University of Perugia
Department of Mathematics and Computer Science

Master's Thesis in Computer Science

Adversarial Attacks on Explainability Methods for Image Classification Models

Candidate
Cristian Cosci

Supervisor
prof.ssa Valentina Poggioni

Co-Supervisor
prof. Alfredo Milani

Academic Year 2022-2023

Ai miei genitori e ai miei cari zii Rita e Quinto.

Contents

Introduction

Deep neural networks have achieved remarkable performance across a wide range of tasks, with notable successes on image classification, speech recognition, and natural language processing. However, one significant challenge associated with these models is their perceived lack of interpretability, i.e the difficulty to understand how they arrive at their predictions. For this reason, we often refer to neural networks as "**black boxes**". This lack of transparency can be a significant issue, particularly in applications where the consequences of incorrect predictions can be severe, such as medical diagnosis or autonomous driving.
To address this issue, researchers have developed various **explainability methods** to provide insights into the decision-making process of deep neural networks. These methods differ according to the type of application in which the models are used. For example, in the image classification task we can use methods such as saliency maps, activation maximization, and perturbation-based approaches, which identify the image areas that contributed most to the prediction. While, in the case of tabular data, there are approaches able to identify the data or features that contributed most in the decision. Furthermore, for other tasks such as natural language processing, attention mechanisms can provide insights into the words and phrases that heavily influence predictions.

In this study, we focus on image classification models due to their prevalence and relevance in various domains. However, it's important to note that the concept of **Explainable Artificial Intelligence (XAI)** extends beyond image classification and encompasses techniques that can provide insights into the decision-making processes of diverse machine learning models.

Nevertheless, recent research has uncovered a potential vulnerability within these ex-

plainability methods, that is their susceptibility to **adversarial attacks**. In such attacks, an attacker manipulates an input image to cause the explainability methods producing incorrect or misleading explanations while preserving the visual appearance of the original image and the correct classification label by the model. This vulnerability poses a significant challenge in achieving reliable and trustworthy interpretability techniques.

Therefore, the purpose of this thesis is to investigate the utilization of adversarial machine learning to attack explainability methods employed in image classification models. Specifically, we aim to examine the impact of adversarial attacks on the interpretability of deep neural networks and compare the performance of different explainability methods in the presence of adversarial examples.

This thesis starts with the introduction to **Adversarial Machine Learning (AML)** and **Explainable Machine Learning (XAI)** research topics (*Chapter* 1 and *Chapter* 2). Then, it proceeds in *Chapter* 3 by reviewing the relevant literature on adversarial attacks for explainability methods. Next, in *Chapter* 4 and in *Chapter* 5 we describe our approach and the experimental setup, which includes the selection of a deep neural network architecture and the creation of adversarial examples using a specific adversarial attack. Finally, in *Chapter* 6 we present our results and analyze the effectiveness of the different explainability methods in the presence of adversarial attacks.

Chapter 1

AML - Adversarial Machine Learning

As the field of machine learning has advanced, it has become increasingly apparent that the extraordinary results of machine learning models come with vulnerabilities. **Adversarial Machine Learning (AML)** is a subfield of Machine Learning that focuses on studying the effects of malicious inputs, known as **adversarial examples**, on the performance and robustness of machine learning models.

This chapter explores the concept of AML, its underlying principles, and the various attack techniques employed in this field.

1.1 Adversarial Attacks

Adversarial attacks on image classification task aim to manipulate machine learning models by introducing carefully crafted **perturbations** to the input data. These perturbations, imperceptible to humans, can cause the model to **misclassify or generate incorrect outputs**. Formally, an adversarial attack can be described as follow:

Let M be a Machine Learning model and C_{true} an input for that model. Assume that C_{true} is correctly classified by the model: $M(C_{true}) = y_{true}$. The AML methods are based on building an adversarial input C_{adv} adding a perturbation to C_{true}, in a manner that the difference between the two input is imperceptible, but permit to product a wrong classification by the model $M(C_{adv}) \neq y_{true}$.

In brief, the scope of AML is to create an input C_{adv} in order to fool the model with

a misclassification.
A first categorization of the attacks is between black box and white box methods and its based on the level of information that an attacker has about the target model:

- **White Box** Attacks: assume that the attacker has full access to the target model's architecture, parameters, training data, and even its internal workings. This detailed knowledge allows the attacker to devise more effective and targeted attacks. White box attacks are often more powerful because the attacker can exploit specific weaknesses in the model's structure and behavior.

- **Black Box** Attacks: assume that the attacker has limited or no access to the internal details of the target machine learning model. They can only interact with the model by providing inputs and observing outputs. The attacker may not know the architecture, parameters, or even the training data used to create the model. Despite this limited knowledge, black box attacks can still be effective. A key concept in black box attacks is the idea of **transferability**. This refers to the ability to generate adversarial examples on one model and then use those adversarial examples to fool a different, unknown model of the same type. This is possible because certain adversarial perturbations are effective across different models.

Furthemore, we can distinguish between three main attack methodologies [9]:

- **Evasion Attacks**
- **Poisoning Attacks**
- **Model Extraction**

Each of them, acts on the various weaknesses of the model. Below is a brief description of each of these approaches.

1.1.1 Evasion Attacks

Evasion attacks, also known as adversarial examples, consists in modifying input data in a way that leads the machine learning model to misclassify it. Adversarial

examples can be generated through various techniques, such as the **Fast Gradient Sign Method (FGSM)**[14], **Jacobian-based Saliency Map Attack (JSMA)**, or the **Carlini-Wagner Attack** [10] . These attacks often exploit the linearity and sensitivity of machine learning models to small changes in input.
Formally, it can be described as an optimization problem [10]:

$$\text{argmin}||\delta_X|| \text{ such that } M(X + \delta_X) = Y^*$$

where δ_X is the perturbation added to the input X, M is the classifier model and Y^* is a target label (it can be a specific label or the goal may just be that Y^* is different than the correct label Y).

1.1.2 Poisoning Attacks

Poisoning attacks, on the other hand, aim to compromise the training process by injecting malicious data into the training set. The attacker's goal is to manipulate the model's behavior during both training and deployment phases. Poisoning attacks can be categorized as either **data poisoning** or **model poisoning** attacks, depending on the attack target. The attack can be launched by adding poisoned samples, modifying existing samples, or even by controlling the availability of training data.

1.1.3 Model Extraction

The technique of model extraction is a methodology that aims to extract information or knowledge from a pre-existing machine learning model. In this technique, an attacker attempts to obtain an approximate copy or a similar model to the original by accessing its responses to queries. In this approach the attacker gains access to the target model and sends a series of queries to obtain the model's predicted responses. The attacker carefully records the responses of the target model and uses them to construct an approximate model or a copy of the original model. There are various motivations for attackers to utilize the model extraction technique. One of the primary objectives is to gain proprietary or secret knowledge behind a machine learning model. For example, an organization may wish to extract a proprietary fraud detection model from a competing bank to gain a competitive advantage. Similarly, an

attacker might seek to extract a facial recognition model used for access to a security system. Model extraction methods can compromise data privacy and security as the attacker can obtain sensitive information from the target model.

1.2 Evasion Attacks examples

Below are some current techniques in literature for generating adversarial examples in image classification tasks.

1.2.1 Fast gradient sign method

Goodfellow et al. (2014) [14] proposed the **fast gradient sign method (FGSM)** as a simple way to generate adversarial examples [22]:

$$X^{adv} = X + \varepsilon \text{ sign } (\nabla_X J(X, y_{true}))$$

where X is the original image, ε is a very small number, ∇_X is the gradient function, J is the loss function and y_{true} is the true label (see Figure 1.1). In other words, it consists of adding a linear amount of imperceptible noise to the image that causes the model to misclassify it. This noise is calculated by multiplying the sign of the gradient with respect to the image we want to perturb by a small constant ε. As ε increases, the model is more likely to be fooled, but the perturbations become easier to be identified.
We can see that the gradient is computed with respect to the input image, this is because the goal is to generate an image that maximizes the loss for the true label y_{true} on the original image. Otherwise, in traditional gradient descent (for model training), the gradient is used to update the weights of the model, since the goal is to minimize the loss for the model on a ground truth dataset.
This method is simple and computationally efficient compared to more complex methods like $LBFGS$ (Szegedy et al., 2014 [37]), however it usually has a lower success rate.

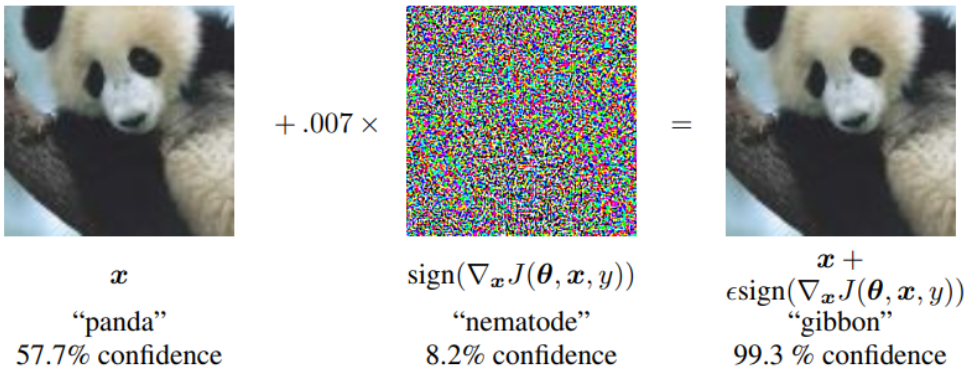

Figure 1.1: A demonstration of adversarial example generation applied to GoogLeNet [38] on ImageNet. By adding an imperceptibly small vector whose elements are equal to the sign of the gradient elements of the cost function with respect to the input, is possible to change GoogLeNet's classification of the image. Here ε of .007 corresponds to the magnitude of the smallest bit of an 8 bit image encoding after GoogLeNet's conversion to real numbers [14].

1.2.2 One-step target class method

An anlternative approach to FGSM is to maximize a probability $p(y_{target}|X)$ for a specific target class y_{target} (which is a different class than the original). For a neural network with *cross-entropy* loss this will lead to the following formula for the one-step target class method [22]:

$$X^{adv} = X - \varepsilon \text{ sign } (\nabla_X J(X, y_{target}))$$

The target class can be, for example, the one with the less probability given by the network (*One-step Least Likely Class* is the method's name for this approach) or a random class.

The difference between **FGSM** and **One-step target class method** is that FGSM finds an adversarial perturbations which increase the value of the loss function, whereas the current approach has the goal to maximize a probability for a different and wrong class.

1.2.3 Basic Iterative Method

The Basic Iterative Method (BIM) [23] is an evolution of the Fast Gradient Sign Method (**FGSM**) that involves a series of iterations to create adversarial examples. Instead of introducing a single small change to the original input, BIM applies the FGSM approach iteratively to create increasingly significant perturbations over the course of iterations.
In BIM, the adversarial example is created through the following iterative steps:

1. Start with the original input X.
2. Calculate the gradient of the loss function with respect to X.
3. Calculate the sign of the gradient.
4. Add a small amount of the perturbation from step 3 to X to create a new input X'.
5. Repeat steps 2 to 4 for a certain number of iterations.

Each iteration adds a bit of noise to the original input, and this gradual process can lead to an adversarial example that is effective at confusing the model, even if the adversarial example may still appear similar to the original input to a human observer.

1.2.4 Iterative least-likely class Method

The Iterative Least-Likely Class Method [23] is a variant of the **One-step target class method** mentioned earlier. Instead of attempting to maximize the probability of a target class in a single step, this method applies the same idea iteratively, similar to BIM. Again, the goal is to create an adversarial example that maximizes the probability of a class different from the correct one, but through an iterative process. Both iterative methods, BIM and the Iterative Least-Likely Class Method, make the attack more powerful than the FGSM method because they gradually introduce perturbations to the input, making it more difficult for the model to recognize and counteract the attack. However, this increased power might require more computational time compared to a single attack like FGSM.

1.3 Attacks using Evolutionary and Multi-objective Algorithms

In addition to the previously discussed iterative attacks, there is another branch of equally powerful and effective methods for carrying out adversarial attacks: **Evolutionary and Multi-Objective algorithms**. These represent an advanced approach which leverage principles inspired by biological evolution and multi-objective optimization to create highly effective and elusive adversarial examples [5, 6].

1.3.1 Attacks using Evolutionary Algorithms

Evolutionary algorithms simulate mechanisms such as **natural selection, mutation, and crossover** found in biological evolution. In the context of Adversarial Machine Learning, these algorithms generate adversarial examples by gradually introducing small perturbations to input data and optimizing it over generation in a manner that alters the model's classification.
A typical process for conducting an attack using evolutionary algorithms may involve the following steps:

1. **Initial Generation**: An initial set of adversarial examples is created, each slightly perturbed from the original input (it can be done using randomic perturbation). This set is called *initial population*.

2. **Selection of the mating pool**: Create a set of pairs of elements from the population. This set of couples is necessary for the next step.

3. **Crossover**: Starting from a couple of elements called *parent* it generates two new elements called *child* which are made up of the genetic heritage of both parents.

4. **Mutation**: Randomly select some elements from the *new population* and randomly modifies their *genome* (the genome is the internal representation of each element).

5. **Evaluation**: All the population elements are evaluted based on the objective function. On an adversarial attacks, the population elements can be the

Adversarial examples and their evaluation is based on the ability to induce classification errors in the target machine learning model.

6. **Selection**: The best elements are selected for the next phase. As adversarial examples, they can be selected using the criteria of causing the greatest classification alteration.

7. **Iteration**: The process of selection, crossover, and mutation is repeated iteratively to create more effective perturbations.

8. **Convergence**: The adversarial example generation process evolves over time, seeking to converge towards optimal perturbations for deceiving the model.

1.3.2 Multi-objective Attacks

In traditional optimization, the objective is to find a single solution that optimizes a specific criterion. However, in many real-world scenarios, there are multiple conflicting objectives that need to be considered simultaneously. Multi-objective optimization addresses this by seeking a set of solutions that represent trade-offs between different objectives, forming what is known as the Pareto front. These solutions are not dominated by any other feasible solution in all objective dimensions.
In the context of adversarial machine learning, it involve considering multiple objectives simultaneously during the adversarial example generation process. Instead of solely focusing on maximizing classification error, multi-objective attacks strive to strike a balance between various objectives, such as attack effectiveness and perturbation stealthiness.
This approach requires defining a **multi-objective fitness function** that evaluates adversarial examples based on variables like classification error, perturbation magnitude, and their difficulty of detection. Multi-objective algorithms like NSGA-II (Non-dominated Sorting Genetic Algorithm II) [6] can be employed to explore solution spaces and find sets of perturbations that satisfy multiple criteria.

1.3.3 Advantages and Complexity

Attacks using evolutionary and multi-objective algorithms offer significant advantages over traditional techniques. They are less reliant on complete model knowledge and

can create adversarial examples that are more robust and difficult to detect by both the model and the human eye. Furthermore, multi-objective attacks can create perturbations that balance attack success with the detectability of the perturbations.
However, these approaches are computationally more complex. The search for optimal solutions requires a significant number of iterations and model evaluations. Additionally, designing multi-objective fitness functions can be intricate and necessitate careful consideration of the attacker's objectives.
In summary, attacks based on evolutionary and multi-objective algorithms introduce a sophisticated dimension to the challenge of Adversarial Machine Learning, enabling attackers to create highly effective and elusive adversarial examples. Nevertheless, their implementation requires careful planning and understanding of the involved complexities.

In this work, to generate adversarial examples which deceive an explainability methods, was used a variant of the NSGA-II algorithms which internally has an evolutionary algorithm. In this way, it was possible to combine the capabilities of multi-objective algorithms to manage conflicting objectives, and the capabilities of evolutionary algorithms to optimize the generation phase of adversarial examples.
All the details of the adversarial attacks are described in Chapter 4.

Chapter 2

XAI - Explainable Artificial Intelligence

As the complexity of Machine Learning models advances and the utilization of Neural Network-based structures increases, the need for a set of techniques aimed at explaining and comprehending their operations becomes imperative. The **black-box** nature of a neural network's structure and functioning makes the interpretation of the produced outcomes challenging. In sensitive domains such as medicine and finance, placing full trust in the black-box behavior of these approaches is not feasible. Therefore, the importance of methods enabling precise understanding of the processes leading to specific outcomes becomes crucial.

The term **Explainable Artificial Intelligence** (**XAI**) refers to a field of study that provides a range of processes and methods enabling humans to understand the outcomes generated by various Machine Learning algorithms and Deep Learning models.

2.1 Need for Transparency and Trust in AI

The incremental performance achieved in various recent applications of artificial intelligence is primarily attributed to **Deep Neural Networks** (**DNN**). These models consist of dozens of layers and millions, in some cases even billions, of parameters, making it difficult to understand why certain predictions are obtained (i.e., it is challenging to grasp the true motivations and logic behind the model's choices) [11].

In many cases, both in real-world applications and research, such as in the medical field, experts require more than just a simple binary prediction; they need much more information to support their diagnoses [39]. Indeed, in general, humans are hesitant to adopt techniques that are not directly and easily interpretable.
XAI methods are born in response to this need. By making the decision processes of machine learning models transparent and interpretable, they allow users to better understand how these algorithms work.
Formally, an **explanation** refers to a set of information or interpretations that can take various forms, such as:

- **Visualizations** that highlight the importance of features used by the model in its decision (see Figure 2.1).
- **Textual descriptions** that explain key factors that influenced the prediction.
- **Comparisons with similar examples** to show how the model used past experiences to make the current decision.
- Insights into **interactions between variables** and how they influenced the model's output.

Figure 2.1: Visual Examples of an explanation for an Image Classification model. On the left image the heatmap represents what makes the model think the image label is "*pug*". Instead on the right image the heatmap represents what makes the model think the image label is "*tabby cat*".

