eman ta zabal zazu

Universidad del País Vasco　Euskal Herriko Unibertsitatea

Konputazio Zientzia eta Adimen Artifizialaren Saila

Departamento de Ciencia de la Computación e Inteligencia Artificial

Department of Computer Science and Artificial Intelligence

Broadening the Horizon of Adversarial Attacks in Deep Learning

by

Jon Vadillo

Supervised by Roberto Santana and Jose A. Lozano

Donostia - San Sebastián, November 2022

"Machines take me by surprise with great frequency."
— Alan Turing

Acknowledgement

These first lines cannot be addressed to anyone other than my advisors, Roberto Santana and Jose A. Lozano, whose invaluable support, trust and guidance during all these years have been essential for the development of this thesis. I hope that we can keep working together in the future. Thank you very much.

I also owe a debt of gratitude to Professor Marta Kwiatkowska for allowing me to complete a research stay as a visitor in her research group at the University of Oxford. This gratitude is also extended to the members of the QAVAS research group, the Computer Science Department and Reuben College, specially to Aleks, Artem, Benjie, Emanuele, Pascale, Pian and Xiyue, for making my stay a really enjoyable and productive one.

Special thanks go also to my friends in the Intelligent Systems Group, and to an endless list of members in the Faculty of Computer Science, whose company and help during this stage has been an important part of it. In addition, I would also like to acknowledge the great work of John Kennedy in proofreading our work, which has contributed greatly to the readability of this dissertation.

I am grateful to the Spanish Ministry of Science, Innovation and Universities for financially supporting this project (through the FPU19/03231 grant). In addition, this work has been partially supported by the Basque Government (through the ELKARTEK program) and by the Spanish Ministry of Economy and Competitiveness MINECO (project PID2019-104966GB-I00).

Last but not least, I would like to thank the endless support of my family and friends. This thesis is also for them.

Contents

Symbols

General nomenclature

X	Input space
Y	Output space
x	Input
y	Output
d	Input dimensionality
k	Number of classes
$\mathcal{P}(X)$	Input data distribution
$\mathcal{P}(Y)$	Probability distribution for the output classes
$\mathcal{P}(X, Y)$	Joint data distribution
$f(\cdot)_i$	Softmax value corresponding to the i-th class
$\hat{f}(\cdot)_i$	Logit value corresponding to the i-th class
x'	Adversarial example
ϵ	Perturbation budget (i.e., distortion threshold)

Acronyms

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
DNN	Deep Neural Network

FGSM	Fast Gradient Sign Method
PGD	Projected Gradient Descent
DF	DeepFool
C&W	Carlini & Wagner attack
dB	Decibel
MFCC	Mel-Frequency Cepstral Coefficient
SVD	Singular Value Decomposition
i.i.d.	Independent and identically distributed
u.a.r.	Uniformly at random

1 Introduction

The advances in the field of Artificial Intelligence (AI) over the past years have improved the capabilities of intelligent systems to unprecedented levels, given rise to flourishing technologies with an ever-increasing presence in real-world scenarios, such as virtual assistants, self-driving vehicles, or computer-aided diagnosis systems. The outstanding success of these technologies is primarily due, along with the availability of large volumes of data and advances in computing systems, to Deep Learning (DL) models such as Deep Neural Networks (DNNs), which are capable of efficiently learning to perform highly complex tasks from data. Indeed, DNN based approaches are currently the state-of-the-art in the great majority of AI tasks, particularly in those involving unstructured data, including, but not limited to, large scale object recognition, image segmentation, face recognition, automatic speech recognition, natural language processing or advanced control.
At the same time, the success of DNNs is due to several reasons. First, the great expressiveness of these models makes them universal approximators [67], implying that any function or input-output mapping can be approximated by a DNN with an arbitrary degree of accuracy. Moreover, unlike other AI paradigms such as symbolic methods or knowledge-based systems, DNNs are able to improve their predictive performance just from instances of the problem to be solved, without requiring hand-crafted or ad-hoc solutions. For these reasons, DL based paradigms represent a powerful and general framework that can be applied in a large number of complex problems for which finding explicit solutions is unfeasible.
Nevertheless, despite their remarkable performance, these models still face several limitations when it comes to their deployment in real-world scenarios. Indeed, given the high standards required in tasks as critical as autonomous driving, healthcare or criminal justice, the acceptance of these models is not subject only to their accuracy, but also to their trustworthiness and fairness. Unfortunately, DNNs are still far from being fully reliable. One of their main disadvantages lies in the complexity of the architectures and computational processes employed. Although this complexity provides the model with the

necessary computational requirements to address difficult tasks with high effectiveness, it also makes it intractable to analyze the model's inference process in detail. Due to this opacity, DNNs are often considered "black-box models", which gives rise to a number of consequences. First, the impossibility of fully understanding how the model determines a prediction for a given input reduces their reliability in scenarios where the "reasoning" is as important as the final prediction. A representative example of these scenarios is healthcare, where it is vital to understand which are the factors that determine a patient's condition in order to be able to make an appropriate diagnosis and treatment. Apart from that, another negative consequence of the opacity of the prediction process is the difficulty of detecting flaws, biases or vulnerabilities in the models, also impeding the provision of provable guarantees for the correctness or fairness of the predictions.

Furthermore, another important limitation of DNNs is their high vulnerability to *adversarial examples*: inputs purposefully manipulated by an adversary in order to change the output classification of the model and, at the same time, ensure that the modifications are imperceptible to humans [18, 158]. An illustration of such an *adversarial attack* is shown in Figure 2.1. The discovery of adversarial examples revealed a surprisingly high instability in these models, which is in conflict with the remarkable accuracy and robustness observed empirically when tested on *natural* inputs, or on inputs with small yet random corruptions, such as white or Gaussian noise. Consequently, these vulnerabilities put even more into question the reliability of DNNs in critical and real-world tasks, in which adversaries might take advantage of those *flaws* or *security issues.*

The concerning implications of adversarial examples and their intriguing nature has resulted in a prolific research field in the past ten years, particularly in offensive strategies, which has led to uncountable attack methods, that is, procedures that exploit different vulnerabilities of the models in order to generate adversarial examples. This wide attack repertoire allows attacks for multiple goals and scenarios to be generated. For instance, adversarial examples can be designed to force a classification model to produce a specific incorrect class, or, in contrast, to ensure only that the provided class is incorrect (without prioritizing any incorrect class). To achieve any of these goals, most methods generate perturbations which are specifically optimized for each input individually, since this allows simple and highly efficient procedures to be employed. Nevertheless, at the cost of higher computational cost and amount of distortion, it is also possible to produce *universal* perturbations capable of fooling the model in an input-agnostic fashion, allowing, for instance, to deploy attacks in real-time, given that the perturbation is already precomputed. In addition, generally, most of the attack strategies assume a scenario where the adversary can access all the information about the model in order to produce the perturbations, including all its parameters or training details. Even so, adversarial examples can be generated even when the adversary only can observe the output of the model (e.g., the confidence assigned to each possible class),

opening the door for attacks even in highly constrained real-world scenarios. Similarly, some attack approaches aim to increase the risk of adversarial examples in other realistic scenarios, such as creating perturbations that can remain effective when deployed in the physical world [24, 44, 89, 181, 185], or perturbations *transferable* across multiple target models or architectures, which have not been used to generate the perturbations [106, 199].
All these research efforts have, therefore, expanded the notions and the scope of adversarial attacks, and, consequently, the capabilities of the adversaries. However, it is worth stating that, somewhat paradoxically, advances in offensive methods contribute directly to a safer use of DNNs. Indeed, this research is fundamental in order to assess the risks of these technologies as well as the possible threats that an adversary might pose or their implications, which is crucial to promote a more aware and secure use of DL based technologies in practice. At the same time, these discoveries frequently expose previously unknown properties of DNNs, which promotes further studies in the field, ranging from theoretical analyses about the functioning of the models to the development of more robust architectures or learning strategies, contributing, ultimately, to more secure and reliable systems. For all these reasons, the study of novel attack strategies and vulnerabilities continues being an urgent and important research field.

1.1 Objectives

Despite this extensive research and the wide range of attack approaches already proposed in the literature, the possibilities of adversarial attacks have not been fully explored yet.
For instance, the literature on adversarial examples has focused on producing misclassifications for each input isolatedly, without considering the potential of coordinating attacks for multiple inputs. In addition, the definition or utility of adversarial examples is unclear for scenarios in which their *undetectability* can be compromised. This is evident in scenarios in which a human assesses no only the input sample, but also the prediction, or even an explanation computed to justify that prediction, since, in such cases, the inconsistencies are likely to be noticed by the human. Thus, the role of the user is a critical factor to be considered in order to design, evaluate or even define adversarial examples against explainable models. Nevertheless, this factor is often overlooked in the literature, and, therefore, its implications have not been fully formalized or developed yet. Finally, as stated above, the discovery of new vulnerabilities often reveals previously unknown or unexplored properties in DNNs. Studying and characterizing these properties is, therefore, essential to determine the extent of their implications, that is, what additional vulnerabilities they may entail or how these properties could be exploited to improve existing attacks.
For all these reasons, this dissertation will be devoted to explore novel vulnerabilities and more general notions of adversarial examples, which allow

an adversary to carry out novel types of attacks in new domains, scenarios and types of problems as well as achieve malicious objectives that cannot be realized by conventional approaches, thereby broadening the horizon of adversarial examples in DL.

1.2 Outlook of the Dissertation

This dissertation begins by presenting the background and preliminary concepts in Chapter 2, focusing on introducing DNNs, which will be the main focus of the study, as well as the field of adversarial examples. Afterwards, the main contributions of the dissertation will be fully developed in Chapters 3, 4 and 5, summarized as follows.

In Chapter 3, we propose a novel probabilistic framework to generalize and extend adversarial attacks in order to produce a desired probability distribution for the output classes when we apply the attack method to a large number of inputs. This novel "multiple-instance" attack paradigm provides the adversary with greater control over the target model, thereby exposing, in a wide range of scenarios, threats that cannot be conducted by the conventional paradigms.

In Chapter 4, we explore the possibilities and limits of adversarial attacks for explainable models. For this purpose, we first extend the notion of adversarial examples to fit in scenarios in which the inputs, the output classifications and the explanations of the model's decisions are assessed by humans. Next, we study whether (and how) adversarial examples can be generated for explainable models under human assessment, identifying and illustrating the attack strategies that should be adopted in each scenario to successfully deceive the model, and even the human querying the model.

In Chapter 5, we investigate an intriguing phenomenon of universal (i.e., input-agnostic) adversarial perturbations, which has been reported previously in the literature, yet without a proven justification: universal perturbations change the predicted classes for most inputs into one particular (dominant) class, even if this behavior is not specified during the creation of the perturbation. In order to justify the cause of this phenomenon, we propose and experimentally test a number of hypotheses. Our analyses reveal interesting properties of universal perturbations, suggest new methods to generate such attacks and provide an explanation of dominant classes, under both a geometric and a data-feature perspective, contributing to a better understanding of these vulnerabilities.

Finally, Chapter 6 concludes the dissertation by summarizing the work and the main contributions, as well as identifying several future research directions derived from them.

2

Background

2.1 Machine Learning

Machine Learning (ML), as a branch of AI, pursues the goal of generating computational models capable of performing tasks that require intelligent reasoning or cognition. In particular, the main distinguishing feature of ML is that the intelligent behavior is learned from data or examples derived from the problem to be solved. This avoids having to formulate an explicit solution for the problem, which has proven to be cumbersome or unfeasible when it comes to tasks with a high level of abstraction or complexity.

In essence, ML models can be formulated as a parametric function $f_\theta : X \rightarrow Y$ mapping inputs from an input space X to an output space Y based on a set of parameters θ.[1] In this context, *inputs* $x \in X$ represent a characterization of a phenomenon, event, signal or object of interest, and are generally formalized as a d-dimensional attribute-vector $x = [x_1, \ldots, x_d]$, where x_i represents the i-th attribute, $i \in \{1, \ldots, d\}$. On the other hand, the *outputs* in Y represent the possible responses for the inputs.

Two main types of problems can be considered depending on the characteristics of the output space Y, or, in other words, the type of output to be produced given an input. In *classification* problems, which will be the focus of this presentation, Y represents a finite set of categories, often denominated *classes* or *labels*, and the goal of the task is to classify inputs in their corresponding categories. The set of output classes will be denoted as $Y = \{y_1, \ldots, y_k\}$, where y_i represents the i-th class and k the total number of classes. Thus, ML models implemented for classification tasks, often denominated *classifiers*, implicitly partition the input space into classification regions, or, equivalently, define decision boundaries that separate inputs corresponding to different classes. In contrast, in *regression* problems, Y is a

[1] Throughout the dissertation, when the use of a parameter set is implicit, the reference to θ will be omitted from f_θ and only the notation f will be used, for the sake of simplicity.

continuous space (e.g., $Y = \mathbb{R}$), and, thus, regression models aim to capture the function that best fits the relationship or correlations between the input attributes and the output value. Popular examples of regression problems are weather forecasting, stock market analysis or predicting the frequency of an event or phenomenon (e.g., the number of people affected by a disease in a future time interval).

As stated above, the key distinguishing feature of ML is its ability to learn from data. Although this can be accomplished based on different data-driven learning paradigms, this dissertation will be focused on the most relevant one in the literature: *supervised learning.*

In the supervised learning paradigm, the availability of a set of representative instances of the problem is assumed, called the training dataset, in which each element consists of a pair $\left(x^{(i)}, y^{(i)}\right)$ formed by an input of the problem $x^{(i)}$ and the corresponding *ground-truth* response $y^{(i)}$ (i.e., the value that the model is expected to return for that particular input). The dataset will be denoted as $\mathcal{D} = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N_\mathcal{D}}$, with $N_\mathcal{D}$ representing its size, that is, the number of samples in the dataset. Generally, the training instances are assumed to be independent and identically distributed (i.i.d.) samples from the true yet unknown data distribution $\mathcal{P}(X, Y)$:

$$\left(x^{(i)}, y^{(i)}\right) \overset{i.i.d.}{\sim} \mathcal{P}(X, Y), \quad i = 1, \ldots, N_\mathcal{D}. \tag{2.1}$$

A convenient characteristic of the supervised learning paradigm is that the availability of the training dataset enables the predictive error of the model to be measured directly, by comparing the predictions with the ground-truth values using a *loss function* $\mathcal{L}(x, y, f_\theta)$. This allows estimating the optimal parameters for the model (i.e., those that minimize the expected error on the true data distribution) by means of finding the set of parameters that minimizes the *empirical* loss on $\mathcal{D}$:

$$\begin{align} \theta^* &= \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}\left[\mathcal{L}(x, y, f_\theta)\right] \tag{2.2} \\ &\approx \underset{\theta}{\operatorname{argmin}} \frac{1}{N_\mathcal{D}} \sum_{i=1}^{N_\mathcal{D}} \left[\mathcal{L}\left(x^{(i)}, y^{(i)}, f_\theta\right)\right]. \tag{2.3} \end{align}$$

Based on this setup, the different ML methods differ in i) the form of the predictive function f_θ, that is, the strategy employed to model the data and process the input samples to determine the prediction, and ii) the particular optimization strategy employed to minimize the empirical loss of the model given $\mathcal{D}$. In this dissertation, we will focus on one particular type of ML method, known as Deep Neural Networks, and which currently constitute the state-of-the-art in the great majority of AI tasks. Due to their relevance and influence, the study of these models currently represents a prominent area within the field of ML, popularly known as Deep Learning.

2.2 Deep Neural Networks

In the context of ML, Deep Neural Networks (DNNs) are computational models inspired by the *biological neural networks* of animal brains. More particularly, these models mimic, in a loosely simplified manner, the propagation of information through the biological neural networks. This process is assumed to be triggered by an inputs stimulus, which is captured by specialized input neurons. The captured information is then propagated through the network by means of triggering or activating other neurons, finally arriving to specialized type of output neurons, which produce a response to the input stimuli.
Formally, DNNs can be defined as computational graphs in which nodes represent *neurons*, while the (directed) edges represent the connections between neurons. More particularly, each neuron acts as a computational unit, taking as input the values of the neurons connected to it, and computing an output value that will be propagated to the neurons to which it is connected. In order to enable an organized flow of information, DNNs are generally arranged in different *layers*, in which a layer represents a subset of neurons. The way in which these layers are arranged and the way in which the information is propagated through them, usually denominated as the *architecture*, leads to different types of DNNs. In fact, the architectures can highly determine the expressivity of the models, and, consequently, the performance of the DNN in a particular task. For instance, some architectures are better suited to extract spatial information in vision tasks, whereas others are more appropriate to process sequential data in which there is an inherent order in their values, such as audio signals or text sentences.
In this dissertation, we will focus on Deep Neural Networks (DNNs), a term coined for those NNs that employ multiple layers to process the inputs and determine the prediction. Furthermore, we will assume *feed-forward* DNNs, in which layers are arranged sequentially so that the values of the neurons in the l-th layer depends only on those of the previous layers. In the following sections, we describe the most common types of feed-forward DNNs, which will also be the focus of this thesis: Multi-Layer Perceptrons and Convolutional Neural Networks.

2.2.1 Multi-Layer Perceptrons

In *Multi-Layer Peceptrons (MLPs)* [33, 67], which constitute the most basic DNN layout, each neuron in the l-th layer is connected only to the neurons in the $l-1$-th layer, and no connections are assumed between neurons of the same layer. Thus, two consecutive layers form a complete bipartite graph, often called *Fully Connected (FC)* layers or *Dense* layers.
Let L represent the number of layers in the network, and let the neuron values in the l-th layer, $l \in \{1, \ldots, L\}$, be denoted as:

$$\mathbf{x}^l = \left[\mathbf{x}_1^l, \mathbf{x}_2^l, \ldots, \mathbf{x}_{n_l}^l\right], \tag{2.4}$$

where n_l denotes the number of neurons in layer l. The values of the first layer $l = 1$, named as the *input layer*, will contain the input information, that is:

$$\mathbf{x}^1 = x. \tag{2.5}$$

Afterwards, the values of each layer $l \in \{2, \dots, L\}$ will be determined based on an affine transformation of the values in the previous layer $l - 1$, followed by an activation function σ:

$$\mathbf{x}^l = \sigma\left(\mathbf{x}^{l-1} \cdot W^l + b^l\right), \tag{2.6}$$

where W^l is a $n_{l-1} \times n_l$ weight matrix and $b^l \in \mathbb{R}^{n_l}$ is known as the *bias* term of layer l. The propagation process described in Equation (2.6) and the role of each term is clarified as follows. First, from the term $\mathbf{x}^{l-1} \cdot W^l$, it can be seen that the value of each neuron $\mathbf{x}_j^l \in \mathbf{x}^l$ is based on a weighted sum of the neurons in the previous layer, in which the contribution of each neuron $\mathbf{x}_i^{l-1}$ is weighted according to the weight $w_{i,j}^l$ of the corresponding connection or edge, that is, $\sum_{i=1}^{n_{l-1}} \mathbf{x}_i^{l-1} \cdot w_{i,j}^l$. Second, the role of activation functions is to introduce more expressiveness in the propagation process. Indeed, non-linear activation functions are conventionally chosen, which enables modeling non-linear transformations of the data. Otherwise, the value of each layer will be a linear transformation of the input, what would drastically limit the expressiveness of the model, as only linear decision boundaries could be modeled. Third, the bias term has the effect of shifting the activation function for each neuron, further enhancing the expressiveness of the model.
The propagation process defined in Equation (2.6) continues until the last layer of the model, known as the *output layer*, designed to contain one neuron per class. Generally, the *softmax function* is applied as the activation function of the output layer:

$$\mathbf{x}_i^L = \text{Softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}, \quad i = 1, \dots, k, \tag{2.7}$$

where $z = \mathbf{x}^{L-1} \cdot W^L + b^L$ represents the pre-activation values of the neurons in the output layer, popularly known as *logits*. Conveniently, the softmax function normalizes the final values so that $\sum_{i=1}^{k} \mathbf{x}_i^L = 1$ and $0 \leq \mathbf{x}_i^L \leq 1$, $i = 1, \dots, k$, allowing them to be interpreted as a categorical probability distribution over the output classes, and $\mathbf{x}_i^L$ as the probability of the input belonging to the class y_i according to the model. Finally, based on the softmax values, the most likely or probable class can be taken as the final prediction:

$$f(x) = y_c, \quad \text{where} \quad c = \underset{i=\{1,\dots,k\}}{\text{argmax}}\ \mathbf{x}_i^L \tag{2.8}$$

For the sake of simplicity, in the remainder of the dissertation, the logit and softmax value corresponding to the class y_i will be denoted as $\hat{f}(x)_i$ and $f(x)_i$, respectively.

2.2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [85, 93, 94] introduce to the architecture of the DNN a particular type of layer known as *convolutional layer*. The key advantage of these layers, first studied in image recognition tasks, is that they enable a more powerful and location-invariant feature extraction process,[2] which results in models with enhanced spatial reasoning capabilities in comparison to MLPs. Indeed, CNNs have proven to be remarkably effective in detecting complex discriminative patterns in inputs with a spatial structure, boosting the performance of DNNs in a wide range of tasks, and constituting one of the major breakthroughs in modern DL.

Let us assume a 3-dimensional input $x \in \mathbb{R}^{d_h \times d_w \times d_c}$, where d_h, d_w and d_c represent its height, width and depth (e.g., the number of channels in an image). Convolutional layers employ at least one *filter* or *kernel* $\kappa \in \mathbb{R}^{d'_w \times d'_h \times d_c}$, with $d'_w \leq d_w$ and $d'_h \leq d_h$, which is convolved[3] across the height and width dimensions of the input. This convolution returns a two-dimensional (single-channel) output, representing a filtered version of the input, known as *activation map*. Each convolutional layer can employ several filters in order to enable the detection of different features in the same input. Thus, the output of each convolutional layer will be a set of stacked activation maps, which will be propagated to the next layer. In addition, convolutional layers are usually complemented by *pooling layers*, which down-sample the activation maps by means of aggregating their values, for instance, by computing the average or maximum value of different patches in each map.

Put together, a sequence of convolutional and pooling layers leads to a powerful feature extraction process in which, as shown in previous works [184, 188], the complexity of the features increases with the depth of the layer (i.e., how far it is from the input layer). While the first convolutional layers often extract simple features such as the presence of edges or corners, deeper layers might be able to detect much more sophisticated patterns such as physiological characteristics or textures. Therefore, the complete architecture of a standard CNN consists of several convolutional and pooling layers, followed by a FC-layer, which, based on the extracted features, computes the final prediction.

Finally, it is worth stating that, although CNNs have been mostly studied in computer vision tasks due to their suitability for image data, they have proven to be effective also for other types of data and classification tasks, such as keyword spotting in audio signals [142].

[2] In this dissertation, unless otherwise specified, *features* are assumed to be abstract representations derived from *patterns* in the input data distribution (e.g., how round the objects in an image are), rather than the set of individual *attributes* that characterize the data (e.g., the set of pixels of an image).

[3] The convolution can be understood as a sliding-operation which, at each step, performs a dot-product between the filter and an input *patch*. Further details on this operation and its application to CNNs can be consulted in [99].

2.2.3 Training Deep Neural Networks

As explained in Section 2.1, the *learning* or *training* phase of a ML model is based on optimizing its set of parameters θ in order to minimize a loss function $\mathcal{L}$, evaluated on a training dataset $\mathcal{D}$. In the context of DNNs, θ represents the set of weights and biases of each layer, assuming that the remaining architectural configurations or *hyperparameters* (e.g., the number of layers, the activation function employed in each layer, etc.) are fixed beforehand.[4]

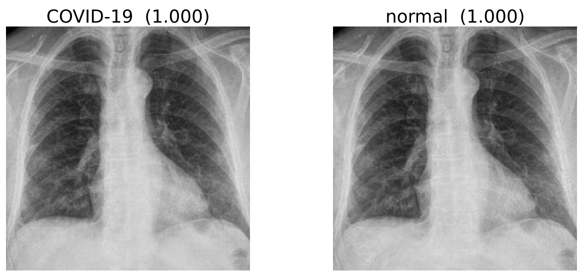
Given that the inference process of standard DNNs is end-to-end differentiable, their learning phase can be realized by gradient descent: an iterative optimization approach based on, first, computing the gradients of the loss function with respect to the parameters of the model, and, second, adjusting the parameter values by moving them in the opposite direction of the gradient, hence minimizing the loss function. Conventionally, the first step is implemented using the prominent *back-propagation* algorithm [93], which allows those gradients to be computed efficiently by means of applying the derivative chain-rule in a backward fashion, (i.e., propagating the computation from the last layer to the first layer). This is, therefore, a central algorithm in DL, which enabled scaling DNNs to large architectures and complex high-dimensional problems.

In its most general form, each gradient descent step is performed by computing the average of the gradients for the entire training dataset, and adjusting the parameters based on the directions determined by this aggregated result. This approach is popularly known as *deterministic* or *batch* gradient descent. However, processing all the training inputs is often impractical when it comes to datasets with a large number of samples. In order to tackle this drawback and increase the computational efficiency of the training process, in practice, only a random subset of samples can processed at each step, at the cost of relying on an estimation of the gradients to adjust the parameters, what generally leads to a more unstable convergence. The particular case in which only one sample is processed at a time is denominated *stochastic gradient descent* or *online learning*, which is particularly well-suited for budget-constrained (e.g., limited memory) learning scenarios. However, using only one input sample leads often to very noisy gradient estimations, which can dramatically slow the convergence of the training process and lead to poor fits in practice. Thus, in most modern scenarios, a better trade-off between efficiency and effectiveness is achieved by *mini-batch* gradient descent, in which a small subset of $N > 1$ samples is employed at each step, thus relying on a more robust estimation of the gradient.

[4] In practice, the hyperparameters can also be experimentally tuned to improve the effectiveness of the model for the task at hand, often referred to as *hyperparameter tuning*.

2.3 Adversarial Examples

Adversarial examples [158] are inputs deliberately manipulated by a malicious actor with the purpose of i) fooling the model into providing incorrect predictions, and ii) ensuring that the perturbations are imperceptible to humans. An illustration of an adversarial example is shown in Figure 2.1. The imperceptibility constraint ensures that the adversarial perturbation introduced to the inputs does not legitimate the change in the output classification. At the same time, the fact that imperceptible perturbations can fool DL models raised alarms regarding the vulnerability of these models. Indeed, adversarial examples have shown to be applicable to a wide range of high stakes and human-centered applications, such as healthcare systems [77], surveillance systems [12, 149, 198], automatic speech recognition [26], machine translation [17, 31, 40], dialogue or question and answering systems [34, 180], and financial applications such as credit loan approval or fraud detection systems [13, 27, 47, 87, 137]. The vulnerability to adversarial attacks has also been exposed in a wide range of popular *Machine Learning as a Service* APIs [20, 71, 126].



(a) Original input (b) Adversarial example

Fig. 2.1: Illustration of an adversarial example generated for a chest X-ray classification task, in which the objective is to categorize the status of the patient as one of three classes: "normal", Covid-19, or (non-Covid) pneumonia (more details in Section 4.4.1). (a) Original input sample in which the patient is diagnosed with Covid-19. (b) Adversarially manipulated input, which is misclassified by the model as "normal" (i.e., no disease found in the patient) despite being perceptually identical to the original image.

2.3.1 Taxonomy of Adversarial Attacks

The extensive research in the last few years has led to uncountable attack methods and paradigms, that is, procedures that exploit different vulnerabilities of the models in order to generate adversarial examples for multiple goals and scenarios. Consequently, different categories of attacks are considered in the literature depending on factors such as the specific type of error to be produced, the scope of the perturbation and the resources available to the adversary [186]. In this section, we present the main categories in order to describe the most common attack paradigms studied in the literature.

- **Type of misclassification**: Two main types of attacks can be differentiated depending on the type of error that wants to be produced: *untargeted* attacks and *targeted* attacks. On the one hand, untargeted attacks aim to produce an adversarial perturbation r capable of producing a misclassification of the model, that is, $f(x+r) \neq f(x)$, without any preference over the incorrect class. In contrast, targeted attacks aim to force the model to produce one particular incorrect class $y_t \neq f(x)$, that is, $f(x+r) = y_t$.
- **Scope of the perturbation**: In addition, different types of perturbations can be considered depending on whether they are generated for one specific input at hand (individual perturbations) [25, 59, 107, 116, 158] or whether they are designed to be input-agnostic and therefore effective with independence of the input in which they are applied (universal perturbations) [79, 114, 118, 120]. Although individual perturbations have been the main focus of study in the literature, universal perturbations allow adversarial attacks to be produced in scenarios where individual perturbations are impractical (for instance, scenarios requiring a fast or real-time computation of adversarial examples), or performing a high number of attacks more efficiently, avoiding having to generate a new (individual) perturbation for each new input. Nevertheless, given the fact that universal perturbations address a more general goal than the individual counterparts, generating effective universal perturbations has proven to be more challenging, requiring higher amounts of perturbation and more computational time.
- **Resources available to the adversary**: The information required by the adversary in order to effectively generate attacks leads to two main scenarios. On the one hand, in the *white-box scenario*, the adversary has full knowledge about the model internals (e.g., its architecture, weights or hyperparameters) and its training details. This allows highly efficient attacks to be generated, most of them relying on gradient-based strategies [25, 59, 107, 114, 116, 158]. On the other hand, in *black-box scenarios* the adversary has no knowledge about the model [9, 22, 71, 126]. More intermediate scenarios (sometimes referred to as gray-box scenarios) can be assumed when the adversary has limited access to the models, such as the output confidences assigned to every possible class or the logit values [71]. The opacity in terms of model details requires more costly strategies than those used in the white-box scenario, such as evolutionary algorithms [9, 133], gradient estimations [29, 71], or the use of surrogate models to generate the attack that are afterwards transferred to the initial model [106, 126].
- **Type of deployment**: Finally, a key aspect of adversarial examples is how those inputs are fed to the model. Generally, the assumed scenarios

in research works allow the input to be modified "digitally", which is afterwards fed to the model. In other cases, *physical* adversarial examples are crafted (e.g., printed traffic signs or malicious speech commands reproduced by a speaker) that are effective even when the signal is captured "over-the-air" and fed to the model [44, 149, 179]. This allows circumventing the possible limitation in real-world scenarios in which the adversary might not have access to the digital files.

A summary of the taxonomy described can be consulted in Table 2.1. We also refer the reader to the work of [186] for a more comprehensive and fine-grained survey on adversarial examples.

Factor	Categories	Explanation / goal
Type of misclassification	Targeted	Produce one particular incorrect output class.
	Untargeted	Produce an incorrect class without any preference over the available classes.
Scope of the Perturbation	Individual	Optimized for one particular input.
	Universal	Optimized to be applicable to multiple inputs.
Resources available to the adversary	White-box scenario	The adversary has full knowledge of the model (e.g., weights, hyperparameters, training details, etc.).
	Black-box scenario	The adversary has none (or very limited) information about the details of the model, which is considered a "black-box".
Type of deployment	"Digital-world"	The adversarial example is crafted digitally (i.e., by manipulating the "digital file") and is sent to the model "as-is".
	"Physical-world"	The adversarial example is generated "physically" in order to fool the model when the input is captured "over-the-air".

Table 2.1: Summary of the main taxonomy used to describe and categorize adversarial attacks.

2.3.2 How to Generate Adversarial Perturbations: Overview of the Main Methods

The first adversarial attack algorithm against DNNs was proposed in [158], in which, given an input x and a target class $y_t \in Y$, the search of the adversarial perturbation r is formulated as the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & ||r||_2 && (2.9)\\ \text{such that} \quad & 1.\ f(x+r) = y_t, && (2.10)\\ & 2.\ x + r \in X, && (2.11) \end{aligned}$$

which was solved by L-BFGS, a general-purpose second-order optimization technique. Since the discovery of these vulnerabilities, more sophisticated methods have been proposed, allowing to generate adversarial examples using more efficient and effective approaches. In this section, we describe the most influential attack algorithms proposed in the literature in order to generate adversarial examples, which will also be employed in this dissertation.

2.3.2.1 Fast Gradient Sign Method

In [59], a single-step gradient ascent approach was proposed, called *Fast Gradient Sign Method* (FGSM), to efficiently generate untargeted adversarial perturbations using a first-order approximation strategy. In particular, the attack strategy is based on linearizing the cross-entropy loss function $\mathcal{L}(x, y_c, f)$, where $y_c = f(x)$, and perturbing the input in the direction determined by the gradient of $\mathcal{L}(x, y_c, f)$ with respect to the input x, $\nabla_x \mathcal{L}(x, y_c, f)$. Thus, the adversarial example x' is generated according to the following closed formula:

$$x' = x + \epsilon \cdot \operatorname{sign}\big(\nabla_x \mathcal{L}(x, y_c, f)\big), \tag{2.12}$$

where $\operatorname{sign}(\cdot)$ is the sign function and ϵ a budget parameter that controls the ℓ_∞ norm of the perturbation. A targeted formulation can be obtained by considering the loss with respect to the target class y_t, $\mathcal{L}(x, y_t, f)$, and perturbing x in the opposite direction of the gradients, that is, $\operatorname{sign}\big(-\nabla_x \mathcal{L}(x, y_t, f)\big)$.

2.3.2.2 Projected Gradient Descent

The drawback of the FGSM is that a single step might not be enough to change the output class of the model. To solve this limitation, this strategy has been extended to iteratively perturb the input in the direction of the gradient:

$$x'_{[i+1]} = \mathcal{B}^x_{\infty,\epsilon}\Big(x'_{[i]} + \alpha \cdot \operatorname{sign}\big(\nabla_x \mathcal{L}(x'_{[i]}, y_c, f)\big)\Big), \quad x'_{[0]} = x \tag{2.13}$$

where $x'_{[i]}$ represents the adversarial example at the i-th step, α controls the step size and the projection operator $\mathcal{B}^x_{\infty,\epsilon}$ ensures that $||x' - x||_\infty \leq \epsilon$. This attack is known as the Projected Gradient Descent (PGD) [107], which is regarded as the strongest first-order adversarial attack [107, 174].

2.3.2.3 DeepFool

The DeepFool algorithm [116] consists of perturbing an initial input x towards the closest decision boundary of the decision space represented by the model. Due to the intractability of computing these distances in high-dimensional spaces, a first-order approximation of the decision boundaries is employed, and the input is iteratively pushed towards the (estimated) closest decision boundary at each step until a wrong prediction is produced. Precisely, being $\hat{f}(\cdot)_j$ the output logit of a classifier $f(\cdot)$ corresponding to the class y_j, $y_c = f(x)$ the correct class, $f'_j = \hat{f}(x'_{[i]})_j - \hat{f}(x'_{[i]})_c$ and $w'_j = \nabla \hat{f}(x'_{[i]})_j - \nabla \hat{f}(x'_{[i]})_c$, the following update-rule is employed:

$$x'_{[i+1]} \leftarrow x'_{[i]} + \frac{|f'_l|}{||w'_l||_2^2} w'_l, \quad l = \operatorname*{argmin}_{j \neq c} \frac{|f'_j|}{||w'_j||_2}, \tag{2.14}$$

in which w'_l represents the direction towards the (estimated) closest decision boundary, corresponding to the class y_l, and $\frac{|f'_l|}{||w'_l||_2}$ the step size. The targeted version of DeepFool can be obtained if, at every iteration i, the sample is moved in the direction of the target class $y_t \neq f(x)$, that is:

$$x'_{[i+1]} \leftarrow x'_{[i]} + \frac{|f'_t|}{||w'_t||_2^2} w'_t. \tag{2.15}$$

In this case, the process stops when the condition $f(x'_{[i]}) = y_t$ is satisfied.

2.3.2.4 Carlini and Wagner Attack

In [25], the problem of generating an adversarial example is formulated as the following optimization problem, hereinafter referred to as the C&W attack:

$$\text{minimize} \quad ||\frac{1}{2}(\tanh(\omega)+1) - x||_2^2 + \tau \cdot \mathcal{L}\left(\frac{1}{2}(\tanh(\omega)+1), y_t, f\right). \tag{2.16}$$

where $\mathcal{L}$ is the following loss function:

$$\mathcal{L}(x, y_t, f) = \max\big(\max_{j \neq t}\{\hat{f}(x)_j\} - \hat{f}(x)_t, -\mu\big). \tag{2.17}$$

The adversarial example is defined as $x' = \frac{1}{2}(\tanh(\omega)+1)$, which allows an unconstrained variable ω to be optimized, while ensuring that each value of the adversarial input is in a valid range, typically $[0,1]$. The parameter μ in equation (2.17) controls the desired confidence in the incorrect target class y_t, and the constant τ in equation (2.16) balances the trade-off between the perturbation norm and the confidence in the incorrect class.

3

Extending Adversarial Attacks to Produce Adversarial Class Probability Distributions

3.1 Introduction

As described in Section 2.3.1, most adversarial example generation methods can be taxonomized in different groups according to the scope or the objective of the adversarial attack. However, overall, the literature on adversarial attacks has mainly focused on "single-instance" scenarios, where the goal is to minimally manipulate the input at hand, so that the model misclassifies that instance in particular. Only a few works have considered "multiple-instance" scenarios, where adversarial attacks are used to achieve malicious goals that can only be realized by considering multiple inputs (e.g., by generating multiple attacks).

In [69, 101, 161], adversarial examples are sequentially created and fed to reinforcement learning models in order to control their behavior in the long run. More particularly, a sequence of adversarial examples is generated in [101] to force the model to take a preferred sequence of actions, which can be used to guide the agent towards a particular state of interest. In [161], the sequence of attacks attempts to impose an adversarial reward of interest on the victim policy at test time (i.e., leading the agent's policy to maximize the imposed adversarial reward). Similarly, the goal of imposing an adversarial target-policy is pursued in [69]. Other works attempted to sequentially generate *adversarial* inputs in order to introduce adversarial concept drifts in streaming classification scenarios [76, 84, 148], which can lead to a drop in the performance of the model. A comprehensive taxonomy of adversarial concept drifts is proposed in [84], where different types of goals are discussed, such as injecting a sequence of *corrupted* instances to make the adaptation to a real concept drift difficult, or injecting a sequence of adversarial instances which form a coherent concept and which are capable of inducing a concept drift.

