Results of the White Box scenario over the image recognition dataset

	100%	50%	25%	12.5%
0.01	69%	52.1%	25.1%	10.5%
0.05	99.9%	98.8%	83.6%	41.3%
0.15	100%	100%	99.3%	89.3%
0.3	100%	100%	100%	99.6%

	100%	50%	25%	12.5%
0.01	76%	62.3%	40.3%	26.6%
0.05	99.9%	99%	86.7%	51.5%
0.15	100%	100%	99.4%	90.9%
0.3	100%	100%	100%	99.6%

Table 6.1: Image Recognition, White Box scenario results: left table targeted attack, right table untargeted attack

6.1.2 Gray Box Scenario

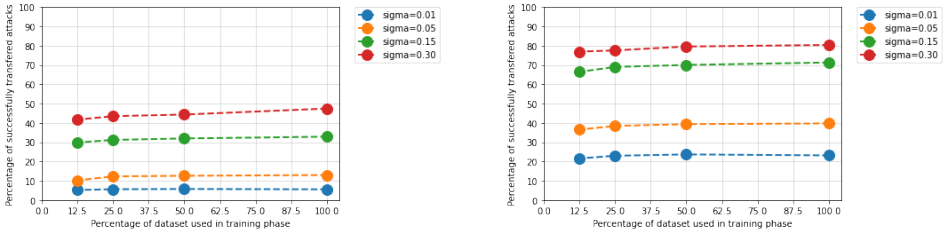
Unlike the previous case, none of the test achieve a 100% of transferability. As is it possible to see from Figure 6.4a, we get an accuracy around 50% in the best case. This is still a really good result since in average it would require only two attempts to fool an adversarial AI system. This result was obtained by the using two networks in the evaluation of the loss. The attack generated in such way, probably focus more on the image and on common aspects of the target class, than on possible outliers. Even after this optimization is unlikely to reach results similar to the one of the White Box scenario, since we do not have any information regarding the target classifier. Unfortunately, as soon as we reduce the quantity of noise the number of transferable adversarial example drastically reduce. In the worst case we achieve a transferability of only 7%. Such a low number is anyway still too high to guaranty security and robustness of an AI system. For example, in a safety related application the errors should be around 0% in order to protect the user life.

Comparing the result of Figure 6.3a with the one of Figure 6.4a we can notice that the line of the second experiment are flatter than the previous one, even without reaching a transferability of 100%. This can be interpreted as the fact that the dataset does not affect the result in the same way. Indeed, in this second scenario, using smaller sub group of the whole dataset generate a drop of only few percentage points. This behaviour can be explained with the fact that we do not know the classifier. Indeed, even in this case

removing data created a less tuned version of the local classifier. Due to the fact the local DNN dose not share the model of the target network, this loss in fine tuning is less relevant since the two classifiers probably differ on more relevant aspects, such as the properties extracted during the classification.

Since, the variability of sigma still influence in a significant way the number of transferable example, this means that the process of generating adversarial solution, works in the same way within the two models.

The untargeted experiment shows better results. Comparing this scenario with the untargeted White Box one, we can notice that untargeted examples significantly increase the successful case. For example, if we consider the results obtained by setting σ equal to 0.01 (blue line) we have a difference of 15 percentage points, which correspond to an increase of the result of 315%. This increment is still significant when considering σ equal to 0.3, where we have a difference of 36 percentage points, with an increment of 167% over the successful case. This results show that if the target is not relevant we can almost achieve the same result as in the White Box scenario. This finding is very relevant e.g. misclassify a gun to another object or lock out a person by classifying face for other people.



(a) Targeted (b) Untargeted

Figure 6.4: Results of the Gray Box scenario over the image recognition dataset

	100%	50%	25%	12.5%
0.01	5.6%	5.7%	5.6%	5.3%
0.05	12.9%	12.6%	12.2%	10.1%
0.15	32.9%	31.9%	31.2%	29.8%
0.3	47.4%	44.3%	43.4%	41.7%

	100%	50%	25%	12.5%
0.01	23.1%	23.7%	23%	21.6%
0.05	39.7%	39.3%	38.4%	36.5%
0.15	71.3%	70%	69%	66.4%
0.3	80.3%	79.6%	77.6%	76.9%

Table 6.2: Image Recognition, Gray Box scenario results: left table targeted attack, right table untargeted attack

6.1.3 Black Box Scenario

Due to the different nature of this scenario, is meaningless to test it with different percentage of dataset. Indeed, since we do not share any part with the dataset of the target network, there is no point trying to use only a smaller percentage of the data that the

attacker has. This would only reduce the accuracy of our network. So, to obtain this results, the networks were trained on different part of the same dataset. Indeed, the two local DNN used 50% of the training set while the target network used the other half. Note that this could be a limitation since the image are all acquired through the same pre-processing, and so are all elaborated in the same way.

Based on the fact that we performed only one experiment for this scenario, the representation of the result used in the previous scenario are not suitable for this one, and therefore the graph are organized in a different way. Figure 6.5 shows on the x axis the maximum perturbation allowed, while on the y axis the percentage of transferable adversarial images. The graph of targeted and untargeted attacker are shown together. As is possible to imagine, the higher line, which correspond to a more successful attack, represent the untargeted version.

