University of Padova

Department of mathematics Tullio-Levi Civita

Master Thesis in Data Science

Investigating Adversarial Attacks on Deep Learning Models for RGB Remote Sensing Image Classification

Supervisor
Professor Lamberto Ballan
University of Padova

Co-supervisor
Doctor Name Surname
Co-advisor affiliation

Master Candidate
Elisa Matteligh

Academic Year
2023-2024

Abstract

This is the abstract of the thesis.

Contents

Listing of figures

Listing of tables

Listing of acronyms

NN . . . . . Neural Network

CNN . . . . . Convolutional Neural Network

FGSM . . . . . Fast Gradient Sign Method

BIM . . . . . B Iterative Method

ILCM . . . . . check

1 Introduction

This is the introduction.

The thesis is organized as follows. Some useful examples are detailed in Chapter 2. Concluding remarks are reported in Chapter 6.

2

Adversarial Attacks

Adversarial attacks can be formally defined as: ”inputs to machine learning models that an attacker intentionally designed to cause the model to make mistakes”.

In the domain of image classification these adversarial attacks are intentionally modified images, often with almost imperceptible differences from the original image, but that can mislead the classifier to provide wrong predictions. One of the first and most famous example is shown at Fig. 2.1.

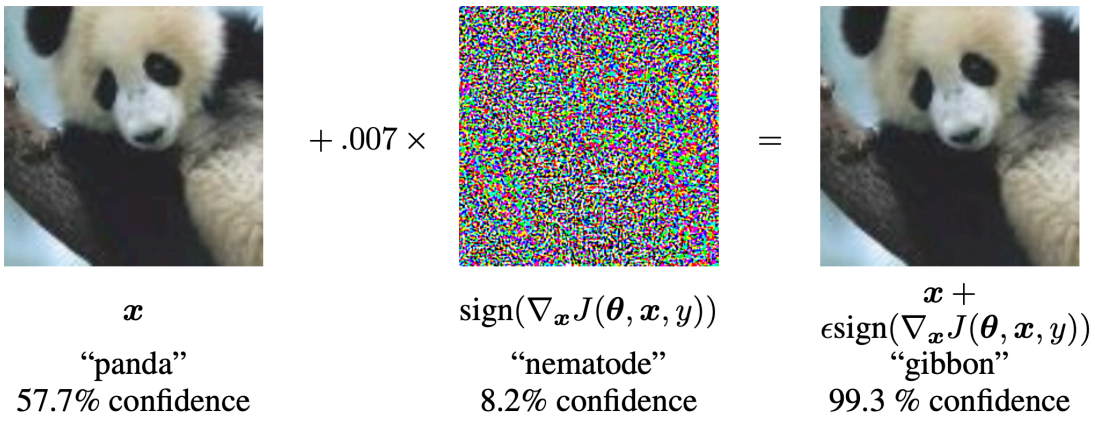

Figure 2.1: A demonstration of adversarial attacks, from [cit].

History and background

These attacks were first described in 2014 by Szegedy et al. [1], who also outlined a method for identifying them. They also observed that the same adversarial example often induce misclassification across a range of classifiers with different architectures and trained on different subsets of the training data.

To construct an adversarial example, a typical procedure is to start with an image that is correctly classified by the neural network and then add a human-imperceptible perturbation to change the predicted label. This is also called evasion attack or test-time attack, since the attacker is trying to add a small perturbation in the test time to make the machine learning model mis-classify without interfering the training procedure.

A more general definition of adversarial attacks are inputs specifically designed to cause a model to make mistakes.

In an increasingly digitized world where reliance on computers and machines is ever-growing, understanding the phenomenon of adversarial attacks becomes of vital importance. These attacks represent a significant challenge to security and must be comprehended.

2.1 Definitions

Adversarial attacks are divided into two main types: white-box attacks and black-box attacks. Additionally, attacks can be further classified as targeted or non-targeted. We will now see each category in details.

Targeted vs non-targeted attack

In a **targeted attack**, when an input image (x, y) is given, where x is the feature vector and $y \in Y$ the ground-truth label of x, the attack induce the model to give the perturbed image x' a specific label $t \in Y$.

On the other hand, in a **non-targeted attack**, the only objective is to induce the model to make incorrect predictions without specifying a particular target label $t \in Y$.

White-box attack

White-box attacks represent the simplest method for constructing adversarial attacks.

In a white-box attack, there is complete knowledge about the victim neural network, including both its architecture and weights. In practice, the white-box assumption is often too restrictive and impractical. Therefore, we also define the black-box scenario.

Black-box attack

In the **black-box scenario**, attackers provide adversarial examples to a targeted model during test time without having knowledge of the model. Various solutions have been proposed to tackle this situation. The two main strategies are:

- to make the problem a white-box setting again (e.g. using a surrogate model);
- estimating the Gradient based on Input-Output pairs (i.e. using the model as an oracle).

In this work we will focus on the first category of black-box methods, which use strategies to go back into the white-box scenario, which is easier to solve.

We are now going to give a more formal definition of adversarial attacks, see the reasons behind their existence, and then how we practically create them.

2.1.1 Formal definition in the white-box setting

Generally, in a white-box setting, for an image (x, y) and a classifier C, the objective is to find an adversarial image that satisfies the following conditions:

$$\begin{aligned} \text{find } x' \text{ satisfying} \quad & \|x - x'\| < \epsilon \\ \text{s.t.} \quad & C(x') = t \neq y = C(x) \end{aligned} \tag{2.1}$$

where $\|\cdot\|$ measure the difference between x and x', typically using a l_p norm bounded by a small value ϵ (which is usually tuned as an hyper-parameter).

2.1.2 Theory behind the existence of adversarial attacks

Adversarial attacks present a significant threat to machine learning models, effectively acting as "hallucinations" for deep neural networks (DNN). Upon their discovery, they prompted a series of questions regarding their nature: why do they occur? Can they be prevented?

In 2014, Goodfellow et al. [2] proposed a theoretical explanation for adversarial attacks in their work "Explaining and Harnessing Adversarial Attacks." They postulate that neural networks are susceptible to adversarial perturbations due to their inherent linearity.

They observed that the architectural design of modern deep neural networks intentionally promotes linear behavior for computational efficiency. Even nonlinear models such as sigmoid networks are "carefully tuned to spend most of their time in the non-saturating, more linear regime" [...] "so they are easier to optimize".

This whole concept is known as the "linear hypotheses".

In high-dimensional settings, such as image classification tasks, even small changes in input can lead to significant variations in output. For instance, consider the dot product between a weight vector w and an adversarial image x':

$$w^T x' = w^T x + w^T \eta \tag{2.2}$$

Here, η represents the perturbation applied to the image. By maximizing η while constraining it within a norm bound, such as

$$\eta = \epsilon \cdot sign(w) \tag{2.3}$$

we can derive

$$w \cdot \eta = \epsilon \cdot w^T sign(w) = \epsilon \left\| w \right\| 1 = \epsilon mn. \tag{2.4}$$

Since $\|\eta\|_\infty$ remains bounded regardless of the problem's dimensionality, while $w^T \eta = \epsilon \cdot m \cdot n$ scales with it, infinitesimal changes to input can result in substantial output variations in high-dimensional scenarios.