In this chapter, we introduce a novel "multiple-instance" adversarial attack strategy. In particular, we propose a method which provides the adversary with the ability not only to deceive the model by adding imperceptible perturbations to the inputs, but also to control the frequency or proportion with

which each class is predicted by the model, even in scenarios where we can only introduce very small amounts of distortion to the inputs.
Let us consider a target ML model that implements a classification function $f : X \to Y$, where $X \subseteq \mathbb{R}^d$ represents the d-dimensional input space, and $Y = \{y_1, \ldots, y_k\}$ the set of possible output classes. The main objective of this chapter is to create an attack method Φ that is able to efficiently produce adversarial examples $x' = \Phi(x)$ not only with the objective of achieving $f(x') \neq f(x)$ for every input, but also to accomplish the objective of producing a specific probability distribution for the classes $\widetilde{\mathcal{P}}(Y) = (\tilde{p}_1, \ldots, \tilde{p}_k)$ after multiple attacks, that is:

$$P_{x \sim \mathcal{P}(X)} \left[f(\Phi(x)) = y_i \right] = \tilde{p}_i \ , \ 1 \leq i \leq k, \tag{3.1}$$

where $\mathcal{P}(X)$ represents the probability distribution of the *natural inputs*.

3.1.1 Applications and Use Cases

The idea of controlling the probability distribution of the classes produced by the adversarial examples (after sending multiple adversarial inputs to the model) provides a novel perspective to design such attacks. First, this attack can be used to produce drifts in the probability distribution of the classes, commonly referred to as *target shift* [192], *label shift* [49, 102] or as *prior probability shift* [19, 134, 141] in the literature. Indeed, it has been shown that such drifts can degrade the performance of the classifiers [102, 141, 168], or imply ethical issues when such changes cause the predictions of the models to be biased or unfair [19, 168]. Thus, strategies have been proposed to detect such changes in the probability distribution of the classes during the prediction phase, and even to correct or "adjust" the decisions of the models to, accounting for those changes, improve the classification performance of the model [102, 141, 168].
Secondly, controlling the output probability of the classes might also be of particular interest for those cases in which the frequency with which each class is predicted for multiple inputs (i.e., the class distribution) is more relevant than the individual predictions given for each input. This is, for instance, the case of the quantification learning paradigm [57, 58, 131]. Representative examples and domains of this paradigm are opinion mining, sentiment analysis or collective information retrieval in social networks [48, 53, 112, 187], where the main focus can be, for instance, on accurately estimating the frequency of a particular opinion among a population. Other sensitive domains where the aggregated results of the output class is relevant is epidemiology [82], for example, to estimate the cause-specific mortality in a population or the prevalence of a specific disease, which might be crucial to tackle it. Thus, in all these applications, maliciously changing the ratios with which each class is predicted for multiple inputs might bring about critical consequences, such as biased estimations of the population opinion or an incorrect screening of the prevalence of a disease, leading to an ineffective action plan.

Finally, whereas, to the best of our knowledge, defensive methods against such "multiple-instance" adversarial attacks have not been proposed (since all of them focus on counteracting attacks in the "single-instance" scenario), recent works have shown that a label-drift might be a clear indicator of adversarial activity [135]. Thus, label-drift detection methods [102, 135] could be straightforwardly applied as defensive countermeasures in order to detect that an adversary is sending multiple adversarial attacks to the models [135]. Therefore, from the adversary's perspective, controlling the frequency with which each class is predicted allows, for instance, the same probability distribution produced by the target DNN on clean inputs to be replicated, making the attacks less likely to be detected by label-drift detection mechanisms.

3.1.2 Contribution

For all these reasons, maliciously controlling the probability distribution of the classes can lead to more ambitious and complex attacks. However, the current adversarial attacks proposed in the literature are not capable of controlling such distributions. The main contribution of this chapter is to fill this gap by introducing a probabilistic framework with which any targeted adversarial attack can be extended to produce not only a misclassification in a DNN for the incoming inputs, but also any target probability distribution of the output classes after multiple attacks. In particular, we propose four different methods to create the optimal strategies to guide such attacks, and we validate them by extending a wide range of adversarial attacks for two exemplary ML problems. The effectiveness of the proposed four strategies is compared under multiple criteria, such as the similarity of the produced probability distributions and the target distributions, the percentage of inputs fooled by the attack or the number of parameters to be optimized for each method.[1]
The rest of this chapter is organized as follows. Section 3.2 provides a detailed description of the proposed adversarial attack strategy, and specifies a number of assumptions and key concepts. Section 3.3 introduces four different approaches to produce a target probability distribution for the output classes. Section 3.4 describes the experimental setups used to evaluate and compare the effectiveness of the approaches introduced. This section also includes the experimental results. Section 3.5 illustrates how our methods can be applied to produce label-shifts in streaming classification scenarios without alerting label-shift detection mechanisms. Section 3.6 concludes the chapter.

3.2 Producing Specific Class Probability Distributions

We focus on defining an attack approach in which the application of the attack for many incoming inputs x, assuming an input data distribution $x \sim \mathcal{P}(X)$,

[1] Our code is publicly available at: `https://github.com/vadel/ACPD`.

can produce not only a misclassification for every x, but also a desired (fixed) probability distribution of the predicted classes by the target model $\widetilde{\mathcal{P}}(Y) = (\tilde{p}_1, \ldots, \tilde{p}_k)$.

3.2.1 Assumptions and Key Concepts

In this section, we specify a number of assumptions and concepts that will be used to develop our methodology.
First of all, we assume that the clean input x is correctly classified by the target classifier, and that $f(\Phi(x) = x') \neq f(x)$, in order to ensure that the attack is actually fooling the model. In addition, being $\varphi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ a distortion metric and ϵ a maximum distortion threshold, we require the adversarial example to satisfy $\varphi(x, x') \leq \epsilon$, to ensure that x' is as similar as possible to x. In the literature, common choices for φ are ℓ_p norms such as the ℓ_2 or the ℓ_∞ norm.
The approach we introduce will use a targeted adversarial attack as a basis, that is, attacks capable of forcing the model to produce a particular target class $f(x') = y_j$. However, setting a maximum distortion supposes that we may not reach every possible target class by adversarially perturbing an input sample. For this reason, we consider that y_j is a *reachable class* from x if it is possible to generate a targeted adversarial example x', so that $\varphi(x, x') \leq \epsilon$ and $f(x') = y_j$, and it will be denoted as $\Phi(x) \to y_j$. We assume that $f(x)$ is always a reachable class. However, if there are no reachable classes $y_j \neq f(x)$, we will consider that we cannot create any *valid* adversarial example for x.

3.2.2 Attack Description

The main rationale of the approach we introduce is to *guide* a targeted adversarial attack method Ψ in order to achieve the global objective of producing any probability distribution of the output classes $\widetilde{\mathcal{P}}(Y)$, while maintaining a high fooling rate and minimally distorted inputs. To enable such attacks, our method consists of generalizing Ψ to be stochastic, so that the target class is randomly selected, and the probability of transitioning from the class y_i to the class y_j depends on the source class y_i and the input x at hand.
These probabilities will be represented by a transition matrix $T = [t_{i,j}]_{i,j=1}^{k}$, where $t_{i,j}$ represents the probability of transitioning from the class y_i to the class y_j. In the event that, given an input x of class y_i, it is not possible to reach all the classes without exceeding the distortion limit, the probability of transitioning to a non-reachable class will be set to zero, and the probability distribution $(t_{i,1}, \ldots, t_{i,k})$ will be normalized accordingly. Thus, being $\mathcal{Y} = \{y_j \mid \Phi(x) \to y_j\}$ the set of reachable classes for one particular input x of class y_i, the probability of selecting y_j as the target class is determined by:

$$t'_{i,j} = \begin{cases} \frac{t_{i,j}}{\sum_{y_r \in \mathcal{Y}} t_{i,r}} & \text{if } y_j \in \mathcal{Y} \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

By modeling the decision to move from one class to another in this way, it is possible to approximate with which probability the model will predict each class after multiple attacks. Algorithm 1 provides the pseudocode of this approach.
Note that $t'_{i,i}$ represents the probability of maintaining an input in its own class y_i, and, therefore, these values should be as low as possible in order to ensure that we maximize the number of inputs that will fool the model.

Algorithm 1 Generating adversarial class probability distributions.

```
Require: A classification model $f$, a set of classes $Y = \{y_1, \ldots, y_k\}$, a targeted adversarial attack method $\Psi$, a distortion metric $\varphi$, a maximum distortion threshold $\epsilon$, a transition matrix $T$, a set of input samples $\hat{\mathcal{X}}$.
 1: for each $x \in \hat{\mathcal{X}}$ do
 2:   $reachable[1, \ldots, k] \leftarrow$ initialize with $False$.
 3:   for $j \in \{1, \ldots, k\}$ do
 4:     $v_j \leftarrow$ use $\Psi$ to generate an adversarial perturbation for $x$ targeting class $y_j$
 5:     if $f(x + v_j) = y_j \wedge \varphi(x, x + v_j) \leq \epsilon$ then
 6:       $reachable[j] \leftarrow True$
 7:     end if
 8:   end for
 9:   $\mathcal{Y} \leftarrow \{y_j \in Y \mid reachable[j] = True\}$
10:   $(t_{i,1}, \ldots, t_{i,k}) \leftarrow$ probability distribution in the row of $T$ corresponding to the source class $y_i = f(x)$.
11:   $t_{sum} \leftarrow \sum_{y_j \in \mathcal{Y}}(t_{i,j})$
12:   if $t_{sum} = 0$ then
13:     Feed $x$ to the model $f$.
14:   else
15:     for $j \in \{1, \ldots, k\}$ do
16:       if $reachable[j] = True$ then
17:         $t'_{i,j} \leftarrow \frac{t_{i,j}}{t_{sum}}$
18:       else
19:         $t'_{i,j} \leftarrow 0$
20:       end if
21:     end for
22:     $y^* \leftarrow$ randomly select a class according to the probabilities $(t'_{i,1}, \ldots, t'_{i,k})$.
23:     Select the adversarial example with the targeted perturbation $v^*$ corresponding to class $y^*$:
        $x' \leftarrow x + v^*$
24:     Feed $x'$ to the model $f$.
25:   end if
26: end for
```

However, depending on the probability distribution of the classes we want to produce, a nonzero value for these probabilities may be needed to achieve such goals, for instance, if we require a high probability for one class but this class is seldomly reached from inputs belonging to the rest of classes. How to obtain transition matrices T that comply all the aforementioned conditions will be discussed in detail in Section 3.3, where four different methods are proposed.
Finally, we would like to point out that our approach is not subject to any particular targeted adversarial attack strategy, and, therefore, it is agnostic with regard to the particularities of the selected strategy (for example, the amount of information about the model that the adversary can exploit to generate the attacks). This allows the adversary to select the most appropriate attack depending on the requirements of the problem or scenario. For instance, for scenarios where the computation time is a critical aspect or for problems with a large number of classes, the adversary can opt for adversarial attacks with low computational cost.[2] On the other hand, in less restrictive scenarios, the adversary can employ more effective attacks, at the expense of higher computational cost.

3.3 Constructing Optimal Transition Matrices to Guide Targeted Attacks

In this section we introduce different strategies to construct the optimal transition matrix T which, used to stochastically decide the class transitions, produces a target probability distribution $\widetilde{\mathcal{P}}(Y) = (\tilde{p}_1, \ldots, \tilde{p}_k)$ for the output classes. Formally, being $\mathcal{X}$ a set of inputs sampled from the input data distribution $\mathcal{P}(X)$, and $\mathcal{P}(Y) = (p_1, \ldots, p_k)$ the original probability distribution of the classes assigned by the target classifier, we want to obtain a transition matrix T that satisfies:

$$(p_1, \ldots, p_k) \begin{pmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,k} \\ t_{2,1} & t_{2,2} & \cdots & t_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ t_{k,1} & t_{k,2} & \cdots & t_{k,k} \end{pmatrix} = (\tilde{p}_1, \ldots, \tilde{p}_k). \tag{3.3}$$

We will define the problem of finding such matrices as a linear program, and, in order to restrict the possible values of T, different strategies will be introduced.
The main objective is to ensure that T satisfies equation (3.3). Therefore, this equation is added as a constraint of the linear program. The second objective

[2] It is worth pointing out that the process of generating an adversarial example for each target class is fully parallelizable, since each targeted attack is independent of the others, making our method applicable in practice even for problems with a large number of classes.

is to maximize the expected fooling rate of the attack, that is, to minimize the probability of keeping the original class predicted by the model unchanged. This will be achieved by adding the sum of the diagonal of T as a component of the objective function of the linear program, which will be minimized.
It is important to note that, although multiple optimal solutions may exist for these problems, it is not expected that all of them will produce the same approximation to the target probability distribution of the classes $\widetilde{P}(Y)$ when applied in the prediction phase of the classifier (i.e., when our attacks are put in practice). For instance, if many of the values of T are zero, then these transitions cannot be carried out, resulting in inaccurate approximations of $\widetilde{P}(Y)$. Similarly, if many of the transitions are not possible in practice (something that can happen for low distortion budgets or problems in which targeted attacks can not always be successfully generated), then different transition matrices could produce very different results. For these reasons, the four methods that will be introduced in this section will rely on different strategies to increase the effectiveness of the matrices in the prediction phase. In addition, they will differ in the amount of information they use from $\mathcal{X}$. While our first method is almost agnostic, the subsequent three use more informative approaches.

3.3.1 Method 1: Agnostic Method (AM)

The first method will follow an almost agnostic approach to generate the transition matrix T, where the only information that will be used is the initial probability distribution $\mathcal{P}(Y)$. Therefore, the results obtained with this method will be used to compare the gain that the following methods imply, in which more informed transition matrices will be created.
Thus, this method consists of directly searching for a transition matrix T that satisfies equation (3.3), while minimizing the sum of the diagonal of T. To avoid a high number of null $t_{i,j}$ probabilities outside the diagonal of T, an auxiliary variable matrix $L = [l_{i,j}]_{i,j=1}^{k}$ will be introduced, so that $l_{i,i} = 0$ and $0 \leq l_{i,j} \leq \xi, i \neq j$, with $\xi \in \mathbb{R}$ and $\xi \ll 1/k$. Each $l_{i,j}$ will be included in the set of restrictions as a lower bound of $t_{i,j}$ to require a minimum probability, $l_{i,j} \leq t_{i,j}$, $i \neq j$. At the same time, the values in L will be maximized in the objective function of the linear program.
Taking into account all these basic requirements, the optimal transition matrix T can be obtained by solving the following linear program:

$$\begin{aligned}
\min \quad & z = \gamma_1 \cdot \sum_{i=1}^{k} t_{i,i} - \gamma_2 \sum_{i=1}^{k} \sum_{\substack{j=1 \\ j \neq i}}^{k} l_{i,j} \\
\text{s.t.} \quad & \mathcal{P}(Y) \cdot T = \widetilde{\mathcal{P}}(Y) \\
& \sum_{j=1}^{k} t_{i,j} = 1 && \forall i \in \{1, \ldots, k\} \\
& t_{i,j} \geq l_{i,j} && \forall i, j \in \{1, \ldots, k\}, i \neq j \\
& 0 \leq t_{i,j} \leq 1 && \forall i, j \in \{1, \ldots, k\} \\
& 0 \leq l_{i,j} \leq \xi && \forall i, j \in \{1, \ldots, k\}.
\end{aligned} \tag{3.4}$$

For the sake of generality, a coefficient $\gamma \in \mathbb{R}$ was included for each of the main terms in the objective function, allowing their importance to be traded off. This will also be done in the subsequent methods.

3.3.2 Method 2: Upper-Bound Method (UBM)

In the second method, we will extend the linear program introduced in the AM to include an additional restriction to the values of T in order to capture more accurate information about the feasible class transitions associated to the perturbations. In this approach, an auxiliary matrix $R = [r_{i,j}]_{i,j=1}^{k}$ will be considered, in which $r_{i,j}$ represents the number of samples in $\mathcal{X}$ that, with a ground-truth class y_i, can reach the class y_j:

$$r_{i,j} = \left|\{x \in \mathcal{X} \mid f(x) = y_i \wedge \Phi(x) \rightarrow y_j\}\right|. \tag{3.5}$$

We assume that the ground-truth class of an input is always reachable, and therefore, $r_{i,i} = \left|\{x \in \mathcal{X} \mid f(x) = y_i\}\right|$, $1 \leq i \leq k$. If we divide each $r_{i,j}$ by the number of inputs of class y_i, the value will represent the proportion of samples in $\mathcal{X}$ which, with a ground-truth class y_i, can reach the class y_j:

$$r'_{i,j} = \frac{r_{i,j}}{\left|\{x \in \mathcal{X} \mid f(x) = y_i\}\right|}. \tag{3.6}$$

Note that we are estimating, by using the set $\mathcal{X}$, the proportion of successful targeted attacks that it is possible to create for the inputs coming from $\mathcal{P}(X)$, for any pair of source class y_i and target class y_j. Therefore, to generate a more informed transition matrix T, we will maintain the following restriction: $t_{i,j} \leq r'_{i,j}$. The aim of this restriction is to avoid assigning transition probabilities so high that, in practice, it will be unlikely to obtain them due to the distortion threshold, which may imply a loss of effectiveness regarding the global objective of producing $\widetilde{\mathcal{P}}(Y)$, as the algorithm may not be able to successfully follow the guidance of T.

However, setting an upper bound to the values of T according to the values of R may imply increasing the values in the diagonal, decreasing the fooling rate expectation. Moreover, those restrictions can be too strict for low distortion thresholds, making it impossible to find feasible solutions in the linear program for a large number of cases (see Table 3.1). Thus, to relax these restrictions, we will consider an auxiliary set of variables $0 \leq \eta_{i,j} \leq 1$, that will act as upper thresholds for the $t_{i,j}$ values in the matrix T, and which will be minimized in the objective function. Based on all these facts, we will generate the optimal transition matrix T by solving the following linear program:

$$\begin{aligned}
\min \quad & z = \gamma_1 \cdot \sum_{i=1}^{k} t_{i,i} - \gamma_2 \cdot \sum_{i=1}^{k} \sum_{\substack{j=1 \\ j \neq i}}^{k} l_{i,j} + \gamma_3 \cdot \sum_{i=1}^{k} \sum_{j=1}^{k} \eta_{i,j} \\
\text{s.t.} \quad & \mathcal{P}(Y) \cdot T = \widetilde{\mathcal{P}}(Y) \\
& \sum_{j=1}^{k} t_{i,j} = 1 && \forall i \in \{1, \ldots, k\} \\
& t_{i,j} \geq l_{i,j} && \forall i, j \in \{1, \ldots, k\}, i \neq j \\
& 0 \leq t_{i,j} \leq 1 && \forall i, j \in \{1, \ldots, k\} \\
& 0 \leq l_{i,j} \leq \xi && \forall i, j \in \{1, \ldots, k\} \\
& t_{i,j} \leq r'_{i,j} + \eta_{i,j} && \forall i, j \in \{1, \ldots, k\} \\
& 0 \leq \eta_{i,j} \leq 1 && \forall i, j \in \{1, \ldots, k\}.
\end{aligned} \tag{3.7}$$

3.3.3 Method 3: Element-Wise Transformation Method (EWTM)

The main drawback of the strategy used in the UBM is that establishing bounds for every value of T can significantly limit the space of possible transition matrices, reducing the range of target probability distributions $\widetilde{\mathcal{P}}(Y)$ that can be produced. Therefore, a relaxation of those restrictions is required in order to achieve feasible solutions, which at the same time could, however, reduce the effectiveness of the approach.

In addition, even if it is estimated, using the set $\mathcal{X}$, that it is not possible to move more than a certain proportion of cases $r'_{i,j}$ from the class y_i to the class y_j, in some cases it can be necessary to assign values higher than $r'_{i,j}$ to $t_{i,j}$, for example, to produce y_j with a very high probability. In such cases, even if reaching the class y_j from the class y_i is unlikely, we can specify that when this transition is possible, it should be produced with a high probability.

Therefore, in order to be able to accurately approximate a wider range of distributions $\widetilde{\mathcal{P}}(Y)$, the EWTM does not impose bound constraints on the values of T. Apart from that, the row-normalized version of R will be used in this method, denoted as $\hat{R}$, which already represents a transition matrix. In particular, the probability distribution $\mathcal{P}(Y)\hat{R}$ is the one that would be achieved if the target class of each input x were uniformly selected in the set of reachable classes for x. As our goal is to produce $\widetilde{\mathcal{P}}(Y)$, we aim to find an auxiliary matrix $Q = [q_{i,j}]_{i,j=1}^{k}$, so that $\mathcal{P}(Y)(\hat{R} \odot Q) = \widetilde{\mathcal{P}}(Y)$, where the operator $\odot$ represents the Hadamard (element-wise) product. If we denote $T = \hat{R} \odot Q$, we can generate T by solving the following linear program:

$$\begin{aligned}
\min \quad & z = \gamma_1 \cdot \sum_{i=1}^{k} \hat{r}_{i,i} \cdot q_{i,i} + \gamma_2 \cdot \eta \\
\text{s.t.} \quad & \mathcal{P}(Y) \cdot (\hat{R} \odot Q) = \widetilde{\mathcal{P}}(Y) \\
& \sum_{j=1}^{k} \hat{r}_{i,j} \cdot q_{i,j} = 1 && \forall i \in \{1, \dots, k\} \\
& 0 \leq \hat{r}_{i,j} \cdot q_{i,j} \leq 1 && \forall i,j \in \{1, \dots, k\} \\
& 0 \leq q_{i,j} \leq \eta && \forall i,j \in \{1, \dots, k\} \\
& 0 \leq \eta
\end{aligned} \tag{3.8}$$

The values in Q will be minimized by including, as a decision variable, an upper bound η for its values, avoiding excessive transformations of the matrix $\hat{R}$ which would lead to losing its influence, and, therefore, resembling an *agnostic* approach similar to the AM.

3.3.4 Method 4: Chain-Rule Method (CRM)

As explained in Section 3.2.2, even if we specify a probability $t_{i,j}$ for every possible transition, in practice, an input sample may not be able to reach any possible class without surpassing the maximum distortion allowed, so we need to normalize those probabilities to consider only the reachable classes from that input. However, the two previous methods have not considered this effect during the optimization process of the transition matrices, which may cause a reduction in the effectiveness of the resulting matrices when they are applied during the prediction phase of the model. For this reason, in this method, we will make use of that information with the aim of achieving a more informed attack.

To construct the transition matrix T, we start by estimating the probabilities that an input x of class y_i can only reach a particular subset of the classes $\mathcal{S} \subseteq Y$ (i.e., $y_j \in \mathcal{S} \Leftrightarrow \Phi(x) \rightarrow y_j$). We will denote these probabilities

$$P_{x \sim \mathcal{P}(X)}(\mathcal{S} | f(x) = y_i), \tag{3.9}$$

or $P(\mathcal{S}|y_i)$ for simplicity. In order to estimate these values, the set of reachable classes $\mathcal{S}$ will be computed for each $x \in \mathcal{X}$, and the frequency of each subset will be calculated.

The next step is to define the probability that an input x of class $f(x) = y_i$ and with a set of reachable classes $\mathcal{S}$ will be moved from y_i to the class y_j, that is,

$$P_{x \sim \mathcal{P}(X)}(y_j | f(x) = y_i, \mathcal{S}), \tag{3.10}$$

or $P(y_j|y_i,\mathcal{S})$ for simplicity. These probabilities will be also denoted as $V_{i,j}^{\mathcal{S}}$ when referring to them as variables in the linear program. All these values will directly define the transition matrix T in the following way:

$$t_{i,j} = \sum_{\mathcal{S}\subseteq Y} P(y_j|y_i,\mathcal{S})P(\mathcal{S}|y_i) = \sum_{\mathcal{S}\subseteq Y} V_{i,j}^{\mathcal{S}} P(\mathcal{S}|y_i) \tag{3.11}$$

As we assume that the ground-truth class of an input is always reachable, for the inputs of class y_i, the probabilities corresponding to those sets $\mathcal{S}$ in which $y_i \notin \mathcal{S}$ will be zero. That is, $P(\mathcal{S}|y_i) = 0$ if $y_i \notin \mathcal{S}$. Similarly, $P(y_j|y_i,\mathcal{S})$ must be zero if $y_j \notin \mathcal{S}$.
In order to find the appropriate values for the variables $V_{i,j}^{\mathcal{S}}$, we will solve the following linear program:

$$\begin{aligned}
\min \quad & z = \sum_{i=1}^{k} t_{i,i} & \\
\text{s.t.} \quad & \mathcal{P}(Y)\cdot T = \widetilde{\mathcal{P}}(Y) & \\
& \sum_{j=1}^{k} V_{i,j}^{\mathcal{S}} = 1 & \forall i \in \{1,\dots,k\}\ ,\ \forall \mathcal{S}\subseteq Y \\
& 0 \leq V_{i,j}^{\mathcal{S}} \leq 1 & \forall i,j \in \{1,\dots,k\}\ ,\ \forall \mathcal{S}\subseteq Y \\
& V_{i,j}^{\mathcal{S}} = 0 & y_j \notin \mathcal{S}
\end{aligned} \tag{3.12}$$

The main disadvantage of this method is that it requires a considerably larger number of decision variables, bounded by $O(2^k k^2)$, assuming that for k classes there are 2^k possible subsets of reachable classes $\mathcal{S}$, each with an associated probability $P(\mathcal{S}|y_i)$, and for each of them another distribution of k probabilities $P(y_j|y_i,\mathcal{S})$, which are optimized in the linear program.
Due to the high number of possible subsets, in practice, $P(\mathcal{S}|y_i)$ will be zero for many of the subsets $\mathcal{S}$. This reduces the number of parameters that can be tuned, and also, as a consequence, the number of probability distributions that can be produced. For this reason, to avoid having multiple null values for those probabilities, in this method we will smooth every probability distribution $P(\mathcal{S}|y_i)$ using the Laplace smoothing [109].
In addition, after a preliminary experiment we discovered that, because of the values of $P(\mathcal{S}=\{y_i\}|y_i)$, the linear problem was infeasible for many target probability distributions, especially for low distortion thresholds. This is because the values $t_{i,i}$ are highly influenced by such probabilities, which, indeed, are lower thresholds for $t_{i,i}$. In addition, those probabilities can be considerably higher than those corresponding to the remaining subsets if there is a sufficiently large proportion of samples that cannot be fooled, especially for low values of ϵ. This also translates into a low fooling rate expectation.
To avoid all these consequences, after the Laplace smoothing, we set every $P(\mathcal{S}=\{y_i\}|y_i)$ to zero and normalize every distribution $P(\mathcal{S}|y_i)$ accordingly,

$i = 1, \dots, k$, even if this can reduce the effectiveness of the method in producing the target probability distribution, as we are not considering the estimated proportion of samples that can not be fooled.

3.3.5 Overview of the Attack Strategies

All the strategies introduced in the previous sections can be used to generate the transition matrices needed to produce adversarial class probability distributions, all of them relying on a different strategy to model the solutions.
Both the UBM and EWTM provide a simple framework that allows the transition matrices T to be directly optimized. In the UBM, the proportion of samples $r'_{i,j}$ that can be moved from each class y_i to another class y_j is estimated, and those values are used as upper bounds for T, assuming that, in practice, it will be unlikely to move a larger proportion of samples. In the EWTM, the aim is to transform the transition matrix $\hat{R}$ in order to meet our particular requirements, without setting boundaries to the values of T. A positive point in both methods is the low number of parameters to be optimized, bounded by $O(k^2)$. The CRM, however, requires a considerably larger number of parameters, bounded by $O(2^k k^2)$, but provides a more comprehensive and general approach to generate the transition matrix. In particular, contrarily to the previous strategies, it allows the particular set of reachable classes for each instance individually to be taken into account, instead of considering *aggregated* information.

3.4 Validating Our Proposals: Setup and Results

In this section, we present the particular task, dataset, model and further details regarding the experimental setup used to validate our proposals. We also report the obtained results, in which we measure the effectiveness of the introduced approaches according to different criteria.

3.4.1 Case of Study: Speech Command Classification

Due to advances in automatic speech recognition technologies based on DL models, and their deployment in smartphones, voice assistants and industrial applications, there has been a rapid increase in the study of adversarial attacks and defenses for such models [26, 42, 96, 111, 143, 157], despite being considerably less studied than computer vision problems. For these reasons, we have decided to validate our proposal in the task of speech command classification, an exemplary and representative task in this domain. Nevertheless, we remark that our methods can be directly applied to any problem or domain, as long as it is possible to generate targeted adversarial attacks, and that this selection is only for illustration purposes.

We use the Speech Command Dataset [175], which consists of a set of WAV audio files of 30 different spoken commands. The duration of all the files is fixed to 1 second, and the sample-rate is 16kHz in all the samples, so that each audio waveform is composed of 16000 values, in the range $[-2^{15}, 2^{15}]$. We use a subset of ten classes, following previous publications [8, 39, 56, 100, 175, 183], so that our results are more comparable with previous works in the literature: *Yes, No, Up, Down, Left, Right, On, Off, Stop*, and *Go*. In order to provide a more realistic setup, two special classes have also been considered: *Silence*, representing that no speech has been detected, and *Unknown*, representing an unrecognized spoken command, or one which is different to those mentioned before. The dataset contains 46.258 samples, accounting for approximately 13 hours of data, and it is split into training (80%), validation (10%) and test (10%) sets, following the standard partition procedure proposed in [175].
Also following previous publications [8, 39, 56, 100, 175], a CNN will be used as a classification model, based on the architecture proposed in [142], which is particularly well-suited for small-footprint keyword recognition tasks. The test accuracy of the model is 85.52%. The input of the model will be the Mel-Frequency Cepstral Coefficients (MFCCs) extracted from the raw audio waveform, which is a standard feature extraction process in speech recognition [122]. Nevertheless, the adversarial examples will be generated directly in the audio waveform representation, as done in previous works [8, 26, 39, 132, 181].

3.4.2 Experimental Details

The ultimate goal is to validate that any desired probability distribution $\widetilde{\mathcal{P}}(Y)$ can be approximated with a low error by guiding a targeted adversarial attack using Algorithm 1 and a transition matrix T, which has been optimized using any of the methods introduced in Section 3.3.
To show that any targeted attack strategy can be extended, we will evaluate our methods using a wide range of attacks, which have been exhaustively employed in the literature and which rely on different strategies to generate the adversarial perturbations: DeepFool [116], Fast Gradient Sign Method [59], Projected Gradient Descent [107] and Carlini & Wagner attack [25]. An introduction to these algorithms is provided in Section 2.3.2. For the sake of simplicity, the experimental results presented in this section will be reported for the DeepFool algorithm[3], whereas the results obtained with the other attack algorithms will be reported in Appendix A.1.
In all the experiments, we will assume a uniform initial probability distribution $\mathcal{P}(Y)$. In Section 3.4.3, the particular case in which $\widetilde{\mathcal{P}}(Y) = \mathcal{P}(Y)$ will be tested, that is, when the aim is to reproduce the original probability distribution $\mathcal{P}(Y)$ obtained by the model (in our case the uniform distribution). Afterward, in Section 3.4.4, a more general scenario will be tested, in which

[3] In order to fit in our specification, we employed a targeted version of DeepFool, as described in Section 2.3.2.3.

different target probability distributions $\widetilde{\mathcal{P}}(Y)$ will be randomly sampled from a Dirichlet distribution of $k = 12$ parameters and $\alpha_i = 1,\ 1 \leq i \leq 12$. A total of 100 different target probability distributions will be sampled, and our methods will be tested in each of them. This general case will be used to provide an exhaustive comparison of the effectiveness of the methods introduced.
To generate the transition matrices T, a set of samples $\mathcal{X}$ will be used, composed of 500 samples per class, which makes a total of 6000 input samples. In particular, $\mathcal{X}$ will be used to generate the auxiliary matrix R required in the UBM and EWTM, as described in equation (3.5), and, for the case of the CRM, to estimate the probabilities described in equation (3.9). The generated transition matrices T will be tested using another set of samples $\hat{\mathcal{X}}$, disjoint from $\mathcal{X}$, also composed of 500 inputs per class. The proportion of samples that has been classified as each particular class after the attack is applied to every input in $\hat{\mathcal{X}}$ will be taken as the *empirical* probability distribution, and will be denoted $\hat{\mathcal{P}}(Y) = (\hat{p}_1, \dots, \hat{p}_k)$. Using this collection of data, we will evaluate to what extent the empirical probability distributions $\hat{\mathcal{P}}(Y)$ match $\widetilde{\mathcal{P}}(Y)$. The similarity between both distributions will be measured using different metrics: the maximum and mean absolute difference, the Kullback-Leibler divergence and the Spearman correlation.
To thoroughly evaluate our methods, we randomly sampled a set $\bar{\mathcal{X}}$ of 1000 inputs per class from the training set of the Speech Command Dataset, and computed a 2-fold cross-validation, using one half of $\bar{\mathcal{X}}$ as $\mathcal{X}$ and the other half as $\hat{\mathcal{X}}$. Moreover, we launched 50 repetitions of the cross-validation process, using in every repetition a different random partition of $\bar{\mathcal{X}}$. An additional evaluation of our methods considering different sizes for the set $\mathcal{X}$ will be provided in Appendix A.2.
The transition matrices will be generated using the four linear programs described in Sections 3.3.1, 3.3.2, 3.3.3 and 3.3.4. The linear programs are solved using the Python PuLP library [4] and the Coin-or Branch and Cut (CBC) solver [5]. For the AM and the UBM, an upper bound of $\xi = 0.01$ will be set for the values in L. For the AM, the UBM and the EWTM, $\gamma_1 = \gamma_2 = 1$ will be set, as well as $\gamma_3 = 10$ for the UBM, in order to avoid the relaxation of the upper-bounds $r'_{i,j}$.
In addition, the ℓ_2 norm of the adversarial perturbation (in the raw audio waveform representation) will be used as the distortion metric φ. The results will be computed under the following maximum distortion thresholds: $\epsilon \in \{0.0005, 0.001, 0.0025, 0.005, 0.01, 0.05, 0.1, 0.15\}$. These values were empirically selected in order to evaluate the behavior of our methods depending on how restricted the adversary is, ranging from scenarios where only very few class transitions can be performed (i.e., low values of ϵ), to scenarios where the number of possible transitions is high (i.e., high values of ϵ).

[4] `https://github.com/coin-or/pulp`
[5] `http://www.coin-or.org/`

Finally, our methods will be compared against two baseline methods. With the first baseline, for each input x, the target class will be selected according to the probabilities defined in the target distribution $\widetilde{\mathcal{P}}(Y)$. Notice that, following the introduced methodology, this method can be modeled as a transition matrix T in which all the rows contain the target distribution $\widetilde{P}(Y)$. Thus, this method will presumably provide a good approximation of the target distribution, but also fooling rates far from the optimum, as it only focuses on producing the target distribution, with no particular incentive to maximize the fooling rate. Hence, we will refer to this baseline as the *Maximum Approximation Baseline* (MAB).

On the other hand, the second baseline will also follow the same strategy as the MAB, with the difference that, in order to maximize the fooling rate of the attack, the diagonal of T (i.e., the probability of staying in the ground-truth class) will be set to zero, and each row will be normalized accordingly:

$$\begin{cases} t_{i,i} = 0, & 1 \leq i \leq k, \\ t_{i,j} = \dfrac{\widetilde{p}_j}{\sum_{\substack{r=1 \\ r\neq i}}^{k} \widetilde{p}_r}, & 1 \leq i,j \leq k, \quad i \neq j. \end{cases} \tag{3.13}$$

Therefore, this baseline provides a maximum fooling rate, but, presumably, at the expense of producing worse approximations to $\widetilde{\mathcal{P}}(Y)$ than the MAB. For this reason, we will refer to this baseline as the *Maximum Fooling Rate Baseline* (MFRB).

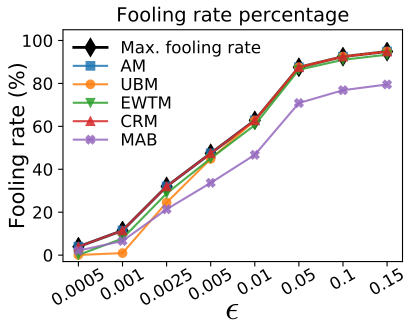
For the purpose of evaluating the baselines under the same conditions as our methods, the normalization described in Equation (3.2) will be also employed before each attack (see Algorithm 1, lines 11-21). In this way, sampling a target class that is not reachable from the input x at hand is avoided, which favors the baselines.

3.4.3 Illustrative Case: Reproducing the Initial Probability Distribution

For illustration purposes, we first report the results obtained for the particular scenario in which we want to produce the same probability distribution that the model produces when it is applied on clean samples. Notice that this distribution is the same as the ground truth distribution of the classes, since we assume that the model produces a correct classification for the original samples. Having the ability to reproduce such distributions allows an adversary to deploy attacks that are less likely to be detected in the long run, for instance, by label-shift detection methods that can warn against the presence of multiple adversarial attacks against the model [135].[6]

[6] We clarify that this does not imply that each individual attack that is sent to the model will also be less detectable, as this depends on the underlying targeted adversarial attack method employed.

To begin with, Figure 3.1 (left) shows the achieved fooling rates for each method, as well as the maximum fooling rate that can be achieved for every ϵ as reference, that is, the percentage of inputs in $\hat{\mathcal{X}}$ for which it is possible to create a targeted attack capable of fooling the model. These results have been averaged for the 50 different 2-fold cross-validations. For the sake of simplicity, and since the MFRB achieves by definition the maximum possible fooling rate, the results corresponding to that baseline are not included in the figure. According to the results, the four attack methods maintained fooling rates very close to the optimal values independently of the distortion threshold, with the exception of the UBM and the EWTM, in which a loss can be observed (of approximately 10% and 4%, respectively) for the lowest values of ϵ evaluated. It can also be noticed that the lowest fooling rates are achieved by the MAB, with a loss of approximately 15% for $\epsilon \geq 0.005$.



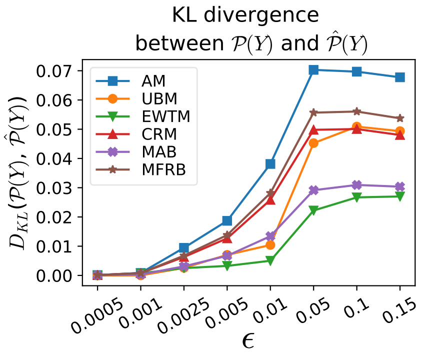


Fig. 3.1: Fooling rates (left) and Kullback-Leibler divergence (right) obtained with each of the proposed methods in 50 2-fold cross-validation trials, for the particular case in which the target distribution $\widetilde{\mathcal{P}}(Y)$ is the initial (uniform) probability distribution $\mathcal{P}(Y)$. The results corresponding to the MFRB have been omitted from the left figure, since that method achieves the maximum fooling rate by definition.