As was possible to predict from the Gray Box scenario, the results of this last case are similar to the previous one. Indeed, this scenario is no different from the previous one if we consider the part of the dataset we know as 0% of the target one. Indeed, if we compare this new graph with the previous one, we can see that the result are in line with what we expect by continuing to reduce the known dataset in the previous experiment until 0%.

On the other hand, when we compare the influence of the untargeted version with the one of the Gray Box scenario we can notice that we obtain different result between the two experiments. Indeed, when knowing even a small part of the data we obtain a much higher increment than the one of the last experiment.

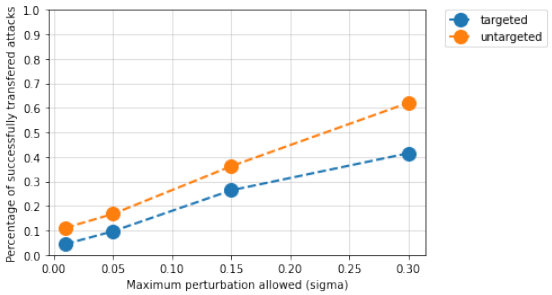


Figure 6.5: Results of the black box scenario over the image recognition dataset

	Targeted	Untargeted
0.01	4.5%	11.1%
0.05	9.7%	16.7%
0.15	26.4%	36.2%
0.3	41.5%	62%

Table 6.3: Image Recognition, Black Box scenario results

6.2 Face Recognition

To work in face recognition area, as already mentioned, was used a derivation of vggface2 dataset. From those data a small partition of only 10 classes was extracted.

The ML models used during those tests were intentionally chosen with a more complex internal structure. This allowed us to evaluate if the higher complexity of the network affects the final result, and to reach a good accuracy even when using a more complex dataset. This would grant us the possibility to have a more direct comparison with future experiments.

The results generally present less transferability percentage with respect to the image recognition. This is probably to the increased complexity of the network. Indeed, we ran few test with the new dataset, but with the model used in image recognition and we obtained similar result to previous ones.

In Figure 6.6 are reported some example of the results obtained by the face recognition experiments. As we can see, the image are much clearer then the one in Figure 6.2. This is due to the fact that the original images have a higher resolution, and therefore the standardization in the number of pixels, applied in the pre-processing phase, does not affect the quality.

(a) Sigma=0.01 (b) Sigma=0.05 (c) Sigma=0.3

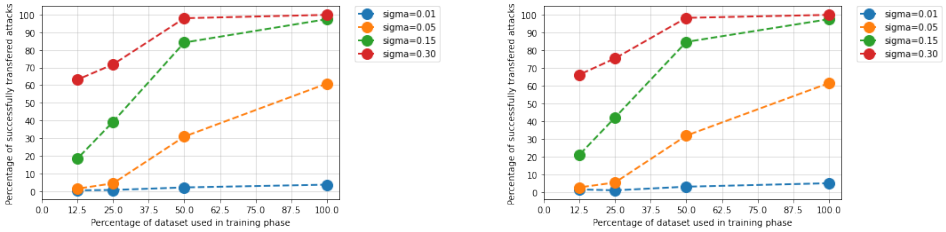
Figure 6.6: Example of the result when applying the algorithm to the face recognition dataset.

6.2.1 White Box Scenario

Regarding the white box scenario we can identify similar pattern with the one obtained by the image recognition experiments. Comparing Figure 6.7a and Figure 6.3a we can see a similar behaviour between the two graph. Moreover, a first view can show that the previous experiments were leading to better results. This can be justified by the introduction of a more complex network. Indeed, increasing the complexity of the model, we also increased the number of element required to fool the network. In addiction, these changes also increased the number of random elements of the network, which will increase the difference between the weights of the two networks, even if they are trained on the same dataset.

On the other hand, analysing the influence of the training set over the successful results, we can see the same behaviour as in the image recognition area. In this case the influence is evident with almost all different values of σ. Indeed, even the red line, which is the only one that reaches a value close to 100%, is strongly dependent on the dataset, as is possible to see by analysing the drop in transferable cases when passing from the use of 50% to only 25%. The blue line, on the other hand, seems to not be affected by the dataset. This is probably due to the fact that when having to few successful cases, which are much more liable to external noise and do not show correlation with the dataset.

Regarding the untargeted version of the attack, Figure 6.7b, we do not see any substantial improvement, even where we have only few successful cases. This could be explained by the fact that creating a targeted attack for the new network requires a higher number of modifications, which reduces the transferability, as previously explained.



(a) targeted (b) untargeted

Figure 6.7: Results of the White Box scenario over the face recognition dataset

	100%	50%	25%	12.5%
0.01	3.7%	2%	0.7%	0.5%
0.05	60.9%	30.9%	4.4%	1.4%
0.15	97.4%	84.1%	39.3%	18.4%
0.3	99.8%	97.9%	71.7%	63.1%

	100%	50%	25%	12.5%
0.01	4.9%	2.9%	0.9%	0.1%
0.05	61.3%	31.7%	5.3%	2.5%
0.15	97.3%	84.6%	41.9%	21%
0.3	99.8%	98.1%	75.2%	66.1%

Table 6.4: Face Recognition, White Box scenario results: left table targeted attack, right table untargeted attack