Now that we have an idea for the reason of their existence, let's understand how they are generated. In the white-box scenario, which is the one we are firstly going to describe, we possess complete knowledge about the victim model, which we are using to construct the attacks. Subsequently, we will explore black-box attack in Section 2.3.

2.2 Generating adversarial examples

Adversarial attacks, as we have seen, are the solutions of the optimization problem in cit. One of the earliest and perhaps simplest algorithm to solve it was proposed by Goodfellow et al. in their seminal work "Explaining and Harnessing Adversarial Examples" (I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," arXiv preprint arXiv:1412.6572, 2014), and it is known as the **Fast Gradient Sign Method** (FGSM).

Over the years, numerous alternative approaches and innovative ideas have been put forth

in the literature. Given the breadth of these methods, it's impossible to cover each one of them. Therefore, we have taken into the analysis a selection of some of the most renowned, diverse, and promising techniques, and these are the ones we are going to present. We will start with Fast Gradient Sign Method (FGSM).

FGSM

As we have previously mentioned, FGSM is one of the easiest and most intuitive ways to create adversarial attacks.

The gradient is computed with the respect to the input data x, and by taking the sign of the gradient, represented as $sign(\nabla_x J(\theta, x, y))$, we obtain the perturbation η. The sign of the gradient is taken to constraint the perturbation and ensures that the magnitude of η does not exceed a predefined limit.

All sum up, given an image x, its true label y and the parameters θ of the model, the adversarial example x_{adv} can be computed as:

$$x_{adv} = clip(x + \epsilon \cdot sign(\nabla_x J(\theta, x, y)) \tag{2.5}$$

where $\nabla_x J(\theta, x, y)$ is the gradient of the loss function J w.r.t. the input x, computable using back-propagation; $sign(\cdot)$ is the sign function, and $clip(\cdot)$ clips the pixel values of the resulting image to assure they stays in the $[0, 1]$ domain.

Goodfellow et al. have asserted that the max-norm is the optimal choice. However, different interpretations exist, such as those involving the l_2 and l_∞ norms:

$$\begin{aligned} l_2 : x_{adv} &= clip\Big(x + \epsilon \frac{\nabla_x J(\theta, x, y)}{\|\nabla_x J(\theta, x, y)\|_2}\Big) \\ l_\infty : x_{adv} &= clip\Big(x + \epsilon \frac{\nabla_x J(\theta, x, y)}{\|\nabla_x J(\theta, x, y)\|_\infty}\Big) \end{aligned} \tag{2.6}$$

This method perturb the original image by taking a step in the direction that increases the loss of the classifier. To ensure the added noise is not visible, it confine it with a norm constraint. FGSM can be easily improved by the following.

BIM

Basic Iterative Method (BIM) is the iterative version of FGSM. Instead of a single step, BIM takes multiple small steps adjusting the gradient at each iteration. Each iteration is defined as

follows:

$$x_{adv}^{i+1} = clip(x_{adv}^{i} + \epsilon \cdot sign(\nabla_x J(\theta, x, y)) \tag{2.7}$$

where x_{adv}^{i} denotes the perturbed image at the i^{th} iteration. The BIM starts with $x_{adv}^{0} = x$ and runs for a number of iterations which is decided by the user. Moreover, the authors of this method have defined a formula for the "best" number of steps, which is computed as $\lfloor min(\epsilon + 4, 1.25\epsilon \rfloor$.

Kurakin et al. (year) further expanded the Basic Iterative Method (BIM) to the Iterative Least-likely Class Method (ILCM). ILCM is the only target method we will see. The target label of the image is automatically assigned as the least-likely class predicted by the classifier for that image.

One-Pixel attack

In 2.1. we have defined the problem of generating the adversarial attacks as perturbed images such that their difference with the original image is bounded in norm. We explored algorithms using the max-norm and the Euclidean norm (FGSM, BIM and ILCM). An intriguing outcome arises with the utilization of the l_0 norm, and the method which is created is known as the One-Pixel attack. Employing the l_0 norm, indeed, restricts the number of pixels altered to create the perturbed image. We will show an example of this algorithm in (specific section), demonstrating the possibility of executing a successful attack by modifying just a single pixel.

DeepFool

DeepFool is a potent algorithm for generating adversarial examples, introduced by Moosavi-Dezfooli et al. in [reference]. It produces smaller perturbations compared to previous methods. The approach of DeepFool is to identify the minimum perturbation required to push an image across the decision boundary, which represents the boundary of the region in the feature space where the image is assigned its ground-truth label. We remind to for extensive details.

CW

Carlini and Wagner attack [cite] is considered, by its own authors, one of the most powerful adversarial attacks algorithm generator.* The formal definition of the attack is the following:

*And this is true in our case too.

$$\begin{aligned} \text{minimize} \quad & \|x - x + \delta\| \\ \text{s.t.} \quad & C(x+\delta) = t \\ & x\delta \in [0,1]^n \end{aligned} \tag{2.8}$$

Here, $\|\cdot\|$ can be either l_0, l_2, or l_∞, and C represents a classifier. The CW algorithm aims to find the perturbation δ that minimizes the distance between the original and the perturbed image, while ensuring that the perturbed image $x + \delta$ is classified as target label t and the pixel values of δ are bounded within the range $[0, 1]^n$.

The methods we have discussed thus far involve algorithms that generate perturbations on individual images. Typically, a perturbed test set is created by modifying each image with a distinct perturbation, tailored specifically for that image. In contrast, the next method, known as the "Universal attack," operates on multiple images simultaneously. It creates a single perturbation that, when applied to the images in the dataset, should cause the model to misclassify almost all of them. Hence, the term "universal.

This is a powerful method because, as we will see, works both in a white-box and black-box scenario.

2.2.1 Universal attack

This attack was introduced in 2017 by Moosavi-Dezfooli et al. Given a set X of images for which we want to create ad adversarial version, we seek a perturbation δ such that

$$C(x+\delta) \neq C(x) \quad \text{for "most" } x \in X. \tag{2.9}$$

This perturbation δ is referred to as *universal*, as it represents a fixed, image-agnostic perturbation. Formally, δ must satisfy the following constraints:

$$\begin{aligned} &1. \quad \|\delta\|_p \leq \epsilon \\ &2. \quad \mathcal{P}(C(x+\delta) \neq C(x)) \geq 1 - \gamma \end{aligned} \tag{2.10}$$

Here, $\mathcal{P}$ denotes a probability, $x \in X$ represents images in a sample space, and γ quantifies the desired proportion of misclassified images (i.e. the *fooling rate*).

Fooling Rate. *The fooling rate serves as a quantitative metric for assessing the effectiveness of adversarial attacks (in general, not only for the universal). It is calculated as the ratio between*

the number of examples where the adversarial attack successfully changes the class label and the total number of examples that were originally correctly classified.

Intuitively, when considering images x_i and their corresponding classification regions R_i, the algorithm, based on $x_1, x_2, and x_3$, manipulates the perturbation η in such a way that the resulting perturbed images $x_1 + \eta$, $x_2 + \eta$, and $x_3 + \eta$ lie outside their respective classification regions. A schematic representation from the original paper is given at Fig. 2.2.