Regarding the effectiveness in reproducing the initial probability distribution, Figure 3.1 (right) includes the average Kullback-Leibler divergence obtained for every ϵ. In order to better assess the similarity between the initial probability distribution and the ones produced after perturbing the inputs with our attacks, Figure 3.2 contains a graphical comparison of these distributions, for one of the folds included in the cross-validation trials, considering three different maximum distortion thresholds ϵ. These figures also include the Kullback-Leibler divergences between both distributions, as a reference to compare the value of this metric and the similarity between the perturbations. According to the results, in all the cases the algorithms were able to maintain a probability distribution very close to the original one, the EWTM being the most accurate, the AM the least accurate, and the remaining approaches achieving intermediate results.

It is noteworthy that, in this particular case, the obtained approximations of the target probability distributions are more accurate for the lowest ϵ values tried. This is due to the fact that, for low distortion thresholds, the number of inputs for which the model can be fooled is lower, and therefore, a larger number of inputs remains correctly classified as their ground-truth class, which makes the empirical probability distribution $\hat{\mathcal{P}}(Y)$ closer to the original. However, note that the results obtained for high values of ϵ also represent close approximations of the target distributions, and at the same time, the model is fooled for almost all the input samples.

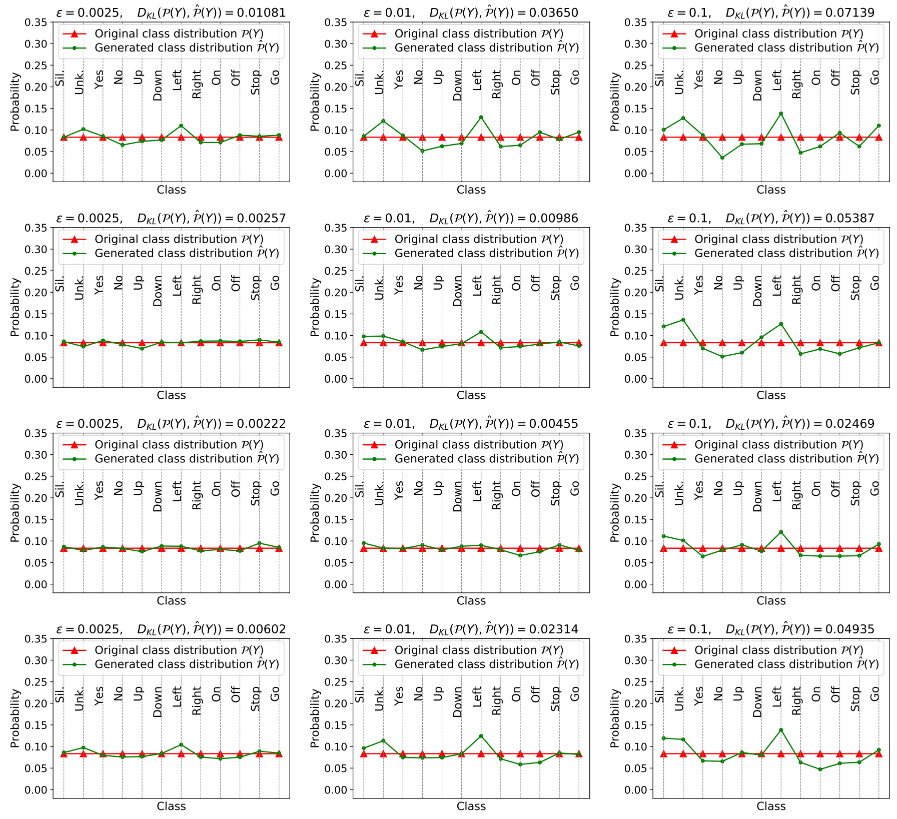


Fig. 3.2: Comparison between the target distribution (in this case the initial probability distribution) $\mathcal{P}(Y)$ and the produced probability distribution $\hat{\mathcal{P}}(Y)$, for the four different methods introduced: AM (first row), UBM (second row), EWTM (third row) and CRM (fourth row). The results are shown for three different values of ϵ, and have been computed for one of the folds of the cross-validation trials. The Kullback-Leibler divergence between both distributions, $D_{KL}(\mathcal{P}(Y), \hat{\mathcal{P}}(Y))$, is also shown above each figure.

3.4.4 Deeper Exploration

In this section, we provide a deeper evaluation of our methods, testing them against 100 probability distributions, randomly drawn from a Dirichlet distribution, as described in Section 3.4.2.
First, we compute the percentage of cases in which the methods managed to generate a valid transition matrix, that is, one which satisfies all the restrictions of the corresponding linear program. This information is shown in Table 3.1, for different values of ϵ. Note that the baselines were not considered for this analysis, since they do not require solving a linear program. In

particular, for each method, the values in Table 3.1 represent the percentage of cross-validation trials in which a valid transition matrix was found, averaged for the 100 target distributions. If a method failed in any of the folds of a cross-validation trial, a failure is reported for the whole cross-validation trial. According to the results, the AM, the UBM and the CRM managed to create a valid matrix for all the cases tried, independently of the distortion threshold. For the EWTM, although it also achieved a total success for values of distortion above or equal to 0.0025, the percentage drops to 38.8% for $\epsilon = 0.0005$. This is due to the larger number of zeros in the matrices $\hat{R}$ for such low distortion thresholds, which makes it impossible to find feasible solutions through an element-wise multiplication with another matrix.

Table 3.1 also includes the success percentages of one variant of the UBM and two variants of the CRM. Regarding the UBM, without relaxing the upper bounds of the transition matrix T, 100% success is achieved for values of distortion $\epsilon > 0.05$, but the percentage of cases for which a valid transition matrix was found drops dramatically for lower values of ϵ. Regarding the CRM, without the Laplace smoothing and without fixing the probabilities $P(\{y_i\}|y_i)$ to zero, the method was not able to generate a valid transition matrix for distortions below 0.05, and even in the maximum distortion threshold tried the method only succeeded in 46.2% of the cases. Applying the Laplace smoothing (without fixing $P(\{y_i\}|y_i) = 0$), those results improve significantly, particularly for the highest distortion thresholds tried, succeeding in more than approximately 80% of the cases for $\epsilon \geq 0.1$, and in 70.6% of the cases for $\epsilon = 0.05$. These results clearly reflect that those corrections are necessary to make the linear programs feasible.

Secondly, the average fooling rates obtained by each method is compared in Table 3.2, for each value of ϵ. In addition, the table includes, for reference purposes, the maximum fooling rate that can be obtained with a maximum distortion ϵ. All the values have been averaged for the 100 target probability distributions considered in the experiment and for the 50 2-fold cross-validations carried out for each of them.[7] The results demonstrate that, whereas, by construction, the MFRB always achieves the optimum fooling rate, the MAB achieves the worst results in the majority of the cases, of approximately 15% below the optimum for $\epsilon \geq 0.005$. In contrast, a very high fooling rate is maintained in the AM, the UBM (for $\epsilon > 0.01$) and the CRM, with a negligible loss with respect to the maximum achievable value. The EWTM, however, achieved slightly lower fooling rates, of approximately 8% below the maximum, independently of the distortion threshold. A similar loss is observed for the UBM when $\epsilon \leq 0.01$.

To conclude the analysis, the average similarity between the target distributions $\widetilde{\mathcal{P}}(Y)$ and the corresponding empirical distributions $\hat{\mathcal{P}}(Y)$ is analyzed

[7] The cross-validation processes in which a method failed in generating a valid matrix T for any of the two folds were discarded, and, therefore, the results might be slightly biased for the EWTM and $\epsilon < 0.001$.

	Maximum distortion amount (ϵ)							
Method	0.0005	0.001	0.0025	0.005	0.01	0.05	0.1	0.15
AM	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
UBM	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
EWTM	38.8	99.9	100.0	100.0	100.0	100.0	100.0	100.0
CRM	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
UBM [1]	0.0	0.0	0.0	9.2	67.9	99.9	100.0	100.0
CRM [2]	0.0	0.0	0.0	0.0	0.0	14.0	30.3	46.2
CRM [3]	10.6	14.0	20.0	25.7	37.5	70.6	79.7	85.3

[1] Without relaxing the upper bound restrictions of T using the auxiliary decision variable η.
[2] Without the Laplace smoothing and without fixing the values of $P(\{y_i\}|y_i)$ to zero.
[3] Without fixing the values of $P(\{y_i\}|y_i)$ to zero.

Table 3.1: Success percentages in generating valid transition matrices for the different methods introduced.

	Maximum distortion amount (ϵ)							
Method	0.0005	0.001	0.0025	0.005	0.01	0.05	0.1	0.15
AM	3.80	11.17	31.58	46.98	62.36	87.29	92.31	94.69
UBM	0.45	2.88	19.06	38.03	57.89	87.05	92.28	94.68
EWTM	1.88	6.87	23.59	38.65	53.60	79.66	85.21	87.84
CRM	3.90	11.29	31.55	46.88	62.23	87.26	92.31	94.70
MAB	2.06	6.55	21.33	33.72	46.87	71.02	76.96	79.64
MFRB	3.93	11.47	32.02	47.44	62.80	87.54	92.48	94.86
Max. FR	3.93	11.47	32.02	47.44	62.80	87.54	92.48	94.86

Table 3.2: Fooling rate (FR) percentages achieved by the different methods introduced.

in Figure 3.3, independently for the different similarity metrics considered and for every maximum distortion threshold. First of all, it is clear that the effectiveness of the methods in reproducing the target distribution increases with the maximum allowed distortion. Apart from that, it can be seen that the AM achieves worse results compared to the rest, thereby validating the hypothesis that the more informed strategies employed in the UBM, EWTM and CRM are capable of increasing the effectiveness of the attack. Indeed, analyzing the results obtained with the remaining methods, the maximum absolute difference between the probabilities of the classes is below 0.09 for

$\epsilon \geq 0.05$, which reflects a very high similarity. In fact, for the mean absolute difference, this value decreases to 0.03. The Kullback-Leibler divergence also shows the same descending trend as the maximum and mean differences. Finally, the Spearman correlation between both distributions is above 0.80 for $\epsilon \geq 0.05$, which indicates that even if there are differences between the values, both distributions are highly correlated.
Comparing the overall effectiveness of the methods in approximating the target distributions, the UBM and the EWTM were the most effective for low and intermediate distortion thresholds ($\epsilon \leq 0.01$), followed by the MAB, while the CRM and the MFRB achieved intermediate results. For high distortion thresholds ($\epsilon \geq 0.05$), in contrast, the EWTM achieved the best results with a notable margin with respect to the other methods, which show a more similar performance.

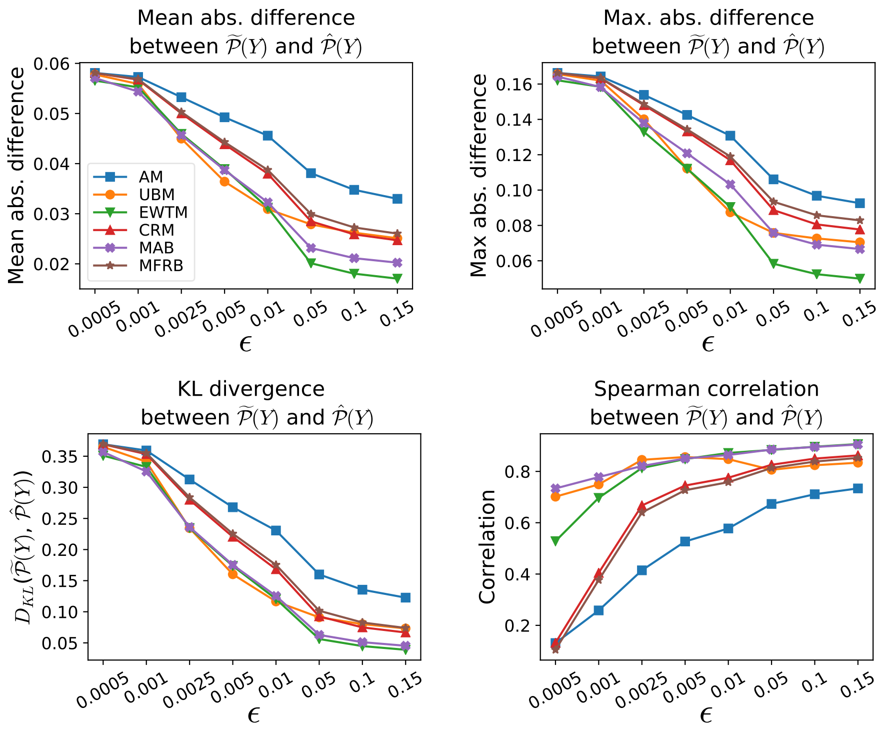


Fig. 3.3: Sensitivity analysis of different similarity metrics between the produced probability distribution $\hat{\mathcal{P}}(Y)$ and the target probability distribution $\widetilde{\mathcal{P}}(Y)$: mean absolute difference, maximum absolute difference, Kullback-Leibler divergence and Spearman correlation.

Comparing our methods with the baselines, on the one hand, the MAB achieves results competitive with the UBM and the EWTM in terms of approximating the target distribution. Nevertheless, it can be noticed that the MAB is outperformed by the EWTM in most cases, and even by the UBM for intermediate values of ϵ, while it is also outperformed in terms of fooling rate by all the remaining methods, with a considerable margin (as shown previously in Table 3.2). Hence, the MAB is dominated by our methods in both factors. On the other hand, whereas the MFRB cannot be outperformed in terms of fooling rate (since it guarantees the optimal value), it is outperformed in terms of the quality of the approximation by our methods. These results corroborate that the proposed methods are capable of taking advantage of the information about the problem provided in order improve their joint effectiveness in the two main goals of the attack: closely approximating the target distribution for the classes while keeping remarkable effectiveness in the objective of fooling the model for any incoming input sample.
Finally, as an overview of the distortion, Table 3.3 shows the average distortion level introduced by the perturbations, in decibels (dB). Following the methodology introduced in previous related works on adversarial perturbations in speech signals [26, 123, 165, 178], the distortion has been computed as

$$dB(x, v) = \max_i 20 \cdot \log_{10}(|v_i|) - \max_i 20 \cdot \log_{10}(|x_i|), \tag{3.14}$$

x being the clean signal and v the perturbation.[8] According to this metric, the lower the value, the less perceptible the perturbation. Even for the highest values of ϵ tried, the mean distortion level is far below -32dB, which is the maximum acceptable distortion threshold assumed in related works [26, 123, 165]. To empirically assess the imperceptibility of the adversarial perturbations, a randomly sampled collection of our adversarial examples can be found in our webpage [9].

ϵ	0.0005	0.001	0.0025	0.005	0.01	0.05	0.1	0.15
dB	-80.69	-78.21	-72.73	-69.13	-65.60	-58.30	-55.82	-54.61

Table 3.3: Average distortion levels introduced by the adversarial perturbations generated, measured in decibels.

[8] Notice that the metric described in Equation (3.14) is used for a post-hoc analysis and not to optimize the adversarial attacks, for which the ℓ_2 norm was used, as described in Section 3.4.2.

[9] `https://vadel.github.io/acpd/AudioSamples.html`

3.4.5 General Comparison of the Introduced Approaches

As a general overview of the effectiveness of the introduced strategies, focusing on the UBM, the EWTM and the CRM, the three of them provided an effective way to find optimal transition matrices, capable of producing the desired target probability distributions. In addition, and considering that the effectiveness of the methods depends on multiple factors, there is no one best method in all the cases. For instance, the EWTM was overall the most effective one in approximating the desired probability distributions, but achieved lower fooling rates than the UBM and the CRM, which achieved values close to the maximum fooling rates.
This can be assessed more clearly in Figure 3.4, in which a graphic comparison of the effectiveness according to the most relevant factors is provided. For a clearer visualization, dominated values (i.e., those corresponding to methods which are outperformed in all factors by at least another method, under the same distortion threshold) have been displayed in white. Moreover, the non-dominated values corresponding to the same distortion threshold have been connected by dashed gray lines. Notice also that some axes are flipped to represent in all the cases that a value is better if it is closer to the bottom-left corner. As can be seen, no method is dominated by the others in all the factors or metrics considered, with the exception of the AM (which is dominated in all the cases) and the MAB (which is dominated in most of the comparisons in which the fooling rate and the similarity metrics are traded-off). Thus, the variety of methods proposed allows us to select the one that best suits the requirements of the adversary, depending on which factors are the most relevant or which are to be optimized the most.

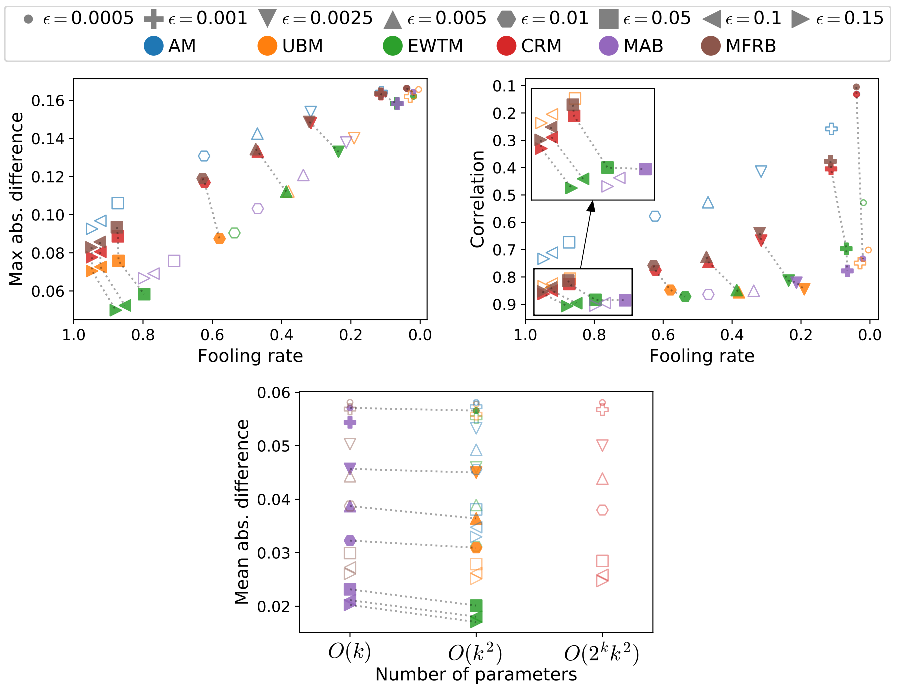


Fig. 3.4: Multifactorial comparison of the effectiveness of the six methods evaluated. For a clearer visualization, dominated values (i.e., those corresponding to methods which are outperformed in all factors by at least another method, under the same distortion threshold) have been displayed in white, whereas the non-dominated values corresponding to the same distortion threshold have been connected by dashed gray lines.

3.5 Counteracting Label-Shift Detection Algorithms in Data Streaming Scenarios

As discussed in Section 3.1, a change in the probability distribution of the classes can lead to a change in the predictive performance of the models or to ethical issues [19, 102, 141, 168]. Therefore, some approaches have been proposed to detect and correct those shifts. In this section, we show the effectiveness of our method in producing a label-shift, even when a label-shift detection method is enforced.

The assumed scenario is as follows. First, we consider a classification model in its deployment phase, which receives a set of unlabeled instances every time unit. We also assume that the initial probability distribution of the classes $\mathcal{P}(Y)$ is known. Finally, we consider the presence of a label-shift detector, which evaluates whether the probability distribution of the classes at the prediction phase, $\mathcal{Q}(Y) = (q_1, \ldots, q_k)$, is different from $\mathcal{P}(Y)$. We assume that this evaluation is done periodically, for instance, after receiving a certain number of new instances. In such a scenario, the goal of our attack will be to maliciously produce a target probability distribution $\widetilde{\mathcal{P}}(Y)$, different from $\mathcal{P}(Y)$, yet preventing the change from being detected by the label-shift detection mechanism. Otherwise, the detection mechanism can alert the user about possible attacks [135] or trigger actions such as retraining or replacing the model. Such actions may force the adversary to recalculate the attack strategy, interrupt the attack process or cause it to fail.

For illustration purposes, we will employ the Black Box Shift Detection (BBSD) approach proposed in [102] as the label-shift detector. This method assumes a realistic scenario in which the probability distribution at prediction time $\mathcal{Q}(Y)$ is unknown, since only unlabeled data is observed, which is a common scenario in practice. To address such scenarios, [102] proposes a methods-of-moments approach to consistently estimate $\mathcal{Q}(Y)$ at prediction time, based on the predictions of the classification model.[10] Once the probability distribution at prediction time is estimated, the shift detection is formulated as a statistical test under the null hypothesis $\mathbf{H_0} : \mathcal{Q}(Y) = \mathcal{P}(Y)$ and the alternative hypothesis $\mathbf{H_1} : \mathcal{Q}(Y) \neq \mathcal{P}(Y)$. As in [135], a Pearson's Chi-Squared test will be used as the statistical test to quantify the significance of the label-shift. We will consider that the null-hypothesis is rejected (i.e., the BBSD method detects a significant shift) when the p-value is below 10^{-5}.

As the underlying task for our experiments, we will consider a Tweet emotion classification problem, which is a popular benchmark in text classification [5, 176], streaming classification [62] and quantification learning scenarios [48, 127], where the probability distribution of the output classes (which might represent, for instance, the overall opinion of the population with respect to a given topic) is of paramount relevance [53]. We selected the *Emotion* dataset proposed in [145], which contains Tweets categorized in 6 emotions: *sadness*, *joy*, *love*, *anger*, *fear* and *surprise*. We also selected a pretrained classifier based on the popular *BERT* language model [36], fine-tuned for this dataset.[11] The resulting model achieves a 92.65% of accuracy in the test set of the Emotion dataset.

As the underlying adversarial attack, we selected the method proposed in [10]. Finally, the Levenshtein Edit Distance [95] between the original and the adversarial text was selected as the distortion metric, normalized by the length of the longest text.[12] We set a maximum distortion threshold of $\epsilon = 0.25$.

Since we assume a label-shift detection mechanism, it is important to note that only those target distributions that are not statistically different from $\mathcal{P}(Y)$ (according to the detection method) can be targeted, to prevent the change

[10] We refer the reader to [102] for further details.

[11] The model is publicly available at: `https://huggingface.co/bhadresh-savani/bert-base-uncased-emotion`.

[12] A randomly sampled collection of our adversarial examples can be found in our webpage: `https://vadel.github.io/acpd/TextSamples.html`.

from being detected. Thus, if a target distribution $\bar{\mathcal{P}}(Y)$ is significantly different from $\mathcal{P}(Y)$, our best option is to find another distribution which, despite being as close to $\bar{P}(Y)$ as possible, will not cause the statistical test to reject the null hypothesis. For instance, such a trade-off can be straightforwardly managed by computing the intermediate distribution

$$\widetilde{\mathcal{P}}(Y) = (1-\tau)\mathcal{P}(Y) + \tau\bar{\mathcal{P}}(Y),\ \tau \in [0,1], \tag{3.15}$$

and finding the maximum value of τ so that $\widetilde{\mathcal{P}}(Y)$ is not significantly different from $\mathcal{P}(Y)$.[13]

To evaluate the effectiveness of our method, we considered three different configurations for the source probability distribution $\mathcal{P}(Y)$. First, a roughly uniform distribution will be tested, similarly to the evaluation in the previous section. Secondly, following a similar approach to [102], we considered two distributions in which a probability p_i is assigned to the i-th class and the remaining probability mass is distributed uniformly among the remaining classes. For our experiments, we will set $p_i = 0.25$ and $i = \{2, 4\}$, and, following the notation of [102], we will refer to these distributions as Tweak-2 and Tweak-4.

For each $\mathcal{P}(Y)$, 1000 random Dirichlet distributions were sampled as the target distributions. We ensured, using the approach described in Equation (3.15), that all of the target distributions are not being identified by the label-shift detector as significantly different from the corresponding source distribution $\mathcal{P}(Y)$.[14] In addition, to generate the transition matrices, we sampled 1000 *training* inputs from the dataset, with a class proportion following $\mathcal{P}(Y)$. The EWTM will be used to optimize the transition matrices in all the cases. Once the transition matrix is generated, its effectiveness will be evaluated on a different set $\hat{\mathcal{X}}$, also composed of 1000 inputs. For the sake of a realistic (and challenging) evaluation, the BBSD will be evaluated in cumulative batches of 100 inputs, and a success will be considered only if, for none of the batches, the detector detects significant differences between the empirical distribution $\hat{\mathcal{P}}(Y)$ and $\mathcal{P}(Y)$.

The results are shown in Table 3.4. As can be seen, our method succeeded in 24.8% to 43.4% percent of the cases depending on the configuration of the source distribution $\mathcal{P}(Y)$, which is a reasonably high percentage considering the presence of a label-shift detection mechanism. Furthermore, in the three cases a high fooling rate was maintained, of approximately 62%, which supposes a loss of approximately 10% in comparison to the maximum fooling rate that can be achieved in each case, which is shown in the fourth column. The fifth column of the table shows the average similarity between $\hat{\mathcal{P}}(Y)$ and $\widetilde{\mathcal{P}}(Y)$ according to the following metrics: the Kullback-Leibler divergence, the maximum absolute difference and the mean absolute difference. Only those

[13] In our experiments, the maximum value of τ was found by means of a binary search on the range $[0, 1]$.

[14] We considered a tolerance of $10^{-4} + 10^{-5}$ during the sampling process.

cases for which the label-shift detector did not detect significant changes were considered. According to the three metrics, our method was capable of closely approximating the target distributions, achieving, for instance, an average Kullback-Leibler divergence of approximately 0.04 in the three cases.
Finally, Figure 3.5 (top row) shows three illustrative label-shifts generated in our experiments, one for each of the source distributions considered (column-wise). The second row of the figure shows, for each case, the evolution in the p-value computed by the BBSD label-shift detector during the attack process, measured for cumulative batches of 100 inputs. For comparison, the p-value has been computed considering i) the adversarial predictions provided by the model when it is attacked, and ii) the original predictions, that is, the ones that would be provided if the model was not attacked. As can be observed, both the target and empirical probability distribution of the classes represent an interpretation that can be considerably different from that of the original distribution. In the first case (left column), in which $\mathcal{P}(Y)$ initially portrays a *uniform* opinion distribution in the population, the adversarially generated distribution portrays a predominantly positive opinion. A similar effect is achieved in the second case (middle column), in which the mode of the distribution is changed from a negative opinion to a positive opinion. Finally, in the third case (right column), the probability assigned to the mode of $\mathcal{P}(Y)$ is further increased (by reducing the probability assigned to some of the other classes), further biasing the distribution in favor of that mode.

$\mathcal{P}(Y)$	Success (%)	FR (%)	Max. FR (%)	Similarity (D_{KL} / Max. / Mean)
Uniform	27.90	60.99	78.30	0.03 / 0.08 / 0.03
Tweak-2	43.40	62.63	77.20	0.05 / 0.10 / 0.04
Tweak-4	24.80	61.87	78.80	0.04 / 0.08 / 0.04

Table 3.4: Attack performance of the EWTM in producing label-shifts in the presence of the BBSD label-shift detection method. The following information is provided, column-wise: source distribution $\mathcal{P}(Y)$, percentage of cases in which the label-shift was not detected by the BBSD, average fooling rate (FR) achieved by the attacks, maximum fooling rate achievable (as reference), and the average similarity between the produced and the target probability distributions. The similarity is reported for three different metrics: Kullback-Leibler divergence (D_{KL}), maximum absolute error and mean absolute error.

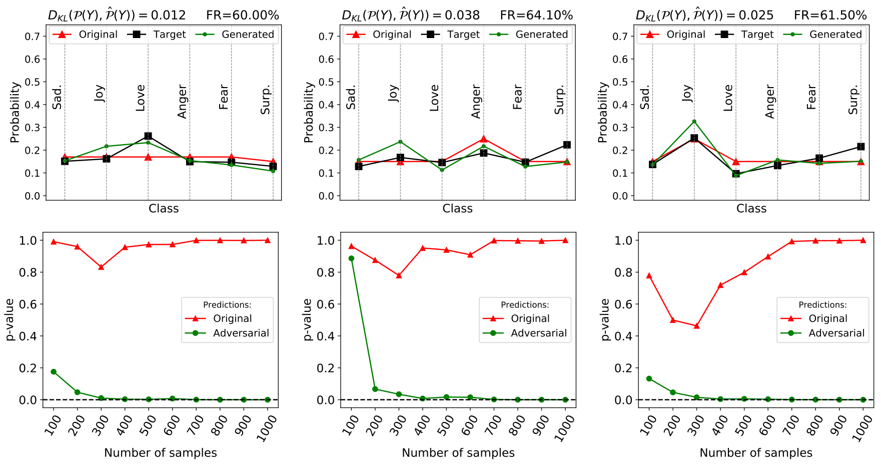


Fig. 3.5: Illustrative label-shifts generated for the Tweet Emotion Classification task (column-wise). The first row provides a comparison of the source, target and produced probability distributions. In each case, the achieved fooling rate (FR) and Kullback-Leibler divergence (D_{KL}) between the target and the generated distributions is shown above the figure. The second row shows, for each case, the evolution of the p-value computed by the BBSD label-shift detector during the attack process, evaluated in cumulative batches of 100 inputs and in both the correct predictions (i.e., when the model is not attacked) and the adversarial predictions (i.e., when our method is applied). The dashed lines mark the detection threshold.

3.6 Conclusions

In this chapter, we have introduced a novel strategy to generate adversarial attacks capable of producing not only prediction errors in ML models, but also any desired probability distribution for the classes when the attack is applied to multiple incoming inputs. This multiple-instance attack paradigm, due to its capability of coordinating multiple attacks to produce more complex malicious behaviors in the models, exposes threats that cannot be conducted by the conventional paradigms, broadening the horizon of adversarial attacks. The proposed attack methodology has been conceived as an extension of targeted adversarial attacks, in which the target class is stochastically selected under the guidance of a transition matrix, which is optimized to achieve the desired goals. We have introduced four different strategies to optimize the transition matrices, which can be solved by using linear programs. Our approach was experimentally validated for the spoken command classification task, using different targeted adversarial attack algorithms as a basis. Furthermore, we also evaluated the success of our methods in preventing the attacks from being detected by label-shift detection methods in a streaming classification scenario. Our results clearly show that the introduced methods are capable of producing close approximations of the target probability distribution for the output classes while achieving high fooling rates.

4

When and How to Fool Explainable Models (and Humans) with Adversarial Examples

4.1 Introduction

As discussed in Chapter 1, DNNs still face several weaknesses that hamper the development and deployment of these technologies, despite their outstanding and ever-increasing capacity to solve complex AI problems. In addition to their vulnerability to adversarial examples, another major shortcoming is their black-box nature, which prevents analyzing and understanding their reasoning process, while such a requirement is ever more in demanded in order to guarantee a reliable and transparent use of AI. To overcome this limitation, different strategies have been proposed in the literature [196], ranging from post-hoc explanation methods, which try to identify the parts, elements or concepts in the inputs that most affect the decisions of trained models [52, 80, 184, 188], to more proactive approaches which pursue a transparent reasoning by training inherently interpretable models [6, 28, 63, 98, 144, 193]. Interestingly, improving the explainability of the models is also a promising direction to achieve adversarial robustness, a hypothesis which is supported by recent works which show that interpretability and robustness are connected [43, 125, 138, 163, 194]. Furthermore, the study of adversarial attacks against explainable models has gained interest in recent years, as will be fully reviewed in Sections 4.2.2 and 4.2.3. In contrast to common adversarial attacks, which focus solely on changing the classification of the model [186], attacks on explainable models need to consider both changes in the classification and in the explanation supporting that classification. Another key difference when considering attacks against explainable models is related to their stealthiness. Generally, the only constraint assumed in order to produce a stealthy attack is that the changes added to the inputs must be imperceptible to humans. However, the use of explainable models implies a different scenario, where it is assumed that a human will observe and analyze not only the input, but also the model classification and explanation. Therefore, uncontrolled changes in both factors may cause inconsistencies, alerting the human. For this reason, the assumption of explainable classification models introduces a new question regarding the definition of adversarial examples: *can adversarial examples be deployed if humans observe not only the input but also the output classification and/or the corresponding explanation?*

4.1.1 Objectives and Contributions

The objective of this chapter is to shed light on this question by extending the notion of adversarial examples for explainable ML scenarios, in which humans can not only assess the input sample, but also compare it to the output of the model and to the explanation. These extended notions of adversarial examples allow us to exhaustively analyze the possible attacks that can be produced by means of adversarially changing the model's classification and explanation, either jointly or independently (that is, changing the explanation without altering the output class, or vice versa). Our analysis leads to a comprehensive framework that establishes whether (and how) adversarial attacks can be generated for explainable models under human supervision. Moreover, we thoroughly describe the requirements that adversarial examples should satisfy in order to be able to mislead an explainable model (and even a human) depending on multiple scenarios or factors which, despite their relevance, are often overlooked in the literature of adversarial examples for explainable models, such as the expertise of the user or the objective of the explanation. Finally, the proposed attack paradigms are also illustrated by adversarial examples generated for two representative image classification tasks, as well as for two different explanation methods. The outline of this chapter is summarized in Figure 4.1.

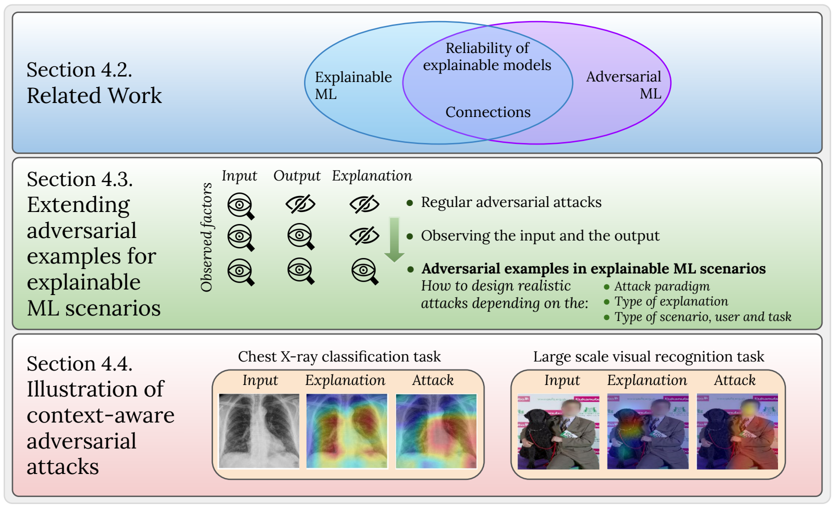

Section 4.2.
Related Work
Explainable ML
Reliability of explainable models
Connections
Adversarial ML
Section 4.3.
Extending adversarial examples for explainable ML scenarios
Observed factors
Input
Output
Explanation
Regular adversarial attacks
Observing the input and the output
Adversarial examples in explainable ML scenarios
How to design realistic attacks depending on the:
Attack paradigm
Type of explanation
Type of scenario, user and task
Section 4.4.
Illustration of context-aware adversarial attacks
Chest X-ray classification task
Input
Explanation
Attack
Large scale visual recognition task
Input
Explanation
Attack

Fig. 4.1: Outline of the exploratory research developed in Chapter 4.

The aim of all these contributions is to establish a basis for a more rigorous study of the vulnerabilities of explainable ML in adversarial scenarios. We believe that these fields will benefit from our work in the following ways.

- Heretofore, studies on adversarial attacks against explainable models have considered very particular or fragmented scenarios and attack paradigms. Thus, there is a lack of a unifying perspective in this field that connects all these works within a general analytical framework and taxonomy, which is a gap that we fill with this work.
- The framework we propose encompasses not only attack paradigms which have already been investigated in the literature, but also paradigms that, to the best of our knowledge, have not yet been studied, paving the way for new research venues.
- In addition, the role of the human is often overlooked in the study of attacks against explainable models, despite being a key factor in these scenarios. In this work, we address this limitation by thoroughly analyzing the requirements that adversarial examples should satisfy in order to be able to mislead an explainable model, and even a human, depending on the attack scenario. This analysis provides a comprehensive road map for the design of realistic attacks against explainable models.
- Furthermore, the fact that our framework considers a wide range of scenarios that an adversary may face allows us to summarize which paradigms are realistic or unrealistic in each of them, which is fundamental to ensure that attack methods are evaluated with an appropriate setting and methodology in future works.
- On another note, our work also contributes to raise awareness about the possible attack types that both models and humans may face in realistic adversarial scenarios, which is important to promote a more aware and secure use of ML based technologies, or even the development of more robust models or explanation methods.

For the above reasons, the aim of this work is to contribute to a more methodical research in this area, delimiting the differences between the possible attack paradigms, identifying limitations in the current approaches and establishing more fine-grained and rigorous standards for the development and evaluation of new attacks or defenses.

4.2 Related Work

This section provides a gentle yet comprehensive introduction to the field of adversarial attacks against explainable models. To begin with, an overview of explanation methods in ML is presented in Section 4.2.1. Subsequently, the reliability of the explanation methods in adversarial scenarios is discussed in Section 4.2.2. Finally, further connections between explanation methods and adversarial examples are discussed in Section 4.2.3.

4.2.1 Overview of Explanation Methods in ML

In this section, we summarize the explanation methods proposed in the literature in order to present the terminology and taxonomy that will be used in the subsequent sections to develop our analytical framework on adversarial examples in explainable models.

4.2.1.1 Scope, Objective and Impact of the Explanations

The objective of an explanation is to justify the behavior of a model in a way that is easily understandable to humans. However, different users might be interested in different aspects of the model, and, therefore, the explanations can be generated for different scopes or objectives.

Overall, the scope of an explanation can be categorized as local or global [196]. On the one hand, local methods aim to characterize or explain the model's prediction for each particular input individually, for example, by identifying the most relevant parts or features of the input. On the other hand, global methods attempt to expose the general reasoning process of the model, for instance, summarizing (e.g., using a more simple but interpretable model) when a certain class will be predicted, or describing to what extent a particular input-feature is related to one class. Since in this chapter we address the vulnerability of explainable models to adversarial examples, we focus on local methods.

In addition, explanations can be used, even for the same model, for different purposes. For instance, users querying the model for a credit loan might be interested in explaining the output obtained for their particular cases only, whereas a developer might be interested in discovering why that model misclassifies certain input samples. At the same time, an analyst can be interested in whether that model is biased against a social group for unethical reasons. At a higher level, all these purposes are based on necessities involving ethics, safety or knowledge acquisition, among others [38]. Based on the purpose of the explanations and the particular problem, domain or scenario in which they are required, another relevant factor should be taken into consideration: the impact of the explanations, which can be defined as the consequence of the decisions made based on the analysis of the explanation. Healthcare domains are clear examples in which the consequences of the decisions can be severe.