6.2.2 Gray Box Scenario

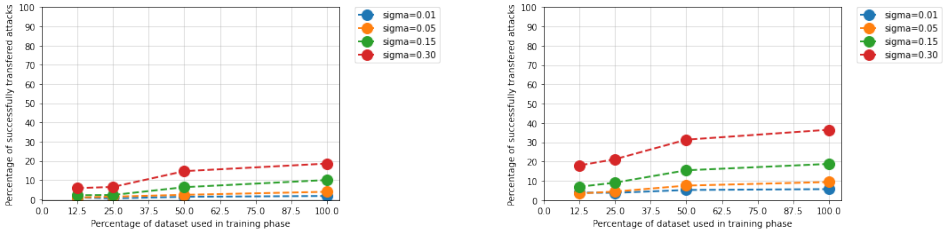
This scenario, as pointed out in the image recognition section, show lower transferability and a lesser influence of the dataset. Indeed, if we analyse Figure 6.8a is possible to notice that we do not even achieve a 20% of transferability in the best case scenario. It was possible to foresee this behaviour by analysing Figure 6.7a, where the blue line do not achieve results higher than a 5% of transferability, and therefore, based on the image recognition results, this scenario should not achieve better results for the same value of σ

On the other hand, comparing these results with the one reported in Figure 6.4a, we can see that the quantity of noise has a lower impact. This is probably due to the fact that

with a more complex network is harder to understand how to modify the image to fool the target network. In other words, the problem is no more the limitation of introducing to few noise, but the fact the attack is probably not targeting the right pixels.

Regarding the influence of the dataset, in Figure 6.8a we can appreciate a higher influence of the data when passing from the usage of 50% of the dataset to only 25%, on the red line. In general, comparing this scenario with the one presented in subsection 6.1.2, we can see a higher influence of the dataset. This could be explained by the fact that more complex network present a higher dependency on the dataset, when creating the solution space.

The untargeted analysis of the attack shows similar improvement, from a relative perspective, to the corresponding scenario in image recognition field, when considering the best case. Indeed, both situations present an improvement of about 100%. On the hand, for lower value of σ we have a much less improvement of the results, showing that to fool a more complex network we require a higher perturbation even if we want to perform an untargeted attack.



(a) targeted (b) untargeted

Figure 6.8: Results of the Gray Box scenario over the face recognition dataset

	100%	50%	25%	12.5%
0.01	1.7%	1.3%	0.4%	0.1%
0.05	3.9%	2.3%	1.3%	0.9%
0.15	10%	6.3%	2.2%	2.1%
0.3	18.5%	14.6%	6.4%	5.8%

	100%	50%	25%	12.5%
0.01	5.7%	5.2%	3.8%	3.9%
0.05	9.4%	7.5%	4.4%	3.5%
0.15	18.7%	15.4%	9.1%	6.9%
0.3	36.5%	31.3%	21.2%	18%

Table 6.5: Face Recognition, Gray Box scenario results: left table targeted attack, right table untargeted attack

6.2.3 Black Box

As pointed out in the image recognition field, the last scenario can be considered a particular case of the Gray Box one where we know 0% the dataset. Moreover, the results are presented with the same structure of Figure 6.5. As is possible to see we achieve a maximum transferability of 5%, which is coherent with the results of previous experiment.

Comparing Figure 6.9 with Figure 6.5, is possible to see a graphical representation of the fact that with the new models the value of σ is less relevant. Indeed, the gradient of the curves gives a visual representation on how the maximum noise affect the result.

Regarding the results of the untargeted attack, we can see that we have an increment of 100% with respect to the targeted version. This increment is higher than the one obtained in the image recognition field, Figure 6.5, but we have to keep in mind that the results are much lower and therefore it is easier to lead to higher improvement. Indeed, if we consider the absolute values of these improvements by comparing the absolute values we increase of only 5 percentage points in Figure 6.9 while we achieve an increment of almost 20 percentage points in Figure 6.5.

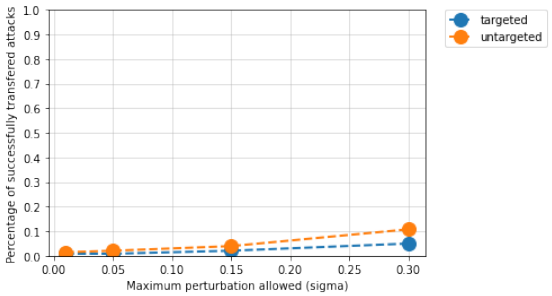


Figure 6.9: Results of the black box scenario over the face recognition dataset

	Targeted	Untargeted
0.01	0.8%	1.4%
0.05	0.8%	2.1%
0.15	2.1%	3.9%
0.3	5%	10.8%

Table 6.6: Face Recognition, Black Box scenario results

Chapter 7

Conclusion

This chapter summarizes the results and shows possible future works.

7.1 Research contribution

In this thesis, we presented how the knowledge of the attacker affects the transferability of adversarial examples. In detail, we analysed how knowing the dataset affects the transferability of adversarial attacks, within different network scenarios. Moreover, during the research we replicated the analyses over different datasets and different network models, which allowed us to discover the relevance of the network model in the transferability of the attacks.