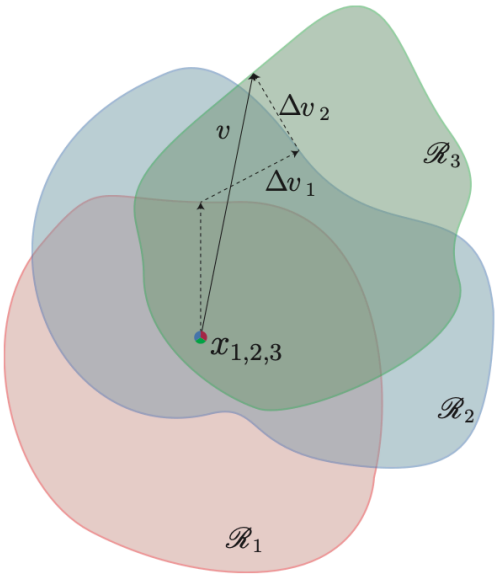

Figure 2.2: Schematics representation of the Universal adversarial attack algorithm.

2.3 Black-box adversarial attacks

We have seen the methods for white-box adversarial generation, we are now going to see some method for the black-box setting, i.e. where we don't have full knowledge about the model. Generally, finding adversarial attacks in this context poses a greater challenge; however, extensive research has been conducted in this area too. As mentioned earlier, black-box attacks can be created using two main approaches: by simulating a white-box scenario or by leveraging input-output pairs. In this discussion, we will focus solely on the first one.

Moreover, recreating a white-box scenario may be done in two ways:

- Approximating the gradient

- Using a surrogate model and the "transferability" property

The second approach, which uses a surrogate model, follows into the category of *transfer-based black-box attacks*.

The main idea behind a transfer-based black-box attack is to create white-box attacks on a surrogate model and adopt the generated adversarial examples to attack the unknown victim model. The more similarities are shared between the two models, the more effective the attack might be.

The phenomenon that adversarial examples generated using white-box attacks will sometimes successfully attack an unrelated model is known as "transferability".

Transferability is a crucial property of adversarial examples, and it was discovered by Szegedy et al. in cit.

Transferability. *The transferability property of adversarial attacks means that the adversarial examples generated to target one model also have a probability of misleading other models.*

As we will see in Chapter 3, directly transferring traditional attack methods can hardly achieve satisfactory performances on the unknown model. We will study the effectiveness of the various white attacks we will create and see when they fail.

To enhance transferability, we will consider the following approaches.

- **Ensemble Method** This strategy is supported by the findings of Liu et al., who observed that ensemble learning enhances the generation of more transferable adversarial attacks
- **Attack on intermediate logits** Zhou et al. and Huang et al. discovered that attacking intermediate layers of the network is more potent than attacking the predicted logits

Ensemble

An ensemble attack with $k \in \mathbb{N}$ models is defined, in general, as follows:

$$\begin{aligned} \text{maximize}_{x'} \quad & J\left(\frac{1}{M}\sum_{k=1}^{M} f_k(x'), y_{true}\right) \\ \text{s.t.} \quad & \|x' - x\|_\infty \leq \epsilon \end{aligned} \tag{2.11}$$

where f_k is the k^{th} classifier, $J(\cdot)$ is the error function, x and x' the original and adversarial images respectively.

Here the loss function is computed as the mean of the individual losses generated by each classifier. As we will see, Model-based Ensembling Attack generally transfers better by avoiding dependence on any specific model, because ensembling attacks are created using multiple models. This approach take advantage of the diversity among the models to generate more robust and transferable adversarial examples.

Attack on intermediate levels

We will see how different models catch similar features for the same images, even when the architecture are quite different. The attack on the intermediate levels aims at creating an attack directly on this features. Specifically, they propose a method that fine-tunes a given adversarial example through examining its representations in intermediate feature maps. The algorithm is called "Intermediate Level Attack" (ILA).

For a full description and motivation behind the method we send to the paper. Here we present the algorithm we have been using for the creation of these attacks. The original image is x, the adversarial created on the softmax layer is x' and the adversarial we are creating with the algorithm is x''; F_l is a function that returns intermediate layer outputs; ϵ is the bound for the infinity norm; lr is the learning rate; iterations n; J is the loss function.

Algorithm 2.1 Intermediate-layer Adversarial Attacks (ILA)

function ILA(x, x', F_l, ϵ, lr, n)
  $x'' = x$
  $i = 0$
  $J = CrossEntropyLoss()$
  while $i \leq n$
    $\Delta y'_l = F_l(x') - F_l(x)$
    $\Delta y''_l = F_l(x'') - F_l(x)$
    $x'' = x'' - lr \cdot sign(\nabla_{x''} J(y'_l, y''_l))$
    $x'' = clip_{\epsilon}(x'' - x) + x$
    $x'' = clip_{[0,1]}(x'')$
    $i = i + 1$
  end while
  return x''

3

White adversarial attacks on Remote Sensing Images

3.1 Remote sensing images

Thanks to the progress of technologies in the past decades, images acquired from the space are more and more used in several areas. Earth Observations (EO) is, nowadays, almost essential in areas such as urban planning, agriculture, disaster management, environmental monitoring and others. The process of acquiring images from a distance, such as from a satellite or an aircraft, is called *remote sensing*.

From a practical point of view, satellites are detecting reflected and emitted radiation coming from the Earth. Each material on the surface, indeed, reflects, absorbs, or transmits electromagnetic radiation differently across the electromagnetic spectrum. Properties contained within different wavelengths of light, captured by the remote sensing sensors, are called *spectral properties*. Depending on the portion of the electromagnetic spectrum captured by the sensor, different images are created. These images are known as Remote Sensing Images (RSI). *Optical images*, specifically, which are the ones we are considering in this dissertation, are those where only wavelengths from the visible part of the spectrum are captured*.

*Sometimes including infrared light, not in our case.

Resolution

The level of detail detectable in the image is part of a measure called *resolution*. The resolution measures the area of the Earth represented by each pixel. Images are classified based on their resolution, as follows:

- **low resolution**: $> 20m$
- **medium resolution**: $> 2m$ to $\leq 20m$
- **high resolution**: $> 50cm$ to $\leq 2m$
- **very high resolution**: $\leq 50cm$

Figure 3.1 shows images taken at different resolutions.

Figure 3.1: Questa immagine è provvisoria, verrà sostituita.

3.1.1 Adversarial attacks in RSIs

In the past decade, thanks to progresses of remote sensing technologies, the automatic interpretation of remote sensing images (RMIs) has greatly improved. With the remarkable advancement made in deep learning within the computer vision field, deep convolutional neural networks (CNNs) have been introduced to automate RSI recognition processes.

As we have previously pointed out, high dimensional data might lead to adversarial examples in CNNs [cite] due to the linear operations of the models. As RMIs images are high dimensional data, this field is, intuitively, not immune from the treat of adversarial attacks.

Previous work

While adversarial attacks on natural images have been extensively studied, less attention has been dedicated to remote sensing images. Interest in this area has increased in recent years, highlighting diverse potential challenges and concerns.

The first paper on adversarial attacks for RSIs was [cite]. They proved the existence of adversarial attacks in the RS domain.

For the authors of the paper, adversarial attacks pose a serious threat to security. Physical adversarial attacks, for example[†], could potentially conceal objects from military object detection systems. This is one of the reasons why studying this field is important. [An addition will be made about adversarial training.]

Challenges

Working on attacking remote sensing images (RSI) differs from attacking natural images. In practice, we often lack any knowledge of the model we intend to attack, i.e., of the victim model. Constructing a white-box attack without the information about the model architecture can present a challenge.