Furthermore, an essential characteristic that every explanation must possess is that of **interpretability**, that is defined as the ability to explain or to provide the meaning

in understandable terms to a human.

Summarizing, the field of Explainable AI aims to create and provide a suite of ML techniques that:

1. Produce more explainable models while still maintaining a high level of performance (e.g., prediction accuracy).

2. Enable humans to understand, trust, and effectively manage the generation and operation of AI models [4].

2.1.1 XAI and Adversarial Machine Learning

In this thesis the concept of Adversarial Machine Learning is closely linked to the Explainability. As mentioned in the Introduction, the objective of this work is precisely to study the reliability of some XAI methods, using adversarial attacks on them. However, outside of this context, AML and XAI are still connected in several aspects. For example, **AML can leverage XAI methods to generate more powerful and effective attacks**.
Ideally, XAI should be able to explain knowledge within an AI model and reason about the model's actions. The information revealed by XAI techniques can be used to generate more effective attacks in adversarial contexts. After learning what specific information should be provided to the system to achieve a particular result, the attacker can use this knowledge to exploit the model.
On the other hand, it's important to leverage the knowledge provided by the XAI methods to make the AI model more secure. By identifying potential weak points where attacks could happen, we can strengthen the model's defenses and make it more resistant to adversarial tricks.

2.2 Explainability Methods Categories

The field of XAI encompasses a variety of methods that can be broadly divided into several categories, each with its own specific approach and benefits:

- **Transparent Machine Learning Models**: This approach takes advantage of the intrinsic properties of some machine learning models, also called **trans-**

parent models. These models, by design, are more interpretable due to their simplicity and limited complexity. Transparent models include **Linear Regression**, **Decision Trees**, and **K-Nearest Neighbors**. Although these models may sacrifice some predictive power compared to more complex counterparts, they offer clear insights into how inputs relate to outputs, making them valuable tools for critical applications where interpretability is paramount.

- **Post-hoc Explainability**: Post-hoc explainability techniques are applied after the model has made predictions. They seek to uncover the factors and relationships that influenced the model's decisions, shedding light on its underlying reasoning process. Post-hoc methods can be further divided into two subcategories:
  - **Model Agnostic**: These methods operate independently of the underlying model architecture. They are designed to work with a wide range of machine learning models without requiring knowledge of their internal structures.
  - **Model Specific**: Model-specific methods, on the other hand, are tailored to the characteristics of particular model types. These methods take advantage of the unique features of a specific model architecture to provide detailed explanations. For instance, **Gradient-weighted Class Activation Mapping** (**Grad-CAM**) (see Section 2.5.1) is applied to **Convolutional Neural Networks** (**CNNs**) to visualize which image regions contributed most to specific predictions.

The various categories and main methods will be described in detail in the next sections.

2.3 Transparent Machine Learning Models

This section presents the main machine learning models that are easily explainable by their nature.

2.3.1 Linear/Logisic Regression

Logistic Regression (**LR**) serves as a classification model aimed at predicting a binary dependent variable (category). On the other hand, when dealing with a continuous dependent variable, the counterpart is **Linear Regression**. LR assumes a linear relationship between predictors and the predicted variables, which limits its flexibility in adapting to data. This specific reason, the model's rigidity, places it within the realm of transparent methods [4].
The application of this model has found extensive use within social sciences for quite a duration, prompting researchers to devise means of explaining model outcomes to non-experts. Numerous authors concur on the different techniques utilized to analyze the robustness of LR [17, 28], including:

- **overall model assessment**, which showcases the improvement brought about by the applied model compared to a baseline, indicating whether it genuinely enhances predictions;
- **goodness-of-fit statistics**, that reveal the quality of the model's fit to the data and its significance;
- **validation of predicted probabilities**, which involves testing whether the model's output aligns with the data.

These techniques provide mathematical ways of representing the model's suitability and behavior.

2.3.2 Decision Trees

Decision trees represent a significant example of models that can easily adhere to transparency requirements.
Decision trees are machine learning models that base decisions on input data conditions. They are employed to facilitate the decision-making process in classification and regression problems. Each tree node corresponds to a question or choice (*if-else*), while the branches represent possible responses or actions. The tree's construction occurs hierarchically, with splits segregating data based on the most relevant features, enabling interpretable predictions and decisions.

Following this hierarchical structure, it becomes straightforward for an individual to comprehend the key decisions that brings the model to a specific prediction.
It is noteworthy that the **manageability of a decision tree by an user depends on its size and on features simplicity**. An increase in dimensions makes the model more intricate, hindering its complete human understanding.
Moreover, in comparison to other models, the limited generalization capabilities of decision trees decrease their attractiveness for applications where the predictive performances are of major importance. Tree ensembles aim to overcome these limitations by aggregating predictions generated by trees learned on different subsets of training data. Unfortunately, such aggregation negates any transparent properties of decision trees, necessitating the implementation of **post-hoc explainability techniques** (see 2.4).

2.3.3 K-Nearest Neighbors

Another method falling within transparent models is the **K-Nearest Neighbors** (**KNN**) algorithm, which tackles classification problems in a straightforward manner: it predicts the class of a test sample by voting based on the classes of its **K** nearest neighbors (where the notion of proximity is induced by a distance measure between samples). When applied in the context of regression problems, the voting is replaced by an aggregation (e.g., average) of the target values associated with the nearest neighbors [4].
In terms of model explainability, it is important to note that the predictions generated by this type of models rely on the notion of distance and similarity between examples, which can be adapted according to the specific problem at hand.
It is important to keep in mind that, as mentioned earlier for Decision Trees models, the **transparency level** of KNN depends on some implementation configuration. In this case, it depends on the **features meaning, the number of neighbors, and the distance function** used to measure similarity between data instances. As these measures take on more complex meaning (for features and distance measures) and larger values (for K neighbors), the difficulty for a human to understand how the model works increases.

The Figure 2.2 shows a graphic illustration of treated models transparency and how a user can approach them to understand how they work.

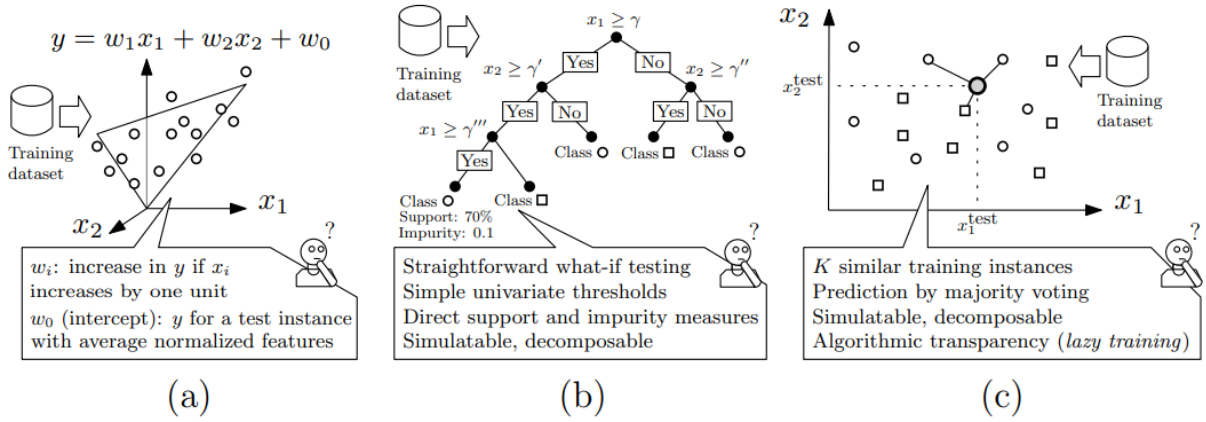

Figure 2.2: Graphical illustration of transparency levels of different ML models: (a) Linear regression; (b) Decision trees; (c) K-Nearest Neighbors [4].

2.4 Post-hoc Explainability Techniques

Post-hoc Explainability Techniques come into play when a model's decision-making process isn't transparent and easy comprehensible to humans. These techniques are designed to give understandable insights into how an existing model arrives at predictions.

This section delves into the primary approaches for post-hoc explainability, distinguishing between **Model-agnostic tecniques** and **Tecniques specifically designed to explain certain ML models**.

2.4.1 Model-agnostic Techniques

Model-agnostic techniques are approaches that can be applied to any machine learning model, regardless of its architecture or complexity. They provide a general way to understand and to interpret the decisions made by a model without needing to know its internal workings. Among the main methods belonging to this category we have **LIME** (**Local Interpretable Model-Agnostic Explanations**) [31] and **SHAP** (**SHapley Additive exPlanations**) [25].

LIME

The goal of LIME is to provide local and interpretable explanations for individual predictions generated by the model, even when the model's complexity makes it challenging to comprehend.
The LIME approach involves generating and trains an easy interpretable model (see *transparent model* in section 2.3) to approximate the behavior of the complex model that we want to explain.

Formally, the authors define an explanation as a new model $g \in G$, where G is a class of potentially interpretable models, such as linear models or decision trees. Furthermore, let f be the model that we want to explain. As not every $g \in G$ may be simple enough to be interpretable, the authors let $\Omega(g)$ be a measure of complexity (as opposed to *interpretability*) of the explanation. For example, for decision trees $\Omega(g)$ may be the depth of the tree, while for linear models, $\Omega(g)$ may be the number of non-zero weights [31].
The authors further used $\pi_x(z)$ as a proximity measure between an instance z to x, so as to define locality around x. Finally, let $L(f, g, \pi_x)$ be a measure of how unfaithful g is in approximating f in the locality defined by π_x. In order to ensure both interpretability and local fidelity, the goal is to minimize $L(f, g, \pi_x)$ while having $\Omega(g)$ be low enough to be interpretable by humans. The explanation produced by LIME is obtained by the following optimization problem [31]:

$$\xi(x) = \text{argmin}_{g \in G}\ L(f, g, \pi_x) + \Omega(g)$$

Since the explainer is model-agnostic, the minimization is done without making any assumptions about f. Thus, in order to learn the local behavior of f as the interpretable inputs vary, the authors use to approximate $L(f, g, \pi_x)$ by generating perturbed samples z, weighted by $\pi_x(z)$ [31]. Subsequently, LIME trains (by optimizing the previous equation) an interpretable model on these perturbed instances to approximate the complex model's behavior around the point of interest.

The primary intuition behind LIME is presented in Figure 2.3, where we sample instances both in the vicinity of x (which have a high weight due to π_x) and far away from x (low weight from π_x). Even though the original model may be too complex to

explain globally, LIME presents an explanation that is locally faithful (linear in this case), where the locality is captured by π_x.

The interpretable model generated by LIME represents a simplified approximation of the complex model. This interpretable explanation of predictions allows to understand the model's decisions even for individuals who are not machine learning experts.

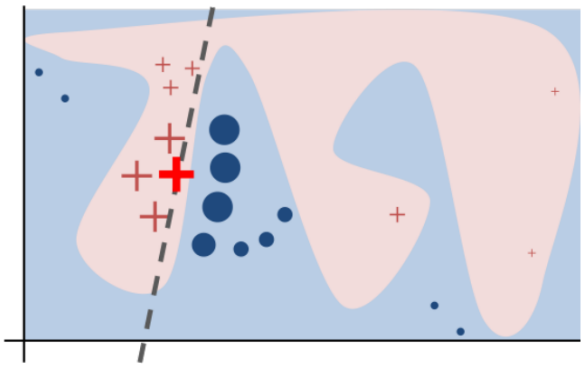

Figure 2.3: Toy example to present intuition for LIME. The black-box model's complex decision function f (unknown to LIME) is represented by the blue/pink background, which cannot be approximated well by a linear model. The bold red cross is the instance being explained. LIME samples instances, gets predictions using f, and weighs them by the proximity to the instance being explained (represented here by size). The dashed line is the learned explanation that is locally (but not globally) faithful [31].

SHAP

The SHAP technique is based on some concepts similar to the LIME method. In the same way as LIME, each explanation is seen as a new model that approximates the behavior of the original one, which, thanks to its transparency, allows easy interpretation even for non-expert users [25].
The theoretical foundation of SHAP comes from cooperative game theory and draws inspiration from the concept of **Shapley value**, which are used to fairly allocate each player's contribution in a cooperative game.
Therefore, SHAP is based on the idea that a prediction can be seen as the cumulative contribution of individual features to that prediction. The goal is to determine how

much each feature has positively or negatively contributed to the difference between the model's prediction and the reference average prediction. To do this, the SHAP approach considers all possible combinations of features (adding and removing features from the input of the model) and evaluates how much each feature contributes to the average Shapley value [25].
The explanation obtained from the SHAP method can be an histogram plot, a visual image or a text form. An example is the one in Figure 2.4, where the importance and contribution of each feature are shown using histograms.

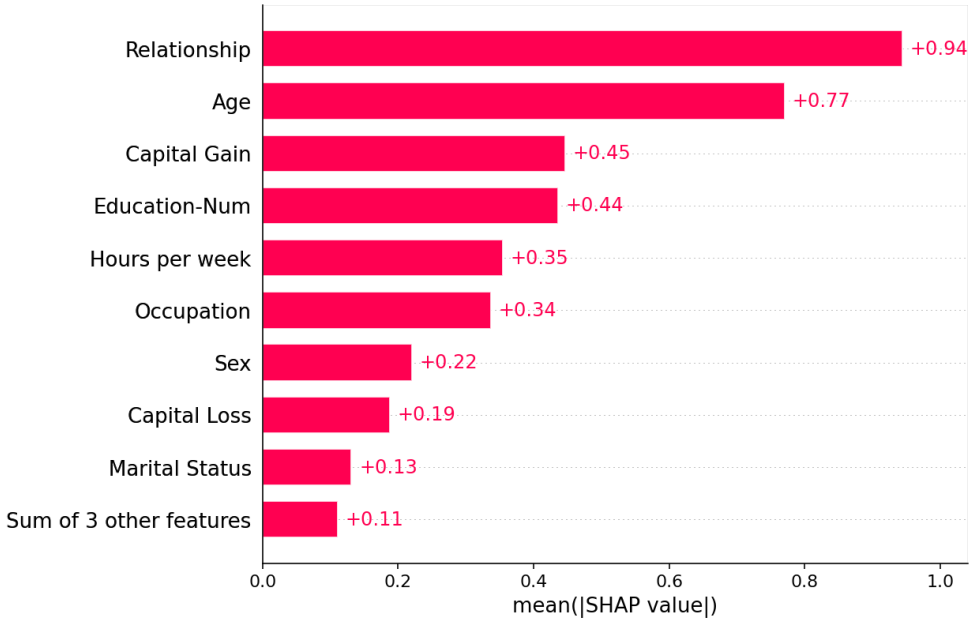

Figure 2.4: Features importance in the income prediction of the SHAP adult dataset according to the explanability model SHAP [1].

These advanced explanations allow users to understand the role of each feature in the model's decision-making process, contributing to building confidence and comprehension in the use of machine learning models.

2.5 Explainability in Deep Learning

When we talk about **Neural Networks** or **Multi-layer Perceptrons** we enter world of **Deep Learning**. These types of models are able to find and infer complex relationships among variables. They are very powerful at a predictive level and for this reason they are so widely used. However, their complexity greatly increases the difficulty of understanding how they work. For this reason, Neural Networks are considered black-box, since it is extremely difficult to understand their behavior and why certain predictions are made.

In these models, the main explainability methods used are post-hoc approaches, in particular those that distinguish the models by type of architecture. Among the most famous, there are specific techniques for Image Classification models based on **Convolutional Neural Networks** (**CNN**). Usually, these models are constructed by arranging convolutional layers and pooling layers in a sequence to autonomously grasp progressively more advanced features. Toward the conclusion of this sequence, one or more fully connected layers are applied to convert the output features map into scores. This structure entails extremely complex internal relations that are very difficult to explain. Remember that Image Classification models take an image as input and return a class label as output.

Fortunately, achieving explainability for CNNs is comparatively simpler than for other model categories, given that human cognitive capabilities naturally incline towards comprehending visual data [4].

In fact the explainability approaches for this type of model are purely visual, in particular they consist in evaluating the contribution/influence of each area/pixel of the image in the prediction made by the model. This can be summed up in a **heatmap** in which each pixel is associated with a value that indicates how relevant it was in the classification process (see an example in Figure 2.1).

One of the most famous approaches, which has also been used in this thesis project, is **Grad-CAM**, discussed in the following section. As the name implies, this method relies on gradient analysis to calculate the importance of image areas. However, **gradient-free** approaches also exist in the literature. Two methods, tested in our experiments, with performances comparable to Grad-CAM are **Ablation-CAM** and **Eigen-CAM**.

2.5.1 Grad-CAM

Gradient-weighted Class Activation Mapping (**Grad-CAM**) is a technique for producing "visual explanations" for decisions made by models based on Convolutional Neural Networks (CNN), with the aim of making them more transparent and interpretable.
Grad-CAM uses gradients, obtained in the last convolutional level of CNNs, of a target class, to produce a localization map highlighting the image region for the intended target. This approach is able to provide an explanation of the model's decisions without directly modifying its structure or parameters [35].
Unlike other methods such as **Guided Backpropagation** [36] and **Deconvolution** [42], Grad-cam has a **class-discriminative** approach. This means that the activation maps differ according to the class on which the explanation is requested, while in the other methods they tend to be very similar for all classes (see Figure 2.5) [35].