Regarding the core analysis, we divided it in three scenarios: the attacker knows the dataset and the model network of the system that he wants to attack (White Box), he only knows the dataset used in the training phase (Gray Box), or he does not know any detail regarding the target network (Black Box). As is possible to see from the Chapter 6, the influence of the dataset set is similar in both face and image recognition challenges. In this context, analyzing the various case individually, is possible to notice that knowing the dataset mainly affects the White Box scenario, as in this case we have almost a linear dependency between the percentage of dataset we know and the transferability of the attack. On the other hand, the knowledge of the dataset became almost irrelevant when we do not know the target network model. Indeed, the Gray Box scenario presents a series of almost constant lines, which explain also why the Black Box case presents similar results to this one.

To perform the test in the face recognition area, we used more complex models, in order to grant a continuity in case of future work. Comparing the two sections of Chapter 6, which report the result of image and face recognition area respectively, is possible to see how the complexity of the network affect the results. Indeed, as it is possible to see by comparing Figure 6.3a and Figure 6.7a, even the best case scenarios differ substantially, mainly when introducing fewer noise. Due to the fact that the two datasets present the same number of classes and the picture are normalized to the same size, this differences are to be attributed to the increased complexity in the network models.

From these analysis we can conclude that the dataset has a minor relevance in the transferability of adversarial examples, unless the attacker has a full knowledge of the target network model. Regarding the value of sigma, we can conclude that it has a high impact when the attack is able to identify the correct pixels to be changed, as in the image recognition experiments. Regarding the network, we can conclude that the knowledge of the model affect the results, but its complexity plays a major role in the transferability of adversarial examples and further analysis could lead to a better understanding of the motivation behind this behaviour.

7.2 Next steps

There are some elements that could be interesting for more in-depth research. We first present some ideas on how to modify the presented research, and then some suggestion that could lead to new research.

The first idea regards the dataset. During the analysis we always used a limited number of classes. Since the network divides the whole solution space into the available classes, increasing the number should affected the transferability of an adversarial attack. Moreover, we think this change should affect only the targeted version which would require a more precise attack to target the correct class. But the untargeted version should compensate this loss, since the noise should still be sufficient to lead to a miss-classification.

Further work could focus on the algorithm used to generate the adversarial attack. Indeed, during my analysis we used an Iterative FGSM algorithm. This method is not the only one that allow us to generate adversarial example, therefore we think that trying other methods could lead into a comparison not only on the absolute number of transferable example, but also with respect to the case we are analysing. Indeed, it could happen that an algorithm better perform on a type of network, or in a specific case scenario.

During my research, we tried to understand if there was a relationship between the generated noise. Moreover, the idea was to find a correlation between the perturbation based on the fact that the images shares the same pair of original - target class. In order to do so, we tried to evaluate the average of the noises, the standard deviation and means of each pixel and we tried to implement a correlation between different noises. Unfortunately, the results did not show any relation, but this could be to the inappropriate mathematical measures. Therefore, a future research could try new approach to this idea, and moreover, could try to apply it again when the noise is generated in a different way, such as with a different algorithm.

The last idea, which we consider would lead to a significant contribution is the idea to transfer the analysis into the real world. As demonstrated in previous work, when moving the attack to a real world scenario, we have to keep in consideration some extra factor that affect the success of an adversarial attack. Moreover, we may have some transformation, such as rotation and translation, or also a change in the light. In order to face those problems, the generation of the adversarial attack has to be modified. Therefore, moving the analysis into the real world, would not only show how much the attack is transferable in this new scenario, but it would also offer the possibility to understand how the changes in the algorithm affect the transferability.

Appendix A

Code

A.1 Training Algorithm

```
def train_model(model, criterion, optimizer, scheduler,dataloaders,
      dataset_sizes, num_epochs=20):
    since = time.time()

    best_model_wts = copy.deepcopy(model.state_dict())
    best_acc = 0.0

    for epoch in range(num_epochs):
        print('Epoch {}/{}'.format(epoch, num_epochs - 1))
        print('-' * 10)

        # Each epoch has a training and validation phase
        for phase in ['train', 'val']:
            if phase == 'train':
                model.train()  # Set model to training mode
            else:
                model.eval()   # Set model to evaluate mode

            running_loss = 0.0
            running_corrects = 0

            # Iterate over data.
            for inputs, labels in dataloaders[phase]:
                inputs = inputs.to(device)
                labels = labels.to(device)

                # zero the parameter gradients
                optimizer.zero_grad()

                # forward
                # track history if only in train
                with torch.set_grad_enabled(phase == 'train'):
                    outputs = model(inputs)
                    _,preds = torch.max(outputs, 1)
```

```
                    loss = criterion(outputs, labels)

                    # backward + optimize only if in training phase
                    if phase == 'train':
                        loss.backward()
                        optimizer.step()

                # statistics
                running_loss += loss.item() * inputs.size(0)
                running_corrects += torch.sum(preds == labels.data)
            if phase == 'train':
                scheduler.step()

            epoch_loss = running_loss / dataset_sizes[phase]
            epoch_acc = running_corrects.double() / dataset_sizes[phase]

            print('{} Loss: {:.4f} Acc: {:.4f}'.format(
                phase, epoch_loss, epoch_acc))

            # deep copy the model
            if phase == 'val' and epoch_acc > best_acc:
                best_acc = epoch_acc
                best_model_wts = copy.deepcopy(model.state_dict())


    time_elapsed = time.time() - since
    print('Training complete in {:.0f}m {:.0f}s'.format(
        time_elapsed // 60, time_elapsed % 60))
    print('Best val Acc: {:4f}'.format(best_acc))

    # load best model weights
    model.load_state_dict(best_model_wts)
    return model, best_acc
```