This is why the world of adversarial attacks targeting remote sensing images focus around two primary aspects, both of which are integral also to this dissertation:

1. Demonstrating the existence of adversarial attacks through the creation of white-box attacks.;
2. Developing black-box attacks to target unknown models.

[†]Physical attacks insert real-world objects into the environment that, when imaged together with the targeted scene element, can bias Depp Neural Networks. The real-world objects are typically image patches whose patterns are optimised in the digital domain before being printed.

This structure explain the framework of this work.

First, we explore various algorithms to generate different adversarial attacks on RSIs. We will compare the results and show their differences and efficiencies.

Secondly, we propose an approach to develop black-box attacks, where we are going to explore the potential of the transferability property of adversarial attacks. Specifically, we are going to generate white-box adversarial attacks and transfer them to a variety of different models. The second part is going to be described in Chapter [].

Before starting describing the first part, we need to define our experimental setup.

3.1.2 Experimental setup

Dataset

To use remote sensing images and data, we used a library called TorchGeo [cite], which provides access to various freely available datasets.

For this experiment, we utilized the UC Merced Land Use dataset [cite]." This dataset contains 2100 RGB images of urban locations extracted from the USGS National Map Urban Area Imagery collection. They have been taken at a resolution of 1 ft. The dataset contains 21 classes, each with 100 examples.[Aggiungere info on resolution]

At Fig [cite] is shown a sample of the dataset (todo). A full list of images from all the classes is shown in the Appendix.

Model

As a victim model for the white black box attacks generation we used ResNet50, pretrained on natural images and fine tuned (with the addition of one linear layer) on the 21 classes from our UC dataset.

On table tot we see the accuracy of the model on the train and test set. A validation set has been also used.[todo]

Also: [todo] informations about the running machine.

3.2 White attacks generation

Now that our experimental setup has been described, we are going to generate white adversarial attacks.

The process usually consists of the following steps:

1. Setting the previously fine-tuned model to evaluation mode.
2. Applying the attacks on the model using a test set.

The last step is achieved by utilizing the model's existing gradients and loss function to compute modifications to the input data (and thus creating the adversarial attacks).

For the majority of these attacks, we utilized a PyTorch library known as *Torch Attacks*. The only exception was for the creation of the Universal Perturbation, where we relied on publicly available code [cite].

3.2.1 Random noise

Before experimenting with various attack methods, we will examine the impact of random noise on images. Specifically, we are applying three types of noise: Gaussian noise (with a specified standard deviation, TODO), Salt and Pepper noise, and Poisson noise. This will serve as a baseline for the next methods. If any of the attacks we later examine perform worse than random noise generation, it suggests that the attack is ineffective. In Figure 3.2, we can observe the visual impact of adding a random noise to the image. The Gaussian and the Salt and Pepper noises are visually discernible in the images, and this makes them unsuitable for our objective. Our aim is, indeed, to introduce a noise that is effective and minimally visible at the same time.

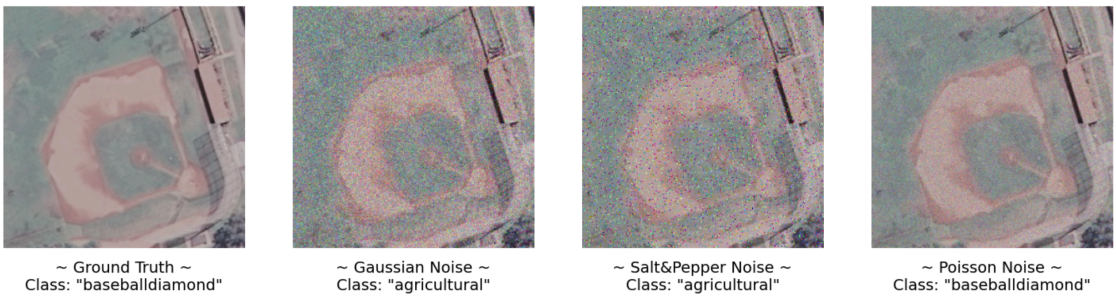

Figure 3.2: Example of caption.

We want, anyway, also to quantitative measure the effectiveness of a method; as previously explained, this is done by using the **fooling rate**. This metric quantifies the number of images incorrectly classified by the model after noise addition and is calculated only on the subset of images that were originally correctly classified.

By looking at the fooling rates for the three random noise types we employed, we gain an initial understanding of their efficiency. Table 3.1 presents the fooling rate values for each method.

Gaussian Noise	Salt and Pepper Noise	Poisson Noise
0.22	0.34	[todo]

Table 3.1

From now on, we will consider these values as our baseline. By the implementation of adversarial attacks methods we will try to beat them. Our ultimate goal is, anyway, to fool the victim model on as many images as possible. This means that we would like to reach the highest possible fooling rate.

3.2.2 FGSM

To start the adversarial attacks creation, we are firstly going to see one of the first algorithms developed for the creation of adversarial examples, the Fast Gradient Sign Method (FGSM). Refer to Chapter [cite] for the theoretical part.

The perturbation generated by the Fast Gradient Sign Method (FGSM) is regulated by the hyper-parameter epsilon. We are selecting a random image from the test set and showing how the adversarial images vary by adjusting the epsilon parameter. An example is shown in Figure 3.3. Increasing the value of epsilon results in a higher level of visible noise. The varying levels of noise also trigger different classifications. While the original image was (correctly) classified as a "beach", the adversarial images are classified as a "river", "chaparral" and "agriculture". At a certain point, the classification stabilize to the last predicted class.

As already mentioned, to create a successful adversarial attack, the generated image should show no discernible differences from the original one, at least to the human eye. To quantitatively measure their difference, we are going to utilize a metric called the *Structural Similarity Index* (SSI) between two images.

Once again in Figure 3.3, we observe the variation in the SSI as the epsilon parameter increases, thereby introducing more noise to the image. The SSI is computed for each image by

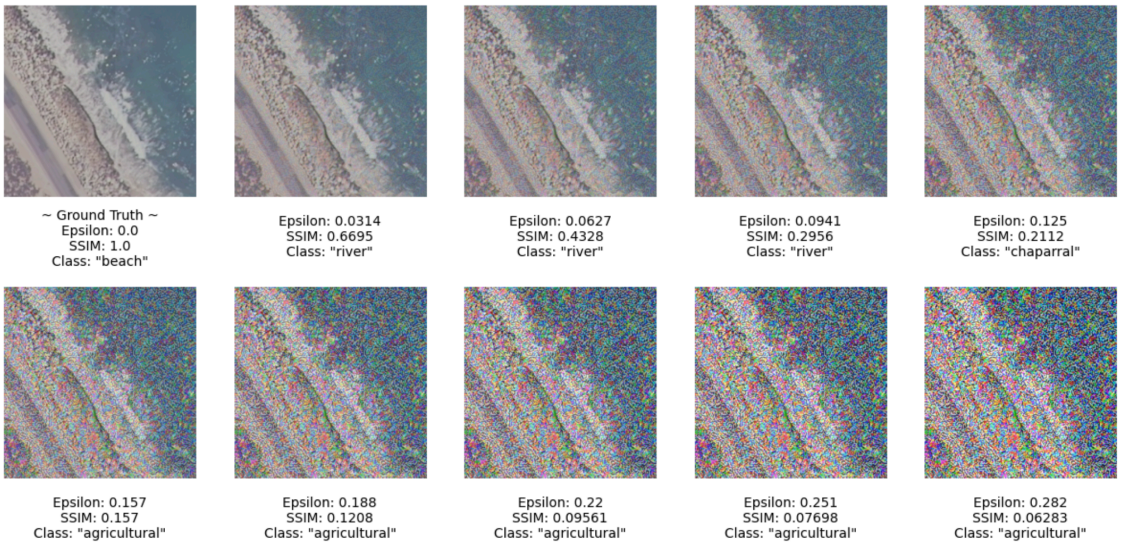

Figure 3.3: Example of caption.

comparing it with the original (clean) image. The greater it is epsilon, the smaller it is the SSI. [todo short theoretical part]

The second image on the left was generated using an epsilon value of approximately 0.0314, which is the default value. This value is considered effective enough to deceive the model while remaining relatively imperceptible. However, starting from the third image onwards, the noise becomes increasingly noticeable, rendering the attack ineffective due to its visibility.