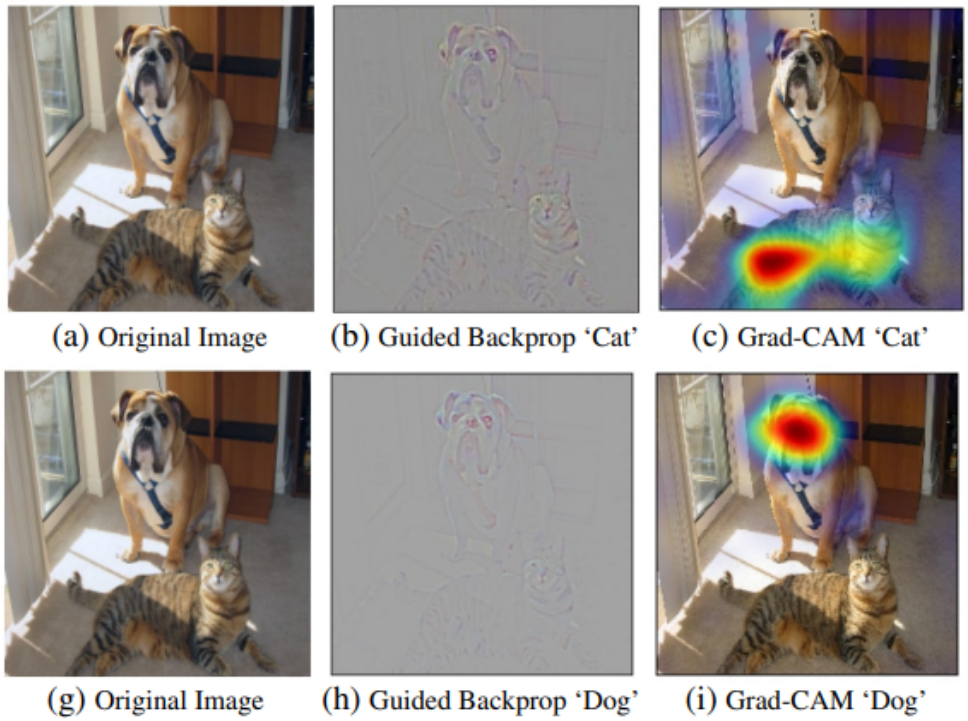

Figure 2.5: Grad-CAM comparison with other explainability methods [35].

One of the most relevant works for the development of the Grad-cam method is the

Class Activation Mapping (**CAM**) approach [43]. This approach modifies the architecture of the classification model (CNN) by replacing the last fully-connected level with a convolutional level and global average pooling, thus obtaining feature-maps specific to the class of interest (this approach has also been tested using global max pooling) [35].
A drawback of the previously mentioned CAM methods is that it requires feature-maps to directly precede the softmax layer, so they are *applicable only to a particular type of CNN architecture* that meets these requirements. This type of architecture could have lower performance than more classic models for tasks such as image classification and could even be inapplicable for tasks such as image captioning. Grad-cam uses a new method to combine feature maps using the gradient **without directly modifying the structure of the models**. In this way, the approach is **generalized** and made applicable to a wider family of convolutional models (thinking of the CNN architecture, CAM becomes a special case of Grad-cam) [35]. To perform localization, Grad-CAM requires only one *forward pass* and one *backward partial pass* [35] per image, which makes it much **more efficient** than many other techniques.

Method Description

Numerous works have claimed that the deeper levels in a CNN capture visual constructs with higher abstraction [26, 8]. Furthermore, convolutional layers naturally retain spatial information that is lost in fully connected ones, so we can expect the **latter convolutional layers to have the best compromise between high-level semantics and detailed spatial information**. The neurons in these layers look for specific information of the semantic class in the image (i.e. specific parts of the objects). **Grad CAM uses gradient information**, flowing into the last convolutional layer of the CNN, **to assign importance values to each neuron for a particular class of interest**. Although the technique is quite general, as it can be used to explain activations at any level of a deep network, in Grad-CAM the main focus is placed on decisions at the **output level**.
As shown in Figure 2.5, to obtain the class-discriminative localization map $L^c_{Grad-CAM} \in R^{u \times v}$ of height u and width v for any class c, the gradient of the class score y^c before the softmax with respect to the activation map A^k of the convolutional layer is first computed, i.e., $\frac{\delta y^c}{\delta A^k}$. These gradients, through backpropagation, are pooled using

global average pooling across the width and height dimensions (indicated by i and j respectively) to obtain the neuron importance weights α_c^k [35].

$$\alpha_c^k = \frac{1}{Z} \sum_i \sum_j \frac{\delta y^c}{\delta A^k} \tag{2.1}$$

While computing α_c^k, during the backward propagation of gradients with respect to activations, the precise calculation involves successive matrix products between weight matrices and gradients with respect to activation functions, until the last convolutional layer where gradients are propagated.
Thus, this weight α_c^k represents a partial linearization of the network and captures the **importance** of the feature map k for a target class c.
To obtain a *heatmap*, a combination of activation maps followed by a ReLU operation is applied to yield:

$$L_{Grad-CAM}^c = ReLU(\sum_k \alpha_c^k A^k) \tag{2.2}$$

A ReLU activation is applied to the linear combination of maps because the focus is solely on the features that have a positive influence on the target class, i.e., the pixels whose intensity should be increased to increase y^c.
It is likely that the negative pixels belong to other categories in the image. Without the ReLU, the localization maps sometimes highlight more than just the desired class and have poorer localization performance.
In general, y^c doesn't necessarily have to be the class score produced by a CNN for Image Classification; it could be any differentiable activation, including words in a caption or a response to a question.

2.5.2 Gradient-free Methods

Since the GradCAM method still represents the state of the art for gradient-based explanation methods, we decided to evaluate the performance of our algorithm also on gradient-free methods. This, with the aim of demonstrating that our attack can work with different explanation techniques.
We have therefore chosen two explanation methods capable of producing class activation maps among the most performing, **Ablation-CAM** [30] and **Eigen-CAM** [27],

described below.

2.5.3 Ablation-CAM

Ramaswamy et al. in [30] propose a class-discriminative localization technique that can generate gradient-free visual explanations for any CNN based architecture. With their experiments, the authors show that this gradient-free approach works better than state-of-the-art Grad-CAM technique.

Method Description

In this method, as with Grad-CAM, the explanation is produced for an image processed by a CNN classifier. During a forward pass through the model, the class activation score y_c for a specific class c is obtained. The authors assumed that this class score is a non-linear function of the activation map A_k of the final convolutional layer, where y_c represents the value of this function when activations in A_k are present [30]. To evaluate the impact of individual activation units in the A_k activation map, all activation cell values were set to zero, and the forward pass was repeated with the same image. The ablation of unit k can lead to a possibly reduced activation score y_k^c, which serves as a baseline for A_k.
The Ablation-CAM approach is robust against both saturation, which indicates an unimportant filter, and explosion, which marks a filter with minimal value as highly important. We calculate the importance of filter k for class c using a variant of the *slope* [30], denoted as w_k^c and defined as:

$$w_k^c = \frac{y^c - y_k^c}{y^c}$$

This importance value can be interpreted as the fraction of drop in the activation score of class c when feature map A_k is removed [30].
The Ablation-CAM can be obtained as a weighted linear combination of activation maps and corresponding weights, similar to Grad-CAM:

$$L^c_{\text{Ablation-CAM}} = \text{ReLU}\left(\sum_k w_k^c A_k\right)$$

The ReLU function ensures that only units with positive drop values, i.e., those units

whose absence causes a drop in the class score y^c, are retained.
It's worth noting that the authors chose to analyze the drop in (unnormalized) class activation scores rather than the drop in confidence scores returned by the softmax layer. This choice is because a drop in confidence scores can result from an increase in activation scores of other classes, whereas a drop in class scores specifically focuses on the class in question. They experimented with drop in confidence scores but found the visualizations to be less reliable.

2.5.4 Eigen-CAM

Muhammad et al. [27] provides a simpler and intuitive way of generating CAM. The proposed Eigen-CAM computes and visualizes the principle components of the learned features/representations from the convolutional layers. The authors demonstrate that Eigen-CAM can robustly and reliably localize objects without the need to modify CNN architecture, or even to backpropagate any computations.

Method Description

The authors assume that all pertinent spatial features in the input image, learned throughout the CNN model's hierarchy, will be preserved during optimization, while non-relevant features will be regularized or smoothed out.
Let I represent the input image of size $(i \times j)$ such that $I \in R^{i,j}$. Let $W_{L=n}$ represent the combined weight matrix of the first k layers of size (m, n).
The class-activated output is the image 'I' projected onto the last convolutional layer $L = k$ and is given by:

$$O_{L=k} = W_{L=k}^T I$$

To compute the principal components of $O_{L=k}$, we factorize it using singular value decomposition [27]:

$$O_{L=k} = U \Sigma V^T$$

Here, U is an M x M orthogonal matrix, with columns as the left singular vectors; Σ is a diagonal matrix of size $M \times N$, with singular values along the diagonal, and

V is an N x N orthogonal matrix, with columns as the left singular vectors [27].
The class activation map, $L_{\text{Eigen-CAM}}$, is derived from the projection of $O_{L=k}$ onto the first eigenvector:

$$L_{\text{Eigen-CAM}} = O_{L=k} V_1$$

Where V_1 represents the first eigenvector in the V matrix.

As can be seen from the description of these three methods, each of them has a different functioning. This allows us to demonstrate how our attack algorithm does not depend on a specific explainability method. Indeed, following the black-box approach, our adversarial attack can be used to generate adversarial examples for any explainability method.

2.6 Evaluating Quality of Explanations

Since we cannot trust the XAI methods blindly, it is necessary to define and have some **metrics capable of providing a quality index of each explanation method**. However, there are still no metrics used as standard in the state of the art. The objective assessment of the explanations quality is still an active field of research. The various metrics defined are not general but depend on the specific model task on which the XAI methods are used.
Many efforts have been made to define **quality measures for heatmaps and saliency maps** which explain individual predictions of an Image Classification model. One of the main metrics used to evaluate explanations in the form of saliency maps is called **FaithFulness**.
The faithfulness of an explanation refers to whether the relevance scores reflect the true importance. The assumption of this evaluation metric is that perturbation of relevant (according to the heatmap) input variables should lead to a steeper decline of the prediction score than perturbation on the less important ones. Thus, the average decline of the prediction score after several rounds of perturbation (starting from the most relevant input variables) defines an objective measure of heatmap quality. If the explanation identifies the truly relevant input variables, then the decline should

be large [33].
There are several methods to compute the Faithfulness. In the work of this thesis we have decided to use a metric called **Area Over the Perturbation Curve** (**AOPC**), described below.

2.6.1 Area Over Perturbation Curve - AOPC

The **AOPC** approach [40, 34] , measures the change in classifier output as pixels are sequentially perturbed (flipped in binary images, or set to a different value for RGB images) in order of their relevance as estimated by the explanation method. It can be see as a greedy iterative procedure that consists of measuring how the class encoded in the image disappears when we progressively remove information from the image x [34].
The classification output should decrease more rapidly for methods that provide more accurate estimates of pixel relevance. This approach can be done in two ways:

- **Most Relevant First** (**MoRF**): The pixels are perturbed starting from the most important according to the heatmap (rapid decrease in classification output).
- **Least Relevant First** (**LeRF**): The least relevant pixels are perturbed first. In this case, the classification output should change more slowly the more accurate the saliency method.

In this work, we decided to use the MoRF version. It is computed as follow:

$$\text{AOPC}_M = \frac{1}{L+1}\left\langle \sum_{k=1}^{L} f(x_M^{(0)}) - f(x_M^{(k)}) \right\rangle_{p(X)}$$

where M is the pixel deletion procedure (MoRF or LeRF), L is the number of pixel deletion steps, $f(x)$ is the output value of the classifier for input image x (i.e. the probability assigned to the highest-probability class), $x_M^{(0)}$ is the input image after 0 perturbation steps (i.e. $x_M^{(0)} = x$), $x_M^{(k)}$ is the input image after k perturbation steps, and $\langle . \rangle_{p(X)}$ denotes the mean over all images in the data set [34].

There exist also other approaches such as **iAUC** (Area Under the Insertion Curve)

[24, 29], which works almost in the opposite way. In **iAUC** method we start from a completely disturbed image and gradually insert areas of the image (starting from the most important ones). Thus it is expected that if the XAI method produced a good explanation, the target class will be found in the first few iterations. However, the logic remains similar to **AOPC** with the MoRF version. In both cases a large change is expected in the first iterations if the explainability method correctly captures the image areas fundamental for classification.

Chapter 3

Work Objectives and State of the art

This chapter describes the objectives of this work and proposes an exploration on the state of the art. In Section 3.1 the idea behind the research, and the used approaches will be described. In Section 3.2 some related works, the techniques and the results obtained will also be presented.

3.1 Aims and Goals

The field of Explainable Artificial Intelligence (XAI) has gradually gained prominence in recent years. This rise can be attributed to the growing adoption of artificial intelligence technologies and the resulting complexity of models, which often pose challenges in terms of interpretability. However, behind this growing enthusiasm, some researchers have begun to pose a crucial question: ***how reliable are XAI methods and can they be fully trusted, especially in very sensitive scenarios?*** And the growing use of these methods in critical areas raises a natural concern: ***shouldn't their reliability be scrupulously tested?***

Kindermans et al. (2019) ([21]) introduced the input **invariance axiom** as a metric for evaluating the reliability of a XAI method of reflecting model's behavior. The concept of input invariance implies that a salience-based method should faithfully capture the responsiveness of the model to various transformations of the input, indicating its adherence to changes in the nature of the input.
Upon subjecting a range of methods to rigorous testing, numerous investigators have

unearthed a stark reality: this axiom does not hold true for every model. An example involves perturbing images slightly and submitting them to image classification models. The outcome is an **accurate label prediction**, yet when the XAI method is tasked with elucidating the rationale behind the prediction, the unveiled **explanation diverges from the model's behavior**. In essence, an explanation is produced without align with what the model actually performed. Considering these discoveries, the basis of trust in these methods is undermined.

This thesis want to address this challenge, aiming to subject some of the most important XAI methods for image classification to a rigorous examination of their reliability.

To achieve this goal, an advanced Adversarial Machine Learning algorithm based on **multi-objective optimization** and **evolutionary** techniques will be used (the algorithm will be described in Chapter 4).
Summarizing, the goal of this work is to demonstrate that some of the most used XAI methods do not reflect the input invariance axiom by exploiting the faithfulness. Then, a general attack strategy that can be transferred to each XAI model and method will be proposed. An example of the expected result for this kind of adversarial attack is shown in Figure 3.1. In this image we can see how the heatmaps (i.e. the explanations) for the original and the attacked images are completely different, despite the same results in terms of classification labels. Moreover, in Figure 3.2 we can see the comparison between the original image and the perturbed one: the application of filters is almost imperceptible despite the large difference produced between the explanations.

Through the synergy of AML and XAI, this study endeavors to push the boundaries of knowledge and establish a nuanced understanding of the ever-evolving relationship between interpretability and security in the realm of AI.

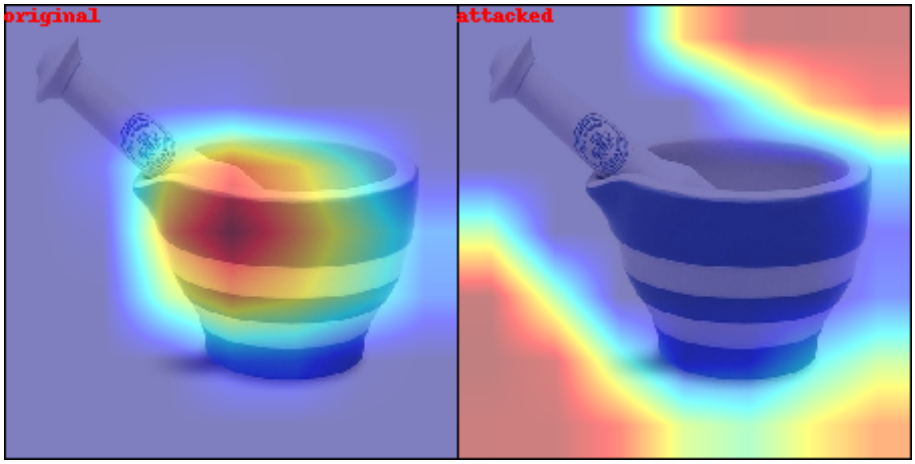

Figure 3.1: Example of expected result for the adversarial attack. The explanation of the original image is on the left, while the explanation for the perturbed image generated by our algorithm is on the right.

Figure 3.2: Example of comparison between original image (left) and perturbed image (right).

3.2 Related Work

Adversarial attacks on explainability methods represent a new problem that has recently emerged in the literature. In particular, Kindermans et al. (2019) [21] were among the first to address this issue. In their work, they presented the *input invariance axiom*, which needs to be satisfied to ensure the reliability of the interpretation

method. They have demonstrated that numerous methods do not satisfy input invariance using a simple transformation like a *constant shift* of the input, that changes the attribution of these methods but does not affect the model prediction or weights. On their work, the authors use two neural networks, in which the second one is trained on the same input of the first except for the addition of a constant vector that represents the shift. They show that the bias of the Network 2 compensates for the constant shift and resulting in two networks with itdentical weights and predictions. They also noted that the gradient of the loss function with respect to the input remains unchanged. So, each method is evaluated by comparing the saliency heatmaps for the predictions of Network 1 and Network 2, where the input on the second network is simply the input of the first one with the addition of a constant shift. Based on their definition, a saliency method that satisfies input invariance will produce identical saliency heatmaps for Network 1 and 2 despite the constant shift in input. In their experiments they found that several explanations methods, for example some of those based on gradients computation does not respect the input invariance axiom.