A.2 Adversarial Example Generation

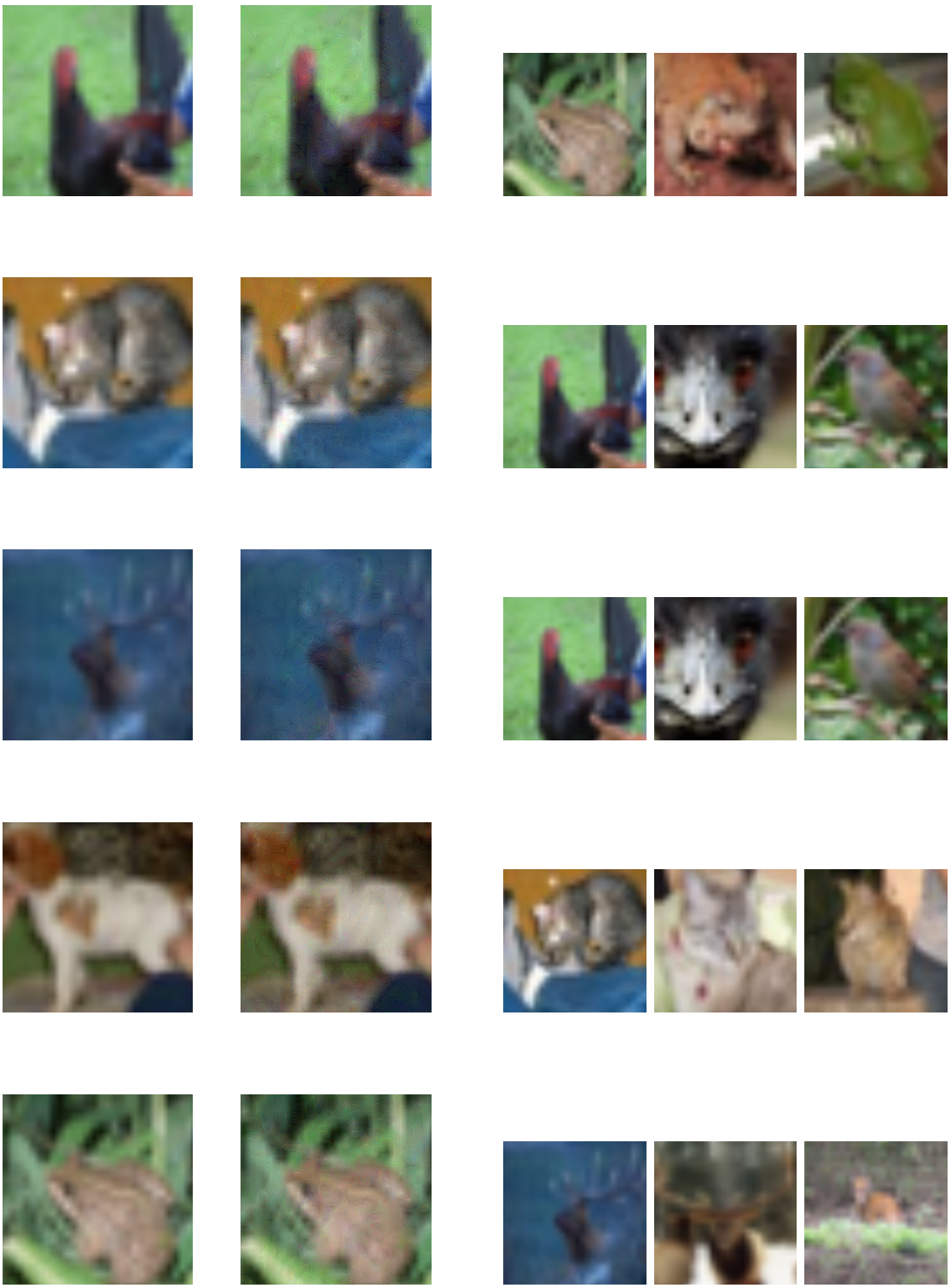
```
def test( model, model2, model3 , device, test_loader, class_len,epsilon ,attemps = 50 ):
    # Accuracy counter
    correct = [0]*class_len
    adv_examples = []
    entered = 0
    transferable = [0]*class_len
    untargeted_adv = [0]*class_len

    # Loop over all examples in test set
    for data, label in test_loader:

        fake_label = torch.tensor([best_target[label]])

        # Send the data and label to the device
        data, label, fake_label = data.to(device), label.to(device), fake_label.to(device)

        # Set requires_grad attribute of tensor. Important for Attack
        data.requires_grad = True

        # Forward pass the data through the model
        output = model(data)
        init_pred = output.max(1, keepdim=True)[1] # get the index of the max log-probability
        # If the initial prediction is wrong, dont bother attacking, just move on
        if init_pred.item() != label.item():
            continue
        entered += 1
        adv_img = data.clone().detach()
        old_grad = 0
        #print(torch.min(data),torch.max(data))

        for i in range(attemps):

            # Set requires_grad attribute of tensor. Important for Attack
            adv_img.requires_grad = True

            # Merge Loss of multiple noises
            output = model(adv_img).mul(2/3)
            output2 = model2(adv_img).mul(1/3)
            output = output.add(output2)

            # Calculate the loss and create the noise
            loss = F.nll_loss(output,fake_label).to(device)
            grad = torch.autograd.grad(loss,adv_img,retain_graph = False, create_graph = False)[0]
            grad = grad/torch.std(grad)
            alfa = epsilon/attemps
```