The left image of Figure 3.4 illustrates the relationship between the epsilon value and the changes in the fooling rate. While increasing the epsilon value results in a higher fooling rate, the majority of the images display visible noise, rendering them unsuitable for our intended purpose.

Hence, in the case of FGSM, only a small value of epsilon can effectively create the attack. To enhance the efficiency of the FGSM attack without altering epsilon (i.e., without adding more noise), one approach is to adjust the magnitude of the logits of the model. Scaling the logits with a parameter, referred to as *temperature*, enhances the model's effectiveness. This is due to the nature of gradient-based learning, where the absence of gradients impedes learning. By scaling the logits, instances with zero gradient are reduced, thereby improving the learning process.

The right image in Fig 3.4 show how the fooling rate is affected by the temperature parameter, and it demonstrates the effectiveness of the scaling in increasing the effects of the attack.

The red point on the image serves as a reference point for the comparison. While increas-

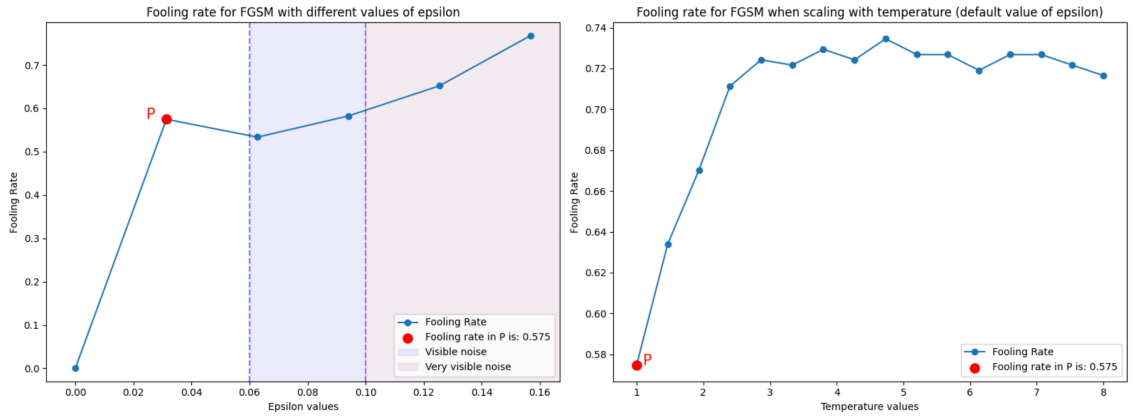

Figure 3.4: Example of caption.

ing the fooling rate by adjusting the epsilon parameter results in images with very visible noise, changing the temperature parameter does not produce the same effect. Despite not altering the amount of added noise, which remains equal to the default parameter, adjusting the temperature is enhancing the model's effectiveness.

With a temperature parameter of around 4-5, the model is fooled on more than 70% of the images in the test set. This result already greatly surpasses those obtained with a random noise. We will now apply several other methods to see if we are able to further better this result.

3.2.3 Fooling rates

A comprehensive list of adversarial attack methods can be found in [cite]. In addition to FGSM, we will now utilize the following techniques: BIM, PGD, Jitter, CW, DeepFool, and OnePixel. While some of these methods have been discussed in Chapter [cite], we recommend referring to the documentation for a complete explanation of the others.

At the following Table 3.2 we compare the values of the fooling rates by using all the methods. We are using GN, i.e. the gaussian noise, as a reference baseline. Everything with a greater score than GN is theoretically working effectively on fooling the victim model.

GN	FGSM	FGSM temperatured (T=7)	BIM	PGD	Jitter	CW	DF	OnePixel
0.226	0.57	0.729	0.997	0.997	0.989	1	0.88	0.01

Table 3.2: Fooling rates on the test set when different adversarial attacks methods are applied.

There are two key considerations to address regarding these results:

1. Which methods produce noise that is imperceptible from a human perspective?
2. How confident is the model (i.e. what is its confidence score) when perturbing the images?
3. Are the execution times similar for each method and, if not, which method is the fastest/slowest?

To assess the first point, we will present a random image to which we apply all the different adversarial attack methods. We will also compute the Structural Similarity Index (SSI) to measure the similarity between the original image and the perturbed images generated by each attack method.

Noise visibility

In Figure 3.5, we present the most notable findings regarding different types of noise. While Gaussian noise is visibly apparent and FGSM is quite apparent, other types are nearly imperceptible to the human eye. The SSI provides a quantitative measure of the dissimilarity between the perturbed and original images. A score of 1, as observed with DeepFool, indicates minimal discernible differences. It is worth noting that despite achieving a high SSI, the DeepFool perturbed image is misclassified, highlighting the potency of this method. It is able, indeed, to induce mis-classification in an image while making minimal alterations to it.

Similarly for the image perturbed with the CW method, which also experiences mis-classification despite exhibiting a high SSI score.

In Figure 3.6, we visually compare some noise additions. It is important to note that this representation prioritizes display over precision and serves to provide a general overview of the noise differences.

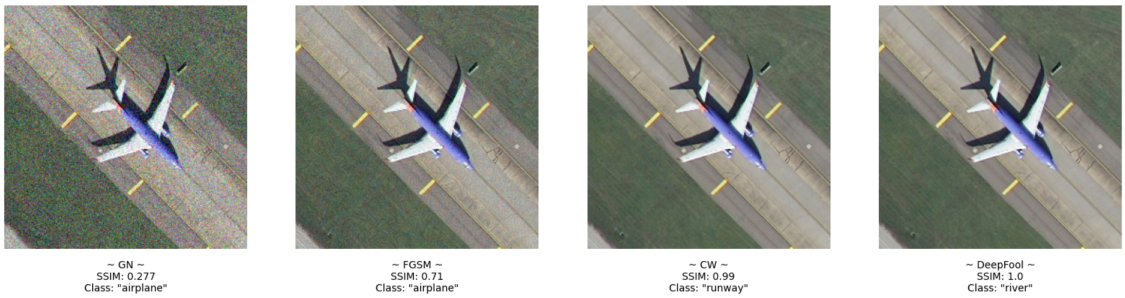

Figure 3.5: Example of caption.