In the same year, also Ghorbani et al. [13] presented a paper in which they demonstrate how to generate adversarial perturbations that produce perceptively indistinguishable inputs (that are also assigned to the *same* predicted label) having very *different* interpretations. With their experiments, the authors show that systematic perturbations can lead to dramatically different interpretations without changing the associated label. In the same way as Kindermans et al. (2019) [21] for the input invariance axiom, they propose a new definition. In particular, they define the *fragility* of the interpretation of neural network. More precisely, given an image, its interpretation is called *fragile* if it is possible to generate an image that is perceptually indistinguishable from the original, that has the same prediction label, yet a substantially different interpretation [13]. Likewise Ghorbani et al., investigate methods that assign importance scores to each feature (this includes gradients based methods, DeepLift and integrated gradients). The authors defined three adversarial attacks against feature importance methods, each of these consists in taking a series of steps in the direction that *maximizes a differentiable dissimilarity function* between the original and perturbed interpretation. In order to evaluate their attack, they used two different metrics for quantifying the similarity between the interpretations of two different images: *Spearman's rank order correlation* and *Top-k intersection.*

Another related work was that proposed by J. Heo et al. [16] in which the authors use a model fine-tuning step in order to fool the neural network interpretation methods. They succeeded by incorporating the interpretation results directly in the penalty term of the objective function for fine-tuning. They showed that the state-of-the-art saliency map based interpreters, e.g., LRP, Grad-CAM [35], and SimpleGrad, can be easily fooled with this type of model manipulation.

There is a great variety of types of adversary attacks on explanation methods and, as viewed before, it is possible to fool XAI methods acting directly to the neural networks structure (i.e. Kindermans et al. (2019) [21] and J. Heo et al. [16]) or only by optimizing a perturbation to apply on the images that are passed as input to the model.

Dombrowski et al. [12], in their work, follow the approach of optimizing perturbations, that are visually imperceptible, keep the network's output approximately constant, while permits to fool an explaination method. Additionally, they introduced the concept of *target interpretation*, which involves an alternative interpretation that the adversarial attack aims to reach. In other words, they were able to generate an attacked image whose interpretation resembles the target. Their manipulation technique has three main properties: the network's output should remain approximately constant; the explanation should be close to the target map; and the norm of the perturbation δx added to the input image should be small. They achieved this manipulation by optimizing a loss function that incorporates all these properties by using three different metrics: the structural similarity index (SSIM), the Pearson correlation coefficient (PCC), and the mean squared error (MSE). These measures were also used to evaluate the similarity between the perturbed image and the original one.

A different approach using the same concept of target explanation is the one proposed by Anders et al. [2]. They based their work on the assumption that, for any classifier, one can always construct another classifier which has the same behavior on the data (same train, validation, and test error) but has arbitrarily manipulated explanation maps. SSIM, MSE and PCC are used to evaluate the similarities also in this work.

In this chapter, we have examined some of the most significant and innovative research

in the field of adversarial attacks on explanation methods for image classification. We have seen, for example, how Dombrowski et al. [12] developed an effective approach for generating adversarial attacks using imperceptible perturbations for humans. Furthermore, we have examined the properties that makes this technique effective and the similarity metrics used to evaluate the effectiveness of an attack. In the next chapters, we will explore in detail a new approach based on evolutionary algorithm and we will evaluate its effectiveness in different image classification scenarios.

Chapter 4

The Attack Algorithm

This Chapter describes the algorithm used to generate the adversarial examples and all its main characteristics: the fitness function and its components, the image classification model and the XAI method used, as well as the possible hyperparameters.

4.1 Problem Definition

Before starting with the description of the algorithm used to craft the attack, it is appropriate to codify what has been previously introduced in Chapter 3, giving a formal definition of the problem.
Our objective can be summarized as follows:

Given a set $S = \{f_1, f_2, ..., f_m\}$ of filters as described in Section 4.3, and a clean image x, we want to find a sequence of n parameterized filters $\{f_{k_1}(\alpha_1, s_1), ..., f_{k_n}(\alpha_n, s_n)\}$ able to produce an adversarial attack against a XAI method starting from the image x. An image classification model will be used to produce a label for the two images y_x and y_{x^*}. The predictions will be explained through a XAI method by producing two heatmaps, referred with h_x and h_{x^*}. The goal is to maximixe the *difference* between the explanation of the original image h_x and the explanation for the perturbed image h_{x^*}. The **quality of the attack** will be verified by evaluating this *difference*. Furthermore, there is the constraint that the modified image x^* must have the same classification label as the original image x, i.e. $l_x = l_{x^*}$.

4.2 Multi-Objective Evolutionary Adversarial Attack

In this work, to generate the adversarial attacks, we propose a technique based on a **Multi-Objective Evolutionary** (**MOE**) approach. The proposed algorithm has been implemented starting from an already existing basis, which was originally used to perform Emotion Adversarial Attack (EAA) [7].
An important feature of our proposed approach is that the algorithm works in a black-box scenario, so it does not require any information about the model's parameters or gradient values nor information about which XAI method you want to attack.
The approach consists in applying perturbations to images in order to deceive the XAI method. To generate the perturbations we use a combination of image filters; in particular, we compose and parametrize popular image enhancing methods, like Instagram filters. To obtain the most effective perturbations, the filter composition is generated using an optimization process implemented by a nested-evolutionary algorithm [7].
The generation of adversarial examples occurs according to the *per-instance* approach presented in [3]. In this manner, the sequence of filters and the relative management of the parameters take place in a specific way for each image. This ensures greater performance as the attack is optimized for each image based on its characteristics.
Moreover, as in [7], performing the attack using well-known filters widely used in social media (e.g., Instagram) makes our filter composition indistinguishable from any other filter composition extensively used every day to enhance photos and images.

4.3 About the Filters

Taking inspiration from Instagram's user-friendly approach to image modification, we present a novel application that mixes various image filters to create personalized adversarial image transformations.
Building upon the ideas presented in previous works such as [7, 3], our algoritmh generates the perturbation starting from a selection of Instagram-style filters: *Clarendon, Juno, Reyes, Gingham, Lark, Hudson, Slumber, Stinson, Rise*, and *Perpetua*. Each of these filters possesses distinctive characteristics that encompass various aspects, including **contrast, saturation, brightness** and **shadow levels**. From the com-

position of these filters it is possible to create a wide range of effects on the image.
The foundation of our approach lies in parameterizing each filter with two key values, **intensity α and strength** s (as for [7]), both optimized through our learning process. The role of α is to alter the intensity of each filter component, e.g., contrast, saturation, brightness, gamma correction, edge enhancement. The s parameter is used to manage the filter impact, defined as the convex interpolation between the input image x and the altered image x^* [7]:

$$\text{strength}(x, x^*, s) = (1.0 - s) \cdot x + s \cdot x^*$$

In case $s = 0$, the image is not altered by the filter, and if $s = 1$ the filter returns a mutated image x^*.
It should be noted that the application of the filters takes place at the global level of the image, i.e. the same perturbation is applied to each pixel of the image.

4.4 The Algorithm

The algorithm used to generate the adversarial examples is implemented by revisiting the MOE approach proposed in [7], appropriately to reach our goal. We decided to use a **multi-objectives algorithm as it allows us to maximize the filters effectiveness for the attack, and minimize their perception to the human eye**, at the same time. This ability of multi-objective algorithms to manage and **optimize at the same time conflicting objective functions** such as these, is an essential and necessary feature in order to achieve our purpose. If a single optimization criterion approach is used, such as attack effectiveness, then the resulting images would be overly modified and distorted, as a result the attack would be easily detectable.
The optimization method consists of two nested evolutionary algorithms: an *outer algorithm*, using a generative adversarial approach based on a **genetic algorithm**, in charge of finding the sequence of filters to use; and an *inner algorithm*, based on **Evolution Strategy** (**ES**), used to choose the values for the filters parameters. Given a set $S = \{f_1, f_2, ..., f_m\}$ of m image filters, the outer algorithm genotype (with length l) is encoded as a **list of filters**, while the inner algorithm genotype is represented by a **list containing the parameters for each selected filter**.

The pseudocode is given in Algorithm 4.4.2.

4.4.1 Outher Algorithm

The outer-algorithm optimization is performed by a genetic algorithm where a population of N candidates is iteratively evolved through the generations. Candidates are randomly selected to compose and give rise to a new generation through crossover and mutation procedures. At the end of each iteration, the best candidates are selected for the next generation, evaluating them on their fitness [7].

- **Initial population**: Generated by randomly selecting l filters from the S available set. Their parameters are initialized to 1.
- **Crossover**: We use a *one-point crossover* to generate new off-springs (i.e., children) from random population members. In thi way every child can inherit genetic information from both parents.
- **Mutation**: one of the filters is replaced (with another filter), based on the probability of mutation. The new filter is initialized with random parameters.
- **Selection**: At each iteration, the N best individuals are chosen from the set of $2N$ candidates (i.e., parents and offsprings), according to their fitness, that is implemented as a multi-objective evolutionary problem based on two criteria, see Section 4.5.

4.4.2 Inner Algorithm

The inner algorithm takes care of optimizing the filter parameters. This is done using an evolution strategy of type $(1, \lambda)$ with $\lambda = 5$ [7], which updates a search distribution following the gradient towards increasing the fitness function.
At each iteration, N new candidates are generated through a perturbation process of the original individuals.
The whole process is repeated until a stopping criterion is met.

Algorithm 1: AGV- attack

Input: Image I, population size N, generations E
Initialize population P of N individuals;
Evaluate each individual of P;
for $e = 0$ **to** E **do**
  Offsprings = $\{\emptyset\}$;
  for $i = 1$ **to** N **do**
    Select randomly $parent_1$,$parent_2$ from P ;
    $\overline{p}_1 \leftarrow encode_1(parent_1)$;
    $\overline{p}_2 \leftarrow encode_1(parent_2)$;
    y_i = crossover($\overline{p}_1$,$\overline{p}_2$);
    $\overline{y_i}$ = mutation(y_i) ;
    $n_i \leftarrow encode_2(\overline{y}_i)$;
    $\overline{n}_i$ = optimizer$_O(n_i)$;
    Offsprings $\leftarrow (\overline{y}_i, \overline{n}_i)$;
  end
  foreach $(\overline{y}_i, \overline{n}_i) \in$ Offsprings **do**
    $b \leftarrow$ decode $(\overline{y}_i, \overline{n}_i)$;
    Evaluate the fitness;
  end
  P = selection(P, Offsprings) ;
end
return: best sequence of filters for image I;

In summary, our algorithm has the following characteristics:

- **Input**: The algorithm takes an image x as input. Following the per-instance approach, a specific sequence of filters is optimized for each image. The algorithm also needs a model to make predictions and an XAI method to produce the explanation on it.
- **How the algorithm works**: The algorithm starts from a random set of filters and tries to optimize their sequence and related parameters in order to have an attack as effective as possible.
- **How the evaluation of each possible solution takes place**: In the optimization phase, every possible solution (parameterized filter sequence) is evaluated according to a fitness function, representing the goodness of the solution.

This takes into consideration the effectiveness of the attack and its difficulty of perception to the human eye.

- **Output**: The algorithm outputs a sequence of filters and their parameters. By applying this perturbation to the input image x, we obtain a perturbed image x^*. If the attack successed, these images x and x^* will have the same class label but different explanations (produced by the same XAI method used during the optimization process).

4.5 The Fitness Function

In the realm of multi-objective optimization, fitness assumes a pivotal role, working as guiding compass that provides the right direction in the complex space of conflicting objectives. These goals might be diverse and interconnected, making challenging the attainment of an ideal solution.
The concept of fitness serves as a quantifiable measure that evaluate the performance of a candidate solution across these various objectives. Essentially, it encapsulates how well a candidate aligns with the problem goals.
In the pursuit of a solution that optimizes multiple objectives, **Pareto dominance** emerges as a cornerstone principle. Pareto dominance establishes a comparative hierarchy between solutions: solution A dominates solution B if it is at least as good as solution B in all objectives, while being strictly better in at least one objective. The **Pareto front**, a collection of non-dominated solutions, defines the boundary of optimal trade-offs where improving one objective comes at the expense of worsening another.
Fitness evaluation in multi-objective problems involves assessing candidates in the context of Pareto dominance. A candidate's fitness hinges on how well it fares against other candidates within the Pareto front. The closer a candidate is to the Pareto front, the more desirable its fitness value becomes, signifying a superior trade-off solution.

In the implemented multi-objective algorithm for generating adversarial examples we decided to use a fitness composed by **two metrics**. In particular, **it is necessary to use a metric that is able to lead the optimization towards a more effective attack and a metric that is able to balance the perturbation with respect**

to its modification to the image. This is because we want an attack that is effective but also not very visible to the human eye. To fulfill this purpose, we have identified and implemented various possible metrics (described below) for each of the two aspects.

4.5.1 Interesting Bounding Box

Before moving on to the objective functions used and their characteristics, it is necessary to introduce and define an important concept.
Since the explanation methods in general return a global heatmap covering the whole original image (see Figure 4.3), we need to define a method to select a refined area of major intereset. It is evident that the comparison must take place only between the areas of greatest importance, so they must be identified and formalized. This area is called **Interesting Bounding Box** and summarize as well as possible the global heatmap.
We know that the heatmaps returned by the explanation method has values from 0 to 255, where values close to 255 are associated to pixels with stronger contribution to the classification. For this reason, the approach we will use consists of identifying a bounding box covering this area. The method devised for constructing these two bounding boxes consists in carrying out a **thresholding** on the explanations, with a pre-established value (which tends to be high as we want to identify only the area of main importance). From this thresholded pixels we then build the perimeter that delimits the remaining area. This gives rise to a interesting bounding box like the ones in the Figure 4.1. This construction process is **parameterized** by a threshold value defined in Section 5.4.

4.5.2 Intersection over Union (IoU)

Intersection over Union (**IoU**) is a famous evaluation metric used mainly in **Image Segmentation** and **Object Detection** methods. In these methods, a model aims to locate an area (a bounding box) around a specific object in an image. IoU allows to evaluate the performance of the models by comparing the area identified by the model with the area provided by the *ground trouth*. To do this, the area of overlap between these two bounding boxes is calculated. In a task like Object Detection, the

model's performance improves as the two areas overlap more.
In our case, the significant regions of the image is identified by the **Interesting Bounding Box** defined in Section 4.5.1. The IoU is calculated by dividing the intersection area between the two regions by the union area of the same two regions:

$$IoU = \frac{A_I}{A_U}$$

where A_I is the area where the two regions overlap, while A_U is the total area covered by both regions.
The IoU value ranges between 0 **and** 1. An IoU of 0 indicates no overlap between the regions, while an IoU of 1 signifies a perfect overlap, meaning the two regions match completely.
We decided to use IoU applied to the bounding boxes corresponding to the explanations as an attack effectiveness evaluation metric. In particular, we consider **the attack the more effective the less the bounding boxes of the explanations overlap**. In our case, unlike Object Detection and Image Segmentation tasks, the goal is to **minimize the value of IoU** (low values IoU indicate a good attack).

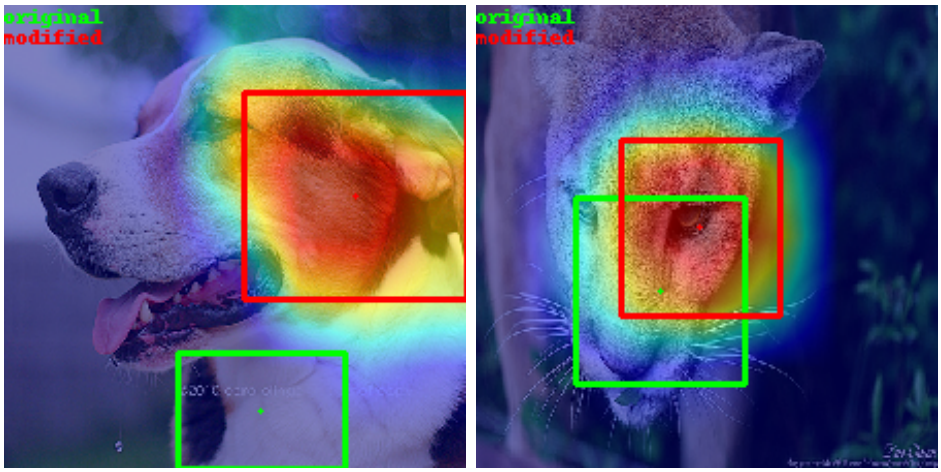

Figure 4.1: Example of perturbed images and their bounding boxes related to the explanation method. Each background image represents the explanation obtained on the perturbed image. The green rectangle indicates the **Interesting Bounding Box (IBB)** of the original image explanation. The red rectangle indicates the IBB of the perturbed image explanation. In the left image we have an IoU value of 0 as the two areas do not overlap (indicates the best possible value for an attack). In the right image we have a less effective attack as the two bounding boxes overlap.

4.5.3 Center Distance

Another possible metric to be used to evaluate the goodness of the attack is called **Center Distance**. This metric consists in calculating the Euclidean distance between the centers of the two interesting bounding boxes delimiting the most important areas of the two heatmaps, as defined in Section 4.5.1. Once the two bounding boxes and their coordinates have been obtained, the centers are identified and the distance between the two is calculated.
Formally, the metric is defined as follows:

$$\text{Center Distance} = \sqrt{(x_{b^*} - x_b)^2 + (y_{b^*} - y_b)^2}$$

where:

- (x_b, y_b) are the center coordinates of the original image interesting bounding box.
- (x_{b^*}, y_{b^*}) are the center coordinates of the perturbed image interesting bounding box.

The obtained value is then normalized between 0 and 1 by dividing it by the maximum possible value for the distance. This value depends on the size of the processed images. In the case of 224 x 224 images, this value is ≈ 315. This is done to standardize it across all fitness objective functions.
In Figure 4.2 is possible to see the two points which represent the centers. The line that connects the two center represents the center distance and is computed as euclidean distance.
In this case a high value for the center distance represents a good attack performance as it indicates that the two bounding boxes are more probably far away, and consequently the explanations are probably more different. For this reason **the center distance value must be maximized by the algorithm, while the IoU must be minimized**.