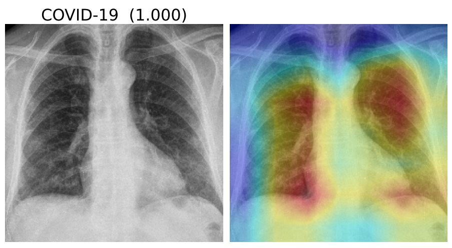
Despite the relevance of these factors, they are often overlooked when local explanation methods are designed or evaluated [38, 196]. The same happens for adversarial attacks in explainable models. We argue that the scope, the objective and the impact of explanations should be key factors when designing adversarial attacks against explainable models, since a different attack strategy needs to be adopted in each context to successfully deceive the model (and the human). This will be discussed in detail in Section 4.3.

4.2.1.2 Types of Explanations

Different types of explanations exist depending on how the explanation is conveyed:

- *Feature-based explanations:* assign an importance score to each feature in the input, based on their relevance for the output classification. Common feature-based explanations (especially in the image domain) are activation or saliency maps [151], which highlight the most relevant parts of the input. Despite their extensive use, previous works have identified that such explanations can be unreliable and misleading [30, 63, 80, 81, 103, 139].
- *Example-based explanations:* the explanation is based on comparing the similarity between the input at hand and a set of *prototypical* inputs that are representative of the predicted class. Thus, the classification of a given input sample is justified by the similarity between it and the prototypes of the predicted class. We will also refer to these types of explanations as *prototype-based explanations* in the dissertation, although different forms of example-based explanation exist, such as the strategy proposed in [83], where influence functions are employed to estimate the training images *most responsible* for a prediction. Recent works have integrated prototype-based explanations directly in the learning process of DNNs, so that the classification is based on the similarities between the input and a set of prototypes [6, 28, 63, 98], achieving a more interpretable reasoning. The prototypes can represent an entire input describing one class (e.g., a prototypical handwritten digit "1" in digit classification) [98], or represent image-parts or semantic concepts [6, 28, 63].
- *Rule-based explanations:* these explanation methods aim to expose the reasoning of a model in a simplified or human-understandable set of rules, such as logic-rules or if-then-else rules, which represent a natural form of explanations for humans [92, 166]. Rule-based explanations are particularly well-suited when the input contains features which are easily interpretable.
- *Counterfactual explanations:* although counterfactual explanations [169] can be considered, in their form, as rule-based explanations, the main difference of these explanations is their conditional or hypothetical reasoning nature, as the aim is suggesting the possible changes that should happen in the input to receive a different (and frequently more positive) output classification (e.g., "a rejected loan request would be accepted if the subject had a higher income").

Some illustrative examples of these four types of explanations are presented in Figure 4.2. Overall, the most suitable type of explanation depends on the domain, the scope and the purpose of the explanation, as well as on the expertise level of the users querying the model. We refer the reader to [196] and [54] for a more fine-grained overview of explanation methods. These surveys also provide an exhaustive enumeration of relevant methods in the literature focused on computing such explanations.



(a) Feature-based explanation

Labrador retriever: 0.949
Chesapeake Bay retriever: 0.030
American Staffordshire terrier: 0.006

(b) Example-based explanation

Income	Age	Job	Country	Gender
x_1	x_2	x_3	x_4	x_5

Output: **Rejected credit loan**

Explanation:

➢ $x_1 < 1500$

➢ and $x_2 < 20$

➢ and $x_3 = \text{None}$

(c) Rule-based explanation

Income	Age	Job	Country	Gender
x_1	x_2	x_3	x_4	x_5

Output: **Rejected credit loan**

Explanation: Would be accepted if:

➢ ($x_1 > 1500$ or $25 < x_2 < 35$)

➢ and ($x_3 \neq \text{None}$)

(d) Counterfactual explanation

Fig. 4.2: Illustrative examples of the four main types of explanations in ML.

4.2.2 Reliability of Explanations Under Adversarial Attacks

Some explanation methods in the literature have proven to be unreliable in adversarial settings. In [7, 37, 51, 88, 195], it is shown that small changes in input samples can produce drastic changes in feature-importance explanations, while maintaining the output classification. In [51], the proposed attacks are also evaluated in the example-based explanations proposed in [83], based on estimating the relevance of each training image for a given prediction by using influence-functions. In [197], adversarial attacks capable of changing the explanations while maintaining the outputs are created for self-explainable (prototype-based) classifiers. In [88, 195], it is shown that adversarial examples can also produce wrong outputs and (feature-importance) explanations at the same time, or change the output while maintaining the explanations [195].

In addition, trustworthy explanations can be produced for a biased or an untrustworthy model, thus manipulating user trust, as shown in [3, 91]. The approaches introduced in these works are, however, not based on adversarial attacks, as they focus on producing a global explanation model that closely approximates the original (black-box) model but which employs trustworthy features instead of sensitive or discriminatory features (which are actually being used by the original model to predict). Similarly, in [152], *adversarial* models are generated, capable of producing incorrect or misleading explanations without harming their predictive performance. In [65], a fine-tuning procedure is proposed to adversarially manipulate models, so that saliency map based explanations drastically change, becoming ineffective in highlighting the relevant regions, whereas the accuracy of the model is maintained.

Some works have also tried to justify the vulnerability of explanation methods to adversarial attacks, or the links between them. In [37, 51], the non-smooth geometry of decision boundaries (of complex models) is blamed, arguing that, due to these properties, small changes in the inputs imply that the direction of the gradients (i.e., normal to the decision boundary) can drastically change. As most explanation methods rely on gradient information, the change in the gradient direction implies a different explanation. In [88, 195], the vulnerability is attributed to a gap between predictions and explanations. It is an open question whether this hypothesis holds for self-explainable models, which have been trained jointly to classify accurately and to provide explanations [6, 28, 63, 98, 144]. Finally, theoretical connections between explanations and adversarial examples are established in [43, 70].

4.2.3 Further Connections Between Adversarial Examples and Interpretability

Paradoxically, using explanations to support or justify the prediction of a model can imply security breaches, as they might reveal sensitive information [153, 167]. For instance, an adversary can use explanations of how a black-box model works (e.g., what features are the most relevant in a prediction) in order to design more effective attacks. Similarly, in this chapter we will show that justifying the classification of the model with an explanation makes it possible to generate types of deception using adversarial examples that, without explanations, it would not be possible to generate (e.g., to convince an expert that a misclassification of the model is correct).

On another note, recent works have shown that robust (e.g., adversarially trained) models are more interpretable [43, 138, 163, 194]. In [43], this is justified by showing that the farther the inputs are with respect to the decision boundaries, the more aligned the inputs are with their saliency maps, thus, being more interpretable. Furthermore, [125] shows that enhancing the interpretability of a model during the training phase increases its adversarial robustness. Moreover, explanation methods have inspired particular defensive strategies against adversarial attacks [68, 74, 105, 137, 160, 172, 182, 191], and, inversely, adversarial attack methods have been proposed as a tool to generate or analyze explanations [41, 61, 113, 130].

Finally, the similarities between interpretation methods and adversarial attacks and defenses are analyzed in [104], showing how adversarial methods can be redefined from an interpretation perspective, and discussing how techniques from one field can bring advances into the other. Our work, however, addresses a different objective. In contrast to [104], which focuses on highlighting the similarities between particular methods from both fields, in this dissertation we propose a comprehensive framework to study if (and how) adversarial examples can be generated for explainable models under human assessment.

4.3 Extending Adversarial Examples for Explainable ML Scenarios

In this section, we extend the notion of adversarial examples to fit in explainable ML contexts. For this purpose, in Section 4.3.1, we start from a basic definition of adversarial examples, and discuss more comprehensive scenarios in which the human subjects judge not only the input sample, but also the decisions of the model. In Section 4.3.2, the human assessment of the explanations is also taken into account. To the best of our knowledge, no prior work has comprehensively addressed this type of generalization of adversarial examples.

This extended definition allows us to provide a general framework that identifies the way in which an adversary should design an adversarial example to deploy effective attacks even when a human is assessing the prediction process. The framework introduced also identifies the most effective ways of deploying attacks depending on factors such as the way in which the explanation is conveyed (Section 4.3.2.1) or the type of scenario, user and task (Section 4.3.2.2). From an adversary perspective, this framework provides a comprehensive road map for the design of malicious attacks in realistic scenarios involving explainable models and a human assessment of the predictions. From the perspective of a developer or a defender, this road map helps to identify the most critical requirements that their explainable model should satisfy in order to be reliable.

4.3.1 Scenarios in Which Human Subjects Are Aware of the Model Predictions

Regular adversarial examples are based on the assumption that an adversary can introduce a perturbation into an input sample, so that:

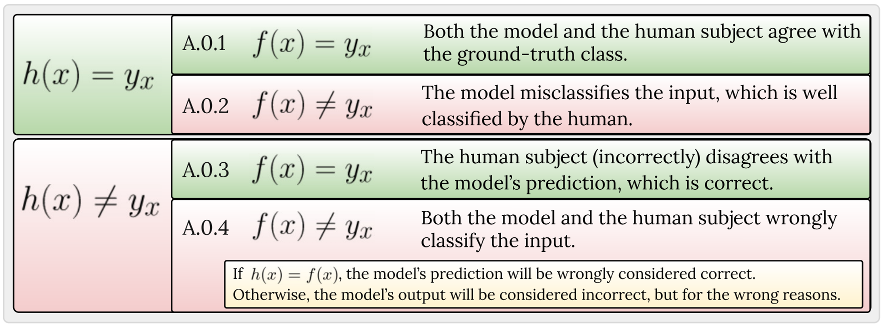
1. The perturbation is not noticeable to humans, and, therefore, the human's perception of which class the input belongs to does not change.
2. The class predicted by a ML model changes.

Note that, according to this definition of adversarial examples, the human criterion is only considered regarding the input sample, without any human assessment of the model's output. However, this definition does not guarantee the stealthiness of the attack in scenarios in which the user observes the output classification, since the change in the output can be inconsistent, alerting the human. For these reasons, the following question arises: *are regular adversarial examples useful in practice when the user is aware of the output?*

To address this question, we start by discussing four different scenarios, based on the agreement of the following factors: $f(x)$, the model's prediction of the input; $h(x)$, the classification performed by a human subject; and y_x, the ground-truth class of an input x (which will be unknown for both the model and the human subject in the prediction phase of the model). For clarification, we assume that a human misclassification ($h(x) \neq y_x$) can occur in scenarios in which the addressed task is of high complexity, such as medical diagnosis [128], or in which the label of an input is ambiguous, such as sentiment analysis [2, 15]. Although a human misclassification might be uncommon in simple problems such as object recognition, even in such cases ambiguous or challenging inputs can be found [155, 162]. Finally, unless specified, we will assume expert subjects, that is, subjects with knowledge in the task and capable of providing well-founded classifications.[1] According to this framework, the four possible scenarios are those described in Figure 4.3.

According to the described casuistry, regular adversarial attacks aim to produce the second scenario (A.0.2, i.e., $f(x) \neq h(x) = y_x$), by imperceptibly perturbing a source input x_s that satisfies $f(x_s) = h(x_s) = y_{x_s}$ (i.e., the first scenario) so that the model's output is changed, but without altering the human perception of the input (which, therefore, implies $h(x) = y_x = y_{x_s}$). However, assuming that the user is aware of the output, the fulfillment of the attack is subject to whether human subjects can correct the detected misclassification, or have control over the implications of that prediction. For example, an adversarial traffic sign will only produce a dramatic consequence in autonomous cars if the drivers do not take control with sufficient promptness.

[1] Different degrees of expertise can be considered for a more comprehensive scenario, such as unskilled subjects, or partially skilled subjects capable of providing basic judgments about the input (for instance, a subject might not be able to visually discriminate between different species of reptiles, yet be able to visually classify an animal as a reptile and not as another animal class).



$h(x) = y_x$	A.0.1	$f(x) = y_x$	Both the model and the human subject agree with the ground-truth class.
	A.0.2	$f(x) \neq y_x$	The model misclassifies the input, which is well classified by the human.
$h(x) \neq y_x$	A.0.3	$f(x) = y_x$	The human subject (incorrectly) disagrees with the model's prediction, which is correct.
	A.0.4	$f(x) \neq y_x$	Both the model and the human subject wrongly classify the input.

If $h(x) = f(x)$, the model's prediction will be wrongly considered correct.
Otherwise, the model's output will be considered incorrect, but for the wrong reasons.

Fig. 4.3: Attack casuistry when the human observes not only the input but also the output classification of the model.

Regarding the remaining cases, they do not fit in the definition of a regular adversarial attack since either the input is misclassified by the human subject ($h(x) \neq y_x$) or the model is not fooled ($f(x) = h(x) = y_x$). Nevertheless, assuming a more general definition, scenarios involving human misclassifications could be potentially interesting for an adversary. Similarly to regular adversarial attacks, which force the second scenario departing from the first one, an adversary might be interested in forcing the fourth scenario departing from the third one. Let us take as an example a complex computer-aided diagnosis task through medical images, in which an expert subject fails in their diagnosis while the model is correct. In such cases, we can induce a human error confirmation attack by forcing the model to confirm the (wrong) medical diagnosis produced by the expert, that is, forcing $f(x) = h(x) \neq y_x$ [21, 45, 55, 75].
Based on the above discussion, we can determine that some types of adversarial attacks can still be effective even when the user is aware of the output. Nonetheless, paradoxically, it is possible to introduce new types of adversarial attacks when the output classification is supported by explanations, as we show in the following section.

4.3.2 Scenarios in Which Human Subjects Are Aware of the Explanations

The scenarios described in the previous section can be further extended for the case of explainable models, as the explanations for the predictions come into play. As a consequence, each of the cases defined above can be subdivided into new subcases depending on whether the explanations match the output class or whether humans agree with the explanations of the models. To avoid an exhaustive enumeration of all the possible scenarios, we focus only on those that we identify as interesting from an adversary perspective. From this standpoint, given an explainable model, adversarial examples can be generated by perturbing a well classified input (for which the corresponding explanation is also correct and coherent) with the aim of changing i) the output class, ii) the provided explanation or iii) both at the same time.

To formalize these scenarios, an explanation $A(x)$ will be defined as a set $A(x) = \{\phi_1, \phi_2, \ldots, \phi_n\}$, where each ϕ_i represents a single explanatory unit which justifies or explains a decision in a human-understandable way. Let us denote $A_f(x)$ as the explanation provided to characterize the decision $f(x)$ of a model, and $A_h(x)$ as the explanation provided by a human according to their knowledge or criteria. The disagreement between $A_f(x)$ and $A_h(x)$, denoted as $A_f(x) \not\approx A_h(x)$, will be formalized as $A_f(x) \cap A_h(x) = \emptyset$, that is, as the lack of common explanatory units in both explanations. The total agreement between the explanations will be denoted as $A_f(x) = A_h(x)$. However, a total agreement is unlikely due to the high number of possible explanations for a given classification. In order to relax this definition, we will consider that there is an agreement between $A_f(x)$ and $A_h(x)$, which will be denoted as $A_f(x) \approx A_h(x)$, when both explanations overlap, that is, $A_f(x) \approx A_h(x) \Leftrightarrow A_f(x) \cap A_h(x) \neq \emptyset$. Similarly, we will denote $A(x) \sim y$ if an explanation $A(x)$ for the input x is consistent with the reasons that characterize the class y (that is, if the explanation correctly characterizes or supports the classification of x as the class y). Formally, $A(x) \sim y \Leftrightarrow A(x) \subseteq A_Y(x)$, being $A_Y(x)$ the set of all explanatory units that justify classifying x as y.

For simplification, unless specified, we assume that given an input x belonging to the class y_x, $h(x) = y_x$ and $A_h(x) \sim y_x$, this is, the human classification of an input into one class is correct and is based on reasons consistent for that class. Similarly, we will also assume that, for a *clean* (unperturbed) input x, $f(x) = y_x$ and $A_f(x) \sim y_x$.

The identified scenarios are as follows:

- A.1: $f(x) = y_x \;\wedge\; A_f(x) \not\approx A_h(x)$. In this case, the model is right but the explanations are incorrect or differ from those that would be provided by a human. Adversarial attacks capable of producing such scenarios have been studied in recent works for post-hoc feature-importance explanations [7, 37, 51, 88, 195] and for self-explainable prototype-based classifiers [197], showing that small perturbations in the input can produce a drastic change in the explanations without changing the output.
  - A.1.1: More particularly, we can imagine a scenario in which $A_f(x) \sim y_x$ despite $A_f(x) \not\approx A_h(x)$, for instance, if $A_f(x)$ points to relevant and coherent properties to classify the input as y_x, but which do not compose a correct or relevant explanation (with respect to the given input) according to a human criterion. From an adversary's perspective, changing the explanations without forcing a wrong classification allows confusing recommendations to be introduced. For illustration, a model can correctly reject a loan request but the decision can be accompanied by a wrong yet coherent explanation (e.g., "the applicant is too young"), preventing the applicant from correcting the actually relevant deficiencies of the request (e.g., "the applicant's salary is too

low") [164]. Similarly, a wrong explanation of a medical diagnosis system might lead to a wrong treatment or prescription [21, 23, 50, 154]. In addition, biased or discriminative explanations could be produced with this attack scheme, for instance, attributing a loan rejection to sensitive features (e.g., gender, race or religion). Such an explanation could make the models look unreliable or untrustworthy for users. Oppositely, biases could be hidden by producing trustworthy explanations to manipulate the trust of the users [3, 91, 152, 171].

- A.2: $f(x) \neq y_x \ \wedge \ A_f(x) \not\approx A_h(x)$. In this case, both the classification and the explanation provided by the model are incorrect. Adversarial attacks capable of producing such scenarios have been investigated in recent works [88, 195]. More particularly, we identify two specific sub-cases as relevant when a human assesses the entire classification process:
  - A.2.1: $A_f(x) \sim f(x)$. In this case, the fact that the provided explanation is coherent with the (incorrectly) predicted class can increase the confidence of the human in the prediction, being therefore interesting from an adversary's perspective. We identify this case as the most direct extension of adversarial examples for explainable models, as the model is not only fooled but also supports its own misclassification with the explanation.
  - A.2.2: $A_f(x) \nsim f(x) \wedge A_f(x) \nsim y_x$. This case is similar to the previous one (A.2.1), with the important difference that the model's explanation is now coherent with a class y' different to $f(x)$ and y_x. Thus, we are in a scenario in which a total mismatch is produced between all the considered factors. Whereas these attacks are an interesting case of study, they are also the most challenging to be deployed in practice without the inconsistencies being noticed.
- A.3: $f(x) \neq y_x \ \wedge \ A_f(x) \approx A_h(x) \ \wedge \ A_f(x) \sim y_x$. In this case, the model's classification is wrong but the provided explanations are coherent from a human perspective with respect to the ground-truth class y_x. The agreement in the explanations can increase the confidence in the model, but, at the same time, the output is not consistent with the explanation. However, the consistency issue might be solved by finding an input for which the explanation not only satisfies $A_f(x) \sim y_x$ but also $A_f(x) \sim f(x)$, for instance, by finding an ambiguous explanation that is applicable to both classes. Such attacks could be employed to convince the user to consider an incorrect class as correct or justified, or to bias the user's decision towards a preferred class (e.g., when there is more than one reasonable output class for an image).

A summary of these scenarios can be found in Table 4.1.

Factors Observed by the User	ID	Classification			Explanation			Attack Category / Description	Representative Examples, Tasks or Use-cases
		Model correct	Human correct	Model-Human agreement	Model coherent with ground-truth	Model coherent with its output	Model-Human agreement		
Input+ Output (Sec. 4.3.1)	A.0.2	✗	✓	✗	–	–	–	Regular attack.	Forcing misclassifications in critical tasks (e.g., traffic-sign recognition, surveillance or finance fraud detection).
	A.0.4	✗	✗	✓	–	–	–	Human error confirmation.	Confirm a wrong diagnosis produced by an expert in health-care domains.
Input+ Output+ Explanation (Sec. 4.3.2)	A.1	✓	✓	✓	*	*	✗	Incorrect explanation (while keeping the correct output).	Reduce human trust in the model.
	A.1.1				✓	✓	✗	Incorrect and coherent explanations (while keeping the correct output).	Confusing recommendations in credit-loan request or medical-diagnosis tasks. Biased or discriminative explanations. Hide inappropriate behaviors of the model.
	A.2	✗	✓	✗	*	*	✗	Incorrect output and explanation.	Reduce human trust in the model.
	A.2.1				✗	✓	✗	Model is wrong but supports its own misclassification.	Increase confidence of the human in the incorrect prediction. Bias the human in favour of a wrong class.
	A.2.2				✗	✗	✗	Total mismatch between the input, the classification and the explanation.	Reduce human trust in the model.
	A.3	✗	✓	✗	✓	✓	✓	Incorrect output while keeping a correct explanation.	Ambiguous explanations applicable to more than one class. Misdirect the attention of the user towards another reasonable class.

Table 4.1: Overview of the attack casuistry described in Sections 4.3.1 and 4.3.2. For the sake of simplicity, we use the following symbols to represent the following terms: ✓ (yes), ✗ (no), – (not applicable). In those paradigms in which subcases are considered, the symbol * is used to represent the term "not specified" (i.e., the choice made for those factors determines the attack subtype).

4.3.2.1 Attack Design Based on the Type of Explanation

Whereas our framework considers the explanations of the models in their most general form, the way in which an explanation is conveyed determines how humans process and interpret the information [38, 196]. This implies that some attack strategies might be more suitable for some types of explanations than for others. Moreover, the way in which an adversarial example is generated for an explainable model will also depend on the type of explanation. For these reasons, in this section we briefly discuss in which way an adversarial example should be designed depending on the type of explanation or the particular type of attack to be produced.

- *Feature-based explanations:* the highlighted parts or features need to be coherent with the classification, and correspond to i) human-perceivable, ii) semantically meaningful and iii) relevant parts. A common criticism to feature-based explanations such as saliency maps is that they identify the relevant parts of the inputs, but not how the models are processing such parts [139]. Thus, an adversarial attack could take advantage of this limitation. First, a particular region of the input can be highlighted to support a misclassification of the model and to convince the user (assuming that the region contributes to predict an incorrect class), which is interesting particularly for targeted adversarial attacks. An attack could also highlight irrelevant parts to mislead the observer, or generate ambiguous explanations, by highlighting multiple regions or providing a uniform map, which are strategies well-suited for untargeted attacks.
- *Prototype-based explanations:* in this case, for the human to accept the given explanation, the key features of the closest prototypes should i) be perceptually identifiable in the given input, and, ideally, ii) contain features correlated with the output class. The contrary should happen for the farthest prototypes, that is, their key features should not be present in the input nor be correlated with the output class (or, ideally, be opposite). In order to achieve these objectives, the more general the prototypes (e.g., if they represent semantic concepts or parts of inputs rather than completely describing an output class), the higher the chances of producing explanations that could lead to a wrong classification while being coherent with a human perception, such as ambiguous explanations.
- *Rule-based explanations* can be fooled by targeting explanations which are aligned with the output of the model (e.g., the explanation justifies the prediction or at least mimics the behavior of the model), but which employ reliable, trustworthy or neutral features [3, 91]. For instance, a model for criminal-recidivism prediction could provide a negative assessment based on unethical reasons, whereas the explanation is taken as ethical [3, 91].
- *Counterfactual explanations:* in this case, the objective of an adversarial attack could be forcing a particular counterfactual explanation, suggesting changes on irrelevant features (thus preventing correcting the deficiencies

which are actually relevant), or forcing biased or discriminatory explanations in detriment to the fairness of the model.

4.3.2.2 Attack Design Based on the Scenario, User and Task

To conclude our framework, we describe the main characteristics or desiderata that an adversarial attack should satisfy in different scenarios in order to be successful. We build on the idea that common tasks, problems or applications share common categories, and that explanations or interpretation needs are different in each of them [38]. Thus, adversarial attacks (or, oppositely, the defensive countermeasures) should also be designed differently for each type of scenario, focusing on the most relevant or crucial factor in each case.
The considered scenarios, summarized in Figure 4.4, comprise different degrees of expertise of the human in supervision of the classification process and different purposes of the explanation. It is important to note that a particular problem or task could belong to more than one scenario. Moreover, we emphasize that some of the scenarios involve factors which are difficult to quantify in a formal way (e.g., the expertise of a user). Nevertheless, we believe that it is necessary to consider such detailed scenarios in order to rigorously discuss which type of adversarial examples can be realizable in practice. In what follows, we describe each scenario and identify the requirements that adversarial attacks should satisfy in order to pose a realistic threat in each of them. This information will be summarized in Table 4.2.
S1 Scenario: The first scenario comprises tasks in which the implications of the decisions made by the model cannot be controlled by the user, or cases in which there is no time for human supervision of the predictions. Despite the relevance of some tasks that fall into this category, such as autonomous driving [46] or massive content filtering [86, 108], humans cannot thoroughly evaluate each possible prediction. For this reason, explanations are not of

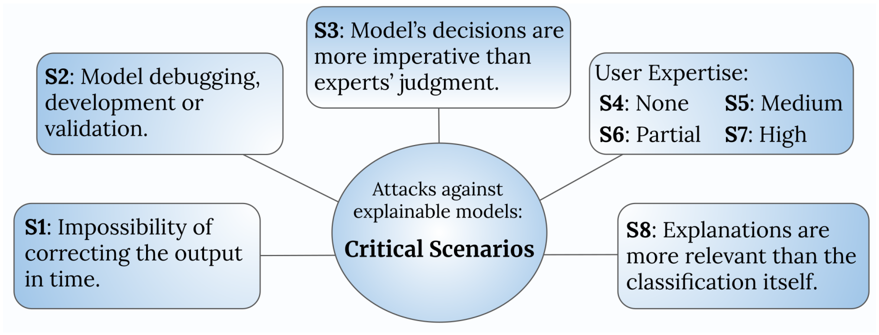

S3: Model's decisions are more imperative than experts' judgment.
S2: Model debugging, development or validation.
User Expertise:
S4: None
S5: Medium
S6: Partial
S7: High
Attacks against explainable models:
Critical Scenarios
S1: Impossibility of correcting the output in time.
S8: Explanations are more relevant than the classification itself.

Fig. 4.4: Critical scenarios to be considered in the study adversarial attacks against explainable models.

Scenario	Representative Examples	Applicable Attacks
S1: Impossibility of correcting the output or controlling the implications of the decision in time.	Fast decision making scenarios (e.g., autonomous cars) or automatized processes (e.g., massive online content filtering).	• Any adversarial attack capable of producing a change in the output class (as the explanations are not of practical use in these cases).
S2: Model debugging, development, validation, etc.	Applicable to any task.	• A.2.1, A.3 (justify misclassifications of the model). • A.1.1 (mask inappropriate behaviors, e.g., hiding biases by producing trustworthy outputs or explanations). • A.2 (produce wrong outputs and explanations jointly).
S3: Decisions of the models are more imperative than experts' judgments.	Risk of criminal recidivism or credit risk management.	
S4: User with no expertise.	Scenarios in which the decision criteria are secret, hidden, or unknown (e.g., banking or financial scenarios, malware classification problems, etc.).	• Any adversarial attack scheme (able to change the classification, the explanation or both at the same time), taking advantage of the user's inexperience.
S5: User with medium expertise (the model is expected to clarify or support the user's decisions).	Challenging scenarios (e.g., complex medical diagnosis) or unforeseeable scenarios (e.g., macroeconomic predictions, risk of criminal recidivism, etc.).	• A.1.1, A.2.1, A.3. • The explanation needs to be consistent with the input patterns and/or consistent with the output class.
S6: User with partial expertise (i.e., expert in some factors but clueless in others).	Hierarchical classification (e.g., large scale visual recognition).	• A.1.1, A.2.1, A.3. • The output and the explanation should be consistent with the factors which are familiar to the user (either regarding input features or the output class).
S7: User with high expertise.	Tasks in which the inputs can be ambiguous (e.g., NLP tasks such as sentiment analysis or multiple object detection in the image domain).	• A.1.1, A.3 (attacks involving generating ambiguous explanations).
S8: Explanations even more relevant than the classification itself.	Predictive maintenance, medical diagnosis or credit/loan approval (e.g., with a wrong explanation users cannot modify or correct the deficiencies).	• A.1, A.2.1 (e.g., maintain the output but produce totally or partially wrong explanations, or produce unethical explanations).

Table 4.2: Possible scenarios in which explainable models can be deployed, and a guideline on how adversarial attacks should be designed in each case in order to pose a realistic threat.

practical use in such cases, so the main (or only) goal of an adversary is to produce an incorrect output.

S2 Scenario: Interpretability or explainability can be desirable properties for ML models (including those developed for the **S1** scenario) in order to debug or validate them [1, 11, 38, 136]. For instance, a model developer might want to explain the decisions of a self-driving car (even if the end-user will not receive explanations when the model is put into practice) to assess why it has provided an incorrect output, to validate its reasoning process or to gain knowledge about what the model has learned [46, 121, 136]. In such cases, an adversary could: justify a misclassification of the model (A.2.1, A.3), hide an inappropriate behavior when the model predicts correctly but for the wrong reasons (A.1.1), or produce wrong outputs and explanations at the same time (A.2).

S3 Scenario: The same attack strategies applicable to the **S2** scenario can be applied in scenarios in which the models' decisions are taken as more relevant or imperative than the experts' judgments. Although this scenario resembles **S1**, the main difference is that, in this case, explanations can be useful or relevant even when the model is deployed or employed by the end-user, and, therefore, the attack should also take the explanations into consideration instead of considering only the output class.

S4 Scenario: Regarding the expertise level of the user querying the model, the case of no expertise is the simplest one from the perspective of the adversary, as any attack scheme can be produced without arousing suspicions, taking advantage of the user inexperience. For the same reason, models deployed in such scenarios should also be the ones with more security measures against adversarial attacks.

S5 Scenario: If the user's expertise is medium, the model might be expected to clarify or support the user's decisions. Thus, the explanation should be sufficiently consistent with the main semantic features in the input (e.g., the user might not be able to diagnose a medical image, but can identify the relevant spots depending on what is being diagnosed, such as darker spots in skin-cancer diagnosis [4]), and/or be sufficiently consistent with the output class (A.1.1, A.2.1, A.3).

S6 Scenario: Similarly to the **S5** scenario, if the user has a partial expertise, that is, if the user is an expert in some factors but clueless in others,[2] then the adversary needs to ensure that the output and the explanations are coherent only with the factors or features that are familiar to the user (A.1.1, A.2.1, A.3).

S7 Scenario: A user with high expertise, by definition, will realize that a model is producing a wrong output or explanation. However, it can be possible to mislead the model and convince the human of a wrong prediction

[2] This could happen in hierarchical classification tasks or large scale visual recognition tasks, as a fine-grained distinction of certain classes might be challenging, whereas the remaining classes are easily classified [35, 63, 124, 140, 150].

by means of ambiguity (A.1.1, A.3). For instance, in an image classification task, two objects can appear at the same time, making it possible to produce a wrong class with a reasonable explanation, for example, by selectively focusing the attention of the explanation on one of the objects or by highlighting the secondary object as the most relevant one [155]. In addition, in problems in which the inputs are inherently ambiguous, such as natural language processing tasks, different but reasonable explanations can be produced for the same input [2, 15].
S8 Scenario: Finally, in some cases the explanations might be more critical, necessary or challenging than the output itself. Some representative tasks are predictive maintenance [147] (e.g., it might be more interesting to know why a certain system will fail than just knowing that it will fail) or medical diagnosis [154] (e.g., discovering why a model has diagnosed a patient as being at high risk for a particular disease might be the main priority to prevent the disease or provide a better treatment). For these reasons, a change in the explanation is critical for such models, which makes them particularly sensitive to the attacks described in A.1, or those described in A.2.1, if the misclassification of the model is difficult to notice by the user.

4.4 Illustration of Context-Aware Adversarial Attacks

In this section, we generate different types of adversarial examples to illustrate the main attack paradigms described in Section 4.3, in terms of both the type of misclassification that wants to be produced (as described in Section 4.3.2) and the "scenario" in which the attack is created (as described in Section 4.3.2.2). To this end, we will consider two representative image classification tasks, assuming an explainable ML scenario. In addition, we will consider two explanation methods, namely feature-based explanations and prototype-based explanations, to illustrate the effect of the attacks in both cases. Our code is publicly available at: `https://github.com/vadel/AE4XAI`.
We remark that the aim of this section is to provide illustrative examples of the attack paradigms described in the proposed framework, and that the focus will be on exemplifying the design of the attacks (i.e., the requirements that they should satisfy in order to pose a legit threat against explainable models) rather than on the methods that could be used to implement them or in their performance. A summary of the illustrated scenarios and the corresponding details is provided in Table 4.3. As can be seen in the table, our illustrations cover all the main attack paradigms and scenarios considered in the framework developed in the previous section.

4.4.1 Selected Tasks, Datasets and Models

We will focus on two image classification tasks to generate the adversarial examples:

- *Medical image classification:* the selected task consists of chest X-ray (CXR) classification, in which the aim is to identify, given an X-ray image, one of the following diseases: Covid-19, (non-Covid) pneumonia or none ("normal"). We used a pretrained Covid-Net model [173], trained on the COVIDx dataset [173], which achieves an accuracy of 92.6%.[3]
- *Large scale visual recognition:* the aim of this task is to classify *real* or *natural* images across a wide range of classes. We selected the ImageNet dataset, which contains images from 1000 different classes such as animals or ordinary objects, and a pre-trained ResNet-50 DNN as a classifier, which achieves a Top-1 accuracy of 74.9%.[4] Both the ImageNet and the ResNet-50 architecture have been widely employed for the study of image classification, as well as in the more particular field of interpretable ML [124, 146, 195].

These two use-cases allow us to illustrate the different scenarios described in Section 4.3.2.2. First, the medical image classification represents a challenging task that requires a high expertise in order to correctly classify inputs or to provide well-funded explanations of the decisions. As discussed, in such a scenario, an adversary has more room to generate adversarial examples that produce incorrect model responses (both in terms of classification and explanation) which, at the same time, may be coherent or acceptable according to a human criterion (particularly for non-expert users). Moreover, the explanations can be critical in this task, as the reason for determining a diagnosis is of high relevance, being therefore representative of the S8 scenario described in Section 4.3.2.2.

Secondly, users with a high expertise can be assumed in the large scale visual recognition task, as the ImageNet dataset contains images containing familiar objects or animals which will be easily recognizable for humans. Thus, a human observing the input as well as the output of the classification should easily detect inconsistencies in the prediction of the model (i.e., whether or not it is correct). However, at the same time, some images might be ambiguous or challenging to classify even for humans (e.g., fine-grained dog breed classification [78, 124]) which therefore can be representative of medium or partial expertise, as the user might be able to effectively discriminate certain classes (e.g., differentiating dogs from other animal species) but not others (e.g., two similar dog breeds). In such cases, the user might expect the prediction of the model or the corresponding explanation to clarify the correct class of the input.

[3] The selected pretrained model (COVIDNet-CXR Small) is accessible at `https://github.com/lindawangg/COVID-Net/blob/master/docs/models.md`.

[4] More information about the pretrained model used can be found at `https://keras.io/api/applications/resnet/`.

Task	Type of Explanation	Possible Scenario	Wrong class	Wrong explanation	Attack description	Figure
X-ray (Sec. 4.4.4)	Feature-based (saliency map)	S2, S3, S4/S5/S6, S8	✗	✗	No attack (i.e., original input)	4.5-(a)
			✓	✓ (conflicting)	Regular attack (i.e., without controlling the explanation)	4.5-(b)
			✓	✗ (non-conflicting)	A.3	4.5-(c)
			✗	✓	A.1.1 (confusing recommendation)	4.5-(d)
			✓	✓ $A_f(x) \sim f(x)$ (non-informative but consistent, i.e., supports prediction)	A.2.1	4.5-(e)
			✓	✓ $A_f(x) \nsim f(x)$ $A_f(x) \nsim y_x$ (non-informative and inconsistent)	A.2.2	4.5-(f)
Large-Scale Visual Recog. (Sec. 4.4.5)	Feature-based (saliency map)	S2, S5/S6	✗	✗	No attack (i.e., original input)	4.6-(a)
			✓	✗	A.3	4.6-(b), 4.6-(c), 4.6-(d)
		S2, S7	✗	✗	No attack (i.e., original input)	4.7-(a)
			✗	✗	No attack (the output is further biased in favour of the correct class, avoiding ambiguities)	4.7-(b)
			✓	✓	A.2.1	4.7-(c)
			✓	✗	A.3	4.7-(d)
	Prototype-based explanation (3 nearest training inputs)	S2, S5/S6	✓	✗ $A_f(x) \sim y_x$ $A_f(x) \sim f(x)$ (ambiguous)	A.3	4.8-(a), 4.8-(b)

Table 4.3: Summary of the illustrative attacks shown in Sections 4.4.4 and 4.4.5. Notice that each attack paradigm and scenario is exemplified at least once. Note also that for the large scale visual recognition task different scenarios can be considered depending on the characteristics or the challengingness of the input.

4.4.2 Explanation Methods

We will consider two representative explanation methods in order to illustrate an explainable ML scenario:

Feature-based explanation The Grad-CAM method [146] will be used to generate saliency-map explanations. The rationale of this method is to employ the activation maps computed by the model in the last convolutional layer to produce the explanations. Given a CNN f and an input x, the Grad-CAM saliency-map S is defined as:

$$S = ReLU\left(\sum_{m}^{M} \alpha^{m,c} \cdot C^{m}\right), \tag{4.1}$$

where C^m, $m = 1, \ldots, M$, represents the (two-dimensional) m-th activation map (for the input x) at the last convolutional layer of f, and $\alpha^{m,c} \in \mathbb{R}$ represents the importance of the m-th map in the prediction of the class of interest y_c (typically $f(x)$, i.e., the class predicted by the model). The importance $\alpha^{m,c}$ of each activation map is estimated as the average global pooling of the gradient of the output score (corresponding to the class y_c) with respect to C^m, which will be denoted as $G^{m,c} = \nabla_{C^m} f(x)_c$:

$$\alpha^{m,c} = \sum_{i}\sum_{j} G^{m,c}_{i,j}, \tag{4.2}$$

where $G^{m,c}_{i,j}$ denotes the value at the i-the row and j-th column. The ReLU non-linearity in (4.1) is applied to remove negative values, maintaining only the features with a positive influence on y_c.