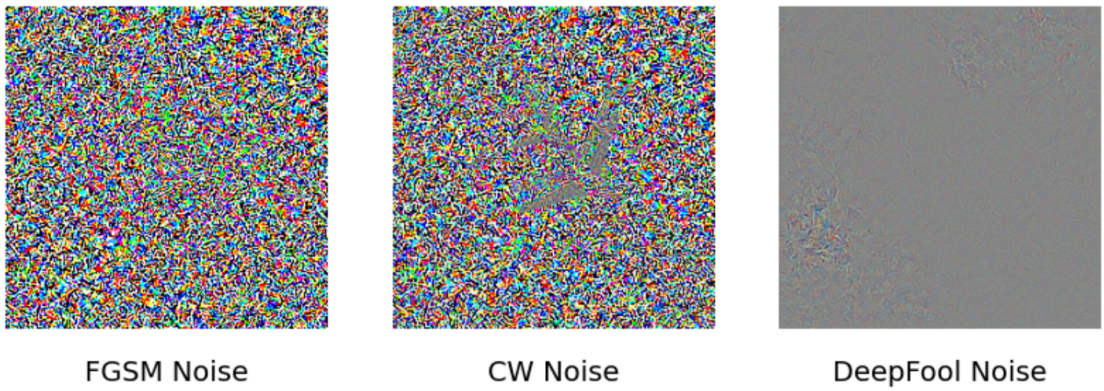

Figure 3.6: Example of caption.

As it can be seen, the results are very interesting. The noise generated from the DeepFool algorithm is very minimal when compared to the others.

Confidence

To address the second point regarding the model's confidence scores, we will compute the softmax probabilities based on the logits for each image both before and after applying an attack. Subsequently, we will reassess the fooling rates for each attack method, focusing solely on images where the model's confidence in its prediction is greater or equal than 70%.

In Table 3.3 below, we present the confidence scores of the model corresponding to each attack method. These scores are derived from all the test set images where the model's prediction was correct and the attack was successful. What stands out is that certain methods instill high confidence in the model's misclassifications. However, it's worth noting that there are instances where the model's confidence score hovers around 40% for images generated by specific methods.

Confidence score	GN	FGSM	FGSM temp (T=7)	BIM	PGD	Jitter	CW	DF	1 Pixel
min	0.36	0.46	0.46	0.89	0.97	0.88	0.54	0.42	0.43
max	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.83	0.57
mean	0.90	0.95	0.95	0.99	0.99	0.99	0.97	0.50	0.50
median	0.98	0.99	0.99	1.0	1.0	1.0	0.99	0.50	0.50

Table 3.3: .

In Table 3.4, we present the fooling rates computed for the subset of images with a confidence score exceeding 70% or 50%. This provides a more nuanced perspective of their effectiveness. Interestingly, it is observed that the number of images misclassified after a DeepFool attack with a confidence score over 70% is negligible. Lowering the threshold for the confidence score to 50% improves DeepFool's performance in terms of the fooling rate, making it comparable to FGSM. However, considering how minimal is the noise introduced by DeepFool, it still overall outperforms FGSM from this point of view.

Confidence	GN	FGSM	FGSM temperatured (T=7)	BIM	PGD	Jitter	CW	DeepFool	OnePixel
>50%	0.244	0.567	0.567	0.994	0.997	0.981	1	0.783	0.015
>70%	0.208	0.543	0.543	0.994	1	0.981	0.97	0.002	0.005

Table 3.4: Fooling rates on the test set when different adversarial attacks methods are applied, confidence score.CHECK

Execution time

The final point we need to address concerns differences in the execution time. For instance, BIM is an iterative version of FGSM; since the latter requires only one iteration while the former requires more, it is intuitive that their execution times will be quite different.

To observe these differences, we will compute the execution time for generating each perturbed image, and then compute the mean values to assess their performance across the entire test dataset. The results are presented in Table 4.3.

At Plot 3.7 we compared all the attack methods based on their fooling rate (setting the confidence score over 50%) and their execution time.

Exe-time	GN	FGSM	FGSM temperatured (T=7)	BIM	PGD Jitter	CW	DeepFool	OnePixel
Mean	0.244	0.567	0.567	0.994	0.9970.981	1	1.378	0.313

Table 3.5: Fooling rates on the test set when different adversarial attacks methods are applied, confidence score.CHECK

Now that we have seen all the standard methods for adversarial attacks generation, we go on with the universal adversarial attack.

3.2.4 Universal perturbation

As discussed in Chapter [cite], the Universal Perturbation is a perturbation that is not tailored to a specific image but rather build on a subset of images with the aim of potentially affecting a large portion of the dataset. From now on, when we are talking about the "subset", we refer to the subset used to build the universal perturbation.

The key points we will address are the following:

1. We will demonstrate the relationship between the selected subset and the fooling rate;
2. We will examine an example, with its noise characteristics;

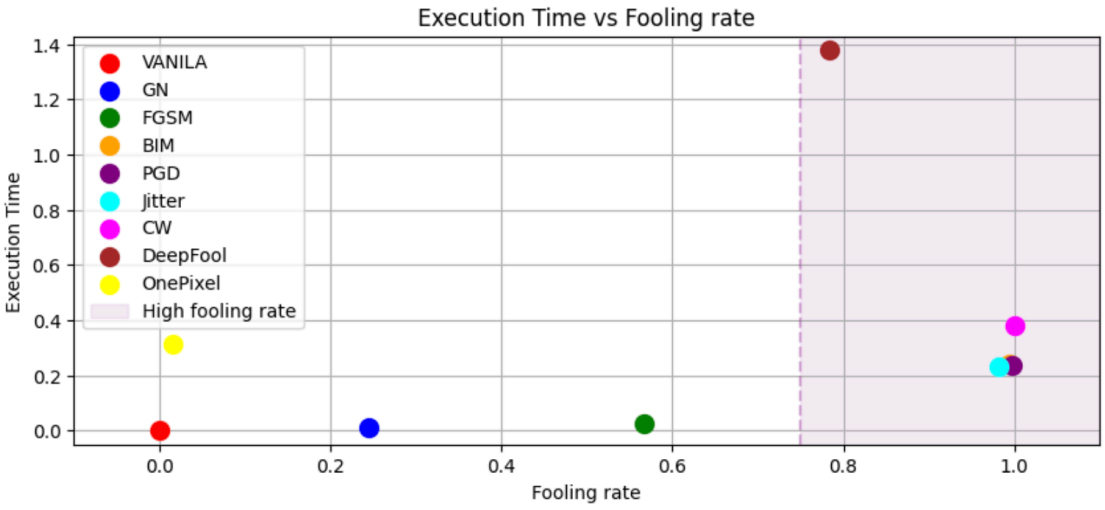

Figure 3.7: Example of caption.

3. We will explore the potential improvement in results by scaling with a temperature parameter in this context.

4. We will see how the images are being moved from a class to another through a network graph.

Subset and Fooling Rate

The test dataset contains 420 images and 21 classes. We are going to try generating the adversarial perturbation doing 5 iterations. We select different subsets to start with:

- A subset of 21 random images (5% of total dataset)
- A subset with one image for each class (5% of the total dataset)
- A subset with two images for each class (10% of the total dataset)
- A subset with three images for each class (15% of the total dataset).

For each subset and for each iteration we are computing the fooling rates as usual. Results are shown in the following Table **??**.