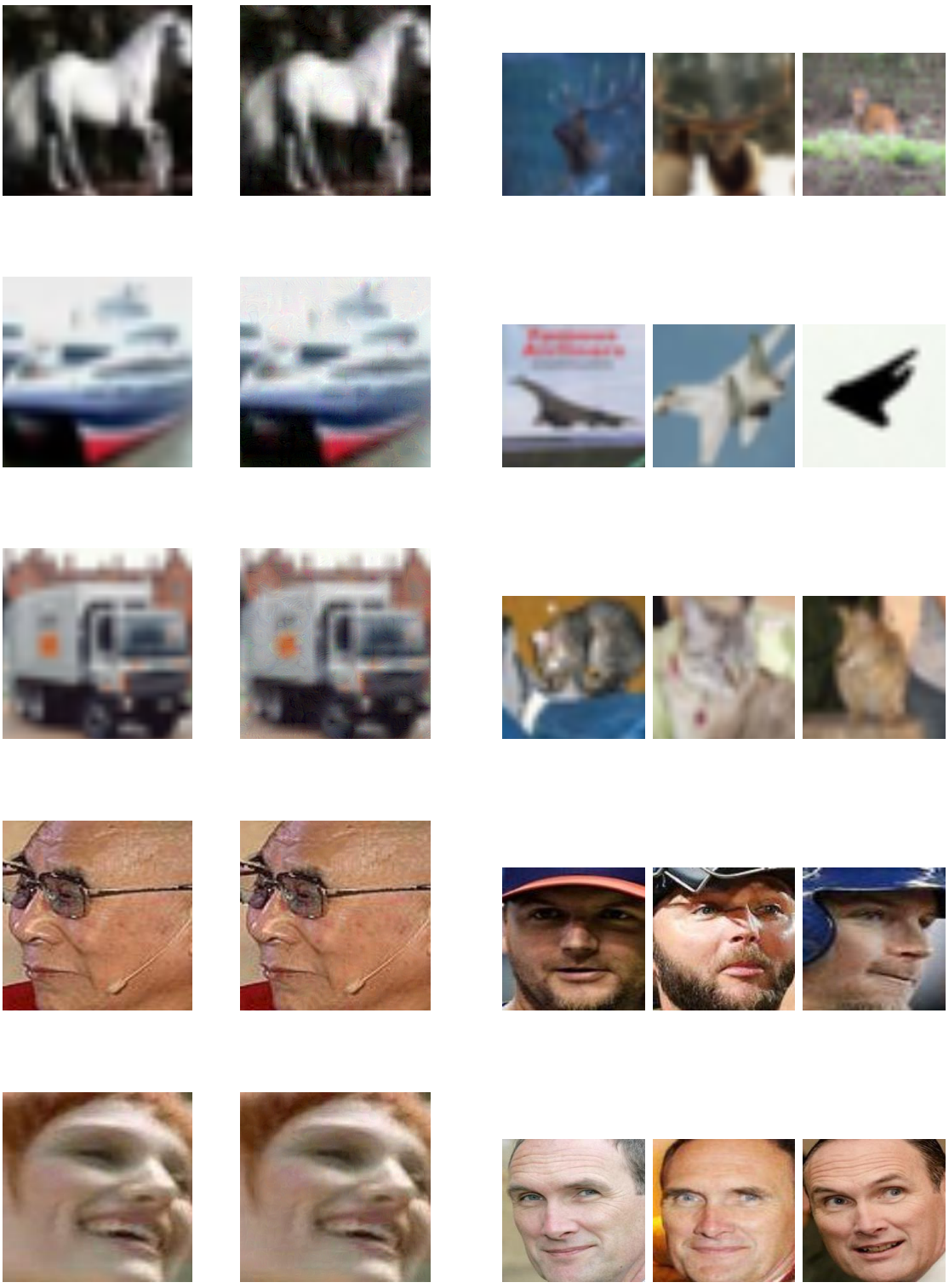
```
47              cost = torch.clamp(torch.round(grad.add(old_grad)), min=-2,
    max=2)
48
49              # Update the adversarial image
50              adv_img = adv_img.detach() - alfa * cost
51              adv_img = torch.clamp(adv_img, min=0, max=1).detach()
52              old_grad = grad.detach()
53
54          # Check for success
55          delta = torch.clamp(adv_img.data - data.data, min=-epsilon, max=
    epsilon)
56          adv_img = data + delta
57          adv_img = torch.clamp(adv_img,min= 0,max = 1)
58          output = model(adv_img)
59          _, final_pred = torch.max(output, 1) # get the index of the max
    log-probability
60          if final_pred.item() == fake_label.item():
61              # If the images is adversarial for the training network 1 is
     tested on the second
62              correct[label.item()] += 1
63              #adv_img = normalize(adv_img.squeeze()).unsqueeze(0)
64              output = model3(adv_img)
65              _,preds = torch.max(output,1)
66              if preds == final_pred.item():
67                transferable[label.item()] += 1
68              elif preds != label.item():
69                untargeted_adv[label.item()] += 1
70
71     print("Epsilon: {}\tTest Accuracy = {} / {} = {}\t Transferable=
    {}/{}\t StillAdv= {}/{}".format(epsilon, correct, entered, 100,
    transferable,correct,transferable+untargeted_adv,correct))
72
73     # Return the accuracy and an adversarial example
74     return correct, entered, transferable,untargeted_adv
```