Example-based explanation We will also consider an example-based explanation in which the n_p training images (which can be considered prototypes representing classes) that are closest to the input which has been classified are provided [124]. The proximity between the inputs will be measured as the Euclidean distance of the d_l-dimensional latent representation $r_f(x) : \mathbb{R}^d \to \mathbb{R}^{d_l}$ learned by the model f in the last layer, that is, the (flattened) activations of the last convolutional layer of the model. This representation captures complex semantic features of the inputs, thus providing a more appropriate representation space for meaningfully comparing input samples according to the features learned by the model. Let X^c_{train} represent the set of *training* inputs belonging to the class of interest y_c (e.g., the class predicted by the model). Given a model f and an input x, the explanation will be a set $P \subseteq X^c_{train}$, with $|P| = n_p$, that satisfies:

$$||r_f(\hat{x}) - r_f(x)||_2 \ > \ ||r_f(x^p) - r_f(x)||_2, \quad \forall \hat{x} \in X^c_{train} - P, \ \forall x^p \in P. \tag{4.3}$$

Note that the two selected methods allow, by definition, explanations to be computed for any class of interest y_c. However, we will consider as the *main* explanation the one corresponding to the predicted class $f(x)$. Finally, we assume that the explanation methods and their parameters are fixed and known to the adversary. Since the focus of our experimentation is illustrative and not performance-based, analyzing the sensitivity of the explanation methods to hyperparameters [14, 37] will be out of the scope of this section.

4.4.3 Attack Method

We will assume a targeted attack for our experiments, in which the aim will be to create, given an input x, an adversarial example x' such that:

$$f(x') = y_t, \tag{4.4}$$

$$A_f(x') = \xi_t, \tag{4.5}$$

$$||x - x'|| \leq \epsilon, \tag{4.6}$$

where y_t represents a target class, ξ_t a target explanation and ϵ the maximum distortion norm. For the case of saliency-map explanations, ξ_t will be a predefined saliency-map S_t. For the case of prototype-based classification, ξ_t will be the set P_t of n_p training inputs (with the value of n_p fixed beforehand by the explanation method) selected by the adversary to be produced as explanations (that is, the training inputs of class y_t that are closer to x should be those in the set P_t). We do not specify any particular order for the n_p target-prototypes in P_t, that is, we assume that the relevance of each of the n_p prototypes in the explanation is the same.
We will use a targeted Projected Gradient Descent (PGD) attack [107] to generate the adversarial examples. A detailed description of this attack can be found in Section 2.3.2.2. In order to produce attacks capable of changing both the classification and the explanation, we will consider a generalized loss function

$$\mathcal{L}(x, y_t, \xi_t, \tau, f) = (1 - \tau) \cdot \mathcal{L}_{pred}(x, y_t, f) + \tau \cdot \mathcal{L}_{expl}(x, \xi_t, f), \tag{4.7}$$

where $\mathcal{L}_{pred}(x, y_t, f)$ represents the classification error with respect to the target class y_t, $\mathcal{L}_{expl}(x, \xi_t, f)$ represents the explanation error with respect to the target explanation ξ_t and $\tau \in [0, 1]$ balances the trade-off between both functions. A close approach can be consulted in [195]. In our experiments, we used the cross-entropy loss as $\mathcal{L}_{pred}$. For the case of saliency-map explanations, we instantiated $\mathcal{L}_{expl}$ as the Euclidean distance between the model's explanation $g(x, f) = S$ and the target saliency map S_t (specified by the adversary):

$$\mathcal{L}_{expl}(x, S_t, f) = ||g(x, f) - S_t||_2. \tag{4.8}$$

For the case of prototype-based explanations, $\mathcal{L}_{expl}$ will be the average Euclidean distance between the latent representation of the (adversarial) input and the latent representation of the n_p prototypes selected by the adversary as the target explanation P_t:

$$\mathcal{L}_{expl}(x, P_t, f) = \frac{1}{n_p} \sum_{x^p \in P_t} ||r_f(x) - r_f(x^p)||_2. \tag{4.9}$$

4.4.4 Illustrative Attacks in the X-ray Classification Task

Figure 4.5 illustrates the results obtained for different adversarial examples generated against the COVID-Net model. The left part of each sub-figure shows the input sample, the model's classification of the input and the confidence score of the prediction, whereas the right part shows the saliency-maps generated with the Grad-CAM explanation (darker-red parts represent a higher relevance). Figure 4.5-(a) shows the original (i.e., unperturbed) input sample, which is correctly classified as its ground-truth class "COVID-19".[5]
Figure 4.5-(b) shows an adversarial example generated using a regular PGD attack (considering only changing the output class, i.e., $\mathcal{L} = \mathcal{L}_{pred}$), targeting the class "normal". Notice that the main parts of the explanation have changed to the central part and to the rightmost part of the image, highlighting mainly irrelevant zones. Therefore, such an explanation might not be taken as consistent. Contrarily, Figure 4.5-(c) shows an adversarial example generated using the loss function described in (4.7), targeting the class "normal" and the original saliency-map (i.e., the one obtained for the original input image), illustrating the attack paradigm A.3 described in Section 4.3.2. We clarify that using a regular adversarial attack might not necessarily imply a change in the explanation (or imply that the explanation, even if changed, will necessarily highlight irrelevant zones). Nevertheless, this example allows us to illustrate the need to control the explanation in order to create adversarial examples that are capable of convincing the human that the model's (mis)classifications and the corresponding explanations are coherent or consistent, as discussed in Section 4.3.
Figure 4.5-(d) illustrates the attack paradigm A.1.1 described in Section 4.3.2, in which the original class is maintained whereas a change in the explanation is produced, in this case, selectively highlighting some regions of the image. Here, we generated the attack setting the target map as the right-half side of the original saliency-map, and setting the left-part values as zero. Such an attack strategy can be extremely concerning for those scenarios in which the explanation is of high relevance, as a misleading adversarial explanation might lead to an incorrect diagnosis, prescription or treatment.
Figure 4.5-(e) illustrates the attack paradigm A.2.1. Notice that both the output class and the explanation are changed. Moreover, the target map set in this case represents a roughly uniform map over the most relevant parts of the image (in this case the two lungs). Therefore, the provided explanation can be taken as coherent, as the main parts of the image are taken into consideration for the prediction. The fact that the predicted class is "normal" also increases the coherence of the explanation, since it can be interpreted from it that the most critical areas are correct (i.e., that there is no evidence

[5] Disclaimer: the authors acknowledge no expertise in CXR classification, and, as the dataset does not contain a ground-truth saliency-map explanation, it will be assumed for illustration purposes that the explanation achieved for the original input is coherent and correct.

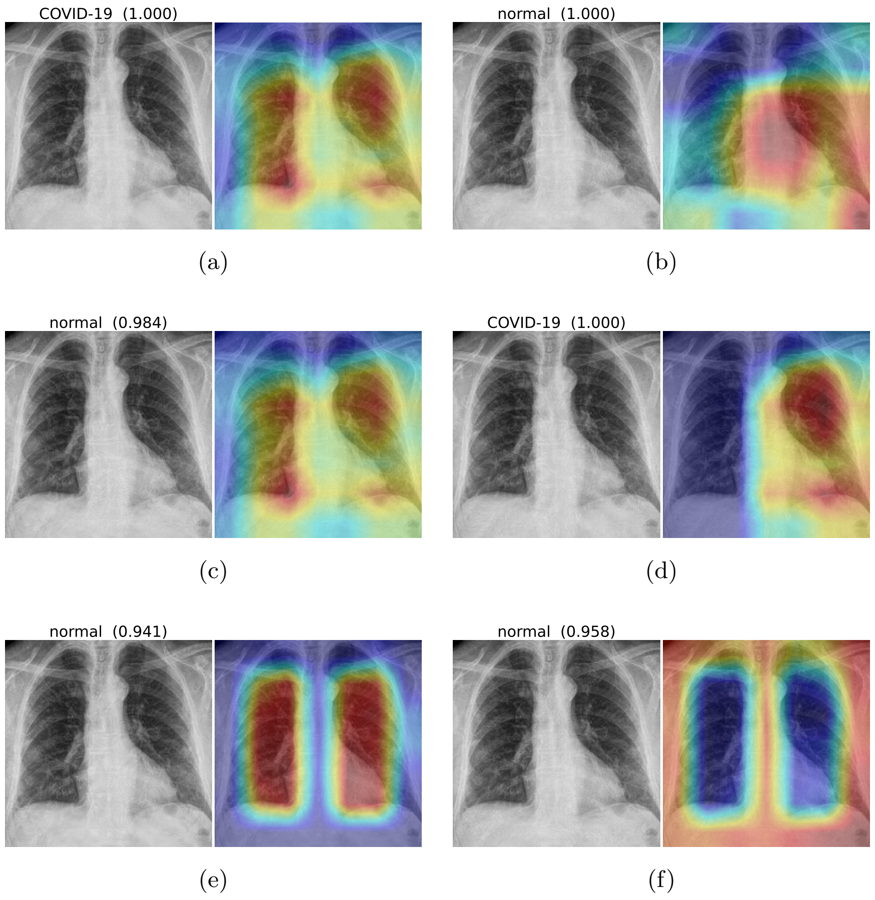
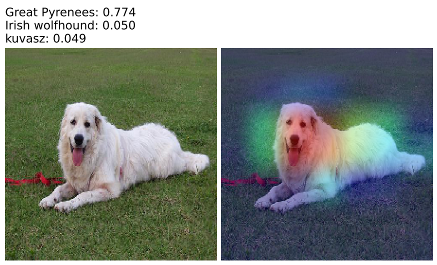


Fig. 4.5: Different types of adversarial attacks for the X-ray medical image diagnosis task. The left part of each image shows the input image as well as the class assigned by the model (jointly with the confidence score in the $[0, 1]$ range), whereas the right part shows the explanation provided by the Grad-CAM method. (a) Original image. (b) Regular adversarial attack (PGD) targeting the class “normal” (i.e., the possible changes that the adversarial perturbation may produce in the explanation are not controlled by the attack). (c) Attack producing the wrong classification “normal” while maintaining the original explanation. (d) Attack maintaining the correct classification while changing the explanation in order to selectively highlight some parts (the right part) but omitting others (in this case, the left part). (e) Attack producing the wrong class “normal” and a wrong explanation which uniformly highlights the relevant parts of the image. (f) Attack producing the wrong class “normal” and a uniform explanation outside the main parts of the image (i.e., highlighting only irrelevant and incorrect parts).

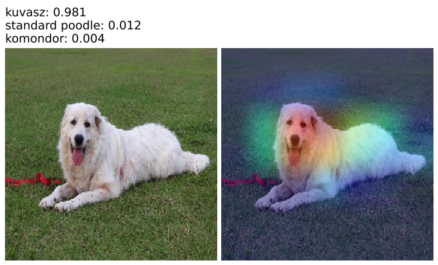
in those areas of a possible disease). Indeed, the same explanation would have a different effect if the prediction had represented a disease (e.g., if the original class had been maintained in this case). This is because a uniform explanation would not provide a precise justification of why the disease is predicted, thus hampering a proper diagnosis, but, at the same time, is coherent with the input features (because the most relevant parts are highlighted), contributing to the user's acceptance that the model prediction is correct. Contrarily, in Figure 4.5-(f) the target map is the opposite: roughly all the relevant parts are considered as not relevant, whereas the remaining regions are considered as relevant, illustrating a case in which the explanation is completely wrong. Since the prediction is also incorrect, and is not supported by the explanation, a total mismatch is produced between all the considered factors, exemplifying the attack paradigm A.2.2.

4.4.5 Illustrative Attacks in the Large Scale Visual Recognition Task

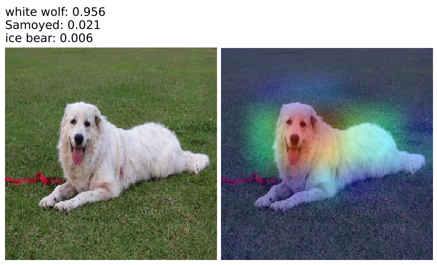
In this section, we illustrate different types of adversarial examples generated taking advantage of class ambiguity. First, in Figure 4.6, the similarity between different classes is used to generate adversarial examples capable of producing a misclassification that could be considered as *coherent* or *reasonable* even for humans. These examples illustrate the attack paradigm A.3 described in Section 4.3. In particular, each attack is generated by setting a target class for which the inputs belonging to that class contain very similar features to those inputs belonging to the source class. Figure 4.6-(a) shows the original input sample used to create the adversarial examples, the top-3 predictions of the model and the corresponding Grad-CAM explanation. Figures 4.6-(b), 4.6-(c) and 4.6-(d) show the adversarial examples targeting the classes, "Kuvasz", "White wolf" and "Labrador Retriever", respectively. These classes represent different dog breeds with very similar features, as is shown in Figures 4.6-(e), 4.6-(f), 4.6-(g) and 4.6-(h), in which the prototypes (for each of the classes) that are closer to the original input image are shown. In all the cases, the saliency map targeted in the attack is the one obtained for the original input (i.e., we maintain the original explanation while changing the classification).
In Figure 4.7, a different type of ambiguity will be considered to generate the adversarial examples: the appearance of multiple concepts or classes in the image. In such cases, adversarial examples can be employed to change the focus of the classification to one of the objects of interest. The explanation is, therefore, a key factor in order to further support the model decision in classifying the input as the class of interest selected by the adversary. The input in Figure 4.7-(a) contains two classes that could be equally relevant: "Curly-coated retriever" dog breed (ground-truth class) and "suit". In Figures 4.7-(b) and 4.7-(c), adversarial examples are generated in order to "untie" this ambiguity, maximizing the confidence of one of the classes ("Curly-coated retriever" and "suit", respectively) and changing the explanation to highlight



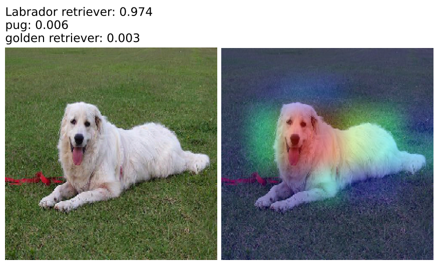
(a) Original input



(b) Target class: Kuvasz



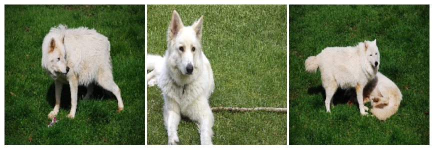
(c) White wolf



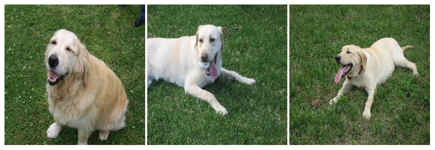
(d) Labrador Retriever

(e) Closest “Great Pyrenees” inputs

(f) Closest “Kuvasz” inputs

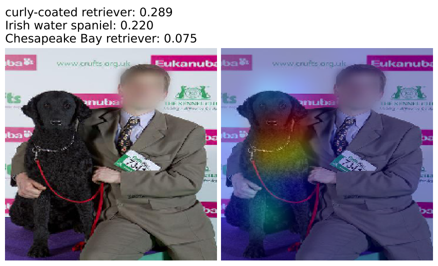


(g) Closest “White wolf” inputs

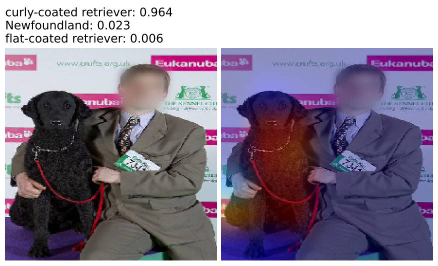


(h) Closest “Labrador Retriever” inputs

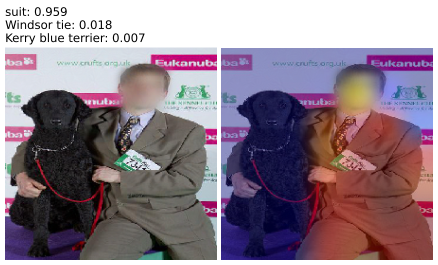
Fig. 4.6: Adversarial examples generated for the ImageNet dataset classification task taking advantage of class ambiguity. The adversarial examples are generated from the input in (a)-left, which belongs to the source class “Great Pyrenees”, targeting different classes that are characterized by features similar to those of the source class: (b) “Kuvasz”, (c) “White wolf”, and (d) “Labrador Retriever”. Each adversarial example is created ensuring that the saliency-map explanation of the original input, shown in (a)-right, is maintained. (e)-(h) show, for each of the four classes considered (source class + 3 target classes), the $n_p = 3$ prototypes closest to the original input, in order to assess their similarity.


curly-coated retriever: 0.289
Irish water spaniel: 0.220
Chesapeake Bay retriever: 0.075

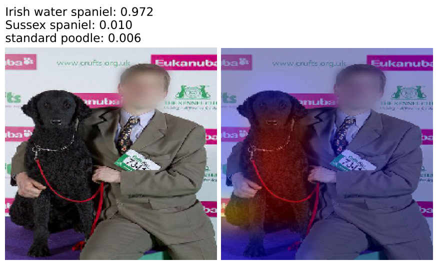
(a) Original input



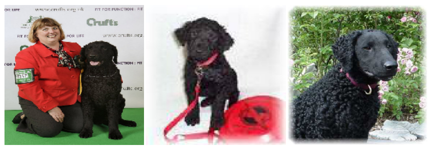
(b) Target class: "Curly-coated retriever"



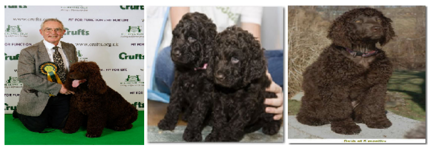
(c) Target class: "Suit"



(d) Target class: "Irish water spaniel"



(e) "Curly-coated retriever"



(f) "Irish water spaniel"

Fig. 4.7: Adversarial examples generated for the ImageNet dataset classification task, taking advantage of the class ambiguity introduced by the appearance of multiple concepts in the image. (a) Original input. (b) Input perturbed in order to maximize confidence in the original class without altering the enhanced region in the explanation. (c) Adversarial example targeting the class "Suit" and a target saliency-map highlighting the region in which this class appears. (d) Adversarial example targeting the class "Irish water spaniel" and a target saliency-map highlighting the region in which the ground-truth class ("Curly-coated retriever") appears. (e) & (f) The 3 training images belonging to the class "Curly-coated retriever" and "Irish water spaniel", respectively, which are closest to the original input.

the selected parts (paradigm A.2.1).[6] As can be seen, the adversarial examples effectively focus the prediction on one of the classes, which can therefore bias the human interpretation of the result, accepting the prioritized output class as the dominant one. Whereas this type of attacks are limited to the objects appearing in the image, different types of ambiguity can be considered at the same time to produce misclassifications that may be taken as "correct" for humans, as shown in Figure 4.7-(d), in which the focus is not only placed on the dog, but also an incorrect class is produced ("Irish water spaniel") taking advantage of the ambiguity of the class similarity (paradigm A.3). This ambiguity is, indeed, also reflected in the output confidence scores provided by the model when the original input is classified, shown above the left part of Figure 4.7-(a), as both classes achieved a similar score. In order to assess the similarity between these two classes, figures 4.7-(e) and 4.7-(f) show the 3 prototypes belonging to each of the two classes that are the closest to the original image, in which it can be seen that both breeds contain very similar features.

Finally, in Figure 4.8, we provide an illustrative example of an attack designed to fool a model whose decisions are explained using a prototype-based explanation. As discussed in Section 4.3.2.1, an adversary can take advantage of prototype-based explanations to support certain misclassifications, for instance, producing an incorrect output class and minimizing the distance with prototypes which, apart from containing features representative of the source class, are representative of the target class as well. Figure 4.8-(a) shows a well-classified image (left) and an adversarial example targeting the class "Doberman". The adversarial perturbation has been optimized in order to reduce the distance (in the latent representation) between the input and the 3 training images (belonging to the target class) shown in Figure 4.8-(b). As can be seen, these training images not only contain features representative of the target class ("Doberman"), but also additional features that resemble those in the original input sample (indeed, a similar dog is present in the selected training images), exemplifying the attack paradigm A.3. Figure 4.8-(c) shows the prototypes belonging to the source class that are the closest to the original image, and 4.8-(d) those prototypes closest to the original image yet belonging to the target class. Note that both Figures 4.8-(b) and 4.8-(d) contain prototypes belonging to the target class, however, those which are adversarially produced appear considerably more coherent due to their ambiguity (in the sense that they contain prototypical features of both the source and target class).

[6] In this case, the target saliency maps have been generated using an image-segmentation model (Mask R-CNN with Inception Resnet v2), which has been used to segment the two desired parts. The pretrained model is accessible at `https://tfhub.dev/tensorflow/mask_rcnn/inception_resnet_v2_1024x1024/1`. Note that, in Figure 4.7-(b), both the classification and the explanation are preserved, thus no attack is carried out.

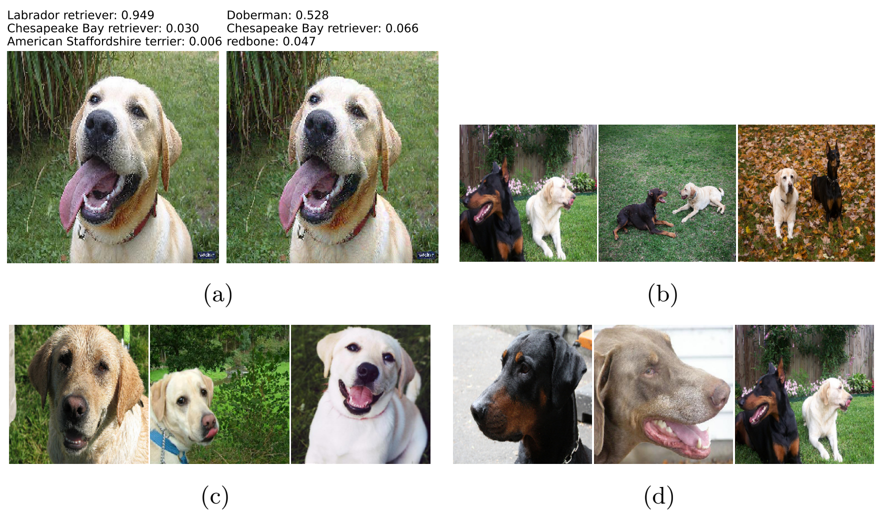

Labrador retriever: 0.949
Chesapeake Bay retriever: 0.030
American Staffordshire terrier: 0.006
Doberman: 0.528
Chesapeake Bay retriever: 0.066
redbone: 0.047
(a)
(b)
(c)
(d)

Fig. 4.8: Adversarial example for the large scale visual recognition task, assuming a prototype-based explanation. (a) Original input belonging to the class “Labrador Retriever” (left) and adversarial example targetting the class “Doberman” (right). (b) Prototype-based explanation of the adversarial example and the class “Doberman” (that is, the 3 training images belonging to the class “Doberman” that are closest to the adversarial example). (c) Prototype-based explanation of the original input and the ground-truth class. (d) Prototype-based explanation of the original input and the target class “Doberman”.

4.5 Conclusions

In this chapter, we have introduced a comprehensive framework to rigorously study the possibilities and limitations of adversarial examples in explainable ML scenarios, in which the input, the predictions of the models and the explanations are assessed by humans. First, we have extended the notion of adversarial examples in order to fit in such scenarios, which has allowed us to examine different adversarial attack paradigms. Furthermore, we thoroughly analyze how adversarial attacks should be designed in order to mislead explainable models (and humans) depending on a wide range of factors such as the type of task addressed, the expertise of the users querying the model, as well as the type, scope or impact of the explanation methods used to justify the decisions of the models. Furthermore, the introduced attack paradigms have been illustrated using two representative image classification tasks and two different explanation methods based on feature-attribution explanations and example-based explanations. Overall, the proposed framework provides a comprehensive road map for the design of malicious attacks in realistic scenarios involving explainable models and a human supervision, contributing to a more rigorous study of adversarial examples in the field of explainable ML.

5

Analysis of Dominant Classes in Universal Adversarial Perturbations

5.1 Introduction

As described in Section 2.3, universal adversarial perturbations [114] are input-agnostic perturbations capable of fooling a DNN while remaining imperceptible for humans. These perturbations are generally created as *untargeted* attacks, so that no preference over the (incorrect) output class is assumed [79, 114, 118, 165]. However, previous work [16, 32, 66, 114] has reported a phenomenon regarding the effect of universal perturbations in the attacked model: the preference of the perturbation to change the class of the inputs into a particular *dominant* class, without this being specified or imposed in the generation of the perturbation. Thus, some classes (or class regions in the decision space) act as *attractors* under the effect of universal perturbations.
In this chapter, we carry out, for the first time, an in-depth study of this phenomenon with the aim of sheding light on the (still misunderstood) vulnerability of DNNs to universal perturbations. The main contributions of this chapter are summarized as follows:

- First, we propose a number of hypotheses to explain and characterize the existence of dominant classes linked to universal adversarial perturbations, and revisit previous hypotheses and open questions in the related work.
- We experimentally test the proposed hypotheses using a speech command classification task in the audio domain as a testbed. To the best of our knowledge, this is the first work in which the analysis of dominant classes is studied for the audio domain. Apart from providing evidence of the validity of the proposed hypotheses, our results reveal interesting properties of the DNN sensitivity to novel types of perturbations, such as perturbations optimized to prevent the main dominant classes.
- Overall, our study exposes the connection between the dominant classes and the sensitivity of the model to i) patterns in the data distribution that the model recognizes as each class with high confidence, and ii) to *vulnerable* directions in the decision space learned by the model. Our findings also

suggest novel approaches to generate universal perturbations, opening the venue for future research on more effective attacks and defenses.

- Finally, we highlight a number of differences between the image domain and the audio domain regarding the analysis of adversarial examples, contributing to a more general understanding of adversarial ML.

5.2 Related Work

Universal adversarial perturbations for DNNs were introduced in [114] for image classification tasks. The goal of such perturbations is to fool a DNN for "most" natural inputs when they are applied to them, and, at the same time, to be imperceptible for humans. Formally, following the notation used in [115], a perturbation v is said to be (ϵ, δ)-universal if the following conditions are satisfied:

$$||v||_2 \leq \epsilon, \tag{5.1}$$

$$P_{x \sim \mathcal{P}(X)} \left[f(x+v) \neq f(x) \right] \geq 1 - \delta, \tag{5.2}$$

being $\mathcal{P}(X)$ the distribution of natural inputs in the d-dimensional input space $\mathbb{R}^d$, and $f(x)$ the output class assigned to an input x by a classifier $f : \mathbb{R}^d \rightarrow \{y_1, \ldots, y_k\}$. Thus, universal perturbations generalize *individual* (i.e., *input dependent*) adversarial perturbations [59, 90, 107, 116, 158], which are optimized to fool a DNN for one particular input of interest.
In the seminal work of Moosavi-Dezfooli et al. [114], an iterative procedure is proposed to generate the universal perturbations. This procedure accumulates *input dependent* perturbations [116] generated for a set of inputs, and projects the universal perturbation after every update in order to bound its norm. Subsequent works have proposed alternative approaches to generate universal adversarial perturbations, such as training generative networks to learn a distribution of universal adversarial perturbations (which, therefore, can be used to sample universal perturbations) [64, 119, 129], or *data-free* approaches capable of generating universal perturbations without any access to the data used to train the target models [117, 118, 120, 190]. Other works pursue more particular objectives, such as generating targeted universal perturbations which change the classification of the model to one predefined label [60, 129, 190], or perturbations that only fool the model for inputs of one particular class [189]. Finally, although image classification tasks have been the main focus of study, universal perturbations have also been reported for tasks such as image segmentation [110, 117], speaker recognition [97], speech recognition [123, 165] or text classification [16, 170].
The discovery of such attacks for state-of-the-art DNNs has led to a deeper study of their properties. In [114], the vulnerability of DNNs to universal perturbations is empirically studied in the image domain, which is attributed in part to the geometry of the decision boundaries learned by the DNNs. In particular, it is shown that, in the vicinity of natural inputs, perturbations normal

to the decision boundaries are *correlated*, in the sense that they approximately span a low dimensional subspace (in comparison to the dimensionality of the input space). Thus, being

$$v_x = \underset{v}{\operatorname{argmin}} ||v||_2 \quad \text{s.t.} \quad f(x) \neq f(x+v) \tag{5.3}$$

the minimal perturbation capable of changing the output of an input x (hence *normal* to the decision boundary at $x + v_x$), it is possible to find a subspace $X_S \subset X$, with $dim(X_S) \ll dim(X)$, so that $v_x \in X_S$ for $x \sim \mathcal{P}(X)$. The existence of such a subspace implies that even random perturbations (with small norms) sampled from X_S are likely to cause a misclassification for a large number of inputs [114]. This hypothesis is further developed in [115], also for the image domain, where the vulnerability of classifiers to universal perturbations is formalized, under the assumption of locally linear decision boundaries in the vicinity of natural inputs. An illustration of a linear approximation of the decision boundary is shown in Figure 5.1 (left).

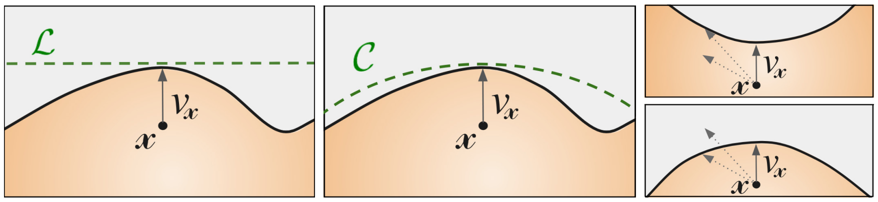


Fig. 5.1: Illustration of the decision boundary approximations introduced in [115]. The left image illustrates the locally linear (flat) decision boundary model, and the middle figure the locally curved decision boundary model. The solid curve corresponds to the actual boundary, and the dashed lines to the approximations. Note that in both cases the approximations are estimated at $x + v_x$, being x an input sample and v_x a vector *normal* to the decision boundary (see Equation 5.3). The right images compare a positively curved boundary (bottom) with a negatively curved boundary (top) along v_x. Two dashed arrows have been included as reference in both images, to highlight that positively curved boundaries require smaller norms to be surpassed.

However, the assumption of locally linear decision boundaries becomes insufficient to comprehensively formalize the vulnerability of DNNs to universal perturbations. Indeed, there is a crucial connection between that vulnerability and the curvature of the decision boundaries [115]: there exist common perturbation directions (i.e., span a low-dimensional subspace) in the input space for which, starting from natural inputs, the decision boundaries are positively curved along these directions. See Figure 5.1 (right) for a comparison between a positively curved boundary and a negatively curved boundary. The positive curvature of the decision boundaries implies small upper bounds for the amount of perturbation required to surpass the decision boundaries, as depicted in Figure 5.1 (right). Thus, those positive curvatures increase the vulnerability of DNNs, as smaller perturbations are required to fool the model. At the same time, the fact that those directions are also *common* for multiple inputs implies the existence of small *input-agnostic* adversarial perturbations. In a further analysis developed in [73], it is shown that the directions in the input space for which the decision boundaries are highly curved are indeed associated by the DNN with class identities (the further we move in one of such directions, the higher - or lower- the confidence of the model in one particular class is). Moreover, it is shown that the class *features* associated to such directions are, indeed, the most relevant ones as far as the classification performance of the model is concerned, what links the accuracy of DNNs with their vulnerability to adversarial attacks. A feature-perspective is also employed in [190] to justify the vulnerability of the models to universal perturbations, experimentally showing that universal perturbations contain features which predominate over the features of natural images. Thus, in the presence of universal perturbations, natural images act like noise, despite being visually predominant.

The aforementioned theoretical frameworks focus, in particular, on the vulnerability to universal perturbations. In this chapter, we focus instead on one particular property of universal perturbations: the existence of *dominant* classes that are significantly more frequently predicted for the perturbed (and misclassified) inputs. This phenomenon was first reported in [114] for image classification tasks. Subsequent works have also reported the existence of dominant classes in image classification tasks [32, 66], and in text classification tasks [16]. Here, we show that this happens also for other domains, such as speech command classification tasks in the audio domain. Although it is hypothesized in [114] that a possible explanation for the *dominant* classes is that they occupy a larger region in the decision space, it is left as an open research question. In the following sections, we tackle this research question and test multiple hypotheses in the search for a deeper understanding of this phenomenon.

Outside the particular field of universal perturbations, multiple theoretical frameworks have been proposed for the explanation of adversarial examples. Whereas most of them focus on the properties of the DNNs [59, 156, 158, 159], other alternative explanations have also been proposed. In this dissertation, special attention is paid to the one introduced in [72], in which adversarial examples are explained in terms of the *robustness* of the features in the data. In particular, it it shown that datasets contain non-robust features which, although being highly discriminative (i.e., that the data is well described by these features), are uncorrelated with the ground-truth classes when they are perturbed by small (adversarial) perturbations. Thus, when a classifier learns to rely on such non-robust features to accurately classify the data, it becomes vulnerable to adversarial perturbations. The small robustness of such features to small perturbations also implies their lack of meaning for humans, which explains the imperceptibility of the attacks. In a similar vein, we hypothesize (Section 5.5.2) that the higher sensitivity of the model to certain features might explain the existence of dominant classes.

5.3 Proposed Framework

Let us consider a classifier $f : X \to Y$, with $X \subseteq \mathbb{R}^d$ and $Y = \{y_1, \dots, y_k\}$, trained to classify inputs $x \in X$ coming from an input data distribution $x \sim \mathcal{P}(X)$ among one of the k possible classes in Y. To formally describe *dominant classes*, let us denote p_j^v the probability of misclassifying an input as the class y_j when a universal perturbation v is added to the inputs:

$$p_j^v = P_{\substack{x \sim \mathcal{P}(X) \\ f(x) \neq y_j}} \left[f(x+v) = y_j \right]. \tag{5.4}$$

Similarly, let $t_{i,j}^v$ represent the probability that, departing from an input of ground-truth y_i, the model incorrectly predicts the class y_j for the perturbed inputs:

$$t_{i,j}^v = P_{\substack{x \sim \mathcal{P}(X) \\ f(x) = y_i}} [f(x+v) = y_j]. \tag{5.5}$$

In practice, if the distribution $\mathcal{P}(X)$ is unknown, these probabilities can be estimated using a finite set of input samples $\mathcal{X}$.

Definition 1. *y_a is an attractor class for another class y_i $(i \neq a)$, under a perturbation v, which will be denoted as $y_i \xrightarrow{v} y_a$, if at least the $\alpha > \frac{1}{k-1}$ proportion of the inputs corresponding to the class y_i are predicted as y_a when they are perturbed with v, that is:*

$$t_{i,a}^v \geq \alpha. \tag{5.6}$$

Notice that the threshold $\frac{1}{k-1}$ represents the proportion that would be achieved if the inputs were evenly distributed among the $k-1$ possible incorrect classes.

Definition 2. *y_b is a dominant class for the universal perturbation v if at least the $\beta > \frac{1}{k-1}$ proportion of the inputs are wrongly classified as y_b when they are perturbed with v, that is:*

$$p_b^v \geq \beta. \tag{5.7}$$

Alternatively, y_b can be defined also in terms of the number of classes that it attracts. Let $Y_b^v = \{y_i \in Y \mid y_i \xrightarrow{v} y_b\}$ represent the set of classes attracted by y_b with the perturbation v, and $|Y_b^v|$ the cardinality of the set Y_b^v. Precisely, y_b is dominant if it is an attractor class for at least the $\zeta > \frac{1}{k-1}$ proportion of the remaining classes:

$$\frac{|Y_b^v|}{k-1} \geq \zeta. \tag{5.8}$$

The choice of the parameters α, β and ζ can determine the existence of multiple attractor and dominant classes. In this dissertation, we assume $\alpha, \beta, \zeta \geq \frac{1}{3}$ since we are interested in those classes which are incorrectly predicted for a significant proportion of inputs, or which attract a significant proportion of other classes.

To study the relationship between universal perturbations and dominant classes, we use the speech command classification problem described in Section 3.4.1 as a testbed. To complement the information on the employed classifier, its architecture and feature extraction process will be further detailed below, as this information will be relevant for the following sections. Firstly, following previous publications [8, 39, 56, 100, 175], the model is based on the architecture proposed in [142], which is composed of two convolutional layers with ReLU activations, a fully connected layer and a final softmax layer. The normalized audio waveforms (in the time-domain) from the input space $\mathbb{R}^{16000}$, which take values in the range $[-1, 1]$, are first converted into spectrograms by dividing the audios into frames of 20ms, with a stride of 10ms, and applying the real-valued fast Fourier transform (retrieving 512 components) for each frame. As the frequency spectrum of a real signal is Hermitian symmetric, only the first 257 components are retained. The dimension of the resulting spectrogram is 99×257. Finally, the MFCCs [122] are extracted from the spectrogram, in the space $\mathbb{R}^{99\times 40}$, before being sent to the network. It is worth pointing out that the adversarial perturbations that are generated for this model are optimized in an end-to-end fashion, directly in the audio waveform representation of the signal.

In addition, to generate the universal perturbations, we selected the UAP-HC algorithm introduced in [165]. This algorithm, which is a reformulation for the audio domain of the one proposed in [114], consists of iteratively accumulating individual untargeted adversarial perturbations, generated using the DeepFool algorithm. The pseudocodes for both the UAP-HC and DeepFool algorithms can be found in Algorithm 2 and Algorithm 3, respectively. These algorithms have been generalized to (optionally) prevent them from reaching certain adversarial classes. This generalization will be further described and motivated in Section 5.4.

Finally, we highlight that the rationale of the DeepFool algorithm relies on a geometric approach. In particular, as discussed in Section 2.3.2.3 a first-order approximation of the decision boundaries is used to move the input towards the estimated closest boundary, being, therefore, an untargeted attack. Thus, the optimization process of the UAP-HC algorithm is not biased towards any particular class, although, in practice, different universal perturbations lead in most of the cases to the same dominant classes.