Subset type	Fooling rates				
	Iter 0	Iter 1	Iter 2	Iter 3	Iter 4
21-random 5%	0.005	0.007	0.018	0.033	0.038
1-per class 5%	0.015	0.018	0.041	0.226	0.412
2-per class 10%	0.018	0.041	0.314	0.670	0.791
3-per class 15%	0.018	0.092	0.376	0.737	0.896

Table 3.6: Fooling rates on the test set when the universal perturbation is created from different subsets.

The significance of subset selection is evident from the results. Despite the first two subsets have the same size, the outcomes vary significantly. The subset with randomly chosen imges barely misclassifies 1% of the test dataset, whereas the subset with one index per class misclassifies around 40% (with 5 iterations). When using the 15% of the dataset to build the universal perturbation, then around 89% of the dataset is misclassified.

This method for generating adversarial attacks exhibits an intriguing behavior by enabling the creation of such attacks without directly accessing the full dataset.

While this approach has its appeal, it does have at least one notable drawback: the execution time required for generating an adversarial perturbation. For instance, utilizing the 15% subset necessitate approximately 665 seconds from iteration 0 to iteration 4. The required time for the DeepFool attack (which was the slowest one) on all the test set was around 578 second. [TODO time is comparable actually, maybe worth invastigating if it changes when the dataset is bigger]

It's important to note the impact on the fooling rate when we set a confidence threshold:

- The fooling rate is 0.811 when computed only on perturbed images where the model has a confidence score greater than 50%
- The fooling rate is 0.711 when computed only on perturbed images where the model has a confidence score greater than 70%

It is also important to note that, for how this approach was developed by its authors, the subset of images utilized for creating the perturbation does not necessarily comprise only images where the model makes correct predictions. Some of the images, i.e. those where the model's

predictions are incorrect, are being discarded **during** the process of creation. Thus, since the choice of the indexes is random, the actual number of images used for the creation of the perturbation varies every time.

Scaling with a Temperature parameter

What does it happen if we use as a subset a set of images where the model starts with a great confidence (> 0.99)?

As previously noted, the algorithm typically utilizes a random subset where some of the model's predictions may be incorrect. However, by selecting a subset where the model begins with high confidence, we guarantee that the entire subset is utilized for perturbation creation.

To amplify the effect of concentrating on images where the model displays high confidence, we can scale the model's logits using a parameter T. This scaling emphasizes the differences in confidence scores between correctly and incorrectly classified images, making the model's confident predictions even more pronounced.

Results are reported in Table 3.7.

Subset: 3-per class 15%	Fooling rates				
	Iter 0	Iter 1	Iter 2	Iter 3	Iter 4
confidence >99%	0.005	0.402	0.721	0.842	0.860
confidence >99% and T=3	0.030	0.324	0.742	0.798	0.835
confidence >99% and T=5	0.048	0.409	todo	todo	todo

Table 3.7: Fooling rates on the test set when the universal perturbation is created from different subsets.

[comments]

Example

By taking some random image from the test set, we can see how the universal perturbation looks like. See Fig 3.8.

Considerations [...].

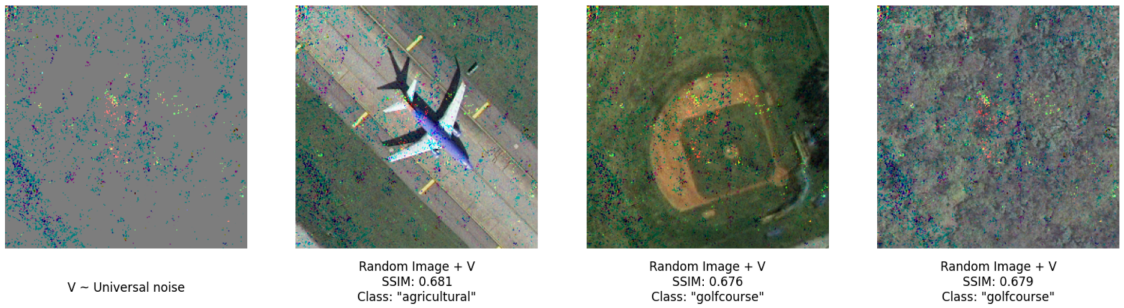

Figure 3.8: Example of caption.

Network class representation

The last point we are addressing is showing how the universal perturbation is moving the classes. A starting insight into this ”issue” may be gleaned from Fig 3.8, where two out of three displayed images are effectively misclassified as ”golfcourse”. Hence, the question naturally arises: how are the class boundaries being altered?

We will see it through network graphs, representing the classes distributions.

[Little display problems – in progress]

In the next chapter we are going to exploit the transferability property.

4

Black box adversarial attacks on Remote Sensing Images

In practical scenarios, we often lack detailed knowledge about the specific model we intend to target. For instance, we might seek to deceive a company utilizing Recognition Systems (RSIs) by manipulating images and sending them some altered versions of them. Alternatively, we might aim to conceal certain objects from autonomous detection systems by designing physical patches that render these objects "invisible" to the detectors.

This is the reason why in the remote sensing scenario the white-box attack generation is often not as important as the black box one.

This chapter focuses on studying the transferability property of adversarial attacks. Essentially, we will utilize adversarial attacks developed in Chapter 2 and apply them in different contexts. This will allow as to understand how they behave against other models, i.e. how well they "transfer". Specifically, our objectives are:

1. Assessing the effectiveness of adversarial attacks across various classification models: we will keep the adversarial we have already created and apply them to other CNNs.

2. Building adversarial attacks with an ensemble of models and evaulate their general performance on all of the models.

A visual representation of the time line of this process is visible at Fig tot.

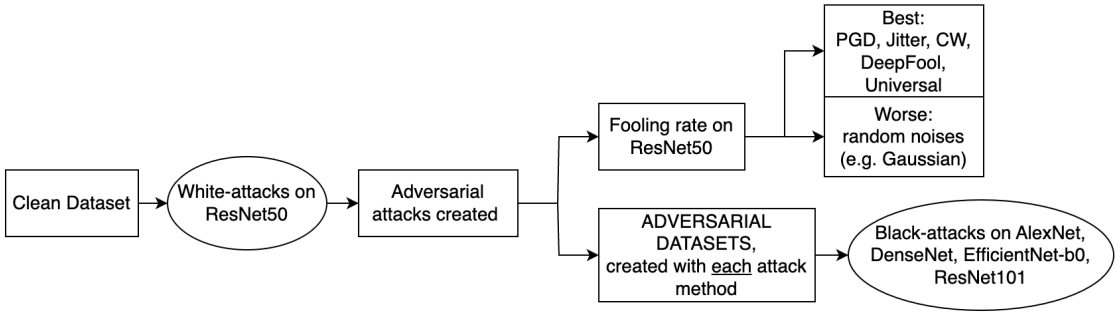

Figure 4.1: Example of caption.

4.1 Transfer Adversarial Attacks Across Different Models

First of all, we want to assess the effectiveness of adversarial attacks on models other than the one they were originally build for. We, indeed, generated the adversarial attacks using only ResNet50. We can now create a new version of the original (clean) dataset, an adversarial one. This dataset will contain the perturbed versions of the test dataset, and we will refer to it as the "noisy dataset" or the "adversarial dataset. For each of the attack methods we have seen before, we are going to have its representative dataset. In the first part we will see the behaviour of the one-image attack (One-Pixel has being removed doe to its complete inneficiency); in the second part we will see the effect of the universal attack.