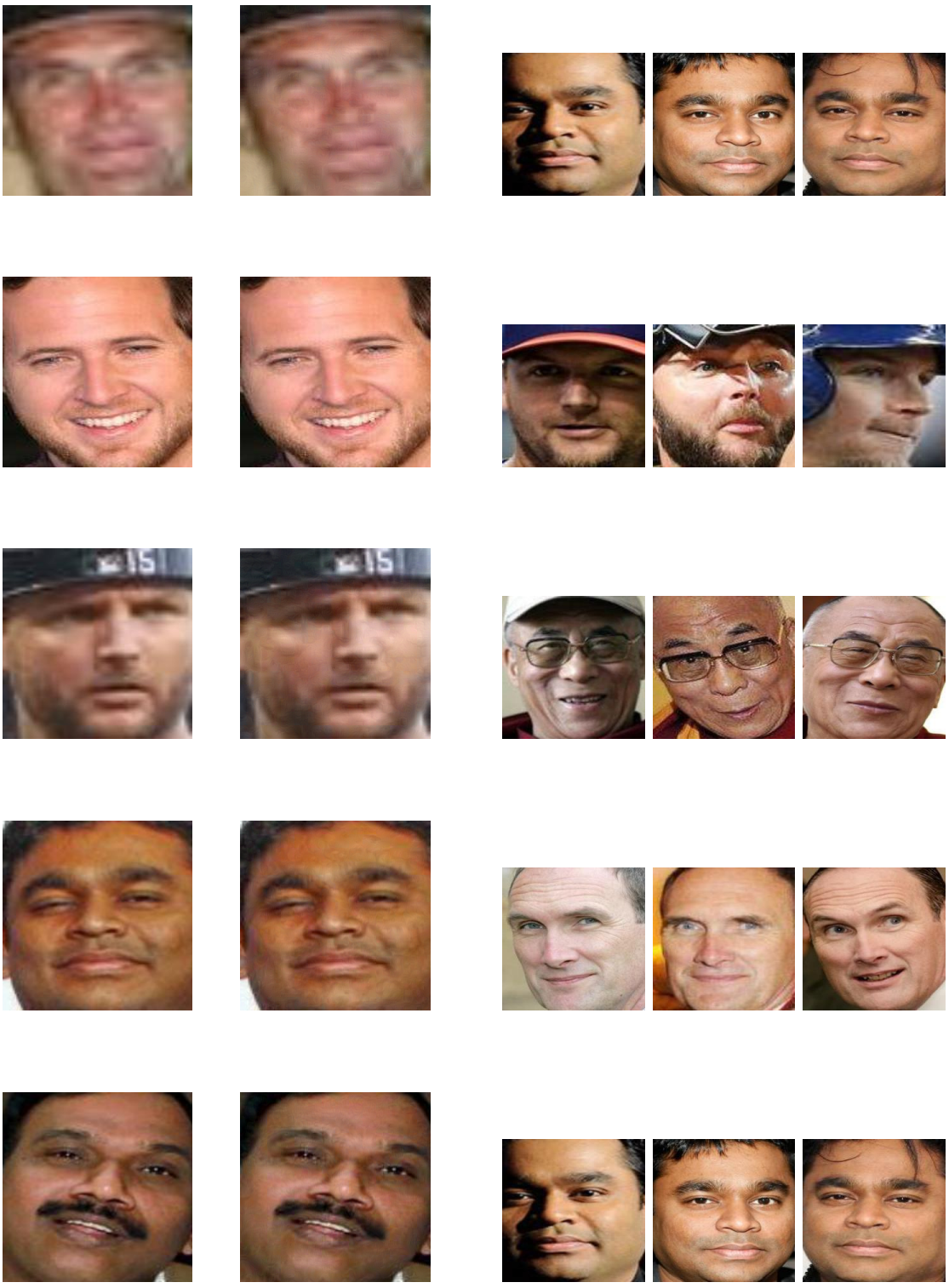
Appendix B

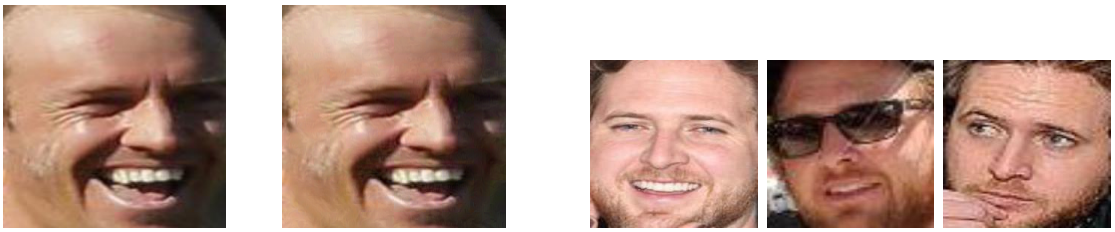
Images

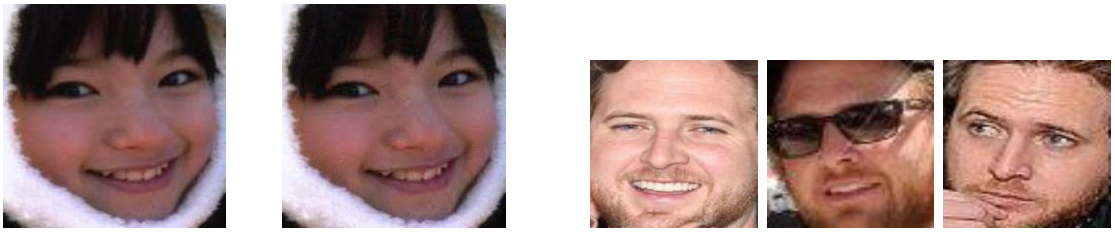
All the group of picture are organized as follow: on the left an image belonging to the clear image, in the middle the modified version with a value of σ of 0.05 and on the right some pictures belonging to the targeted class.

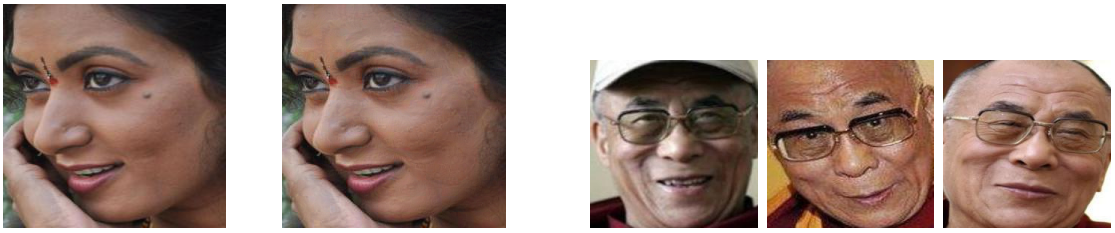












Bibliography

[1] Report of ai research. https://www.theverge.com/2018/12/12/18136929/artificial-intelligence-ai-index-report-2018-machine-learning-global-progress-researc

[2] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[3] Nicholas Carlini. A complete list of all (arxiv) adversarial example papers, 2019.

[4] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 acm sigsac conference on computer and communications security*, pages 1528–1540, 2016.

[5] Miles Brundage, Shahar Avin, Jack Clark, Helen Toner, Peter Eckersley, Ben Garfinkel, Allan Dafoe, Paul Scharre, Thomas Zeitzoff, Bobby Filar, et al. The malicious use of artificial intelligence: Forecasting, prevention, and mitigation. *arXiv preprint arXiv:1802.07228*, 2018.

[6] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25, 2006.

[7] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[8] Florian Tramèr, Pascal Dupré, Gili Rusak, Giancarlo Pellegrino, and Dan Boneh. Adversarial: Perceptual ad blocking meets adversarial machine learning. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2005–2021, 2019.

[9] Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 1–7. IEEE, 2018.

[10] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[11] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1):3–24, 2007.

[12] Robert Gentleman and Vincent J Carey. Unsupervised machine learning. In *Bioconductor case studies*, pages 137–157. Springer, 2008.

[13] Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

[14] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[15] Richard S Sutton. Introduction: The challenge of reinforcement learning. In *Reinforcement Learning*, pages 1–3. Springer, 1992.