Algorithm 2 UAP-HC [165]

```
Require: A classification model $f$, a set of input samples $\mathcal{X}$, a projection
    operator $\mathcal{B}_{p,\epsilon}$, a fooling rate threshold $\delta$, a maximum number of iterations
    $I_{\max}$, a set of restricted classes $Y_R \subset Y$
Output: A universal perturbation $v$
 1: $v \leftarrow$ initialize with zeros
 2: $FR \leftarrow 0$    ▷ Fooling rate.
 3: $iter \leftarrow 0$    ▷ Iteration number.
 4: while $FR < 1 - \delta \wedge iter < I_{\max}$ do
 5:     $\mathcal{X} \leftarrow$ randomly shuffle $\mathcal{X}$
 6:     for $x^{(i)} \in \mathcal{X}$ do
 7:         ▷ Check that $x^{(i)}$ is not already fooled by $v$:
 8:         if $f(x^{(i)} + v) = f(x^{(i)})$ then
 9:             $\triangle v \leftarrow$ DeepFool($x^{(i)} + v$, $f$, $Y_R$)
10:             $v' \leftarrow \mathcal{B}_{p,\epsilon}(v + \triangle v)$    ▷ Project $(v + \triangle v)$ in the $\ell_p$ ball of radius $\epsilon$
                and centered at 0.
11:             $FR' \leftarrow P_{x \in \mathcal{X}} [f(x) \neq f(x + v')]$
12:             ▷ Update $v$ only if adding $\triangle v$ increases the FR and if the current
                class is not in $Y_R$:
13:             if $FR < FR' \wedge f(x^{(i)} + v + \triangle v) \notin Y_R$ then
14:                 $v \leftarrow v'$
15:                 $FR \leftarrow FR'$
16:             end if
17:         end if
18:     end for
19:     $iter \leftarrow iter + 1$
20: end while
```

Algorithm 3 DeepFool [116]

```
Require: An input sample $x$ of class $y_i$, a classifier $f$, a set of restricted
    classes $Y_R \subset Y$.
Output: An individual perturbation $r$.
 1: $x' \leftarrow x$
 2: $r \leftarrow$ initialize with zeros
 3: $Y' \leftarrow Y - (Y_R \cup \{y_i\})$
 4: while $f(x') = y_i$ do
 5:     for $y_j \in Y'$ do
 6:         $f'_j \leftarrow \hat{f}(x')_j - \hat{f}(x')_i$
 7:         $w'_j \leftarrow \nabla \hat{f}(x')_j - \nabla \hat{f}(x')_i$
 8:     end for
 9:     $l \leftarrow \text{argmin}_{j \in Y'} \frac{|f'_j|}{||w'_j||}$
10:     $r \leftarrow r + \frac{|f'_l|}{||w'_l||_2^2} w'_l$
11:     $x' \leftarrow x + r$
12: end while
```

5.4 Dominant Classes in Speech Command Classification

In this section, we generate different universal adversarial perturbations for the speech command classification task specified in Section 5.3, in order to investigate whether in this domain dominant classes are also produced.
We start by generating 10 different universal perturbations using the UAP-HC algorithm, without restricting any class ($Y_R = \varnothing$). We set $\epsilon = 0.1$ as threshold for the perturbation ℓ_2 norm, and restricted the UAP-HC algorithm to a maximum of five iterations. To generate the perturbations, we used a *training* set of 100 inputs per class, which makes a total of 1200 inputs. Once the perturbations are generated, their effectiveness will be measured in a *test* set, containing samples that were not used during the generation of the perturbations. The initial accuracy of the model in this set is 85.52%.[1]
According to the results, the algorithm led to universal perturbations with *left* and *unknown* as dominant classes for almost all the experiments. This can be seen in Figure 5.2 (top), which shows the frequency with which each class is wrongly predicted when the perturbation is applied to the audios in the test set. We only considered those inputs that were initially correctly classified by the model, but misclassified when the perturbation is applied. The frequencies are shown individually for the ten universal perturbations, with each row corresponding to one perturbation. As can be seen, both *left* and *unknown* arise as dominant classes in 9 of the 10 experiments, sometimes even at the same time.
It is important to highlight that dominant classes arise without being imposed in the universal perturbation crafting procedure. For this reason, an interesting property to study is whether dominant classes remain dominant even if we explicitly avoid them during the optimization process (see Algorithms 2 and 3). To shed light on this question, we start by preventing the algorithm from considering those directions that point to the decision boundaries of the class *left*. The results obtained for ten new perturbations generated with this restriction are shown in Figure 5.2 (bottom left). As can be seen, the most frequent adversarial class is now *unknown* for 9 of the 10 perturbations created.
We went another step further and repeated the experiment, this time, however, restricting the boundaries corresponding to both *left* and *unknown* classes. The results are shown in Figure 5.2 (bottom right). In this case, the two restricted classes were no longer dominant classes, but different dominant classes were obtained, precisely, *up*, *right* and *go*. It is also worth emphasizing that, although dominant classes were obtained in all the experiments, they were different depending on which other classes were restricted. For instance, whereas the class *up* rarely appeared as dominant without restrictions, it is the most frequent dominant class when both *left* and *unknown* classes are restricted.

[1] The number of samples per class in the test set and the accuracy of the model in each class is reported in Table B.1.

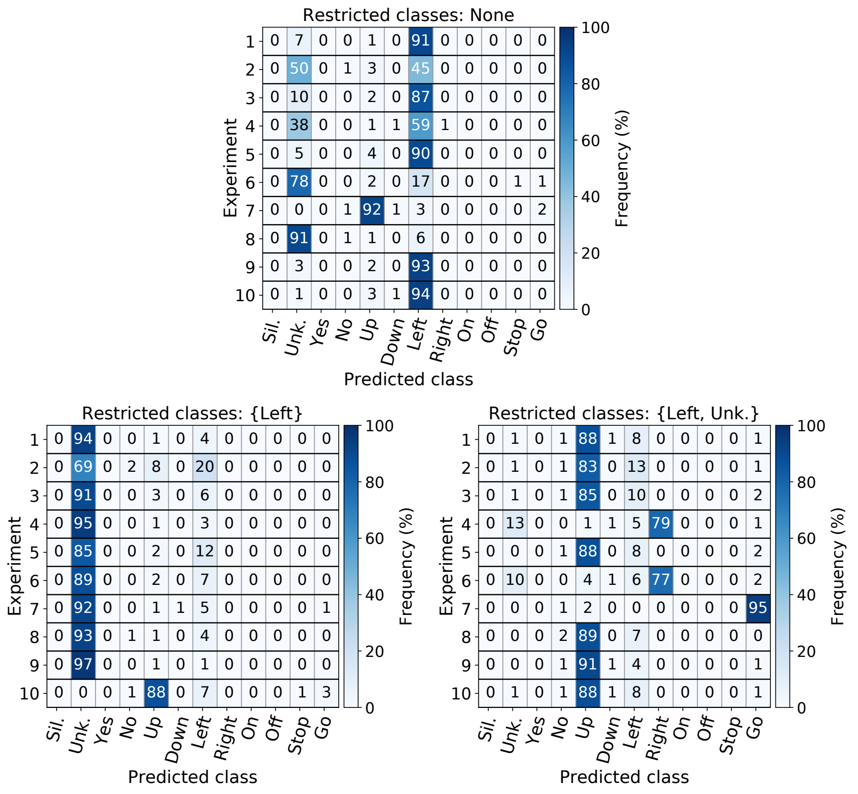


Fig. 5.2: Overview of the frequency with which each class was assigned to the inputs misclassified as a consequence of universal perturbations. The frequencies have been computed individually (row-wise) for the 10 perturbations generated in each of the following configurations of the UAP-HC algorithm: default algorithm (top), restricting the algorithm to follow the class *left* (bottom left) and restricting the algorithm to follow the classes *left* and *unknown* (bottom right).

Regarding the effectiveness of the attacks, the fooling rate of every perturbation (i.e., the percentage of inputs that are misclassified when the perturbation is applied) is shown in Figure 5.3, for each class independently.[2] The fooling rates have been computed considering the inputs that were initially correctly classified. As can be seen, the effectiveness of each perturbation is higher in some classes than in others, achieving up to ≈69% in some cases. The fooling rates corresponding to the dominant classes, which have been highlighted in the figure, are practically zero for most of the perturbations, which reveals

[2] The average effectiveness of each perturbation can be consulted in Table B.2.

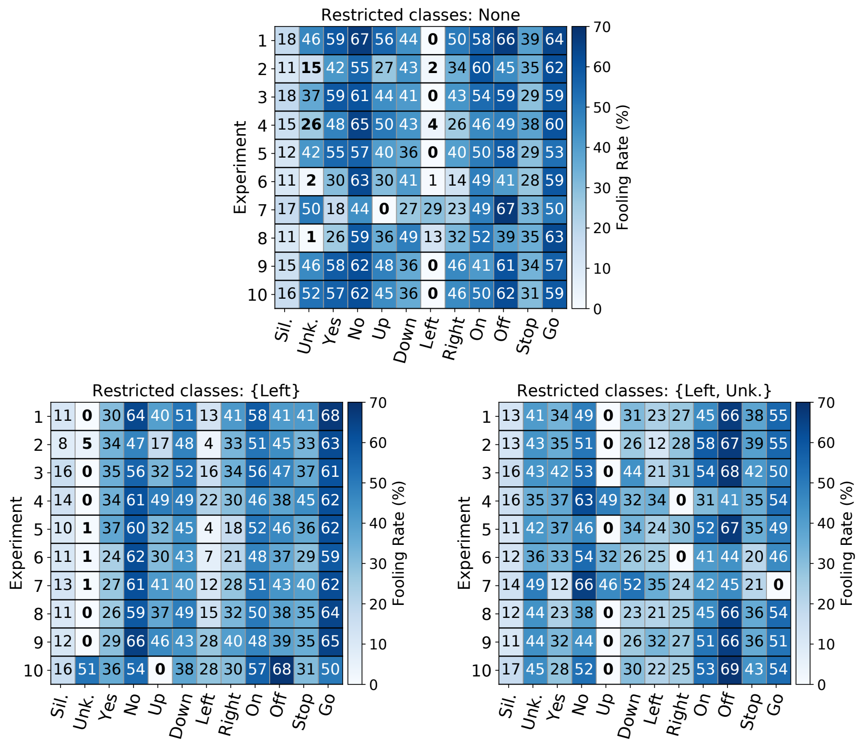


Fig. 5.3: Fooling rate percentage, computed individually for each class, of the 10 perturbations generated in each of the following configurations of the UAP-HC algorithm: default algorithm (top), restricting the algorithm to follow the class *left* (bottom left) and restricting the algorithm to follow the classes *left* and *unknown* (bottom right). In the three figures, the results corresponding to the dominant classes (for each experiment) have been highlighted using bold text.

that the perturbation does not change the prediction of the model for those inputs.
For more informative results, the mean and maximum fooling rate of all the perturbations are shown in Table 5.1. To avoid biases, these aggregated fooling rates have been computed in three different ways: i) considering all the inputs, ii) without considering the inputs corresponding to the dominant classes, and iii) without considering the dominant classes and the class *silence*. The reason for not considering the inputs belonging to the dominant classes is because the perturbation reinforces the confidence on those classes, and, as a consequence, there are practically no misclassifications in those inputs. On the contrary, the results for the class *silence* are clearly lower than for the rest of the classes, which biases the results. Comparing the average effectiveness of the universal

Restricted classes in UAP-HC	Fooling Rate					
	Considering all the classes		w/o considering dominant classes		w/o considering dominant & *Silence*	
	Mean	Max.	Mean	Max.	Mean	Max.
None	37.94	46.34	41.68	50.84	44.97	54.76
{*Left*}	34.90	37.73	37.39	40.60	40.32	43.71
{*Left*, *Unk.*}	33.75	37.49	37.08	41.36	39.90	44.37

Table 5.1: Effectiveness of the UAP-HC algorithm in a set of *test* samples, not seen during the generation of the perturbations.

perturbations, we can notice that the average fooling rate achieved by the perturbations decreases when the dominant classes are restricted in the UAP-HC algorithm. We confirmed using the Wilcoxon signed-rank test [177] (with a significance level of 0.05) that, in comparison to the results obtained when no class is restricted (i.e., $Y_R = \varnothing$), the decrease is significant when the set of classes $Y_R = \{Left\}$ or $Y_R = \{Left, Unknown\}$ is restricted. According to the same test, the differences observed between the cases in which the sets of restricted classes are $Y_R = \{Left\}$ and $Y_R = \{Left, Unknown\}$ were not statistically significant.
Overall, these results confirm the existence of dominant classes in audio tasks, and reveal a number of properties that, to the best of our knowledge, have not been reported before in related works. First, we have shown that it is possible to prevent one class from being dominant during the optimization of the universal perturbation. However, doing so leads to different dominant classes. Moreover, the fact that the effectiveness of the universal perturbations decreases when the *most frequent* dominant classes are restricted might suggest that some classes are more *dominant* than others. All these findings and properties will serve as a basis to further study the cause of this phenomenon in the following sections.

5.5 Hypotheses About the Existence of Dominant Classes

In this section, we propose a number of hypotheses to explain and characterize the relationship between universal adversarial perturbations and dominant classes. The proposed hypotheses are also experimentally tested using the framework described in Section 5.3.

5.5.1 Dominant Classes Occupy a Larger Region in the Input Space

In [114], the existence of dominant classes is attributed to a larger region of such classes in the image space. Nevertheless, due to the high dimensionality of the input spaces in current ML problems, exploring the volume that each decision region occupies in the whole input space is intractable in practice.
Even so, to test this hypothesis, we randomly sampled and classified 1,000,000 inputs from the input space. The values of the inputs were sampled uniformly at random in the range $[-1, 1]$. We found that all the samples were classified as the class *silence*, which is not a dominant class in our experiments, as shown in Section 5.4 (see Figure 5.2). Therefore, our results suggest that there is not necessarily a connection between the *volume* occupied by the decision regions of different classes and the frequency with which inputs perturbed by universal perturbations reach the regions corresponding to the dominant classes.

5.5.2 Class Properties of Universal Perturbations

Universal perturbations are capable of changing the output class of a large number of inputs, and the majority of the misclassified inputs are moved unintentionally towards a dominant class. In this section, we show that the perturbation itself is predicted by the model as the dominant class with high confidence.
In fact, we noticed that the following three factors are positively correlated during the generation process of a universal perturbation v: the fooling rate ($\mathcal{F}_1$), the percentage of inputs misclassified as the dominant class y_b ($\mathcal{F}_2$), and the confidence with which the model considers that the perturbation belongs to the dominant class ($\mathcal{F}_3$):[3]

$$\mathcal{F}_1(v) = P_{x\in\mathcal{X}}\left[f(x) \neq f(x+v)\right], \tag{5.9}$$

$$\mathcal{F}_2(v) = P_{x\in\mathcal{X}}\left[f(x+v) = y_b\right], \tag{5.10}$$

$$\mathcal{F}_3(v) = f(v)_b, \tag{5.11}$$

where $\mathcal{X}$ is a set of inputs and $f(\cdot)_b$ represents the output confidence of the classifier f corresponding to the class y_b. An example of the evolution of these factors during the optimization process of a universal perturbation, using the UAP-HC algorithm, is shown in Figure 5.4. These results correspond to the first experiment of Section 5.4, for the case in which no class was restricted. In particular, the left figure shows the evolution of the frequency with which each class is (wrongly) predicted for the misclassified inputs, and the right figure shows the output confidences of the model when the universal perturbation

[3] For those perturbations in which there are two dominant classes at the same time, the class $f(v)$ has been considered as the dominant (i.e., the class assigned to the perturbation by the model).

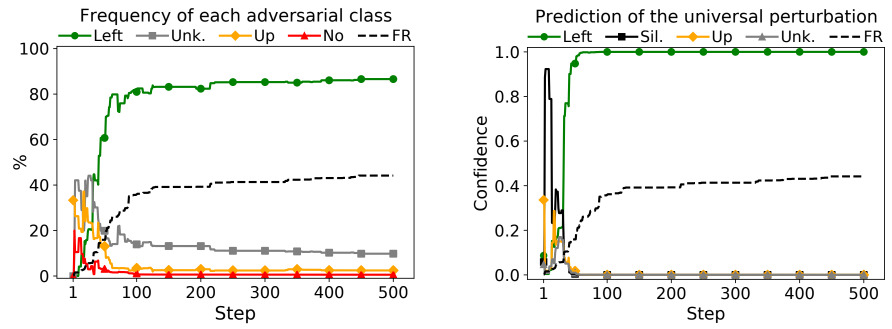


Fig. 5.4: Evolution of three different factors during the optimization process of a universal perturbation using the UAP-HC algorithm: the frequency with which the inputs are classified as the dominant class (left), the confidence of the model in the dominant class when the perturbation is predicted (right), and the evolution of the fooling rate (FR), which is shown in both plots as a reference. These results have been computed on the training set, and correspond to the first experiment reported in Section 5.4, for the case in which no class was restricted. For the sake of clarity, only the information of the four most relevant classes are plotted in each plot.

is classified. The fooling rate of the perturbation has been included in both figures as a reference, represented by a dashed line.
More generally, for the 10 different universal perturbations generated in Section 5.4 (without restricting any class), the average Pearson correlation coefficient between $\mathcal{F}_1$ and $\mathcal{F}_3$ during the first iteration of Algorithm 2 is 0.79. Similarly, the average correlation between $\mathcal{F}_1$ and $\mathcal{F}_2$ is 0.87, and the average correlation between $\mathcal{F}_2$ and $\mathcal{F}_3$ is 0.91. These results confirm that the three factors are being maximized jointly during the optimization process of the universal perturbation, even if such behavior is not specified by the model.
Motivated by this finding, we studied whether any perturbation v that is classified by the model as one particular class with high confidence is capable of producing the same effect as a universal perturbation, that is, to force the misclassification of a large number of inputs by pushing them to the class $f(v)$. For this purpose, we defined the following optimization problem, in which the objective is to find a perturbation v, with a constrained norm, that maximizes the confidence of the model in one particular class y_t, $f(v)_t$, that is:

$$\max_{v} \quad f(v)_t \quad \text{s.t.} \quad ||v||_2 \leq \epsilon. \tag{5.12}$$

We launched 100 trials for each possible target class, starting from random perturbations.[4] We used a gradient descent approach to optimize the pertur-

[4] The initial perturbations were randomly sampled from the input space $\mathbb{R}^{16000}$, where each value was sampled uniformly at random in the range $[-10^{-3}, 10^{-3}]$.

bation, restricting the search to 100 gradient descent iterations, and setting a threshold of $\epsilon = 0.1$ for the perturbation norm.
The mean and maximum fooling rates obtained with the generated perturbations are shown in Table 5.2, computed independently for each target class. The fooling rate for each class individually is shown in Figure 5.5 (left). As can be seen in Table 5.2, for the classes *left* and *unknown*, both the most frequent dominant classes associated to the universal perturbations generated using the UAP-HC algorithm (see Figure 5.2), a significantly higher effectiveness is achieved than for the rest of classes. We confirmed this using the Wilcoxon signed-rank statistical test [177], under a significance level of 0.05. Apart from that, with independence of the target class, the majority of the samples fooled by these perturbations were classified as the target class. This is shown in Figure 5.5 (right), in which the average frequency with which each class is predicted under the effect of the perturbations is computed, independently for each target class.
These results reveal that a perturbation which is optimized only to maximize the confidence of a model into one class can behave as a universal perturbation, and, more relevantly, that their effectiveness is maximized when the target class is a dominant class. Based on these findings, we can hypothesize that the model is more sensitive to some class *features* than to others, and that, ultimately, the sensitivity degree to each class feature is what determines the dominant classes. In other words, a class y_j will have a greater dominance the more sensitive the model is to the patterns in the data distribution that are associated to y_j (by the model itself).[5]

5.5.3 Singular Value Decomposition

In [114], the existence of universal perturbations for image classification DNNs is attributed, in part, to the presence of similar patterns in the geometry of decision boundaries around different points of the decision space. In particular, as described in Section 5.2, perturbations *normal* to the decision boundaries in the vicinity of natural inputs approximately span a very low-dimensional subspace, revealing that *similar* perturbations are capable of changing the output class of different input samples. This was assessed experimentally for state-of-the-art DNNs, by computing the Singular Value Decomposition (SVD) of a matrix A collecting normalized individual untargeted perturbations generated using the DeepFool algorithm. The SVD provides a set of *singular vectors* $\{s^{(1)}, s^{(2)}, \ldots, s^{(r)}\}$, which represent a basis for the subspace spanned

[5] These results are consistent with previous explanations proposed for the vulnerability of universal adversarial perturbations. For instance, these results could be related to the non-robust data-feature framework introduced in [72], to the predominance of the features of universal perturbations over the features of natural inputs [190], or to the link between the class-identity associations of the model and the most vulnerable directions in the input space studied in [73] (see Section 5.2 for more details).

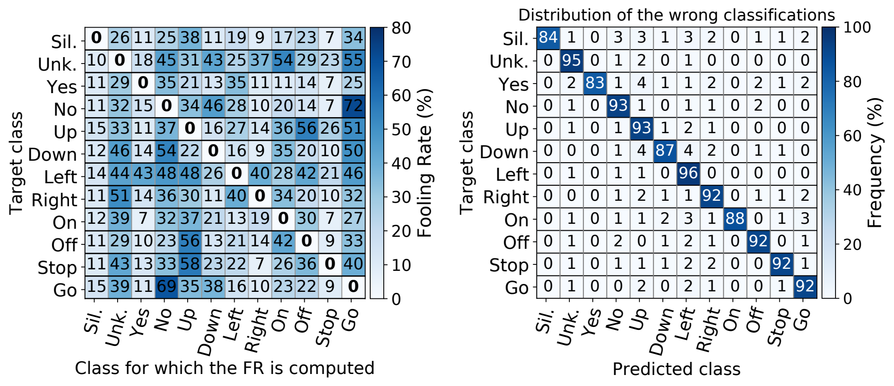


Fig. 5.5: Overview of the effectiveness of the perturbations found by solving the optimization problem defined in (5.12). In both figures, the results are reported independently for each target class (row-wise), and are averaged for the 100 trials generated for each target class. Left: average fooling rate (FR) obtained by the 100 perturbations found for each target class, computed for each class individually. Right: Average frequency with which each class is wrongly assigned to the fooled inputs by the model.

Target class	Fooling Rate					
	Considering all the classes		w/o considering dominant classes		w/o considering dominant & *Silence*	
	Mean	Max.	Mean	Max.	Mean	Max.
Sil.	17.85	21.71	19.77	24.05	19.77	24.05
Unk.	30.31	33.88	32.40	36.21	35.00	39.14
Yes	16.91	20.40	18.67	22.52	19.59	23.89
No	23.46	25.82	25.28	27.84	26.91	29.74
Up	25.53	28.19	28.16	31.10	29.79	32.97
Down	22.56	24.68	24.45	26.75	25.95	28.28
Left	32.57	37.25	35.73	40.87	38.37	44.22
Right	23.25	27.28	25.38	29.78	27.07	31.88
On	19.50	22.43	21.25	24.45	22.40	25.94
Off	21.56	24.46	23.39	26.54	24.83	28.48
Stop	25.07	27.21	27.61	29.97	29.64	32.32
Go	22.99	25.66	24.84	27.72	26.03	29.24

Table 5.2: Effectiveness of the perturbations generated using Equation (5.12), averaged for the 100 perturbations generated for each target class.

by the adversarial perturbations in A. Each $s^{(i)}$ is related to a *singular value* $\sigma^{(i)}$, which indicates the *importance* or contribution of that singular vector. As shown in [114], considering that the singular values are arranged in decreasing order $\sigma^{(1)} \geq \sigma^{(2)} \geq \cdots \geq \sigma^{(r)}$, the decay of the singular values was considerably faster in comparison to the decay obtained from the SVD of random perturbations (sampled from the unit sphere). This implies that the subspace spanned just by the first $d' \ll d$ singular vectors (i.e., those corresponding to the highest singular values) contained vectors normal to the decision boundaries in the vicinity of natural samples. Indeed, random perturbations sampled from such a subspace were capable of achieving a fooling rate of nearly 38% on unseen inputs, whereas random perturbations (of the same norm) in the input space only achieved a fooling rate of approximately 10% [114].

In this section, we take this approach as a framework to study the existence of dominant classes. First, we will replicate the previous experiment to assess whether, in the audio domain, it is also possible to find a low-dimensional subspace of the input space collecting vectors normal to the decision boundaries of DNNs. The existence of such a subspace would allow us to test a number of hypotheses, for example, whether the directions in such subspaces mainly point towards the decision boundaries corresponding to the dominant classes. This would explain why most of the inputs are (incorrectly) classified as the dominant class when they are adversarially perturbed.

Nevertheless, due to the input transformation process required to convert the raw audio signal into the MFCC representation (see Section 5.3), the results might differ depending on the data representation in which the analysis is done. For this reason, we need to assess first which audio representation is the most informative one in our case. Thus, we computed the SVD for a set of individual perturbations and different sets of random perturbations, under the three main representations for audio signals: raw audio waveform, spectrogram and MFCC coefficients.

5.5.3.1 Analysis of the SVD of Audio Perturbations

Let us consider a set of n natural input samples $\mathcal{X} = \{x^{(1)}, \ldots, x^{(n)}\}$. The individual perturbations were generated using the DeepFool algorithm, in the raw audio waveform representation:

$$\mathcal{V} = \left\{ v^{(i)} \mid v^{(i)} = \text{DeepFool}\big(x^{(i)}\big),\ i = 1, \ldots, n \right\}. \tag{5.13}$$

The perturbations that these raw waveforms produce in both the spectrogram and MFCC representations are computed as $v_g^{(i)} = g\big(x^{(i)} + v^{(i)}\big) - g\big(x^{(i)}\big)$, being g the input transform function, which maps the raw audio waveforms into either a spectrogram or the MFCC features:

$$\mathcal{V}_{\text{SPEC}} = \left\{ v_{\text{spec}}^{(i)} \mid v_{\text{spec}}^{(i)} = g_{\text{SPEC}}\big(x^{(i)} + v^{(i)}\big) - g_{\text{SPEC}}\big(x^{(i)}\big),\ i = 1, \ldots, n \right\}, \tag{5.14}$$

$$\mathcal{V}_{\text{MFCC}} = \left\{ v_{\text{mfcc}}^{(i)} \mid v_{\text{mfcc}}^{(i)} = g_{\text{MFCC}}\big(x^{(i)} + v^{(i)}\big) - g_{\text{MFCC}}\big(x^{(i)}\big),\ i = 1, \ldots, n \right\}. \tag{5.15}$$

The random perturbations were sampled uniformly at random from the raw input space:

$$\mathcal{R} = \left\{ r^{(i)} \mid\ r^{(i)} \text{ is sampled u.a.r. from } [-1, 1]^{16000},\ i = 1, \ldots, n \right\}. \tag{5.16}$$

As in the case of adversarial perturbations, the corresponding perturbations in the frequency-domain representation are computed as:

$$\mathcal{R}_{\text{SPEC}} = \left\{ r_{\text{spec}}^{(i)} \mid r_{\text{spec}}^{(i)} = g_{\text{SPEC}}\big(x^{(i)} + r^{(i)}\big) - g_{\text{SPEC}}\big(x^{(i)}\big),\ i = 1, \ldots, n \right\}, \tag{5.17}$$

$$\mathcal{R}_{\text{MFCC}} = \left\{ r_{\text{mfcc}}^{(i)} \mid r_{\text{mfcc}}^{(i)} = g_{\text{MFCC}}\big(x^{(i)} + r^{(i)}\big) - g_{\text{MFCC}}\big(x^{(i)}\big),\ i = 1, \ldots, n \right\}. \tag{5.18}$$

In this case, the random perturbations were scaled to have a fixed ℓ_2 norm of 0.1 before being applied to the inputs in Equations (5.17) and (5.18).
Finally, for a more representative analysis, we considered two additional sets of random perturbations, sampled uniformly at random from the space of spectrograms and the space of MFCC coefficients:

$$\mathfrak{R}_{\text{SPEC}} = \left\{ \mathfrak{r}^{(i)} \mid \mathfrak{r}^{(i)} \text{ is sampled u.a.r. from } [-1, 1]^{99\times 257},\ i = 1, \ldots, n \right\}, \tag{5.19}$$

$$\mathfrak{R}_{\text{MFCC}} = \left\{ \mathfrak{r}^{(i)} \mid \mathfrak{r}^{(i)} \text{ is sampled u.a.r. from } [-1, 1]^{99\times 40},\ \ i = 1, \ldots, n \right\}. \tag{5.20}$$

All the perturbations described in Equations (5.13)-(5.20) were normalized before computing the SVD. It is worth highlighting the key difference between the random perturbations defined in (5.17) and (5.18) and those defined in (5.19) and (5.20). The former represent the changes that randomly perturbing a raw signal produces on the spectrogram (or MFCC) space. In contrast, the random perturbations in (5.19) and (5.20) are directly generated in the spectrogram space or in the MFCC space, respectively. In other words, the perturbations considered in (5.19) and (5.20) are analogous to those in (5.13), but in the spaces corresponding to the spectrograms or to the MFCC coefficients instead of the space of raw audio waveforms. Considering all these types of perturbations and representations is important to better study which of them are the most informative ones in the audio domain, and to ensure that our subsequent analyses will be carried out using the most appropriate representation.

Figure 5.6 compares the decay of the singular values (sorted in decreasing order), for all the sets of perturbations considered in Equations (5.13)-(5.20). The results corresponding to the raw waveform, spectrogram and MFCC representations are shown in the first, second and third row of the figure, respectively. Whereas the left column shows the singular values obtained with the SVD for each data representation, in the right column the decays are characterized by fitting exponential curves (depicted as dashed lines) with the following form:[6]

$$y = \rho \cdot e^{-x\lambda} + \omega \quad , \quad \rho, \lambda, \omega \in \mathbb{R}. \tag{5.21}$$

A higher value of the decay factor λ represents a faster decay, as is illustrated in Figure 5.7, which shows the behavior of the exponential curves for different values of the decay factor λ. As can be seen in the figure, for low values of λ (e.g., $\lambda \leq 1$) the obtained curves are close to a *constant* or *linear* decay (i.e., $y = 1 - x$), whereas for $\lambda > 1$ the values decay much faster (i.e., exponentially). Regarding the results in the raw waveform representation (i.e., $\mathcal{V}$ and $\mathcal{R}$), the decay of the singular values is mainly linear for both individual and random perturbations, which can be assessed by their decay factor λ (see Figure 5.6), since in both cases $\lambda < 1$ is obtained. This means that, in both cases, there is not a set of singular vectors that is considerably more *informative* than the rest, and, as a consequence, a large set of vectors would be needed to provide an approximate basis for the perturbations. Thus, the perturbations do not show meaningful correlations in this representation. The same conclusion can be drawn from the perturbations sampled uniformly at random in the space of spectrograms ($\mathfrak{R}_{\text{SPEC}}$) and in the space of MFCC coefficients ($\mathfrak{R}_{\text{MFCC}}$). However, considering the perturbations in the frequency domain produced by the raw waveform perturbations, either random or adversarial (i.e., $\mathcal{V}_{\text{SPEC}}$, $\mathcal{R}_{\text{SPEC}}$, $\mathcal{V}_{\text{MFCC}}$ and $\mathcal{R}_{\text{MFCC}}$), the singular values decay exponentially, achieving decay factors which are at least of one order of magnitude greater than for the previous cases. For instance, in the MFCC representation (i.e., $\mathcal{V}_{\text{MFCC}}$ and $\mathcal{R}_{\text{MFCC}}$), the values obtained are $\lambda = \frac{1}{0.131}$ and $\lambda = \frac{1}{0.001}$, respectively. These results indicate, first, that even if the perturbations are generated in the raw audio waveform representation, it is necessary to go to the frequency-domain to observe informative patterns. This might be a fundamental difference between the image domain and the audio domain, as most of the analyses done in the former can be done directly in the *raw* image space. Secondly, the effect of audio perturbations in the frequency-domain can be characterized by just a small (in comparison to the dimensionality of the corresponding spaces) number of singular vectors. For instance, for the MFCC representation, the most relevant information is captured in less than the ~150 first singular vectors (that is, those corresponding to the highest singular values). The fact that this happens for both random or adversarial perturbations could imply, however, that the captured correlations are uninformative about the

[6] Note that the singular values have been scaled in the range $[0, 1]$ before fitting the exponential curves, for a more uniform comparison.

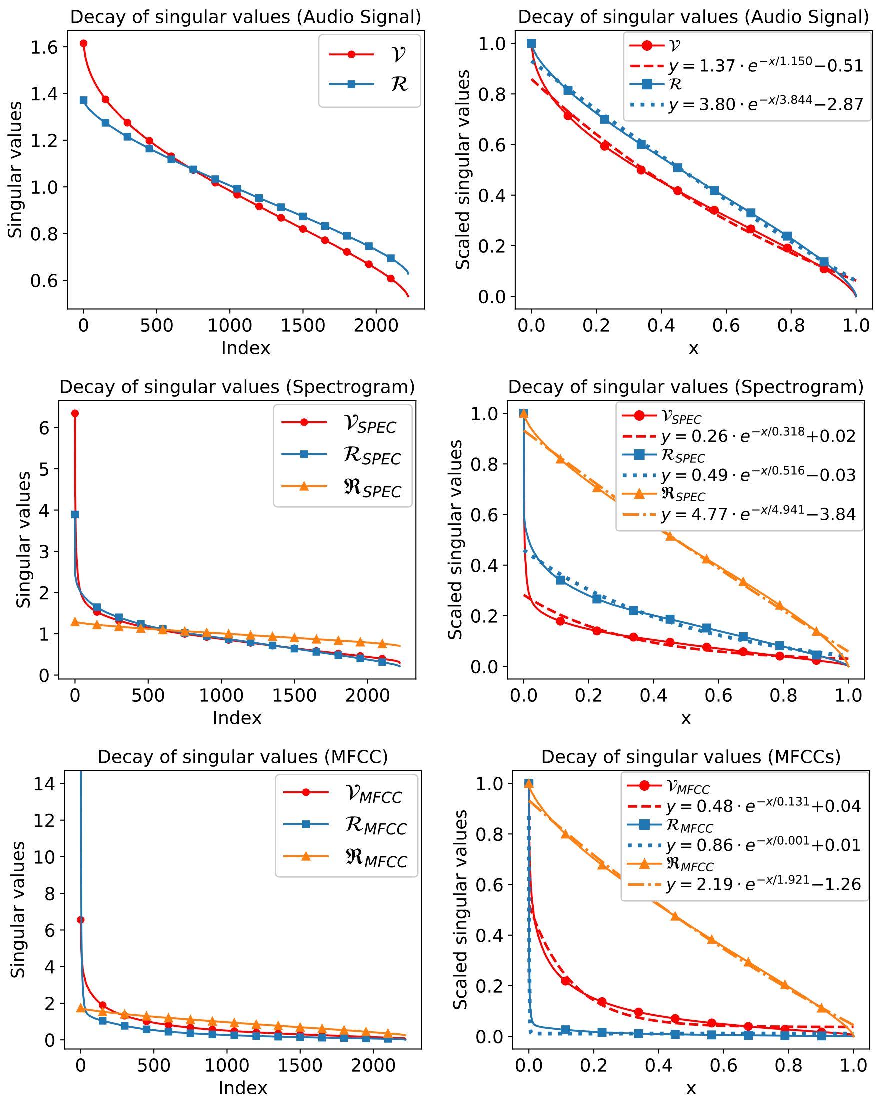


Fig. 5.6: Left column: singular values obtained in the SVD of individual adversarial perturbations and random perturbations, computed in three feature representations: raw audio waveforms (top), spectrograms (center) and MFCCs (bottom). Right column: characterization of the decay of the singular values by fitting an exponential curve (the values in both axes have been scaled in the range [0,1]).

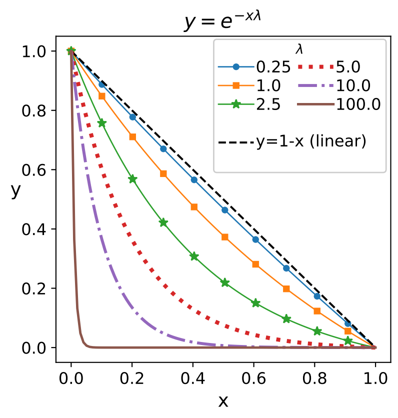


Fig. 5.7: Illustration of an exponential decay $y = \rho e^{-x\lambda} + \omega$ for different values of the *decay factor* λ. For a more uniform comparison, the values $\rho = 1$ and $\omega = 0$ were used in all the cases, and the curves were normalized in the range $[0, 1]$.

geometry of the decision boundaries around natural inputs, or, alternatively, about the vulnerability of the network to adversarial attacks. Nevertheless, in the reminder of this section we show that the SVD of individual adversarial perturbations not only provides a representative basis for input-agnostic perturbations, but also that this basis is strongly connected with the dominant classes. For the previous reasons, the rest of the analysis will focus on the MFCC feature space.

We start evaluating the fooling rate of randomly sampled perturbations in the subspace spanned by the first $N = \{10, 50, 100, 200, 500\}$ singular vectors, for the cases in which the SVD is computed for individual perturbations ($\mathcal{V}_{\text{MFCC}}$) and random perturbations ($\mathcal{R}_{\text{MFCC}}$). Given a value of N, the sampled perturbations will be produced as a linear combination of the first N singular vectors $s^{(1)}, \dots, s^{(N)}$ (computed for either $\mathcal{V}_{\text{MFCC}}$ or $\mathcal{R}_{\text{MFCC}}$). All the sampled perturbations were normalized, and the fooling rate was evaluated for different scaling factors under the ℓ_2 norm, in the range $[-200, 200]$. Note that, given a unit vector v, for any scalar $c \in \mathbb{R}$, $||v \cdot c||_2 = |c|$. For reference, the median ℓ_2 norm of the perturbations (in the MFCC) produced by the 10 universal attacks generated in Section 5.4, measured in the test set, is approximately 100.

Figure 5.8 shows, for each value of N, the average fooling rates obtained for 100 trials (i.e., 100 random perturbations). The fooling rates have been computed in the test set. The results clearly show that, when the SVD is computed for individual perturbations ($\mathcal{V}_{\text{MFCC}}$), the fooling rates are remarkably higher than for the case of random perturbations ($\mathcal{R}_{\text{MFCC}}$), even for norms close to zero. For instance, taking as reference the results corresponding to an ℓ_2 norm

of 100, the average fooling rate is approximately 48% for the case of individual perturbations, when $N \leq 100$. For the case of random perturbations, in the same conditions, the average fooling rate is only 17%.
However, the fooling rate corresponding to individual perturbations considerably decreases when a large number of singular vectors are considered. Indeed, for $N \geq 200$, the fooling rates get closer to those obtained for random perturbations. For instance, when $N = 500$, the average fooling rate (with an ℓ_2 norm of 100) is approximately 18%. This reveals that, whereas the singular vectors corresponding to the highest singular values are capturing directions normal to the decision boundaries around natural inputs (being, therefore, effective in fooling the model for a large number of inputs), the remaining singular vectors do not provide additional or relevant information.