Our objective will than be to evaluate how different models perform on this noisy datasets, i.e. to compute their accuracy. The models we are considering are:

1. AlexNet
2. DenseNet 121
3. EfficientNet - b0
4. ResNet 101

Here's a table illustrating the accuracy scores for each model on both the original test dataset and its noisy version (i.e., the dataset containing adversarial images). Keep in mind that ResNet50 is the model for which the adversarial attacks have been created.

Model	Accuracy on							
	Clean	Perturbed with:						
	Test Set	GN	FGSM	BIM	PGD	Jitter	CW	DF
ResNet50	92.38%	72.38%	39.29%	0.48%	0.24%	2.14%	0%	14.29%
AlexNet	86.43%	68.81%	80.00%	83.33%	85.00%	83.57%	86.19%	86.67%
DenseNet	88.81%	70.71%	77.14%	76.43%	77.86%	80.95%	87.62%	88.81%
EfficientNet	92.14%	76.67%	79.29%	84.76%	85.48%	86.67%	91.67%	92.14%
ResNet101	83.10%	64.52%	67.86%	70.95%	74.52%	79.29%	82.62%	83.33%

Table 4.1: .

The results offer some valuable insights.

Firstly, it's important to highlight that the accuracy scores in the first row, representing the performance on the noisy dataset, don't precisely mirror the fooling rates observed in Chapter 2. This discrepancy arises from the fact that in this evaluation, all the images of the dataset have been attacked, including those where the model initially made incorrect predictions. This explain the higher results compared to those observed in Chapter 2.

By examining the results, we see that the models are quite stable to the treat of *foreigner* adversarial attacks. However, it is interesting to note that the introduction of Gaussian noise to the images consistently impacts the accuracy of the models. This method, previously considered relatively benign and less effective, now consistently induces a drop in model performances, ranging between 10-20% accuracy points.

For this reason, it is worth trying to introduce other forms of random noise, to ascertain if they pose similar challenges to the model. Let's consider two random noises already encountered in Chapter 2: Salt and Pepper and Poisson.

Model	Accuracy on:		
	Test Set	Test set + Salt-Pepper-noise	Test set + Poisson-noise
ResNet50	92.38%	–%	–%
AlexNet	86.43%	51.19%	77.62%
DenseNet	88.81%	75.24%	84.29%
EfficientNet	92.14%	71.19%	83.57%
ResNet101	83.10%	46.19%	68.10%

Table 4.2: .

These two forms of random noise are consistently causing a decrease in the accuracy scores, demonstrating their efficacy in perturbing the models successfully.

From a intuitive perspective, this outcome makes sense. The adversarial attacks specifically crafted for ResNet50 prove highly effective when deployed against ResNet50 itself, because they were build on it. They are working for ResNet101 to some extent, possibly due to shared architectural characteristics between the two models.

On the contrary, random noise poses a challenge for the models. Unlike adversarial attacks crafted with a specific model in mind, random noise is agnostic to model architecture. Hence, it exhibits high transferability across different models, making it a potent threat to model performance.

The good news is that random noise, such as the Gaussian noise, tends to be easily detectable within the images, making it relatively easy to eliminate.

Based on the results obtained from this dataset and the evaluated models, it appears that adversarial attacks may pose a threat when transferred across models that share architectural properties. However, their effectiveness may diminish in scenarios where such architectural similarities are absent or minimal. On the other hand, random noise has demonstrated the capability to impair model performance, but it is easily detectable within the images, and therefore quite manageable.

4.1.1 Applying the Universal Perturbation

We now want to see want does it happen by using the Universal perturbation. We remind that until now this perturbation has been build on ResNet50 (the only model used in Chapter2) from a subset of images. In particular, since we have seen the results with different subsets, here we are considering the subset where three examples for each class has been used (so 15% of the total test set).

Since, in this Chapter, we are considering the perform of the attacks on different models, we are also using AlexNet to CREATE the Universal perturbation.

This means that we are using two universal perturbations, namely "v-resnet" and "v-alexnet". We are applying them to all the test set, creating two noisy versions of it, and then passing both adversarial-datasets to the other models (DenseNet, EfficientNet, ResNet101).

The following table shows the results:

Model	Accuracy on:		
	Test set	Test set + v-resnet	Test set + v-alexnet
DenseNet	88.81%	6.19%	29.52%
EfficientNet	92.14%	7.62%	43.81%
ResNet101	83.10%	5.48%	23.57%

Table 4.3: .

The results are promising. The drawback of this approach is, however, the noise visibility. An example of one of the images created from the test set and adding v-resnet is Fig 4.2.

The attack is working, but it is too visible.

4.2 Model Ensemble

Fatto, da scrivere

Figure 4.2: Example of caption.

4.3 Attack on intermediate levels

Fatto, da scrivere

5

Adversarial Training– forse

So far we have explored adversarial attacks for a dataset of sensory images. Directly attacking the model is the best way to give damage. However, most of the attacks are not quite transferable across models, except for the case of model-similarity.

In this Chapter we are not going to see any more way to directly attac a model. Indee we are shortly showing the impact of the so called adversarial traning.

Adversarial training means we are going to train the model on a dataset which includes normal and adversarial images. Given the impact of the universal attack, we are adding the universal version on it.

The procedure will be the following:

1. First, we are going to train the model on a dataset with both clean and perturbed images; we will see the impact of gaussian e universal noise
2. Second we are going to directly attack the dataset with some methods to see their impact
3. Third we are going to attack the model with a "imported" perturbation, to see how it behave of transferred attacks.

... Adversarial training
training solo su gn, oppure training su diversi attaccki

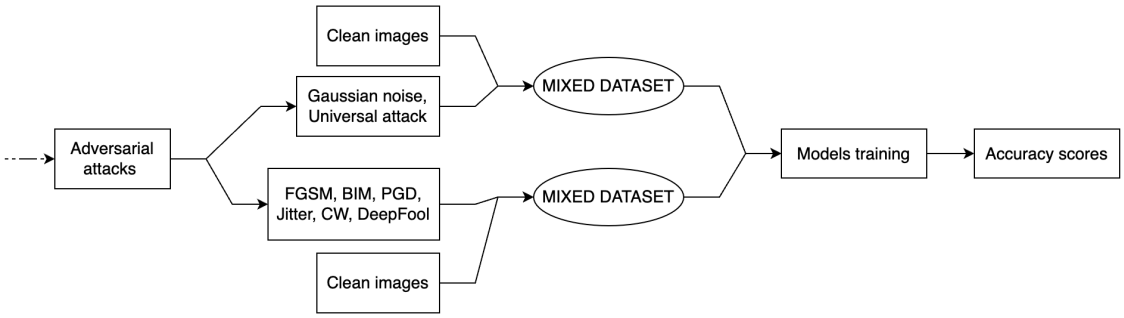

Figure 5.1: Example of caption.

6 Conclusion

The conclusion goes here.

References

[1] Szegedy, Christian, Zaremba, Wojciech, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfelow, Ian J., and Fergus, Rob. Intriguing properties of neural networks. [Online]. Available: https://arxiv.org/abs/1312.6199

[2] Goodfelow, Shlens, Szegedy. Explaining and harnessing adversarial examples. [Online]. Available: https://arxiv.org/abs/1412.6572

Acknowledgments

This is the acknowledgments section.