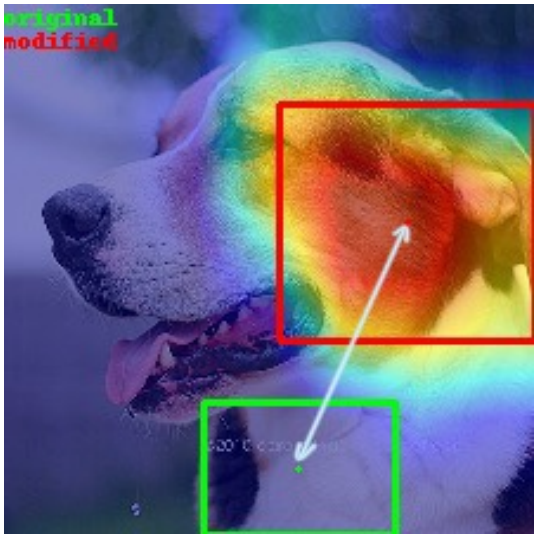

Figure 4.2: Example of center distance. The point in the center of each bounding box represents the *center*, and the distance between the two (seen as the line connecting them) is the value for the center distance metric.

4.5.4 SSIM

The **Structural Similarity Index** (**SSIM**) [41] is a metric used to assess the structural similarity between two images. It's often employed to evaluate image quality, quality degradation during image compression, or the effectiveness of noise reduction algorithms.

SSIM takes into account various factors contributing to human visual perception, including brightness, contrast, and structure. More specifically, SSIM calculates the similarity between the intensity distributions of pixels in two images.

The formula for SSIM is relatively complex and involves three components: **luminance** (L), **contrast** (C), and **structure** (S). The overall SSIM is a weighted combination of these components.

The general equation to calculate SSIM between two images x and y is:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

Where:

- μ_x and μ_y are the means of the pixels in images x and y (**luminance compar-**

ison);

- σ_x^2 and σ_y^2 are the variances of the pixels in images x and y (**contrast comparison**);

- σ_{xy} is the covariance between pixels of images x and y and C_1, C_2 are constants to ensure stability in the calculation (**structure comparison**).

In essence, SSIM indicates the structural similarity between two images and provides a more sophisticated and human-perception-friendly measure compared to simpler difference metrics like Mean Squared Error (MSE). **The SSIM value can range from -1 to 1, but in practice, it typically falls between 0 and 1**, where:

- A SSIM value of 1 indicates that the two images are structurally identical, meaning they are perfectly similar.

- SSIM values close to 0 indicate a lack of structural similarity between the images, meaning they are very different.

Given its characteristics, the SSIM metric can be used in two ways: as an attack effectiveness evaluation and as a balancing metric to avoid too strong perturbations.

- **Attack effectiveness evaluation**: In the phase of evaluating the effectiveness of the attack, the SSIM metric can be used to maximize the difference between the two explanations. Therefore, in this case, the goal will be **to minimize the SSIM value between the explanation of the original image and the one computed for the perturbed image**. This optimization function is called $SSIM_{heatmap}$ and it is defined as

$$SSIM_{heatmap}(x, x^*) = SSIM(h_x, h_{x^*}) \tag{4.1}$$

  In the rest of the thesis, when it cannot induce any mistake, we will indicate this value just with $SSIM_{heatmap}$. For example, for two explanations showed in Figure 4.3, the $SSIM_{heatmap}$ value is low.

- **Perturbation strength evaluation**: By allowing to evaluate the structural similarity between two images, the SSIM metric can also be used to mitigate the effect of the perturbation. This is done by **maximizing the SSIM value between the original image and the perturbed one**. In this case we define the objective function

$$SSIM_{image}(x, x^*) = SSIM(x, x^*) \tag{4.2}$$

that in the following we will indicate just with $SSIM_{image}$. For example, the two very similar images in Figure 3.2 can reach a very high $SSIM_{image}$.

From a technical point of view, since our algorithm is a minimization algorithm, the objective function used for attack detectability is computed as: $1 - SSIM_{image}$. In this way, we have to minimize the objective function, for this metric too.

In the tests carried out during the experiments, different combinations of these fitness metrics have been applied. One of the main goals was precisely to identify the combination able to reach the best performance. Some results are reported in Chapter 6, where also some problems raised during the experiments are discussed.

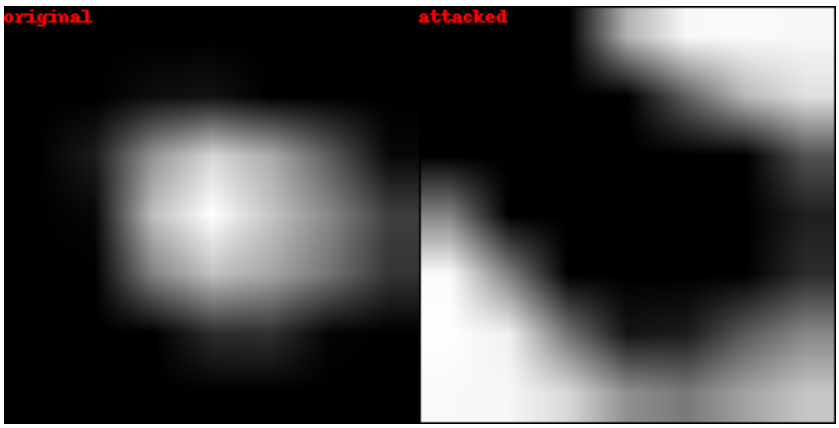

Figure 4.3: Comparison between the explanations obtained using the Grad-CAM method. On the left there is the explanation on the classification for the original image, on the right that for the perturbed image.

Chapter 5

Experiments Setup

This Chapter contains other important information about this work, including the hyperparameters setting , the reference classification model and the dataset used for the experiments.

5.1 Image Classification Model

To carry out the attack with our algorithm, an Image Classification model is needed. The model is used to associate a **class label** to the input image, which must remain unchanged even after the perturbation is applied to the image. For each image there are therefore several classification phases: the initial one to establish the class label of the original image and some other ones to run the algorithm optimization phase. In the latter, the predictions serve two primary purposes. Firstly, they allow to filter out perturbations that result in a class change; additionally, they are needed to generate the explanations during the optimization phase, which are subsequently evaluated using various fitness metrics to assess the attack's effectiveness.
The Image Classification model used is **ResNet50** [15]. This model was chosen for its excellent performance in the Image Classification task and for its efficiency [18, 20]. In fact, it is very fast to make inference, this allows to speed up this phase as much as possible and not burden the already expensive computation of the algorithm.
In particular, ResNet-50 is a deep neural network architecture, which is part of the ResNet (Residual Networks) model family, known for addressing the problem of train-

ing deep networks by utilizing residual blocks.
The ResNet-50 architecture was introduced by Kaiming He et al. [15]. It has become one of the benchmark models in the field of image classification and is often used in competitions.
Key features of ResNet-50:

- **Residual Blocks**: The main innovation of ResNet models, including ResNet-50, lies in the residual blocks. These blocks enable the model to learn the differences ("residuals") between expected and intermediate outcomes. This addresses the problem of degradation in performance in deep neural networks.
- **Depth**: ResNet-50 is a deep model with 50 layers (hence the "50" in the name). This makes the network powerful enough to learn complex representations from large amounts of data.
- **Block Structure**: ResNet-50 is divided into different blocks, each containing multiple layers. Each block includes a series of convolutions, batch normalizations, and activation functions.
- **Pooling and Fully Connected Layers**: After the convolutional blocks, ResNet-50 uses a global average pooling layer and then one or more fully connected layers for final classification.

The structure of ResNet-50 is quite intricate, but its usage is simplified through deep learning libraries like TensorFlow and PyTorch that allow us to import the pretrained model and use it in a few lines of code.

5.2 Images Dataset

As regards the set of images used to carry out the tests, we have chosen a sample of 200 images taken from **ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012)** [32]. It is a subset of the larger ImageNet dataset [19], which is widely used as a benchmark for image classification and object detection tasks. The complete ILSVRC2012 dataset consists of approximately 1.2 million training images, 50,000 validation images, and 150,000 test images. These images cover 1,000 categories of objects, animals, and scenes.

5.3 XAI Method tested

The XAI methods whose reliability we wanted to test are Grad-CAM, Ablation-CAM and Eigen-CAM (described in the Section 2.5.1 and 2.5.2). These methods have been chosen above all because they are widely used: being able to invalidate their reliability lays the foundations for continuous research in this direction. It should also be noted that methods such as Eigen-CAM and Grad-CAM are very fast in producing the explanation, and since they are called many times for each iteration, this allows not to slow down the execution of the algorithm. On the contrary, as described in Chapter 6 Ablation-CAM is particularly expensive from a computational point of view.
However, given the characteristics of our algorithm, it is possible to test any other method that produces explanations in the form of a heatmap.

5.4 Algorithm Hyperparameters

The developed algorithm has several hyperparameters to set. Using the correct values can lead to better attack performance. In the various tests performed, different values for each of the hyperparameters were tested. The most important parameters and the values used for each are described below.

- **Number of epochs**: The number of epochs indicates the number of times that the external cycle of the Algorithm 4.4.2 will be executed. This value defines a limit to the number of generations in which the individuals of the population will be evolved. The number of epochs will greatly influence the computational cost of the algorithm as it grows. It is obviously necessary to have a large enough number of epochs to guarantee sufficient optimization of the population. For our tests we used a value of 10 epochs per image.

- **Population size**: As with the number of epochs, the population size is another very important parameter regarding the research and optimization phase of the attack algorithm. In particular, population size refers to the number of individuals that are evolved from generation to generation through mutation, crossover and selection. Each individual corresponds to a possible solution. Also in this case, this parameter greatly influences the computational cost of the algorithm as it grows. Nevertheless, it is still essential to establish an

appropriate parameter value in order to optimize research space exploration and fully harness the capabilities of evolutionary algorithms. For our tests, we used a value of 10 individuals per population.

- **Number of Filters**: This parameter determines the number of filters that will compose the perturbation applied to the target image during the attack. This value will therefore influence the composition of each individual in the population, i.e. the genotype of the evolutionary algorithm. The higher the number of filters, the greater the complexity of the effects that can be applied to the image. Obviously, this will also involve a greater number of parameters to be optimized for the inner algorithm, since two parameters are associated with each filter. At the same time, too many filters can produce a too visible attack and a small number may not produce an effective one. In our tests we used a value of 3, 5 and 6 filters.

- **Threshold value**: This parameter is used just in the cases where Center Distance and IoU are used in the fitness function. The threshold value is used in the building phase of the interesting bounding box. The possible range for this value is from 0 to 255. The higher the value, the more important the area that will be selected to form the bounding box. **For low values** we will have a **too large interesting bounding box**, this could cause errors in the suitability evaluation phases as we are also evaluating pixels of low importance. At the same time, **too high values** lead to a **very small bounding box**, reducing the evaluation area too much. After a few preliminary tests, evaluating the bounding boxes obtained, we decided to use a value of 170 for thresholding phase. In this way we will have a bounding box representing the most important area of the explanation, but which is large enough for a correct evaluation phase.

5.5 Semantic Evaluation of the Attack

To evaluate the results obtained from the tests, we decided not to rely only on fitness metrics. In particular, we decided to also carry out an evaluation using a metric capable of analyzing the results of the attack more deeply. For this we used the

AOPC described in Section 2.6.1. This metric is also able to evaluate the attack from a **semantic point of view**, measuring how much the area identified as important by the explanation is actually important in the prediction.
In our case, **we measured the difference in the AOPC value before and after the attack for each explanation. The results are averaged over the entire dataset in order to have a single reference value.** Also in this case, we needed to specify some parameters. In particular, we needed to define how many pixels had to be removed from the image and the method for doing so. In our approach, having images of size 224 x 224 pixels, we have decided to divide the image into 8 x 8 pixel blocks (each block corresponds to 0,12% of the image). In this way, the image is seen as a 28 x 28 grid of pixel blocks (784 blocks). From here, we need to select the order in which the blocks are perturbed, and we decided to remove blocks in decreasing order of importance. To do this, to each block is assigned an importance value, which is given by the sum of the importance values of each pixel within it. Pixel importance values are taken from the heatmap produced by the explainability method. The pixel values can range from 0 (for the minimum importance) to a maximum of 255. Once the importance values of the blocks have been computed, they are sorted in descending order and the blocks are perturbed by proceeding iteratively, in this order. The perturbation operation can be performed in several ways: by assigning a constant value to the pixels of the block (for example by setting everything to 0); by adding a random noise or by assigning the pixels the mean value of the pixels within the block. In this work, we follow this last strategy in order to minimize the influence of randomness.

5.6 Objective Function combination for the Fitness

We opted to conduct multiple tests using various combinations of metrics to use in the fitness function.. The obtained results are reported and analyzed in Chapter 6. In particular we have decided to use the combinations described in the table below:

Attack effectiveness evaluation metric	Metric to reduce attack visibility
IoU	$SSIM_{image}$
Center Distance	$SSIM_{image}$
$SSIM_{heatmap}$	$SSIM_{image}$

5.7 How to keep the same classification label

As previously mentioned, **a very important constraint of our attack definition (in Section 4.1)** is that the class label between the original image and the perturbed image remains the same. For this reason, only perturbations that do not modify the class label are admitted as solutions. To do this, in the solution evaluation phase (at each generation) we discard all those solutions that violate this constraint. This way, only admissible solutions will be continuously maintained and optimized.

Chapter 6

Experiments and Results

In this chapter, we present and analyze the obtained results with the aim of providing a foundation for future research and studies in the development of explanation techniques and to assess their reliability.
Three main aspects define the basis of the analysis:

- **the network output should remain approximately constant**;
- **the explanation on the perturbed image must diverge from the original one**;
- **the difference between the original image and the perturbed image must be minimal**.

We now begin by reviewing the key results of our experiments, illustrating our findings and the emerging challenges in this topic.

6.1 Tests Description

The first tests were conducted with the aim of determining the best hyperparameters for the attack algorithm and the best combination of objective functions to compose the fitness. A subset of 30 images from the original 200 images dataset was used to optimize the parameters. Below there are the results obtained from the first tests and the choices made for each hyperparameters.

As described in Section 5.4, the *number of epochs*, the *population size* and the *threshold value* (when necessary), have already been properly determined. Concerns remain about the number of filters and the best combination of goal functions for the fitness.

6.2 Hyperparameters Tuning

6.2.1 Number of Filters

The initial tests conducted aim to determine the optimal number of filters achieving the best performance. In these tests we decided to use the SSIM function for both the objectives: $SSIM_{heatmap}$ and $SSIM_{image}$, the first to evaluate the effectiveness of the attack and the second to mitigate the impact of the perturbation. The results followed our expectations; in fact, as you can see on Table 6.1, the higher the number of filters, the lower the $SSIM_{heatmap}$ will be, but the higher the $SSIM_{image}$. From the results we can assume that the best number of filters in terms of global performance is 5.
With 5 filters it is possible to create complex effects that are, at the same time, soft and difficult to perceive by the human eye. The performances obtained in terms of attack detectability are slightly worse than the ones with 3 filters (approximately between 4% and 6%), but the effectiveness of the attack is higher (approximately between 8% and 12%). Using 6 filters instead we have again a decrease for $SSIM_{image}$ (approximately between 1% and 6%) but not so high to justify the decrease for $SSIM_{image}$ (approximately between 5% and 6%). From here, the next test run always with **5 filters**.

Method	No. of Filters	$SSIM_{heatmap}$ (lower is better)	$SSIM_{image}$ (higher is better)
Grad-CAM	3	0.7415	0.8904
Grad-CAM	5	0.6808 | +8.18%	0.8539 | -4.10%
Grad-CAM	6	0.6354 | +6.66%	0.8027| -6.00%
Eigen-CAM	3	0.7143	0.9040
Eigen-CAM	5	0.6278 | +12.10%	0.8482| -6.17%
Eigen-CAM	6	0.5766 | +8.15%	0.8049| -5.10%
Ablation-CAM	3	0.7038	0.8978
Ablation-CAM	5	0.6263 | +11.01%	0.8547| -4.80%
Ablation-CAM	6	0.6197 | +1.05%	0.8135| -4.82%

Table 6.1: Performance comparison using different number of filters across all methods. Each row presents the value of the objective functions and the percentage difference compared to the previous one. The option with 3 filters acts as baseline.

Figure 6.1: Comparison images to analyze the differences on the number of filters used. Starting from the left, the first column represents the original image, the following columns show the images produced with 3, 5 and 6 filters (in this order).

AOPC drawdown	Effectiveness Function	Detectability function
-9.62%	IoU	$SSIM_{image}$
-16.78%	Center Distance	$SSIM_{image}$
-18.56%	$SSIM_{heatmap}$	$SSIM_{image}$

Table 6.2: Average AOPC drawdown comparison.

6.2.2 Fitness function

To identify the optimal combination of objective functions , we conducted a performance comparison, focusing only on the fastest method, Eigen-CAM. This decision was made to accelerate the testing process.
Since the objective functions operate in different ranges, we employed the AOPC metric to determine the best combination. Essentially, the combination that results in a more significant reduction in the XAI method's performance, according with AOPC, would be considered the most suitable. As demonstrated in Table 6.2, the best performance is achieved by combining two SSIMs, followed closely by the Center Distance measurement.
The use of IoU yields significantly worse results, although the objective function values of the attack effectiveness seem to be better (**0.62 for IoU against 0,75 for Center Distance**). Hence, we decided to investigate the reasons for this contraddiction and found that **this metric is not suitable for our case**, since the metric tends to drive optimization toward enlarging the bounding box rather than relocating it. This leads to a reduction in the IoU value (which may appear favorable to the algorithm optimization phase) but doesn't align with our actual objective. To emphasize a significant distinction between the two explanations, it is logical to position the two bounding boxes 'apart' from each other. Enlarging one of them to decrease the IoU value doesn't serve this purpose. This also explains why we achieve significantly improved results when using Center Distance and SSIM.
This demonstrates how the choice and definition of appropriate objective functions is a crucial factor in evolutionary and multi-objective algorithms.
In Figure 6.2 and in Figure 6.3 you can see some examples of how the algorithm fails to perform a good attack: the bounding boxes are exactly in the same position (both for IoU and Center Distance). However, analyzing only the numerical values of the two metrics, IoU would seem to perform better (0.72 against a 0.95 of the Center

Distance reported in the same range). However, this is not apparent by observing the image, in fact, the numerical improvement is only due to the fact that the bounding box of the perturbed image is larger.