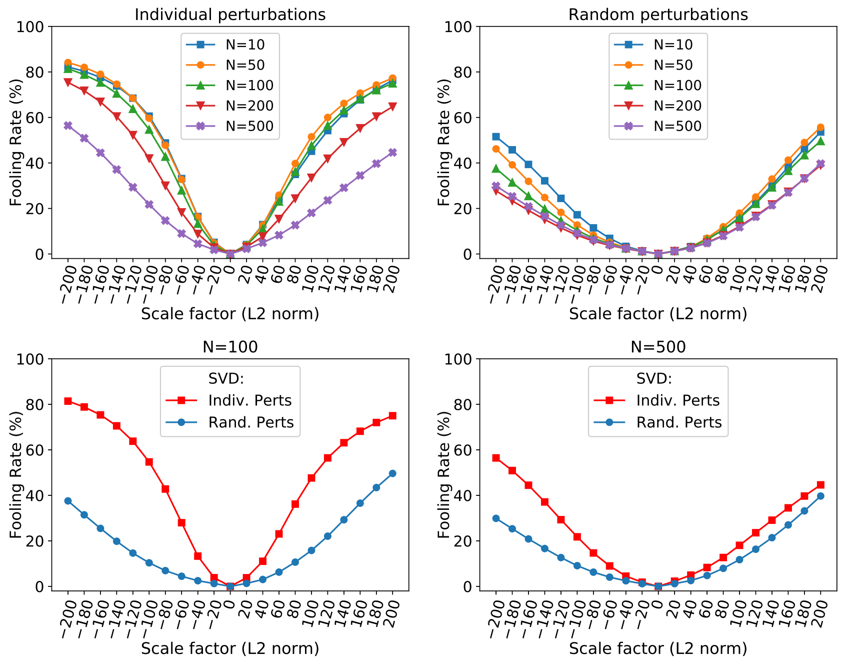


Fig. 5.8: Fooling rate produced by random perturbations sampled from the subspace spanned by the first N singular vectors. The results are averaged for 100 random perturbations. Each perturbation v was normalized and multiplied by different scale factors $c \in \mathbb{R}$ (horizontal axis), so that $||v||_2 = |c|$. The SVD is computed for individual perturbations (top left) and for random perturbations (top right), in the MFCC feature space. The bottom row shows a direct comparison between the average effectiveness of individual and random perturbations for $N = 100$ (bottom left) and $N = 500$ (bottom right).

5.5.3.2 Connection With Dominant Classes

In the previous section, we have shown that, also for speech command classification models, it is possible to find a low dimensional subspace X_S containing (*input-agnostic*) vectors normal to the decision boundaries in the vicinity of natural inputs. Therefore, a reasonable hypothesis is that dominant classes can be explained in terms of the geometric similarity of the decision boundaries in regions surrounding natural inputs, information that is captured by the basis of X_S, that is, by the singular vectors obtained from the SVD of individual perturbations.

The first hypothesis is that the first singular vectors are also normal to decision boundaries corresponding to the dominant classes. To validate this hypothesis, we first computed the fooling rate that each singular vector can achieve individually. This is shown in Figure 5.9 (top left), in which the fooling rate of the first 250 singular vectors is reported for different ℓ_2 norms. For reference, the results corresponding to a norm of 100 are also shown independently in the bottom-left part of the figure. The results clearly show that the first singular vectors are capable of fooling the model for a considerable number of test inputs, particularly for the first 50 vectors (approximately), for which an average fooling rate of 56.3% is achieved. These fooling rates are also remarkably higher than the ones obtained when the SVD is computed for random perturbations, which are also shown in Figure 5.9 (right column). Indeed, the average fooling rate obtained in the latter case (considering the first 50 vectors) is 18.7%, which represents a difference of 37.6%.

To continue with the analysis, we computed the frequency with which each class is (wrongly) predicted, considering only the inputs that were misclassified when the singular vectors were used as perturbations. The aim of this analysis is to assess if there exists a direct connection with the dominant classes. The results are shown in Figure 5.10, considering the first 100 singular vectors, scaled to have an Euclidean norm of 100. As can be seen, considering the singular vectors with the highest fooling rate (those corresponding to the vectors approximately in the range [1,50]), the most frequent wrong classes are *unknown* and *left*. Indeed, for 84% of the singular vectors in [1,50], the sum of the frequency corresponding to those two classes exceeds 50%, that is, at least 50% of the misclassified inputs are classified as *left* or as *unknown*. Moreover, for 62% of the singular vectors, the total frequency corresponding to those two classes exceeds 80%. Therefore, we now know that the singular vectors (with a high fooling rate) not only point towards decision boundaries in the close vicinity of natural inputs, but also that those decision boundaries correspond mainly to the dominant classes.

We repeated the experiment using the singular vectors obtained when the SVD is computed for random perturbations. The results are shown in Figure 5.11. In this case, it is evident that the results are more uniform along all the singular vectors, particularly for those singular vectors with a higher fooling rate (precisely, those in the range [1,50], as shown in Figure 5.9). For reference,

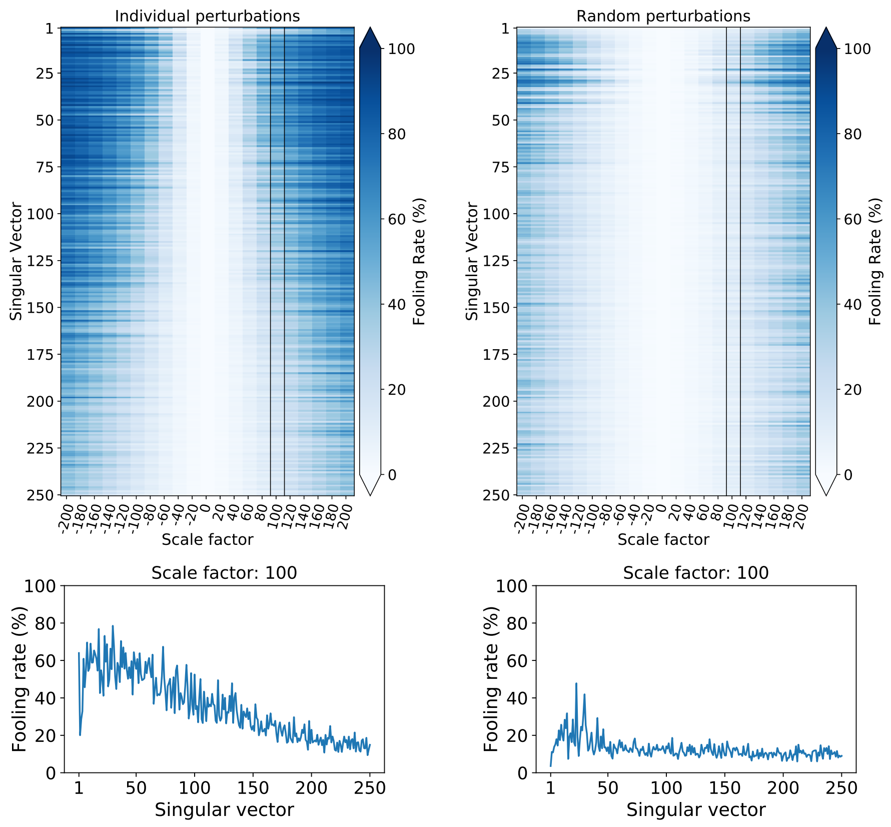


Fig. 5.9: Fooling rate percentage achieved when the inputs are perturbed with the first singular vectors computed for individual perturbations (left column) and for random perturbations (right column), in the MFCC feature space.

in this case, only for 32% of the singular vectors in the range [1,50] the total frequency corresponding to *unknown* or *left* exceeds 50%, and only for 2% of the singular vectors the total frequency exceeds 70%.

Overall, the SVD of individual perturbations has shown that the obtained singular vectors are input-agnostic perturbations directions for which the model is highly vulnerable: even when the inputs are slightly pushed in those directions, they surpass the decision boundary of the model. This reveals that the geometry of the decision boundary has *patterns* that are repeated in the vicinity of multiple natural inputs. Apart from that, we have shown that such *adversarial* directions mainly point towards the decision boundaries corresponding to the dominant classes. Therefore, it can be concluded that the universal perturbation optimization algorithms implicitly exploit the *shared* geometric patterns of decision boundaries to increase the effectiveness of the perturbations, leading to the same dominant classes in the majority of the cases.

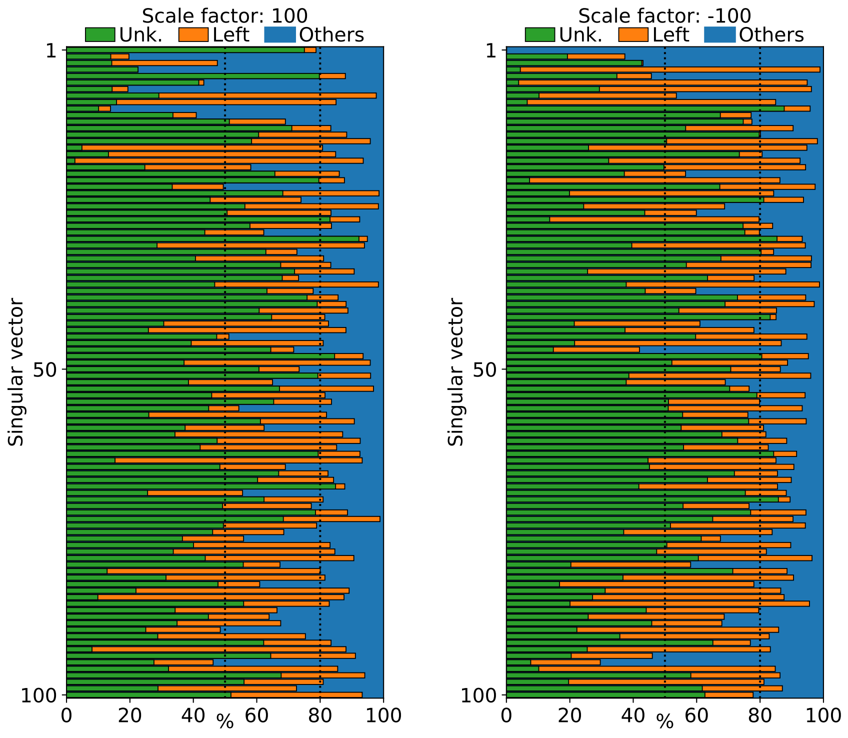


Fig. 5.10: Frequency with which each class is assigned to the misclassified inputs under the effect of singular vectors (computed for **individual perturbations**, see Equation (5.15)). The (unit) singular vectors have been scaled using two different scale factors: 100 (left) and −100 (right). For the sake of clarity, the frequencies are shown individually for the classes *unknown* and *left*, while the total frequency corresponding to the rest of classes has been grouped (*others*).

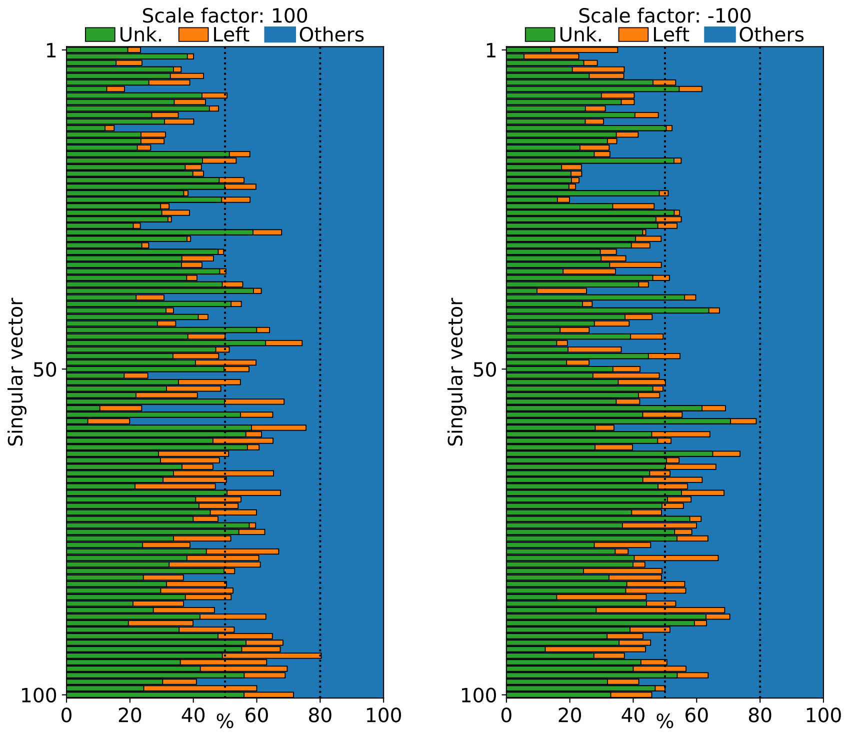


Fig. 5.11: Frequency with which each class is assigned to the misclassified inputs under the effect of singular vectors (computed for **random perturbations**, see Equation (5.18)). The (unit) singular vectors have been scaled using two different scale factors: 100 (left) and −100 (right). For the sake of clarity, the frequencies are shown individually for the classes *unknown* and *left*, while the total frequency corresponding to the rest of classes has been grouped (*others*).

5.6 Conclusions

In this chapter, we have proposed and experimentally validated a number of hypotheses to justify the intriguing phenomenon of why universal adversarial perturbations for DNNs are capable of sending the majority of inputs towards the same wrong class (i.e., dominant classes), even if such behaviour is not specified during the optimization of the perturbations. These hypotheses were studied in the audio domain, using a speech command classification task as a testbed. To the best of our knowledge, previous work has examined this effect only in the image domain, proposing open explanations that we revisit. The results obtained from our analysis revealed multiple interesting facts regarding the vulnerability of DNNs to adversarial perturbations. On the one hand, we have shown that universal perturbations can be created just by optimizing a perturbation to be recognized by the model as one particular class with high confidence. This establishes a new perspective to create universal perturbations, while explains that a class is dominant if it contains patterns in the data distribution for which the model has a higher sensitivity. On the other hand, we demonstrated that the geometry of the decision boundaries of audio DNNs contains similar patterns in the vicinity of natural inputs, and that the most *vulnerable* directions in the decision space point to the regions corresponding to the dominant classes. Finally, our work highlights a number of differences between the image domain and the audio domain, which contribute to a better and more general understanding of the field of adversarial examples in DL.

6 Conclusions and Future Work

This chapter summarizes the main contributions drawn from this thesis dissertation. In addition, a number of general directions for further research are discussed.

6.1 Contributions

DNNs are currently the core of a wide range of technologies applied in critical tasks, and effectiveness and robustness are therefore two fundamental requirements for these models. However, DNNs can be easily deceived by inputs perturbed imperceptibly for humans, known as adversarial examples, which imply a security breach that can be maliciously exploited by an adversary. Given that these vulnerabilities directly affect the integrity and reliability of multiple systems which are, progressively, being deployed in real-world applications, it is crucial to determine the scope of these vulnerabilities and how an adversary could exploit them for illegitimate purposes, in order to make a more responsible, aware and secure use of those systems.
For these reasons, this dissertation has been devoted to explore novel adversarial attack paradigms and vulnerabilities in DNNs. As a result, throughout this dissertation we have introduced new attack paradigms that are beyond the capabilities of methods currently available in the literature, as they are capable of achieving more general, complex or ambitious objectives. At the same time, based on these extended capabilities, our thesis has revealed new security gaps in use cases and scenarios where the consequences of adversarial attacks had not been investigated before. Our work has also shed light on properties of these models that make them more vulnerable to adversarial attacks, indirectly exposing more effective ways to exploit such vulnerabilities, but, at the same time, contributing to a better understanding of these intriguing phenomena.
A more detailed overview of the particular contributions derived from this dissertation is provided as follows.

In Chapter 3, a novel *multiple-instance* adversarial attack paradigm has been proposed, capable of coordinating multiple adversarial examples to achieve goals that cannot be realized by launching individual attacks isolatedly. In essence, the introduced paradigm focuses on achieving the following two objectives jointly: i) produce a misclassification for any incoming input, and ii) control the probability or frequency with which the model will predict each of the classes after multiple attacks. This novel attack paradigm enables multiple adversarial goals. On the one hand, it can be employed to carry out a large number of adversarial attacks without altering the probability distribution of the output classes $\mathcal{P}(Y)$, what could be an indicator of a recurring attack against the model, ensuring, therefore, a more stealthy way to fool the model for several inputs. On the other hand, the introduced paradigm also allows the adversary to maliciously modify $\mathcal{P}(Y)$, enabling new attack types in domains in which the application of adversarial examples is still considerably understudied, such as scenarios in which the aggregated predictions for multiple inputs are highly relevant (e.g., opinion mining or collective information retrieval) and scenarios susceptible to face concept drifts (e.g., streaming classification problems). The proposed attack paradigm is accomplished by means of a probabilistic policy capable of carefully coordinating adversarial attacks in order to achieve the aforementioned two objectives in the long run. More specifically, four different methods have been introduced to determine these policies, which rely on different strategies to determine the solutions. These methods have been validated in different problems and scenarios and compared based on several criteria. Our experimental evaluation has shown that the introduced approaches are capable of successfully achieving the aforementioned objectives, even in highly constrained scenarios where the adversary can only introduce very small amounts of distortion to the inputs.

In Chapter 4, a study has been conducted to explore the possibilities and limits of adversarial attacks in scenarios where explainable models are employed, whose predictions and explanations are supervised by humans. First, we have proposed a more general notion of adversarial examples, in order to fit in such scenarios. Based on this generalized notion, we have developed a comprehensive framework to study whether and how adversarial examples can be generated for explainable models under human assessment, introducing and illustrating novel attack paradigms. In particular, the proposed framework considers a wide range of relevant yet often ignored factors such as the type of problem, the user expertise or the objective of the explanations, in order to identify the attack strategies that should be adopted in each scenario to successfully deceive the model, and even the human querying the model. Interestingly, the research conducted demonstrates that, despite the restriction of assuming a human supervising the predictions and the corresponding explanations, which compromises the stealthiness of the attack, the use of adversarial attacks remains a realistic threat in multiple scenarios. For all these reasons, we consider that these contributions lay a foundation for a more rigorous and realistic study of adversarial examples in the field of explainable ML.

Finally, Chapter 5 addressed the vulnerability of DNNs to universal (i.e., input-agnostic) adversarial perturbations. More particularly, this chapter has been devoted to study the phenomenon of dominant classes in universal perturbations, that is, the preference of the perturbation to change the class of the inputs into a particular set of (dominant) classes, even if the perturbations are crafted in an untargeted fashion, that is, without any *a priori* imposition regarding the incorrect class to be produced. In order to shed light on this intriguing phenomenon, we have proposed and experimentally validated several hypotheses. This research has exposed the connection between the dominant classes and the sensitivity of the model to two different factors: i) to patterns in the input data distribution that the model recognizes as each class with high confidence, and ii) to *vulnerable* directions in the decision space learned by the model. Furthermore, this analysis has revealed interesting properties of the DNN sensitivity to novel types of perturbations, such as perturbations optimized to prevent the main dominant classes, hence providing a deeper understanding of this phenomenon. These findings have also suggested novel approaches to craft input-agnostic perturbations, opening the venue for future research on more effective and efficient attacks.

6.2 Future Work

The following sections summarize several future research lines derived from this dissertation.

6.2.1 Extending the Introduced Attacks

To begin with, we devise possible extensions or modifications to the introduced attacks. Firstly, we plan to extend the *multi-instance* attack approach introduced in Chapter 3 in order to address more challenging scenarios, such as highly imbalanced classification problems or scenarios where the source probability distribution of the classes changes over time, which is often the case in practice. In addition, the introduced methods can be further extended to generate attacks with more particular characteristics or behaviors. For instance, an adversary might be interested in approximating a target probability distribution of the classes while fooling the model the least possible times, which can be achieved by maximizing the values in the diagonal of the transition matrices employed by the proposed attack policy (see Section 3.2.2). Similarly, by including simple restrictions in the linear programs used to optimize those transition matrices, the adversary can choose to fool the model more often for inputs of some classes than for others, decide not to fool inputs of some classes, or specify beforehand other kinds of transition patterns. Furthermore, these methods could also be extended to generate adversarial class probability distributions using a single *universal* perturbation. In this way, a single perturbation may not only cause the misclassification of every input, but also

produce a desired probability distribution for the output classes when applied to a large number of inputs.
Secondly, regarding the attacks against explainable models introduced in Chapter 4, an interesting research line could be developing a general and unifying offensive method capable of addressing all the attack paradigms described in our framework, that is, an approach capable of automatically generating adversarial examples which satisfy the most important requirements depending on the scenario, explanation method or attack paradigm that wants to be produced. Finally, the properties exposed in Chapter 5 regarding the vulnerability of DNNs to adversarial attacks can be used, for instance, to create more effective universal adversarial perturbations. Indeed, as shown in Section 5.4, the existence of dominant classes reduces the effectiveness of universal perturbations, since the fooling rate in the inputs of those classes is practically zero. Therefore, preventing the appearance of dominant classes during the generation of the perturbation could lead to more effective attacks. We plan to develop these hypotheses in future research.

6.2.2 Further Studies in the Properties of DNNs

Whereas the analytical frameworks proposed in Chapter 5 have shown to be effective in revealing the connections between dominant classes and universal perturbations, there are a number of properties and open lines that could be further investigated in order to achieve a deeper understanding of the behavior of universal perturbations. First, focusing on the framework proposed in Section 5.5.2, an interesting future line of research could be trying to identify the data-features that the model recognizes as each class with high confidence, for instance, following the methodologies proposed in recent related works [72]. Similarly, the analysis of the geometry of the decision space carried out in Section 5.5.3 could be further extended by considering the curvature of the decision boundaries, which has proven to be highly informative for the analysis of universal perturbations [73, 115]. Moreover, it could be interesting trying to unify the data-feature perspective used in Section 5.5.2 and the one used in Section 5.5.3, relying on the geometry of the decision boundaries of the DNN. Finally, a deeper understanding of the decision spaces of DNNs is necessary to comprehensively explain why decision boundaries contain large geometric correlations around natural inputs, as well as many other fundamental questions regarding the learning process of DNNs.

6.2.3 Defensive Approaches

As stated before, the study of novel types of adversarial attacks contributes to raising awareness of new security breaches and vulnerabilities in current DNNs. Thus, a natural follow-up to this work is the development of defenses against the introduced attacks. Regarding this goal, we find two major gaps in the literature on defensive methods. First, most of the existing adversarial attack detection methods rely on analyzing, separately, whether each input is an adversarial example or not. Thus, research on methods based on assessing the risk that the model is under attack by means of analyzing multiple inputs or its long-run behavior is very limited. Nevertheless, such methods could allow a more robust, informed and comprehensive defensive diagnosis to be carried out, as they could provide an additional security layer or enable the detection of *multiple-instance* attacks against the model, such as those introduced in Chapter 3. We plan to investigate the possibility of developing such defenses in the future. Apart from that, conceiving strategies to improve the reliability and robustness of explanation methods continues to be an urgent line of research, as still limited research has been conducted on the adversarial robustness of different explanation methods such as prototype-based approaches. Thus, a deeper analysis of the vulnerability of current explanation methods is an important step in order to increase the reliability and trustworthiness of explainable models in practice.

Above all, as new forms of adversarial attacks are introduced, it is essential to revisit what notions of robustness are appropriate for the different types of threats or scenario, in a similar way to what was done in this thesis regarding offensive notions. We consider this important to rigorously determine which the most appropriate or effective defensive measures given a particular use case are, and, in the absence of general defenses capable of defending DNNs against any adversarial example, to provide security measures more tailored to the use case at hand.

7

Publications

7.1 Main Research Line

The research work carried out during this thesis has produced the following publications and submissions:

7.1.1 Referred Journals

- **Vadillo, J.**, Santana, R., and Lozano, J. A. (2020). Extending Adversarial Attacks to Produce Adversarial Class Probability Distributions. *Journal of Machine Learning Research.* Submitted (Second Review Round).
- **Vadillo, J.**, Santana, R., and Lozano, J. A. (2021). When and How to Fool Explainable Models (and Humans) with Adversarial Examples. *WIREs Data Mining and Knowledge Discovery.* Submitted (Second Review Round).
- **Vadillo, J.**, Santana, R., and Lozano, J. A. (2022). Analysis of Dominant Classes in Universal Adversarial Perturbations. *Knowledge-Based Systems*, 236:107719. Elsevier. DOI: 10.1016/j.knosys.2021.107719.

7.1.2 Awards

- 1st Award to the Best Thesis Project. *Doctoral Consortium, XIX Conference of the Spanish Association for Artificial Intelligence.* 2021.

7.2 Other Developed Work

Apart from the publications listed in the previous section, throughout these years, other research works have been developed in the field of adversarial examples, specifically, on the study of audio universal perturbations and decision boundaries. Although these works have not been included in this dissertation so as not to overextend its scope, the corresponding publications are listed below:

7.2.1 Referred Journals

- **Vadillo, J.**, and Santana, R. (2022). On the Human Evaluation of Universal Audio Adversarial Perturbations. *Computers & Security*, 112:102495. Elsevier. DOI: 10.1016/j.cose.2021.102495.

7.2.2 Conference Communications

- **Vadillo, J.**, Santana, R., Lozano, J. A. (2020). Exploring Gaps in DeepFool in Search of More Effective Adversarial Perturbations. In *Machine Learning, Optimization, and Data Science (LOD)*, volume 12566 of Lecture Notes in Computer Science, pages 215-227. Springer, Cham. DOI: 10.1007/978-3-030-64580-9_18.
- **Vadillo, J.**, Santana, R. (2022). Universal Adversarial Examples in Speech Command Classification. In *Proceedings of the XIX Conference of the Spanish Association for Artificial Intelligence (CAEPIA)*, pages 642-647.
- Garciarena, U., **Vadillo, J.**, Mendiburu, A., Santana, R. (2022). Adversarial Perturbations for Evolutionary Optimization. In *Machine Learning, Optimization, and Data Science (LOD)*, volume 13164 of Lecture Notes in Computer Science, pages 408-422. Springer, Cham. DOI: 10.1007/978-3-030-95470-3_31.

7.2.3 Awards

- 3rd Best Paper Award, Universal Adversarial Examples in Speech Command Classification. *XIX Conference of the Spanish Association for Artificial Intelligence*. 2021.

References

[1] Adadi, A. and Berrada, M. (2018). Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6:52138–52160.

[2] Agirre, E. and Edmonds, P. (2006). *Word Sense Disambiguation: Algorithms and Applications*, volume 33 of *Text, Speech and Language Technology*.

[3] Aivodji, U., Arai, H., Fortineau, O., Gambs, S., Hara, S., and Tapp, A. (2019). Fairwashing: The Risk of Rationalization. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 161–170.

[4] Al-masni, M. A., Al-antari, M. A., Choi, M.-T., Han, S.-M., and Kim, T.-S. (2018). Skin Lesion Segmentation in Dermoscopy Images via Deep Full Resolution Convolutional Networks. *Computer Methods and Programs in Biomedicine*, 162:221–231.

[5] Alshahrani, A., Ghaffari, M., Amirizirtol, K., and Liu, X. (2021). Optimism/Pessimism Prediction of Twitter Messages and Users Using BERT with Soft Label Assignment. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

[6] Alvarez-Melis, D. and Jaakkola, T. (2018a). Towards Robust Interpretability with Self-Explaining Neural Networks. In *Advances in Neural Information Processing Systems*, volume 31, pages 7775–7784.

[7] Alvarez-Melis, D. and Jaakkola, T. S. (2018b). On the Robustness of Interpretability Methods. In *Proceedings of the 2018 ICML Workshop on Human Interpretability in Machine Learning (WHI 2018)*, pages 66–71.

[8] Alzantot, M., Balaji, B., and Srivastava, M. (2018a). Did you hear that? Adversarial Examples Against Automatic Speech Recognition. *arXiv preprint arXiv:1801.00554*.

[9] Alzantot, M., Sharma, Y., Chakraborty, S., Zhang, H., Hsieh, C.-J., and Srivastava, M. B. (2019). GenAttack: Practical Black-Box Attacks with Gradient-Free Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, GECCO '19, pages 1111–1119.

[10] Alzantot, M., Sharma, Y., Elgohary, A., Ho, B.-J., Srivastava, M., and Chang, K.-W. (2018b). Generating Natural Language Adversarial Examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896.

[11] Anders, C. J., Weber, L., Neumann, D., Samek, W., Müller, K.-R., and Lapuschkin, S. (2022). Finding and Removing Clever Hans: Using Explanation Methods to Debug and Improve Deep Models. *Information Fusion*, 77:261–295.

[12] Bai, S., Li, Y., Zhou, Y., Li, Q., and Torr, P. H. (2021). Adversarial Metric Attack and Defense for Person Re-Identification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(6):2119–2126.

[13] Ballet, V., Renard, X., Aigrain, J., Laugel, T., Frossard, P., and Detyniecki, M. (2019). Imperceptible Adversarial Attacks on Tabular Data. In *NeurIPS 2019 Workshop on Robust AI in Financial Services: Data, Fairness, Explainability, Trustworthiness, and Privacy (Robust AI in FS)*.

[14] Bansal, N., Agarwal, C., and Nguyen, A. (2020). SAM: The Sensitivity of Attribution Methods to Hyperparameters. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8670–8680.

[15] Beck, C., Booth, H., El-Assady, M., and Butt, M. (2020). Representation Problems in Linguistic Annotations: Ambiguity, Variation, Uncertainty, Error and Bias. In *Proceedings of the 14th Linguistic Annotation Workshop*, pages 60–73.

[16] Behjati, M., Moosavi-Dezfooli, S.-M., Baghshah, M. S., and Frossard, P. (2019). Universal Adversarial Attacks on Text Classifiers. In *Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7345–7349.

[17] Belinkov, Y. and Bisk, Y. (2018). Synthetic and Natural Noise Both Break Neural Machine Translation. In *International Conference on Learning Representations (ICLR)*.

[18] Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., and Roli, F. (2013). Evasion Attacks against Machine Learning at Test Time. In *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, pages 387–402.

[19] Biswas, A. and Mukherjee, S. (2021). Ensuring Fairness under Prior Probability Shifts. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '21, pages 414–424.

[20] Borkar, J. and Chen, P.-Y. (2021). Simple Transparent Adversarial Examples. In *ICLR 2021 Workshop on Security and Safety in Machine Learning Systems*.

[21] Bortsova, G., González-Gonzalo, C., Wetstein, S. C., Dubost, F., Katramados, I., Hogeweg, L., Liefers, B., van Ginneken, B., Pluim, J. P. W., Veta, M., Sánchez, C. I., and de Bruijne, M. (2021). Adversarial Attack Vulnerability of Medical Image Analysis Systems: Unexplored Factors. *Medical Image Analysis*, 73:102141.

[22] Brendel, W., Rauber, J., and Bethge, M. (2018). Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. In *International Conference on Learning Representations (ICLR)*.

[23] Bussone, A., Stumpf, S., and O'Sullivan, D. (2015). The Role of Explanations on Trust and Reliance in Clinical Decision Support Systems. In *Proceedings of the 2015 International Conference on Healthcare Informatics (ICHI)*, pages 160–169.

[24] Carlini, N., Mishra, P., Vaidya, T., Zhang, Y., Sherr, M., Shields, C., Wagner, D., and Zhou, W. (2016). Hidden Voice Commands. In *Proceedings of the 25th USENIX Security Symposium (USENIX Security 16)*, pages 513–530.

[25] Carlini, N. and Wagner, D. (2017). Towards Evaluating the Robustness of Neural Networks. In *Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57.

[26] Carlini, N. and Wagner, D. (2018). Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 1–7.

[27] Cartella, F., Anunciação, O., Funabiki, Y., Yamaguchi, D., Akishita, T., and Elshocht, O. (2021). Adversarial Attacks for Tabular Data: Application to Fraud Detection and Imbalanced Data. In *Proceedings of the 2021 AAAI Workshop on Artificial Intelligence Safety (SafeAI)*.

[28] Chen, C., Li, O., Tao, D., Barnett, A., Rudin, C., and Su, J. K. (2019). This Looks Like That: Deep Learning for Interpretable Image Recognition. In *Advances in Neural Information Processing Systems*, volume 32, pages 8930–8941.

[29] Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. (2017). ZOO: Zeroth Order Optimization Based Black-Box Attacks to Deep Neural Networks without Training Substitute Models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (AISec)*, pages 15–26.

[30] Chen, Z., Bei, Y., and Rudin, C. (2020). Concept Whitening for Interpretable Image Recognition. *Nature Machine Intelligence*, 2(12):772–782.

[31] Cheng, M., Yi, J., Chen, P.-Y., Zhang, H., and Hsieh, C.-J. (2020). Seq2Sick: Evaluating the Robustness of Sequence-to-Sequence Models with Adversarial Examples. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3601–3608.

[32] Co, K. T., Muñoz-González, L., Kanthan, L., Glocker, B., and Lupu, E. C. (2020). Universal Adversarial Perturbations to Understand Robustness of Texture vs. Shape-biased Training. *arXiv preprint arXiv:1911.10364*.

[33] Cybenko, G. (1989). Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals and Systems*, 2(4):303–314.

[34] Deng, E., Qin, Z., Li, M., Ding, Y., and Qin, Z. (2021). Attacking the Dialogue System at Smart Home. In *Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Work-*

sharing, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pages 148–158.

[35] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255.

[36] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

[37] Dombrowski, A.-K., Alber, M., Anders, C., Ackermann, M., Müller, K.-R., and Kessel, P. (2019). Explanations Can Be Manipulated and Geometry Is to Blame. In *Advances in Neural Information Processing Systems*, volume 32, pages 13589–13600.

[38] Doshi-Velez, F. and Kim, B. (2018). Considerations for Evaluation and Generalization in Interpretable Machine Learning. In *Explainable and Interpretable Models in Computer Vision and Machine Learning*, The Springer Series on Challenges in Machine Learning, pages 3–17.

[39] Du, T., Ji, S., Li, J., Gu, Q., Wang, T., and Beyah, R. (2020). SirenAttack: Generating Adversarial Audio for End-to-End Acoustic Systems. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, ASIA CCS '20, pages 357–369.

[40] Ebrahimi, J., Lowd, D., and Dou, D. (2018). On Adversarial Examples for Character-Level Neural Machine Translation. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 653–663.

[41] Elliott, A., Law, S., and Russell, C. (2021). Explaining Classifiers Using Adversarial Perturbations on the Perceptual Ball. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10693–10702.

[42] Esmaeilpour, M., Cardinal, P., and Koerich, A. L. (2021). Cyclic Defense GAN Against Speech Adversarial Attacks. *IEEE Signal Processing Letters*, 28:1769–1773.

[43] Etmann, C., Lunz, S., Maass, P., and Schoenlieb, C. (2019). On the Connection Between Adversarial Robustness and Saliency Map Interpretability. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 1823–1832.

[44] Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D. (2018). Robust Physical-World Attacks on Deep Learning Visual Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1625–1634.

[45] Finlayson, S. G., Bowers, J. D., Ito, J., Zittrain, J. L., Beam, A. L., and Kohane, I. S. (2019). Adversarial Attacks on Medical Machine Learning. *Science*, 363(6433):1287–1289.

[46] Fujiyoshi, H., Hirakawa, T., and Yamashita, T. (2019). Deep Learning-Based Image Recognition for Autonomous Driving. *IATSS Research*, 43(4):244–252.

[47] Fursov, I., Morozov, M., Kaploukhaya, N., Kovtun, E., Rivera-Castro, R., Gusev, G., Babaev, D., Kireev, I., Zaytsev, A., and Burnaev, E. (2021). Adversarial Attacks on Deep Models for Financial Transaction Records. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pages 2868–2878.

[48] Gao, W. and Sebastiani, F. (2016). From Classification to Quantification in Tweet Sentiment Analysis. *Social Network Analysis and Mining*, 6(1):19.

[49] Garg, S., Wu, Y., Balakrishnan, S., and Lipton, Z. C. (2020). A Unified View of Label Shift Estimation. In *Advances in Neural Information Processing Systems*, volume 33, pages 3290–3300.

[50] Ghassemi, M., Oakden-Rayner, L., and Beam, A. L. (2021). The False Hope of Current Approaches to Explainable Artificial Intelligence in Health Care. *The Lancet Digital Health*, 3(11):e745–e750.

[51] Ghorbani, A., Abid, A., and Zou, J. (2019a). Interpretation of Neural Networks Is Fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3681–3688.

[52] Ghorbani, A., Wexler, J., Zou, J. Y., and Kim, B. (2019b). Towards Automatic Concept-based Explanations. In *Advances in Neural Information Processing Systems*, volume 32, pages 9277–9286.

[53] Giachanou, A. and Crestani, F. (2016). Like It or Not: A Survey of Twitter Sentiment Analysis Methods. *ACM Computing Surveys*, 49(2):28.1–28.41.

[54] Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., and Kagal, L. (2018). Explaining Explanations: An Overview of Interpretability of Machine Learning. In *2018 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89.

[55] Goddard, K., Roudsari, A., and Wyatt, J. C. (2012). Automation Bias: A Systematic Review of Frequency, Effect Mediators, and Mitigators. *Journal of the American Medical Informatics Association*, 19(1):121–127.

[56] Gong, Y., Li, B., Poellabauer, C., and Shi, Y. (2019). Real-Time Adversarial Attacks. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI-19)*, pages 4672–4680.

[57] González, P., Castaño, A., Chawla, N. V., and Coz, J. J. D. (2017a). A Review on Quantification Learning. *ACM Computing Surveys*, 50(5):1–40.

[58] González, P., Díez, J., Chawla, N., and del Coz, J. J. (2017b). Why Is Quantification an Interesting Learning Problem? *Progress in Artificial Intelligence*, 6(1):53–58.

[59] Goodfellow, I., Shlens, J., and Szegedy, C. (2015). Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations (ICLR)*, pages 1–11.

[60] Gupta, T., Sinha, A., Kumari, N., Singh, M., and Krishnamurthy, B. (2019). A Method for Computing Class-wise Universal Adversarial Perturbations. *arXiv preprint arXiv:1912.00466.*

[61] Haffar, R., Jebreel, N. M., Domingo-Ferrer, J., and Sánchez, D. (2021). Explaining Image Misclassification in Deep Learning via Adversarial Examples. In *Proceedings of the International Conference on Modeling Decisions for Artificial Intelligence (MDAI)*, Lecture Notes in Computer Science, pages 323–334.

[62] Hasan, M., Rundensteiner, E., and Agu, E. (2019). Automatic Emotion Detection in Text Streams by Analyzing Twitter Data. *International Journal of Data Science and Analytics*, 7(1):35–51.

[63] Hase, P., Chen, C., Li, O., and Rudin, C. (2019). Interpretable Image Recognition with Hierarchical Prototypes. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 7, pages 32–40.

[64] Hayes, J. and Danezis, G. (2018). Learning Universal Adversarial Perturbations with Generative Models. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 43–49.