[16] Bayya Yegnanarayana. *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[17] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[18] Tensorflow vs pytroch. https://realpython.com/pytorch-vs-tensorflow/.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[20] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[21] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

[22] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[23] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 67–74. IEEE, 2018.

[24] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[25] Inceptionresnetv1 pretrained github. https://github.com/timesler/facenet-pytorch#references.

[26] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[27] How apply transfer learning. https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html, note=.

[28] Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108, 2004.

[29] Parsa Saadatpanah, Ali Shafahi, and Tom Goldstein. Adversarial attacks on copyright detection systems. In *International Conference on Machine Learning*, pages 8307–8315. PMLR, 2020.

[30] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 103–117, 2017.

[31] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[32] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[33] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[34] Malhar Jere, Briland Hitaj, Gabriela Ciocarlie, and Farinaz Koushanfar. Scratch that! an evolution-based adversarial attack against neural networks. *arXiv preprint arXiv:1912.02316*, 2019.

[35] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

[36] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018.

[37] Kaidi Xu, Gaoyuan Zhang, Sijia Liu, Quanfu Fan, Mengshu Sun, Hongge Chen, Pin-Yu Chen, Yanzhi Wang, and Xue Lin. Adversarial t-shirt! evading person detectors in a physical world. *arXiv*, pages arXiv–1910, 2019.

[38] Stepan Komkov and Aleksandr Petiushko. Advhat: Real-world adversarial attack on arcface face id system. *arXiv preprint arXiv:1908.08705*, 2019.

[39] Alexey Kurakin, Ian Goodfellow, Samy Bengio, Yinpeng Dong, Fangzhou Liao, Ming Liang, Tianyu Pang, Jun Zhu, Xiaolin Hu, Cihang Xie, et al. Adversarial attacks and defences competition. In *The NIPS'17 Competition: Building Intelligent Systems*, pages 195–231. Springer, 2018.