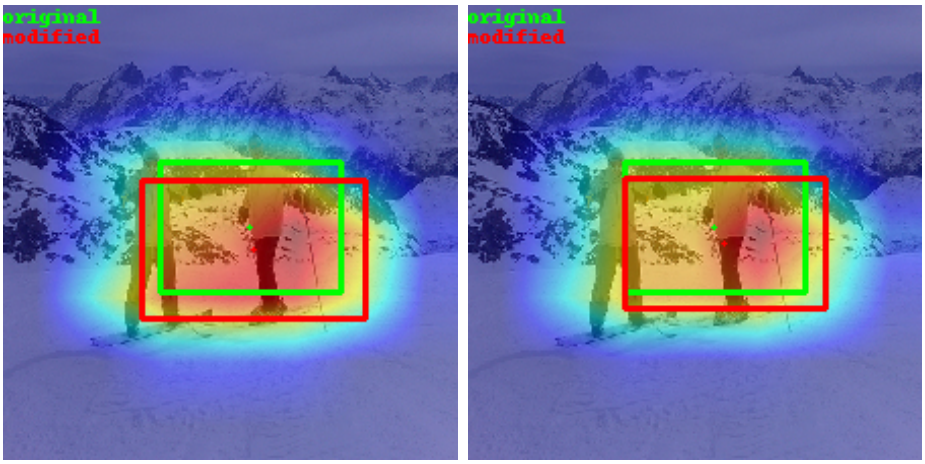

Figure 6.2: Comparative example of the behavior of the attack algorithm driven by two different objective functions: IoU and Center Distance. The **first image** shows the result of the bounding boxes produced by the algorithm with **IoU** objective function in the fitness. In the **second image** those with **Center Distance**.

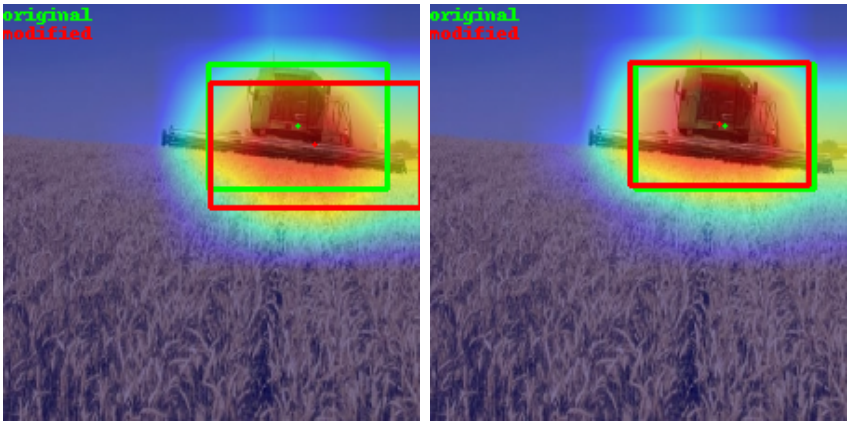

Figure 6.3: Comparative example of the behavior of the attack algorithm driven by two different objective functions: IoU and Center Distance. The **first image** shows the result of the bounding boxes produced by the algorithm with **IoU** objective function in the fitness. In the **second image** those with **Center Distance**.

Further confirmation of our suppositions can also be seen in the examples shown in Figure 6.4. It is possible to observe how this behavior persists even when the attack

seems to perform better. The examples provided illustrate that the IoU metric fails to create two conceptually different explanations, as the two bounding boxes always remain very close. We can assume that the algorithm finds easier to minimize the IoU by chosing filters that enlarge the bounding box. However, this does not necessarily result in a significant difference between the two explanations, as they tend to remain quite similar. In contrast, the Center Distance metric better captures this logic and allows for the creation of significantly different explanations. You can see how the bounding boxes are far from each other.

While it's true that the behavior behind bounding boxes is quite similar between the IoU and Center Distance metrics, their difference lies in what is being optimized. For our purpose, using Center Distance is much more accurate. From this point, we will only compare fitness composed of $\boldsymbol{CD}$ & $\boldsymbol{SSIM}_{image}$ to fitness consisting of $\boldsymbol{SSIM}_{heatmap}$ & $\boldsymbol{SSIM}_{image}$.

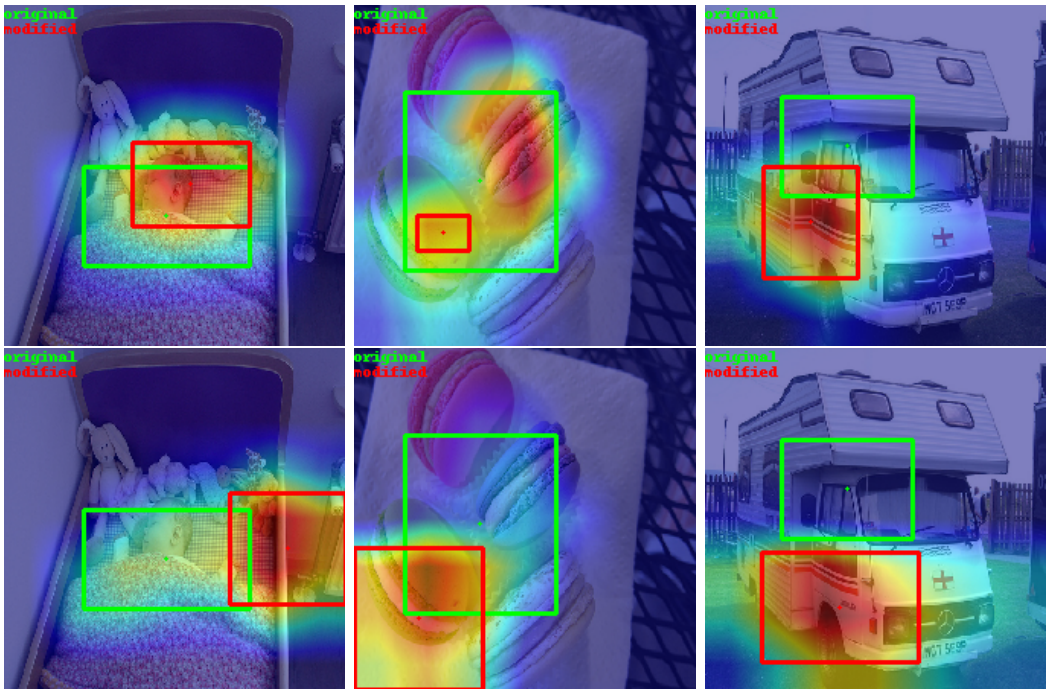

Figure 6.4: Comparative example of the behavior of the attack algorithm driven by two different objective functions: IoU and Center Distance. The **first row** shows the results of the bounding boxes produced by the algorithm with the **IoU** objective function in the in fitness. In the **second row** those with **Center Distance**.

Method	Center Distance	$SSIM_{image}$
Eigen-CAM	0.8291	0.1199
Grad-CAM	0.8536	0.1243
Ablation-CAM	0.8340	0.1387

Table 6.3: Performance using a fitness composed by Center Distance and SSIM.

6.2.3 XAI Methods comparison

As evident from the results presented in Table 6.1, the algorithm performs similarly across all three methods when utilizing a fitness composed by the two SSIM functions. The performances obtained using the fitness consisting of CD & $SSIM_{image}$ are shown in Table 6.3. This leads us to conclude that our algorithm performs nearly identically (with slightly lower performance on Grad-CAM) across all three tested explanation methods.
However, in addition to the performance achieved, it is crucial to consider the time required to generate the attacks. The tests were conducted using two ***Nvidia 3090 GPUs and a 13th-generation Intel Core I7 processor.*** The timing for producing each test on 30 images to determine the hyperparameters for each method was as follows:

- *200 minutes* for Eigen-CAM
- *220 minutes* for Grad-CAM
- *360 minutes* for Ablation-CAM

Given the performance and computational costs, we decided to extend the tests to the entire dataset consisting of 200 images, initially only to Eigen-CAM and Ablation-CAM.

6.3 Full Tests

This section shows the tests performed on all the 200 images of our dataset. Here is a summary of the hyperparameters used:

- **Number of filters**: 5

- **Fitness function**: *CD* & $SSIM_{image}$; $SSIM_{heatmap}$ & $SSIM_{image}$
- **Epoch**: 10
- **Population Size**: 10
- **Threshold for bounding box building**: 170

Table 6.4 shows the results obtained on the entire dataset with the hyperparameters mentioned above. The results are presented in terms of the average drawdowns for the AOPC and the average values of the two fitness components (remembering that this is a minimization problem).

Method	Fitness	AOPC Drawdown	Obj func 1	Obj func 2
Eigen-CAM	*SSIM + SSIM*	-17.19%	0.6079	0.1523
Ablation-CAM	*SSIM + SSIM*	-20.18%	0.5726	0.1751
Eigen-CAM	*CD + SSIM*	-10.05%	0.82913	0.1199
Ablation-CAM	*CD + SSIM*	-14.65%	0.8034	0.1048

Table 6.4: Adversarial Algorithm Results (Averages Over 200 Images)

As expected, the performances are in line with those of the first tests on the dataset subset. The Eigen-CAM method, using the fitness combination of $SSIM_{heatmap}$ and $SSIM_{image}$, exhibited a noticeable AOPC drawdown of -17.19%. This indicates a significant impact on the XAI method's performance. Also for Ablation-CAM, employing the fitness combination of $SSIM_{heatmap}$ and $SSIM_{image}$, displayed a more substantial AOPC drawdown of -20.18%, than the combination of *CD* (Center Distance) and $SSIM_{image}$ (-14.65%). When using the fitness combination *CD* and $SSIM_{image}$, Eigen-CAM achieved an AOPC drawdown of -10.05%. This combination performed relatively worse compared to $SSIM_{heatmap}$ and $SSIM_{image}$. These results provide insights into how different fitness functions and methods impact the performance of the adversarial algorithm.
It is important to note that our algorithm works regardless of the XAI method used. However, it is evident that the choice of the fitness function, as well as the specific XAI method used, can lead to different levels of performance in the explanations

generated by the XAI model.
In this direction, we can conclude that Ablation-CAM seems to be less robust to our attack method and the use of SSIM also to evaluate the quality of the attacks seems to be the best choice.

6.4 Results Analysis

In this section we comment and analyze the results obtained. Starting from an analysis of the results expressed in terms of AOPC, we proceed with an in-depth analysis of the fitness optimization processes. Finally, will be shown some typical examples of the attacks generated by our algorithm.

6.4.1 AOPC

In terms of AOPC, the better our algorithm performs, the higher the drawdown.
The following figures present notable examples of the results obtained. In Figure 6.5 and Figure 6.6, we observe two scenarios where the AOPC of the explainability method is significantly reduced. This doesn't mean directly that the attack was really successfull, because we have to consider how many steps the algorithm needed to reach a significant drawdown (1 is the maximum) and how sharp the drawdown is. In fact, the more forward with the iterations the curve reaches the maximum point, the more effective the attack is. This means that the algorithm needs more steps to find the really relevant parts (blocks) of the heatmap, i.e. these blocks received a low importance value by the explanation method (remembering that the blocks are perturbed in decreasing order of importance). In Figure 6.5 this happens slowly after the 20th iterations, while in Figure 6.6 this happens near the end. The idea behind this curve is that if it quickly reaches its maximum, it implies that the areas most crucial for classification were immediately disturbed.
We expect this occurring for the original explanations, while we aim to delay this effect as much as possible for the explanations on the perturbed images.
In Figure 6.7 we can observe an example where our algorithm had absolutely no effect. In this instance, perturbing the initial regions of the image caused the model's output for the designated class **to drop to 0 immediately**. This, by observing the

formula in Section 2.6.1, translates to a value equal to $f(x_M^{(0)})$, bringing the curve to the maximum level. This suggests that the explainability method was highly accurate also in the perturbed image, which is contrary to our intended outcome.

To better understand the performance variations due to an attack produced by our algorithm, see Figure 6.8. In this graph two curves are compared, one representing the AOPC obtained on the explanation of the original image (blue line) and one the same value computed for the perturbed image (orange line). The perturbed image is obtained using the combination of CD and $SSIM_{image}$ in the fitness function. As you can see, the AOPC drawdown for the perturbed image (orange graph) is much lower than the original.

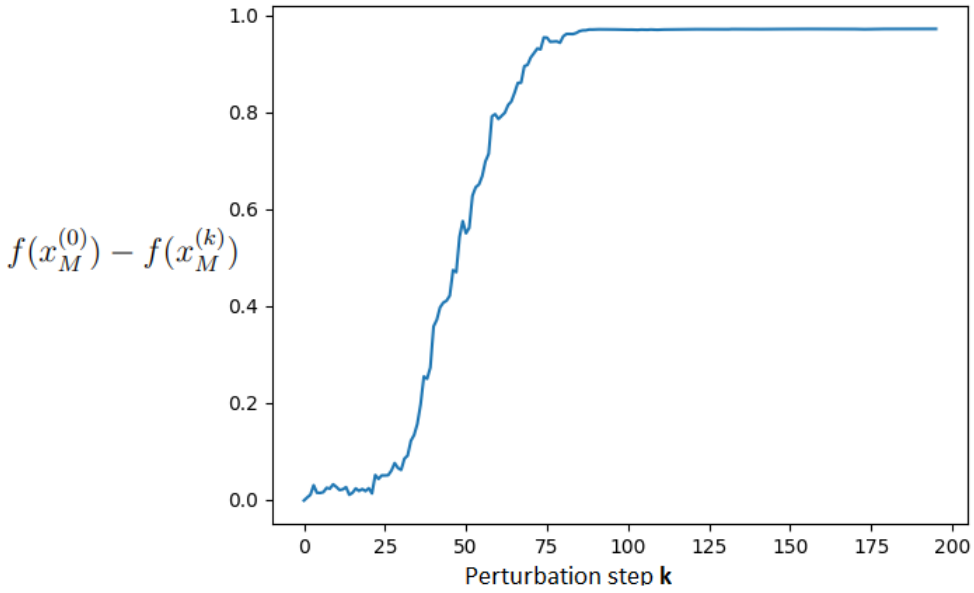

Figure 6.5: Example of AOPC with moderate convergence.

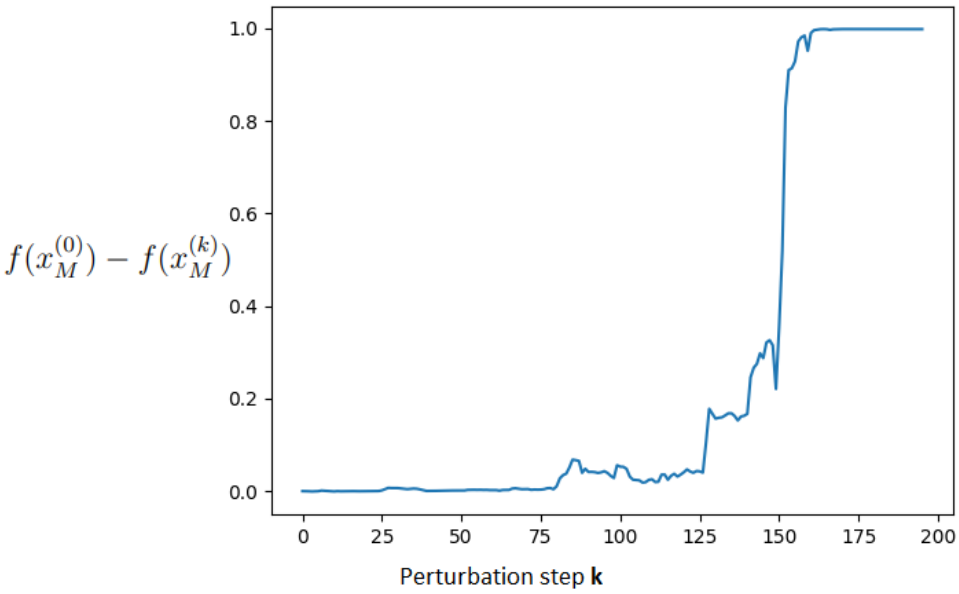

Figure 6.6: Example of AOPC with slow convergence (this means the attack was successful).

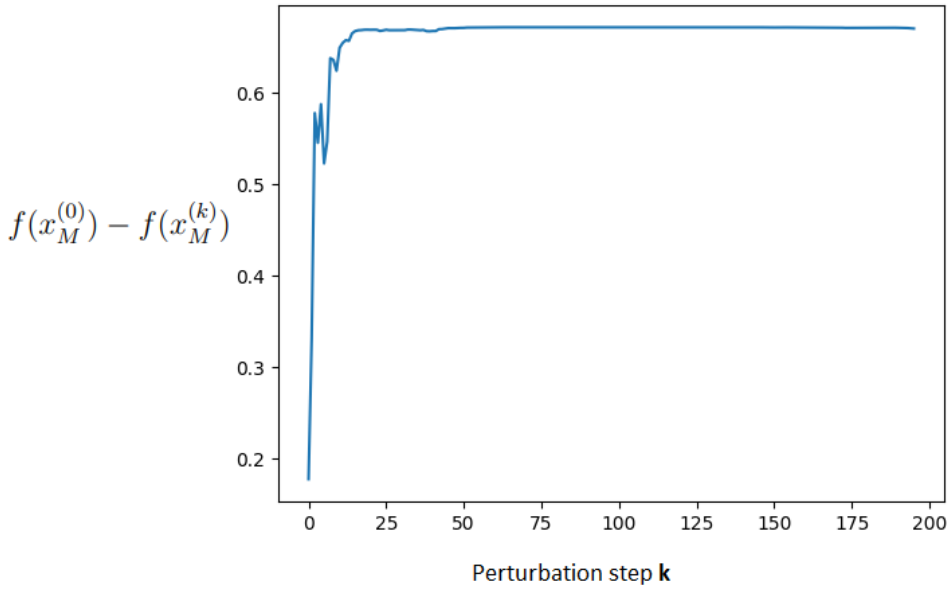

Figure 6.7: Example of AOPC with immediate convergence (the attack was not successful).

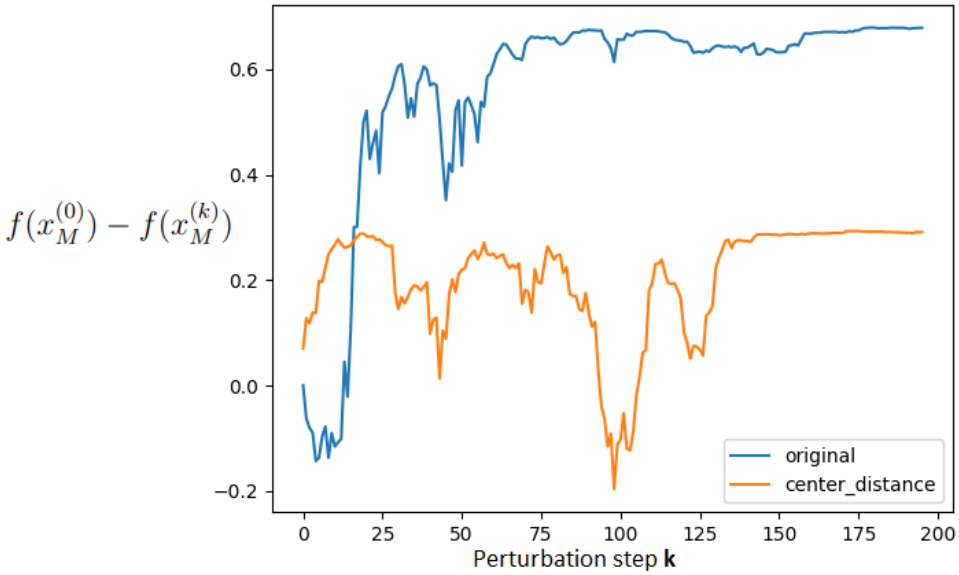

Figure 6.8: Example of comparison between the AOPC of an explanation on the original image (blue line) and on a perturbed image using Center Distance and SSIM (orange line).

6.4.2 Optimization Process

In addition to evaluating the overall effectiveness of our evolutionary algorithm in generating adversarial attacks, it's crucial to delve deeper into the optimization process. By examining how the algorithm refines its strategies across successive generations, we acquire valuable insights into its capacity to improve the quality of the attack.
To gain a comprehensive understanding of this process, we closely examine the evolutionary trajectory of the fitness. This involves tracking the dynamic changes in the fitness function's individual components across multiple generations.
In Figures 6.9 and 6.10, we present visual representations of the convergence curve for the two key fitness functions employed in our experiments. Specifically, we analyze the performance concerning the SSIM metric, which is crucial for gauging the quality of the generated adversarial images.

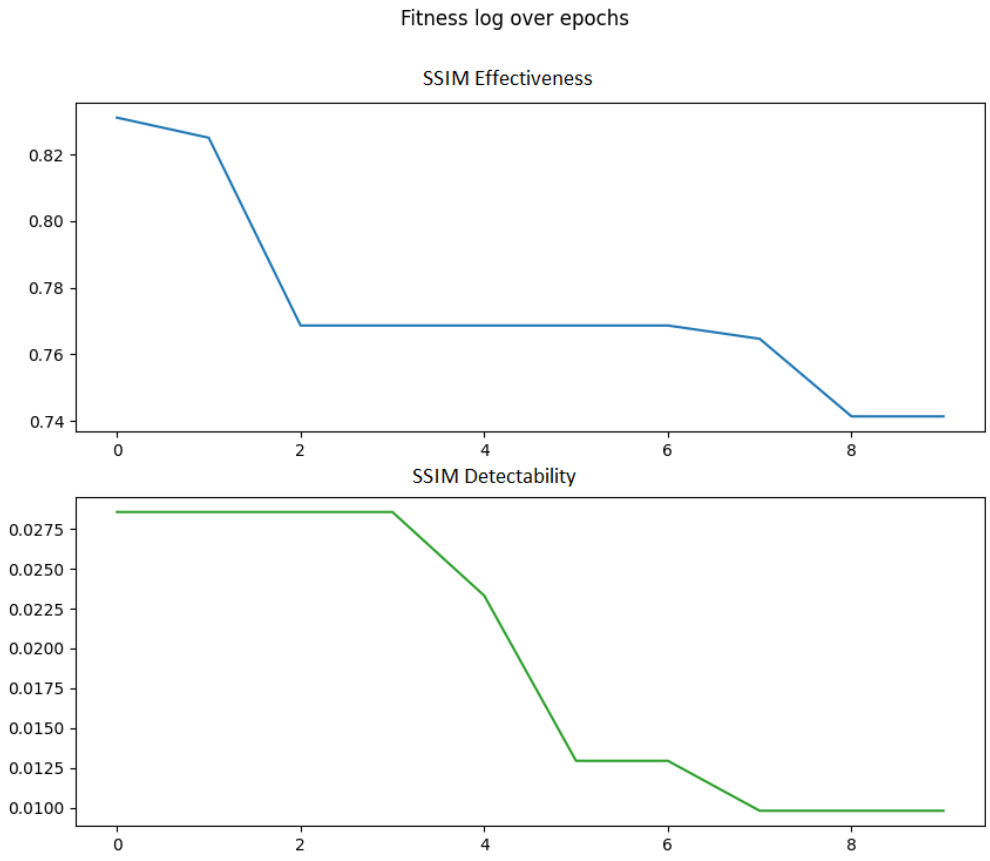

Figure 6.9: Example of how fitness components are optimized across generations. In the upper plot there is the $\text{SSIM}_{heatmap}$ to measure the attack effectiveness, in the lower one the SSIM_{image} to measure the attack detectability.

In Figure 6.9, we notice an successful pattern where the algorithm progressively enhances the attack quality across generations. This outcome underscores the algorithm's ability to fine-tune its strategies and produce more powerful adversarial images. Nevertheless, it's important to recognize that not all scenarios follow a similarly smooth trajectory. In some instances, the algorithm encounters challenges in improving the fitness of an image, as depicted in Figure 6.10. In this case, despite an initial improvement in the objective function for detectability, the algorithm eventually reaches a plateau, encountering challenges in further enhancing its performance. These graphical representations of the optimization process offer a view of how our evolutionary algorithm adapts and evolves over time. Such insights are valuable for

understanding the algorithm's strengths and limitations, contributing to a more comprehensive assessment of its effectiveness.

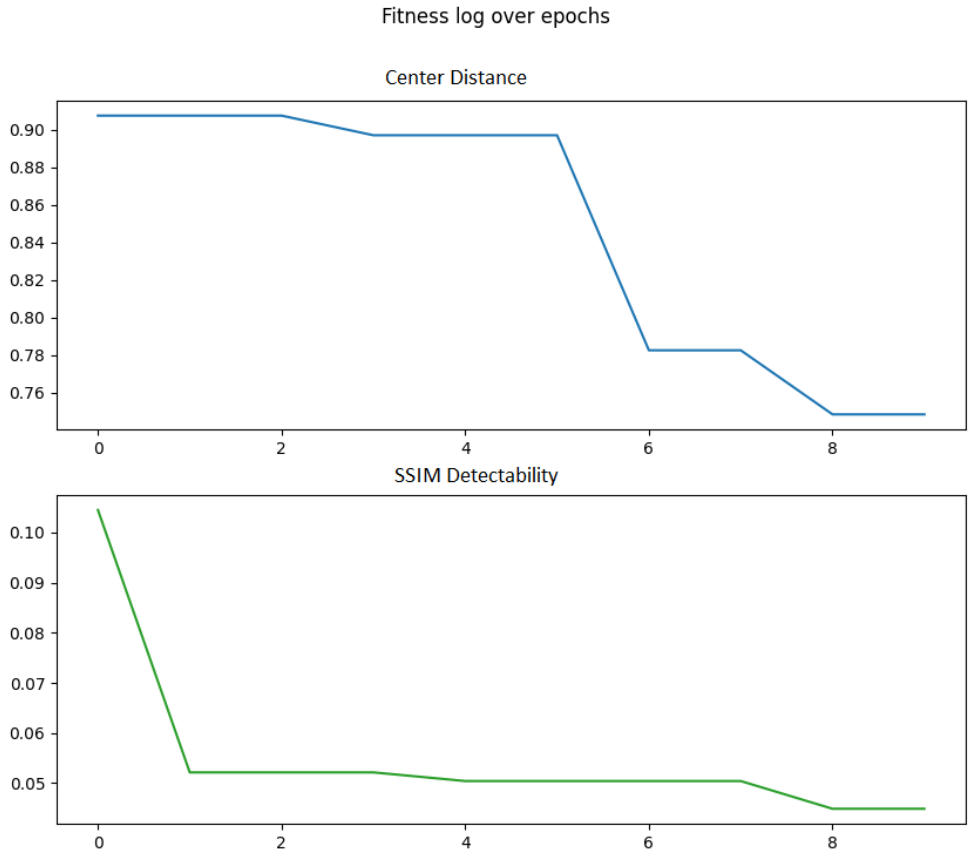

Figure 6.10: Example of how fitness components are optimized across generations. In the upper plot there is the Center Distance to measure the attack effectiveness, in the lower one the SSIM to measure the attack detectability.

6.4.3 Adversarial Examples

In this section, we present a variety of attack examples generated by our evolutionary algorithm, covering both successful and unsuccessful cases.
We will specifically explore four typical scenarios resulting from our attack algorithm:

1. A highly effective attack that remains imperceptible to the human eye.

2. An attack that, while effective, is perceptible to the point of rendering it ineffective.

3. An effective attack where the perturbation is visible but unlikely to be associated with an adversarial attack.

4. An unsuccessful attack.

In Figure 6.11, we illustrate the first scenario mentioned earlier. The starfish (the primary class label in the classification model's output) is initially correctly identified by the explainability method. However, after applying a perturbation generated by our algorithm, which is imperceptible to the human eye, the resulting explanation becomes entirely different. This serves as a classic example of the type of attack we aim to achieve.
However, achieving such ideal scenarios is not always feasible. In cases like Figure 6.12, even though the algorithm has worked successfully, resulting in a radically different explanation, the applied perturbation is so significant that it renders the attack ineffective. The image is nearly unrecognizable due to the obvious perturbation.
Furthermore, in Figure 6.13, we demonstrate an example of an effective attack where the two explanations differ. However, in this case, the perturbation is visible, yet the image remains quite similar to the original. In such a situation, it would be challenging to associate the application of the perturbation with an adversarial attack. This peculiarity is the reason why we chose to generate perturbations using combinations of filters typically associated with image enhancement, similar to those found in Instagram. In fact, here, the perturbation appears to be applied to improve the visual quality of the image, rather than indicating adversarial intent.
The final scenario, depicted in Figure 6.14, showcases an example in which the explanations for the original and perturbed images are identical, signifying a complete failure of the attack.
These diverse scenarios highlight the complexities of crafting effective adversarial attacks and underscore the importance of considering perceptibility when evaluating their success.

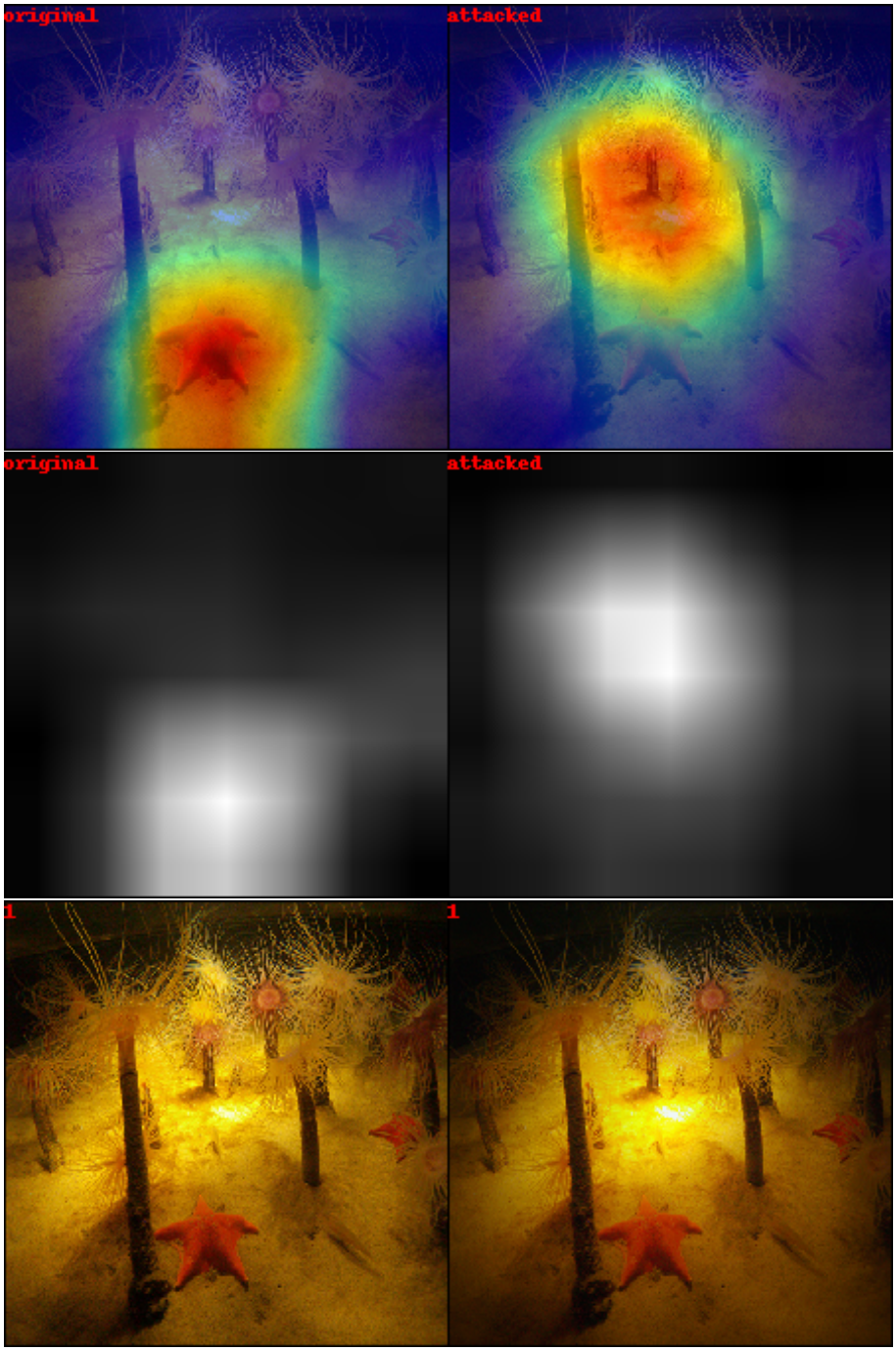

Figure 6.11: Example of an effective adversarial attack that is difficult for the human eye to perceive.

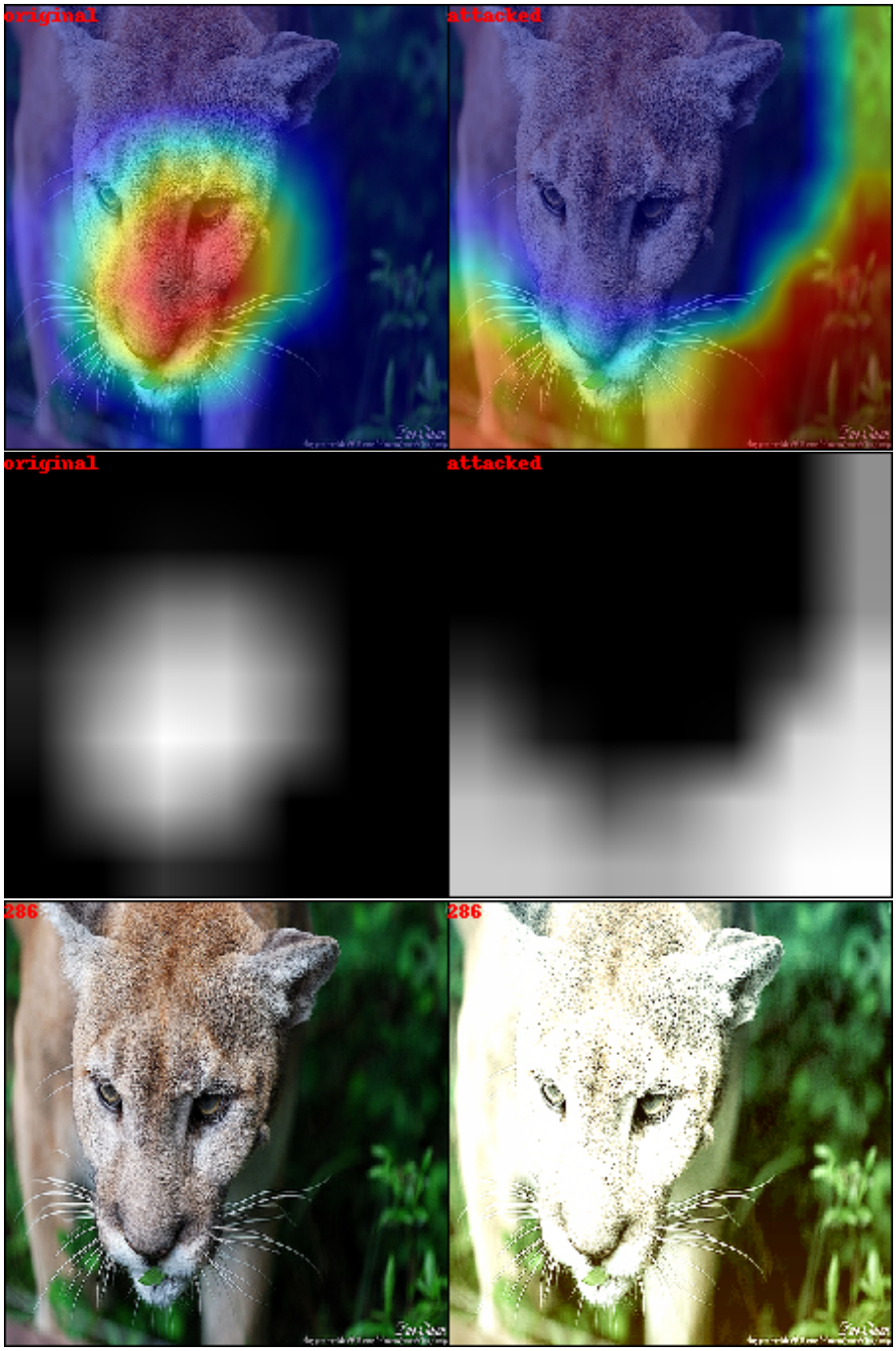

Figure 6.12: Example of an effective adversarial attack but easily detectable to the human eye.

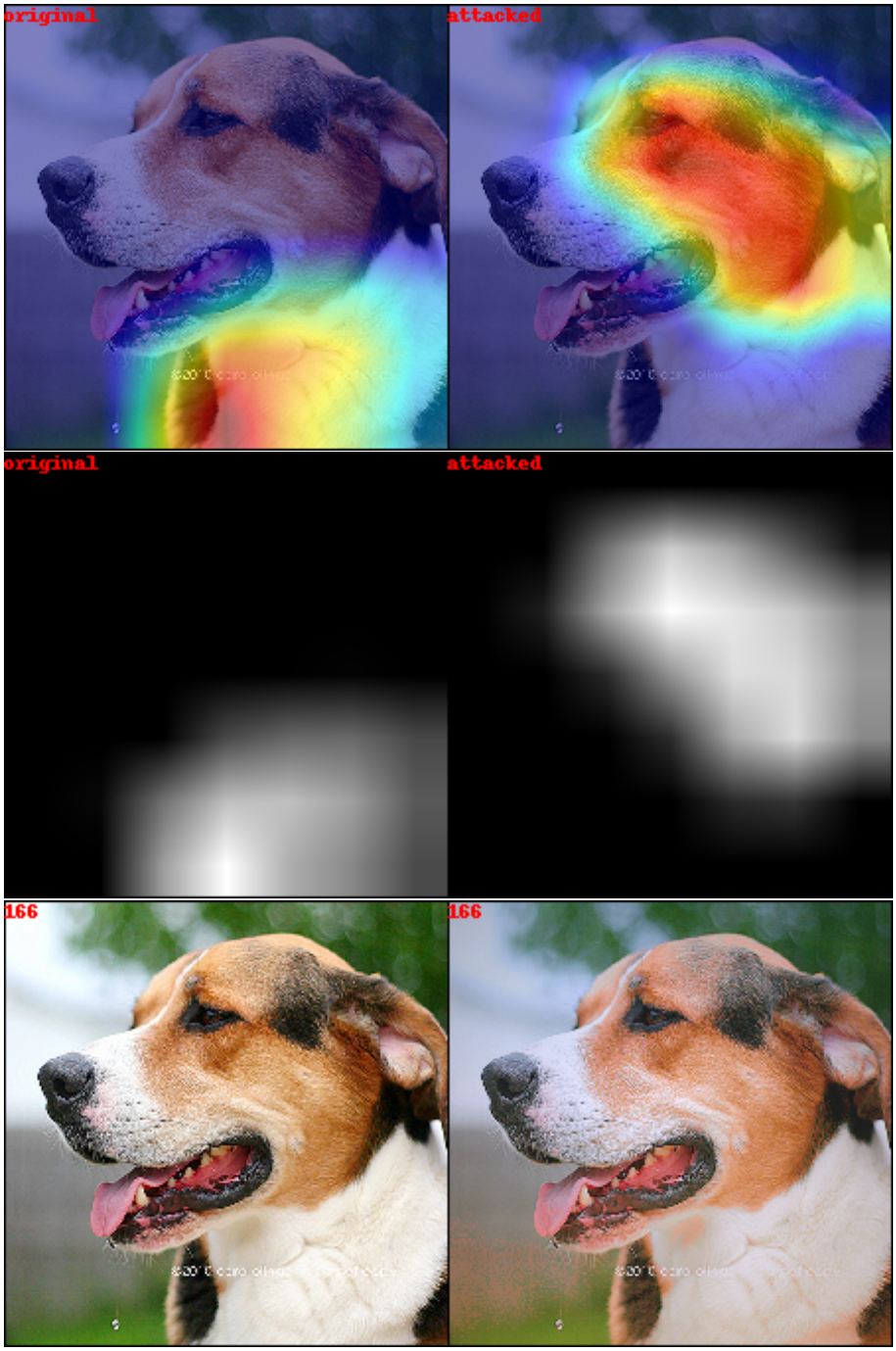

Figure 6.13: Example of an effective adversarial attack that is easily detectable by the human eye but can be exchanged with the application of image enhancement filters.

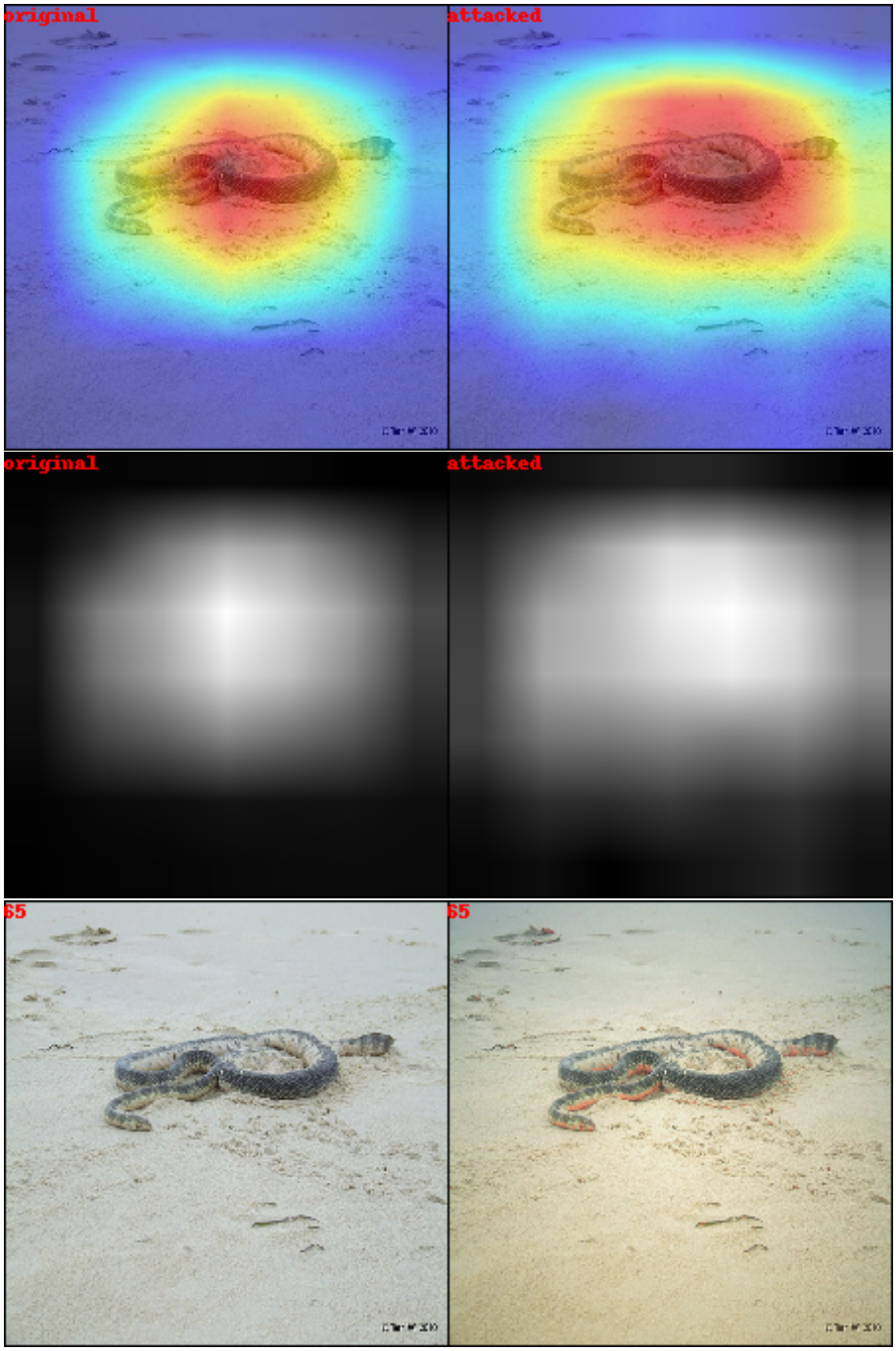

Figure 6.14: Example of an ineffective adversarial attack.

Conclusion and Future Work

In conclusion, this work has delved deeply into the domains of Adversarial Machine Learning and Explainable Artificial Intelligence. Through a comprehensive analysis of attack scenarios and their perceptibility, we have gained valuable insights into the complexities and challenges in this field. Our findings contribute to a better understanding of the interplay between model explainability and adversarial robustness, paving the way for further research and advancements in this critical area of study.

The central focus of this work has been the development of a multi-objective evolutionary algorithm capable of generating imperceptible adversarial examples. These examples were crafted to challenge the reliability of various XAI methods. This endeavor has underscored the complexity of crafting adversarial perturbations that remain invisible to the human eye while significantly altering the explanation produced by the XAI method. Through our algorithm, we have successfully optimized these perturbations, yielding remarkable outcomes. Additionally, following the black-box approach, we've demonstrated the feasibility of constructing an algorithm capable of interfacing with diverse XAI methods.

Our tests have effectively validated the foundational assumptions of our research problem. This validation aligns with the findings of other researchers across various studies (as detailed in Section 3.2). These findings collectively underscore that XAI methods can face challenges related to their faithfulness, ultimately compromising their reliability. This realization further emphasizes the necessity for continued research and deeper insights into this domain.

Furthermore, we have carefully examined how human perception plays a crucial role in the effectiveness of adversarial attacks. Our research has demonstrated that, even when an attack successfully alters the XAI method output, if the perturbation is visible or raises suspicion, its practical effectiveness in real-world scenarios may be

compromised.
This thesis project has led to the creation of an evolutionary adversarial attack algorithm that takes these challenges into account and aims to generate more sophisticated and imperceptible attacks. However, many open research opportunities remain in this ever-evolving field. Improving human perception evaluation metrics and analyzing vulnerabilities in explainability algorithms are just a few of the future directions that warrant further investigation.

Nonetheless, further research and in-depth investigations are indispensable. In addition to testing various XAI methods, it is imperative to enhance the performance of our algorithm. There are multiple viable approaches to achieve this objective. One noteworthy possibility arises from scrutinizing the limitations of ineffective adversarial examples. Our current attack algorithm applies perturbations (and consequently, filters) globally to the entire image, resulting in the same effect being applied to each pixel. Therefore, it may be worthwhile to explore the development of a variant of our algorithm capable of executing this process locally at the pixel level, optimizing the perturbation for each pixel individually. Such an approach could potentially lead to improved performance and enhance the effectiveness of the attack.
This work makes a valuable contribution to the ongoing discourse on the necessity of transparency in the field of Deep Learning and the urgency of developing more robust and resilient explainability methods in the face of adversarial attacks. As neural networks continue to find applications in critical domains, comprehending and defending against such attacks assume ever-greater significance in ensuring the safety and reliability of AI-driven systems.

The algorithm for generating attacks and the evaluation algorithm for AOPC are available in the following Github repositories: AGV-Attack, AOPC_MoRF.

Bibliography

[1] An introduction to explainable AI with Shapley values. An introduction to explainable ai with shapley values, 08 2023. URL `https://shap.readthedocs.io/en/latest/example_notebooks/overviews/An%20introduction%20to%20explainable%20AI%20with%20Shapley%20values.html`.

[2] Christopher Anders, Plamen Pasliev, Ann-Kathrin Dombrowski, Klaus-Robert Müller, and Pan Kessel. Fairwashing explanations with off-manifold detergent. In *International Conference on Machine Learning*, pages 314–323. PMLR, 2020.

[3] Diego Arcelli, Alina Elena Baia, Alfredo Milani, and Valentina Poggioni. Enhance while protecting: Privacy preserving image filtering. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 647–652, 2021.

[4] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020.

[5] AEB Baia, Alfredo Milani, and Valentina Poggioni. Combining attack success rate and detection rate for effective universal adversarial attacks. In *Proceedings of the ESANN*, 2021.

[6] Alina Elena Baia, Gabriele Di Bari, and Valentina Poggioni. Effective universal unrestricted adversarial attacks using a moe approach. In *Applications of Evolutionary Computation: 24th International Conference, EvoApplications 2021,*

Held as Part of EvoStar 2021, Virtual Event, April 7–9, 2021, Proceedings 24, pages 552–567. Springer, 2021.

[7] Alina Elena Baia, Giulio Biondi, Valentina Franzoni, Alfredo Milani, and Valentina Poggioni. Lie to me: shield your emotions from prying software. *Sensors*, 22(3):967, 2022.

[8] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[9] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13*, pages 387–402. Springer, 2013.

[10] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, 2017. doi: 10.1109/SP.2017.49.

[11] Davide Castelvecchi. Can we open the black box of ai? *Nature News*, 538(7623): 20, 2016.

[12] Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can be manipulated and geometry is to blame. *Advances in neural information processing systems*, 32, 2019.

[13] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3681–3688, 2019.

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

[15] 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.

[16] Juyeon Heo, Sunghwan Joo, and Taesup Moon. Fooling neural network interpretations via adversarial model manipulation. *Advances in Neural Information Processing Systems*, 32, 2019.

[17] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013.

[18] Image Classification on ImageNet. Image classification on imagenet, 08 2023. URL `https://paperswithcode.com/sota/image-classification-on-imagenet`.

[19] ImageNet. Imagenet, 08 2023. URL `https://www.image-net.org/`.

[20] Keras Applications. Keras applications, 08 2023. URL `https://keras.io/api/applications/`.

[21] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. *Explainable AI: Interpreting, explaining and visualizing deep learning*, pages 267–280, 2019.

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

[23] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018.

[24] Xiao-Hui Li, Yuhan Shi, Haoyang Li, Wei Bai, Yuanwei Song, Caleb Chen Cao, and Lei Chen. Quantitative evaluations on saliency methods: An experimental study. *arXiv preprint arXiv:2012.15616*, 2020.

[25] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[26] Aravindh Mahendran and Andrea Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision*, 120:233–255, 2016.

[27] Mohammed Bany Muhammad and Mohammed Yeasin. Eigen-cam: Class activation map using principal components. In *2020 international joint conference on neural networks (IJCNN)*, pages 1–7. IEEE, 2020.

[28] Chao-Ying Joanne Peng, Kuk Lida Lee, and Gary M Ingersoll. An introduction to logistic regression analysis and reporting. *The journal of educational research*, 96(1):3–14, 2002.

[29] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018.

[30] Harish Guruprasad Ramaswamy et al. Ablation-cam: Visual explanations for deep convolutional network via gradient-free localization. In *proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 983–991, 2020.

[31] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[32] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[33] Wojciech Samek and Klaus-Robert Müller. Towards explainable artificial intelligence. *Explainable AI: interpreting, explaining and visualizing deep learning*, pages 5–22, 2019.

[34] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural

network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2016.

[35] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[36] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[37] 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.

[38] 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.

[39] Erico Tjoa and Cuntai Guan. A survey on explainable artificial intelligence (xai): Toward medical xai. *IEEE transactions on neural networks and learning systems*, 32(11):4793–4813, 2020.

[40] Richard Tomsett, Dan Harborne, Supriyo Chakraborty, Prudhvi Gurram, and Alun Preece. Sanity checks for saliency metrics. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 6021–6029, 2020.

[41] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[42] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014.

[43] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

Acknowledgements

Vorrei dedicare questo spazio per esprimere la mia sincera gratitudine alle persone straordinarie che hanno reso possibile questo traguardo.

Innanzitutto, ai miei cari genitori, che mi hanno sempre sostenuto con amore, fiducia e sacrificio. Il vostro incoraggiamento è stato il mio faro attraverso le sfide e le vittorie di questo percorso.

A Federica, voglio dire grazie per essere la mia fonte di ispirazione, conforto e gioia. La tua costante presenza è per me di grande valore.

Alla mia relatrice, la Prof.ssa Valentina Poggioni, desidero esprimere la mia profonda riconoscenza per la sua guida esperta, la sua pazienza e la sua dedizione nell'aiutarmi a crescere come studente e come persona.

Un ringraziamento speciale va anche a Fabrizio ed ai miei altri colleghi di università. Le nostre discussioni stimolanti, le sessioni di studio condivise e il sostegno reciproco hanno reso questo percorso accademico ancora più significativo.

Infine, a tutti coloro che, in modi diversi, hanno contribuito al mio percorso di apprendimento, sia familiari che amici, grazie per essere stati al mio fianco in questo viaggio. Questo traguardo non sarebbe stato possibile senza di voi.

Grazie di cuore a tutti.

Cristian