[65] Heo, J., Joo, S., and Moon, T. (2019). Fooling Neural Network Interpretations via Adversarial Model Manipulation. In *Advances in Neural Information Processing Systems*, volume 32, pages 2925–2936.

[66] Hirano, H., Minagi, A., and Takemoto, K. (2021). Universal Adversarial Attacks on Deep Neural Networks for Medical Image Classification. *BMC Medical Imaging*, 21(1):1–13.

[67] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer Feedforward Networks Are Universal Approximators. *Neural Networks*, 2(5):359–366.

[68] Hossam, M., Le, T., Zhao, H., and Phung, D. (2021). Explain2Attack: Text Adversarial Attacks via Cross-Domain Interpretability. In *Proceedings of the 25th International Conference on Pattern Recognition (ICPR)*, pages 8922–8928.

[69] Hussenot, L., Geist, M., and Pietquin, O. (2020). CopyCAT: Taking Control of Neural Policies with Constant Attacks. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, pages 548–556.

[70] Ignatiev, A., Narodytska, N., and Marques-Silva, J. (2019). On Relating Explanations and Adversarial Examples. In *Advances in Neural Information Processing Systems*, volume 32, pages 15883–15893.

[71] Ilyas, A., Engstrom, L., Athalye, A., and Lin, J. (2018). Black-Box Adversarial Attacks with Limited Queries and Information. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pages 2137–2146.

[72] Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. (2019). Adversarial Examples Are Not Bugs, They Are Features. In *Advances in Neural Information Processing Systems 32*, pages 125–136.

[73] Jetley, S., Lord, N., and Torr, P. (2018). With Friends Like These, Who Needs Adversaries? In *Advances in Neural Information Processing Systems 31*, pages 10749–10759.

[74] Jiang, W., Wen, X., Zhan, J., Wang, X., and Song, Z. (2022). Interpretability-Guided Defense Against Backdoor Attacks to Deep Neural Networks. *IEE E Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(8):2611–2624.

[75] Johnson, S. L. J. (2019). AI, Machine Learning, and Ethics in Health Care. *Journal of Legal Medicine*, 39(4):427–441.

[76] Kantchelian, A., Afroz, S., Huang, L., Islam, A. C., Miller, B., Tschantz, M. C., Greenstadt, R., Joseph, A. D., and Tygar, J. D. (2013). Approaches to Adversarial Drift. In *Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security*, AISec '13, pages 99–110.

[77] Kaviani, S., Han, K. J., and Sohn, I. (2022). Adversarial Attacks and Defenses on AI in Medical Imaging Informatics: A Survey. *Expert Systems with Applications*, 198:116815.

[78] Khosla, A., Jayadevaprakash, N., Yao, B., and Li, F.-F. (2011). Novel Dataset for Fine-Grained Image Categorization: Stanford Dogs. In *CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*.

[79] Khrulkov, V. and Oseledets, I. (2018). Art of Singular Vectors and Universal Adversarial Perturbations. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8562–8570.

[80] Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., and Sayres, R. (2018). Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 2668–2677.

[81] Kindermans, P.-J., Hooker, S., Adebayo, J., Alber, M., Schütt, K. T., Dähne, S., Erhan, D., and Kim, B. (2019). The (Un)reliability of Saliency Methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700 of *Lecture Notes in Computer Science*, pages 267–280.

[82] King, G. and Lu, Y. (2008). Verbal Autopsy Methods with Multiple Causes of Death. *Statistical Science*, 23(1):78–91.

[83] Koh, P. W. and Liang, P. (2017). Understanding Black-Box Predictions via Influence Functions. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894.

[84] Korycki, Ł. and Krawczyk, B. (2020). Adversarial Concept Drift Detection under Poisoning Attacks for Robust Data Stream Mining. *arXiv preprint arXiv:2009.09497*.

[85] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105.

[86] Kuchipudi, B., Nannapaneni, R. T., and Liao, Q. (2020). Adversarial Machine Learning for Spam Filters. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ARES '20, pages 1–6.

[87] Kumar, N., Vimal, S., Kayathwal, K., and Dhama, G. (2021). Evolutionary Adversarial Attacks on Payment Systems. In *Proceedings of the 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 813–818.

[88] Kuppa, A. and Le-Khac, N.-A. (2020). Black Box Attacks on Explainable Artificial Intelligence(XAI) Methods in Cyber Security. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

[89] Kurakin, A., Goodfellow, I., and Bengio, S. (2016). Adversarial Examples in the Physical World. In *Workshop of the 2017 International Conference on Learning Representations (ICLR)*.

[90] Kurakin, A., Goodfellow, I. J., and Bengio, S. (2017). Adversarial Machine Learning at Scale. In *International Conference on Learning Representations (ICLR)*, pages 1–17.

[91] Lakkaraju, H. and Bastani, O. (2020). "How do I fool you?": Manipulating User Trust via Misleading Black Box Explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, pages 79–85.

[92] Lakkaraju, H., Kamar, E., Caruana, R., and Leskovec, J. (2019). Faithful and Customizable Explanations of Black Box Models. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '19, pages 131–138.

[93] LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L. (1989). Handwritten Digit Recognition with a Back-Propagation Network. In *Advances in Neural Information Processing Systems*, volume 2, pages 396–404.

[94] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[95] Levenshtein, V. I. (1966). Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Soviet Physics Doklady*, 10(8):707–710.

[96] Li, J., Qu, S., Li, X., Szurley, J., Kolter, J. Z., and Metze, F. (2019). Adversarial Music: Real world Audio Adversary against Wake-word Detection System. In *Advances in Neural Information Processing Systems*, volume 32, pages 11931–11941.

[97] Li, J., Zhang, X., Jia, C., Xu, J., Zhang, L., Wang, Y., Ma, S., and Gao, W. (2020a). Universal Adversarial Perturbations Generative Network For Speaker Recognition. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6.

[98] Li, O., Liu, H., Chen, C., and Rudin, C. (2018). Deep Learning for Case-Based Reasoning Through Prototypes: A Neural Network That Explains Its Predictions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1):3530–3537.

[99] Li, Z., Liu, F., Yang, W., Peng, S., and Zhou, J. (2021). A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21.

[100] Li, Z., Wu, Y., Liu, J., Chen, Y., and Yuan, B. (2020b). AdvPulse: Universal, Synchronization-free, and Targeted Audio Adversarial Attacks via Subsecond Perturbations. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, pages 1121–1134.

[101] Lin, Y.-C., Hong, Z.-W., Liao, Y.-H., Shih, M.-L., Liu, M.-Y., and Sun, M. (2017). Tactics of Adversarial Attack on Deep Reinforcement Learning Agents. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, IJCAI'17, pages 3756–3762.

[102] Lipton, Z., Wang, Y.-X., and Smola, A. (2018). Detecting and Correcting for Label Shift with Black Box Predictors. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 3122–3130.

[103] Lipton, Z. C. (2018). The Mythos of Model Interpretability: In Machine Learning, the Concept of Interpretability Is Both Important and Slippery. *Queue*, 16(3):31–57.

[104] Liu, N., Du, M., Guo, R., Liu, H., and Hu, X. (2021). Adversarial Attacks and Defenses: An Interpretation Perspective. *ACM SIGKDD Explorations Newsletter*, 23(1):86–99.

[105] Liu, N., Yang, H., and Hu, X. (2018). Adversarial Detection with Model Interpretation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 1803–1811.

[106] Liu, Y., Chen, X., Liu, C., and Song, D. (2017). Delving into Transferable Adversarial Examples and Black-Box Attacks. In *International Conference on Learning Representations (ICLR).*

[107] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations (ICLR).*

[108] Mahdavifar, S. and Ghorbani, A. A. (2019). Application of Deep Learning to Cybersecurity: A Survey. *Neurocomputing*, 347:149–176.

[109] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval.* Cambridge University Press.

[110] Metzen, J. H., Kumar, M. C., Brox, T., and Fischer, V. (2017). Universal Adversarial Perturbations Against Semantic Image Segmentation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2774–2783.

[111] Michel Koerich, K., Esmailpour, M., Abdoli, S., Britto, A. d. S., and Koerich, A. L. (2020). Cross-Representation Transferability of Adversarial

Attacks: From Spectrograms to Audio Waveforms. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7.

[112] Milli, L., Monreale, A., Rossetti, G., Pedreschi, D., Giannotti, F., and Sebastiani, F. (2015). Quantification in Social Networks. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10.

[113] Moore, J., Hammerla, N., and Watkins, C. (2019). Explaining Deep Learning Models with Constrained Adversarial Examples. In *PRICAI 2019: Trends in Artificial Intelligence*, Lecture Notes in Computer Science, pages 43–56.

[114] Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. (2017). Universal Adversarial Perturbations. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 86–94.

[115] Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., Frossard, P., and Soatto, S. (2018). Robustness of Classifiers to Universal Perturbations: A Geometric Perspective. In *International Conference on Learning Representations (ICLR)*.

[116] Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2016). DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582.

[117] Mopuri, K. R., Ganeshan, A., and Babu, R. V. (2019). Generalizable Data-Free Objective for Crafting Universal Adversarial Perturbations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(10):2452–2465.

[118] Mopuri, K. R., Garg, U., and Babu, R. V. (2017). Fast Feature Fool: A Data Independent Approach to Universal Adversarial Perturbations. In *Proceedings of the 2017 British Machine Vision Conference (BMVC)*, pages 30.1–30.12.

[119] Mopuri, K. R., Ojha, U., Garg, U., and Babu, R. V. (2018a). NAG: Network for Adversary Generation. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 742–751.

[120] Mopuri, K. R., Uppala, P. K., and Babu, R. V. (2018b). Ask, Acquire, and Attack: Data-Free UAP Generation Using Class Impressions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Lecture Notes in Computer Science, pages 20–35.

[121] Mori, K., Fukui, H., Murase, T., Hirakawa, T., Yamashita, T., and Fujiyoshi, H. (2019). Visual Explanation by Attention Branch Network for End-to-End Learning-based Self-Driving. In *Proceedings of the 2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1577–1582.

[122] Muda, L., Begam, M., and Elamvazuthi, I. (2010). Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques. *Journal of Computing*, 2(3):138–143.

[123] Neekhara, P., Hussain, S., Pandey, P., Dubnov, S., McAuley, J., and Koushanfar, F. (2019). Universal Adversarial Perturbations for Speech Recognition Systems. In *Interspeech*, pages 481–485.

[124] Nguyen, G., Kim, D., and Nguyen, A. (2021). The Effectiveness of Feature Attribution Methods and Its Correlation With Automatic Evaluation Scores. In *Advances in Neural Information Processing Systems*, volume 34, pages 26422–26436.

[125] Noack, A., Ahern, I., Dou, D., and Li, B. (2021). An Empirical Study on the Relation Between Network Interpretability and Adversarial Robustness. *SN Computer Science*, 2(1).

[126] Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. (2017). Practical Black-Box Attacks against Machine Learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, pages 506–519.

[127] Pérez-Gállego, P., Quevedo, J. R., and del Coz, J. J. (2017). Using Ensembles for Problems With Characterizable Changes in Data Distribution: A Case Study on Quantification. *Information Fusion*, 34:87–100.

[128] Pillai, R., Oza, P., and Sharma, P. (2019). Review of Machine Learning Techniques in Health Care. In *Proceedings of the 2019 International Conference on Recent Innovations in Computing (ICRIC)*, Lecture Notes in Electrical Engineering, pages 103–111.

[129] Poursaeed, O., Katsman, I., Gao, B., and Belongie, S. (2018). Generative Adversarial Perturbations. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4422–4431.

[130] Praher, V., Prinz, K., Flexer, A., and Widmer, G. (2021). On the Veracity of Local, Model-Agnostic Explanations in Audio Classification: Targeted Investigations With Adversarial Examples. In *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, pages 531–538.

[131] Qi, L., Khaleel, M., Tavanapong, W., Sukul, A., and Peterson, D. (2021). A Framework for Deep Quantification Learning. In *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, pages 232–248.

[132] Qin, Y., Carlini, N., Cottrell, G., Goodfellow, I., and Raffel, C. (2019). Imperceptible, Robust, and Targeted Adversarial Examples for Automatic Speech Recognition. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 5231–5240.

[133] Qiu, H., Custode, L. L., and Iacca, G. (2021). Black-Box Adversarial Attacks using Evolution Strategies. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO)*, GECCO '21, pages 1827–1833.

[134] Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. (2009). When Training and Test Sets Are Different: Characterizing Learning Transfer. In *Dataset Shift in Machine Learning*, pages 3–28.

[135] Rabanser, S., Günnemann, S., and Lipton, Z. (2019). Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift. In *Advances in Neural Information Processing Systems*, volume 32, pages 1396–1408.

[136] Ras, G., van Gerven, M., and Haselager, P. (2018). Explanation Methods in Deep Learning: Users, Values, Concerns and Challenges. In *Explainable and Interpretable Models in Computer Vision and Machine Learning*, The Springer Series on Challenges in Machine Learning, pages 19–36.

[137] Renard, X., Laugel, T., Lesot, M.-J., Marsala, C., and Detyniecki, M. (2019). Detecting Potential Local Adversarial Examples for Human-Interpretable Defense. In *Proceedings of the 2018 ECML PKDD Workshop on Recent Advances in Adversarial Machine Learning*, Lecture Notes in Computer Science, pages 41–47.

[138] Ros, A. S. and Doshi-Velez, F. (2018). Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing Their Input Gradients. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1):1660–1669.

[139] Rudin, C. (2019). Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Nature Machine Intelligence*, 1(5):206–215.

[140] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252.

[141] Saerens, M., Latinne, P., and Decaestecker, C. (2002). Adjusting the Outputs of a Classifier to New a Priori Probabilities: A Simple Procedure. *Neural Computation*, 14(1):21–41.

[142] Sainath, T. N. and Parada, C. (2015). Convolutional Neural Networks for Small-Footprint Keyword Spotting. In *Interspeech*, pages 1478–1482.

[143] Sallo, R. A., Esmaeilpour, M., and Cardinal, P. (2021). Adversarially Training for Audio Classifiers. In *Proceedings of the 25th International Conference on Pattern Recognition (ICPR)*, pages 9569–9576.

[144] Saralajew, S., Holdijk, L., Rees, M., Asan, E., and Villmann, T. (2019). Classification-by-Components: Probabilistic Modeling of Reasoning over a Set of Components. In *Advances in Neural Information Processing Systems*, volume 32, pages 2792–2803.

[145] Saravia, E., Liu, H.-C. T., Huang, Y.-H., Wu, J., and Chen, Y.-S. (2018). CARER: Contextualized Affect Representations for Emotion Recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697.

[146] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626.

[147] Serradilla, O., Zugasti, E., Rodriguez, J., and Zurutuza, U. (2022). Deep Learning Models for Predictive Maintenance: A Survey, Comparison, Challenges and Prospects. *Applied Intelligence*, 52(10):10934–10964.

[148] Sethi, T. S. and Kantardzic, M. (2018). Handling Adversarial Concept Drift in Streaming Data. *Expert Systems with Applications*, 97:18–40.

[149] Sharif, M., Bhagavatula, S., Bauer, L., and Reiter, M. K. (2016). Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, CCS '16, pages 1528–1540.

[150] Silla, C. N. and Freitas, A. A. (2011). A Survey of Hierarchical Classification Across Different Application Domains. *Data Mining and Knowledge Discovery*, 22(1):31–72.

[151] Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *Workshop of the 2014 International Conference on Learning Representations (ICLR).*

[152] Slack, D., Hilgard, S., Jia, E., Singh, S., and Lakkaraju, H. (2020). Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186.

[153] Sokol, K. and Flach, P. (2019). Counterfactual Explanations of Machine Learning Predictions: Opportunities and Challenges for AI Safety. In *Proceedings of the 2019 AAAI Workshop on Artificial Intelligence Safety (SafeAI)*, pages 95–99.

[154] Stiglic, G., Kocbek, P., Fijacko, N., Zitnik, M., Verbert, K., and Cilar, L. (2020). Interpretability of Machine Learning-Based Prediction Models in Healthcare. *WIREs Data Mining and Knowledge Discovery*, 10(5):e1379.

[155] Stock, P. and Cisse, M. (2018). ConvNets and ImageNet Beyond Accuracy: Understanding Mistakes and Uncovering Biases. In *Computer Vision – ECCV 2018*, volume 11210 of *Lecture Notes in Computer Science (LNCS)*, pages 504–519.

[156] Stutz, D., Hein, M., and Schiele, B. (2019). Disentangling Adversarial Robustness and Generalization. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6969–6980.

[157] Subramanian, V., Pankajakshan, A., Benetos, E., Xu, N., McDonald, S., and Sandler, M. (2020). A Study on the Transferability of Adversarial Attacks in Sound Event Classification. In *Proceedings of the 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 301–305.

[158] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing Properties of Neural Networks. In *International Conference on Learning Representations (ICLR).*

[159] Tanay, T. and Griffin, L. (2016). A Boundary Tilting Perspective on the Phenomenon of Adversarial Examples. *arXiv preprint arXiv:1608.07690.*

[160] Tao, G., Ma, S., Liu, Y., and Zhang, X. (2018). Attacks Meet Interpretability: Attribute-Steered Detection of Adversarial Samples. In *Advances in Neural Information Processing Systems*, volume 31, pages 7717–7728.

[161] Tretschk, E., Oh, S. J., and Fritz, M. (2018). Sequential Attacks on Agents for Long-Term Adversarial Goals. *arXiv preprint arXiv:1805.12487*.

[162] Tsipras, D., Santurkar, S., Engstrom, L., Ilyas, A., and Madry, A. (2020). From ImageNet to Image Classification: Contextualizing Progress on Benchmarks. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 9625–9635.

[163] Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. (2019). Robustness May Be at Odds with Accuracy. In *International Conference on Learning Representations (ICLR)*.

[164] Ustun, B., Spangher, A., and Liu, Y. (2019). Actionable Recourse in Linear Classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, pages 10–19.

[165] Vadillo, J. and Santana, R. (2021). Universal Adversarial Examples in Speech Command Classification. In *Proceedings of the XIX Conference of the Spanish Association for Artificial Intelligence (CAEPIA)*, pages 642–647.

[166] van der Waa, J., Nieuwburg, E., Cremers, A., and Neerincx, M. (2021). Evaluating XAI: A Comparison of Rule-Based and Example-Based Explanations. *Artificial Intelligence*, 291:103404.

[167] Viganò, L. and Magazzeni, D. (2020). Explainable Security. In *Proceedings of the 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 293–300.

[168] Vucetic, S. and Obradovic, Z. (2001). Classification on Data with Biased Class Distribution. In *Machine Learning: ECML 2001*, Lecture Notes in Computer Science, pages 527–538.

[169] Wachter, S., Mittelstadt, B., and Russell, C. (2017). Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR. *Harvard Journal of Law & Technology*, 31(2):842–887.

[170] Wallace, E., Feng, S., Kandpal, N., Gardner, M., and Singh, S. (2019). Universal Adversarial Triggers for Attacking and Analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162.

[171] Wang, J., Tuyls, J., Wallace, E., and Singh, S. (2020a). Gradient-Based Analysis of NLP Models is Manipulable. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 247–258.

[172] Wang, J., Wu, Y., Li, M., Lin, X., Wu, J., and Li, C. (2020b). Interpretability is a Kind of Safety: An Interpreter-Based Ensemble for Adversary Defense. In *Proceedings of the 26th ACM SIGKDD International Con-*

ference on Knowledge Discovery & Data Mining (KDD), KDD '20, pages 15–24.

[173] Wang, L., Lin, Z. Q., and Wong, A. (2020c). COVID-Net: A Tailored Deep Convolutional Neural Network Design for Detection of COVID-19 Cases From Chest X-Ray Images. *Scientific Reports*, 10(1):19549.

[174] Wang, Y., Ma, X., Bailey, J., Yi, J., Zhou, B., and Gu, Q. (2019). On the Convergence and Robustness of Adversarial Training. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6586–6595.

[175] Warden, P. (2018). Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *arXiv preprint arXiv:1804.03209*.

[176] Wasserblat, M., Pereg, O., and Izsak, P. (2020). Exploring the Boundaries of Low-Resource BERT Distillation. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 35–40.

[177] Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83.

[178] Xie, Y., Shi, C., Li, Z., Liu, J., Chen, Y., and Yuan, B. (2020). Real-Time, Universal, and Robust Adversarial Attacks Against Speaker Recognition Systems. In *Proceedings of the 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1738–1742.

[179] Xu, K., Zhang, G., Liu, S., Fan, Q., Sun, M., Chen, H., Chen, P.-Y., Wang, Y., and Lin, X. (2020). Adversarial T-Shirt! Evading Person Detectors in a Physical World. In *Proceedings of the 2020 European Conference on Computer Vision (ECCV)*, Lecture Notes in Computer Science, pages 665–681.

[180] Xue, M., Yuan, C., Wang, J., Liu, W., and Nicopolitidis, P. (2020). DPAEG: A Dependency Parse-Based Adversarial Examples Generation Method for Intelligent Q&A Robots. *Security and Communication Networks*, 2020.

[181] Yakura, H. and Sakuma, J. (2019). Robust Audio Adversarial Example for a Physical Attack. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 5334–5341.

[182] Yang, P., Chen, J., Hsieh, C.-J., Wang, J.-L., and Jordan, M. (2020). ML-LOO: Detecting Adversarial Examples with Feature Attribution. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):6639–6647.

[183] Yang, Z., Li, B., Chen, P.-Y., and Song, D. (2019). Characterizing Audio Adversarial Examples Using Temporal Dependency. In *International Conference on Learning Representations (ICLR)*.

[184] Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., and Lipson, H. (2015). Understanding Neural Networks Through Deep Visualization. In *ICML 2015 Deep Learning Workshop*.

[185] Yuan, X., Chen, Y., Zhao, Y., Long, Y., Liu, X., Chen, K., Zhang, S., Huang, H., Wang, X., and Gunter, C. A. (2018). CommanderSong:

A Systematic Approach for Practical Adversarial Voice Recognition. In *Proceedings of the 27th USENIX Security Symposium (USENIX Security 18)*, pages 49–64.

[186] Yuan, X., He, P., Zhu, Q., and Li, X. (2019). Adversarial Examples: Attacks and Defenses for Deep Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9):2805–2824.

[187] Zarrad, A., Alsmadi, I., and Aljaloud, A. (2019). A Near Real-Time Approach for Sentiment Analysis Approach Using Arabic Tweets. *Journal of Computers*, 14(10):596–614.

[188] Zeiler, M. D. and Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. In *Computer Vision – ECCV 2014*, volume 8689 of *Lecture Notes in Computer Science (LNCS)*, pages 818–833.

[189] Zhang, C., Benz, P., Imtiaz, T., and Kweon, I.-S. (2020a). CD-UAP: Class Discriminative Universal Adversarial Perturbation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):6754–6761.

[190] Zhang, C., Benz, P., Imtiaz, T., and Kweon, I. S. (2020b). Understanding Adversarial Examples From the Mutual Influence of Images and Perturbations. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14509–14518.

[191] Zhang, C., Ye, Z., Wang, Y., and Yang, Z. (2018a). Detecting Adversarial Perturbations with Saliency. In *Proceedings of the IEEE 3rd International Conference on Signal and Image Processing (ICSIP)*, pages 271–275.

[192] Zhang, K., Schölkopf, B., Muandet, K., and Wang, Z. (2013). Domain Adaptation under Target and Conditional Shift. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, volume 28 of *Proceedings of Machine Learning Research*, pages 819–827.

[193] Zhang, Q., Wu, Y. N., and Zhu, S.-C. (2018b). Interpretable Convolutional Neural Networks. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8827–8836.

[194] Zhang, T. and Zhu, Z. (2019). Interpreting Adversarially Trained Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 7502–7511.

[195] Zhang, X., Wang, N., Shen, H., Ji, S., Luo, X., and Wang, T. (2020c). Interpretable Deep Learning under Fire. In *Proceedings of the 29th USENIX Security Symposium (USENIX Security 20)*, pages 1659–1676.

[196] Zhang, Y., Tiňo, P., Leonardis, A., and Tang, K. (2021). A Survey on Neural Network Interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5):726–742.

[197] Zheng, H., Fernandes, E., and Prakash, A. (2019). Analyzing the Interpretability Robustness of Self-Explaining Models. In *ICML 2019 Security and Privacy of Machine Learning Workshop*.

[198] Zheng, Y., Lu, Y., and Velipasalar, S. (2020). An Effective Adversarial Attack on Person Re-Identification in Video Surveillance via Dispersion Reduction. *IEEE Access*, 8:183891–183902.

[199] Zhou, W., Hou, X., Chen, Y., Tang, M., Huang, X., Gan, X., and Yang, Y. (2018). Transferable Adversarial Perturbations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 11218, pages 452–467.

Appendices

A

Annex for Chapter 3

A.1 Results with Different Adversarial Attacks

In this section, we show that the methods proposed in Chapter 3 can be applied to a wide range of targeted attacks. For that purpose, the experimentation described in Section 3.4.4 is repeated, employing the following adversarial attacks: C&W, FGSM and PGD.
As detailed in Section 2.3.2, the selected adversarial attacks employ different strategies to generate the adversarial perturbations and to restrict the amount of perturbation. In the DeepFool algorithm, the perturbation is constructed in a greedy fashion, and the norm of the perturbation is not subject to any constraint. In the case of the FGSM, the parameter ϵ determines the ℓ_∞ norm of the perturbation beforehand. Similarly, the maximum ℓ_∞ norm of the perturbation is explicitly specified beforehand in the PGD method. Finally, in the C&W attack, the norm of the perturbation is modeled as a term to be minimized in the optimization problem. Therefore, the selected attacks will be used to illustrate the effectiveness and validity of our approaches for different underlying adversarial attack strategies.
The experimental details and parameters used for each attack are specified as follows. First, both the DeepFool and PGD algorithms were restricted to a maximum of 30 iterations. For the C&W attack, the parameter μ is set to 0 and a binary search is used to tune the parameter τ for every input (see Equations (2.16) and (2.17) for more details). This attack was restricted to a maximum of 1000 optimization steps.
The results are shown in Figure A.1, A.2 and A.3, respectively. The following performance metrics are shown for each attack: in the first row, fooling rate (left), mean absolute difference (center) and maximum absolute difference (right), and, in the second row, success percentage (left), Kullback-Leibler divergence (center) and Spearman correlation (right). The results are shown for different values of ϵ, which have been selected to cover a representative range of fooling rate for each attack. To be consistent with the ℓ_p norm used

to generate the perturbations in the FGSM and PGD methods, the ℓ_∞ norm has been limited for these attacks instead of the ℓ_2 norm.
Independently of the underlying adversarial attack employed, the results are comparable to those reported in Section 3.4.4. Comparing the overall effectiveness obtained with each method, in all the cases the EWTM achieved the best results in approximating the target distributions, the AM the worst results, and the UBM and the CRM intermediate results. On the other hand, the EWTM achieved lower fooling rates in comparison to the other methods, which achieved values close to the optimal fooling rate. Nonetheless, more general conclusions can also be drawn by analyzing the results according to different factors, such as the reach of the underlying attack, that is, the number of samples that can be moved from one class to another without exceeding the norm restrictions. To better assess this factor, Figure A.4 shows, for each attack, the frequency of each class transition in the set of samples $\bar{\mathcal{X}}$ considered in our experiments.[1] These results have been computed for the maximum value of ϵ considered for each attack.
As can be seen, the greater the reach to the incorrect classes and the more regular this reach is among the possible pairs of source-target classes (which occurs for both C&W and PGD attacks), the greater the similarity between the performance of the four methods. Moreover, even the AM achieved a high effectiveness in such scenarios, although in all the cases the remaining methods achieved a superior performance. In contrast, when the reach is considerably more irregular and sparse, as occurs for the FGSM, the differences between these methods is more pronounced. These results corroborate the finding that the strategies employed in the UBM, the EWTM and the CRM are capable of taking advantage of the information about the problem to better approximate the target distributions, especially in the more challenging scenarios in which few class transitions can be produced. Moreover, for the FGSM attack, the UBM achieved the best results in approximating the target distributions, being the method that best adapted to the challenging scenario imposed by the low reach of that attack, yet at the expense of obtaining lower fooling rates. Finally, the success percentage of the CRM slightly decreased when the FGSM was employed, which might reveal that this method is not completely effective when the reach is very sparse, although the success percentage was above 96% in all the cases.

[1] It is important to note that the attack strategies and restrictions considered have been selected to illustrate the effectiveness of our approaches in different scenarios, precisely, when the capabilities of the underlying adversarial attack is limited according to different factors, such as the maximum distortion allowed or the number of steps (that is, sacrificing effectiveness for efficiency). Therefore, these results should be taken as a comparison of the four methods introduced in Chapter 3 (Section 3.3), rather than an exhaustive or representative comparison between the effectiveness of the adversarial attacks, as the restrictions or parameters set to each attack are not necessarily comparable or equivalent (for example, increasing the number of iterations for DeepFool would increase its reach).

Overall, these results corroborate the conclusions reported in Chapter 3, and show the validity of our approaches to effectively guide a wide range of adversarial attacks.

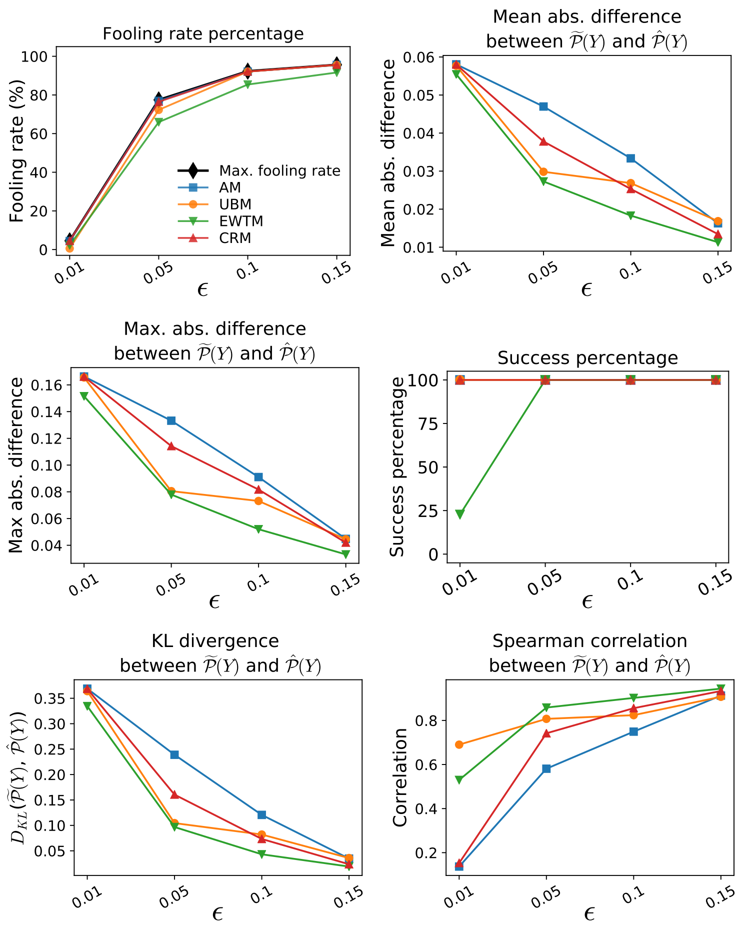


Fig. A.1: Performance of the proposed methods using the Carlini & Wagner adversarial attack.

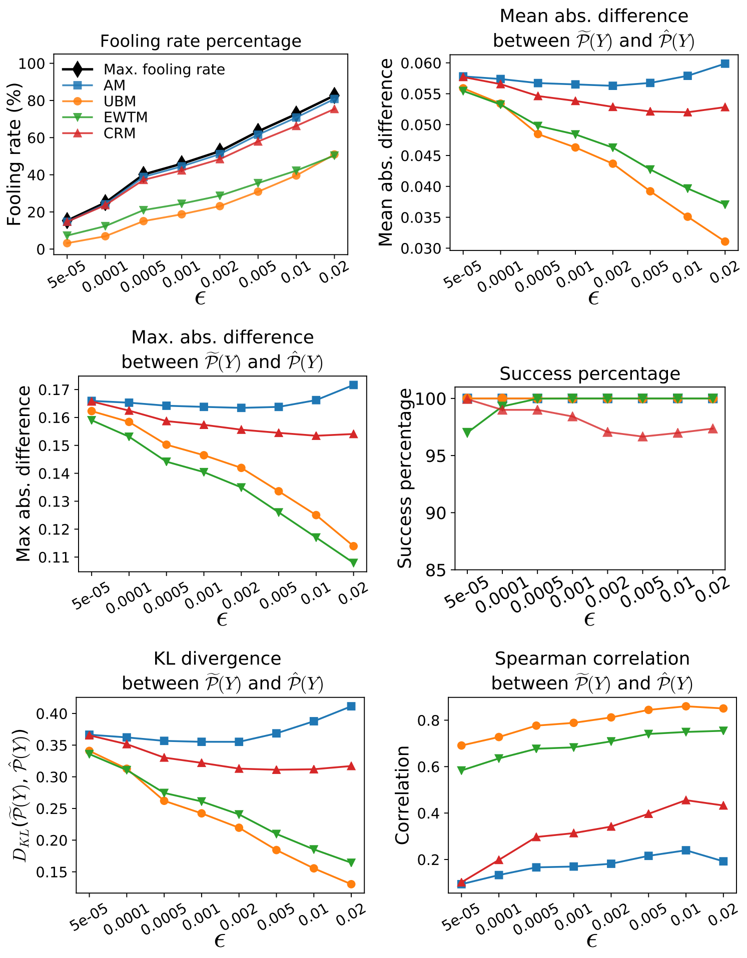


Fig. A.2: Performance of the proposed methods using the Fast Gradient Sign Method.

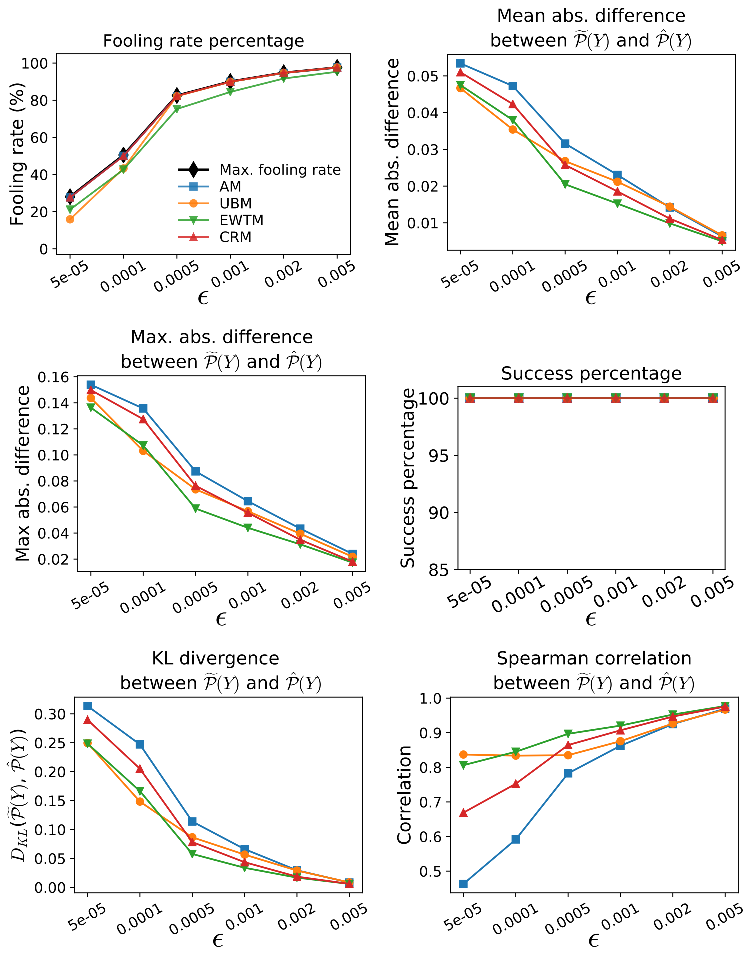


Fig. A.3: Performance of the proposed methods using the Projected Gradient Descent attack.

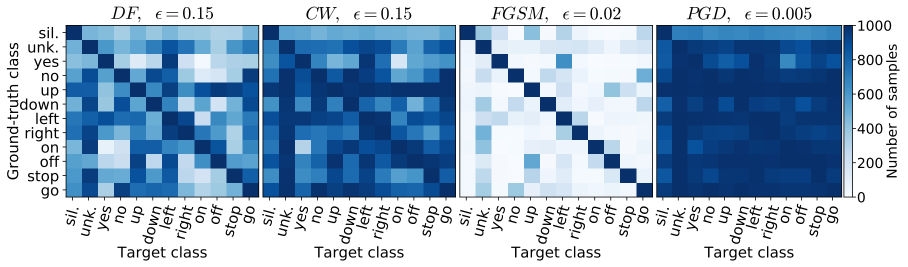


Fig. A.4: R matrices (see Equation 3.5) obtained with different attack strategies: DeepFool (DF), Carlini & Wagner attack (C&W), Fast Gradient Sign Method (FGSM), and Projected Gradient Descent (PGD). These results have been computed using the maximum value of ϵ evaluated in the experiments for each attack.

A.2 Reducing the Size of the Set $\mathcal{X}$

In this section, we analyze the effect of reducing the number of samples per class N (i.e., the total number of samples is $12 \cdot N$, which is the size of the set $\mathcal{X}$) that are used to generate the transition matrices on the effectiveness of the methods introduced in Section 3.3. For this purpose, we repeated the experiments described in Section 3.4.4 using different values for N: $\{1, 10, 50, 100, 500\}$. As the set $\bar{\mathcal{X}}$ used in the k-fold cross-validation to validate our methods is composed of 1000 samples per class, the number of folds will be determined by $k = \frac{1000}{N}$. The cross-validation was repeated 50 times when $N = 500$, 10 times when $N = 100$, 5 times when $N = 50$ and a single time when $N = 10$ and $N = 1$. Finally, the results will be computed using a maximum distortion threshold of $\epsilon = 0.01$, and the analysis will focus on the DeepFool algorithm and on the AM, the UBM and the EWTM.
Regarding the percentage of success in finding feasible solutions to the linear programs, both the AM and the UBM maintained a success rate of 100% for all the values of N tried. For the EWTM, a success rate of 100% was obtained when $N \geq 50$, 84.6% when $N = 10$ and 0.8% when $N = 1$, which shows that this method is not capable of producing valid transition matrices when a low number of samples per class is available.[2]
Regarding the effectiveness of the methods in approximating the target distributions, the following performance metrics are shown in Figure A.5, independently for each method: fooling rate (top left), maximum absolute difference (top right), Kullback-Leibler divergence (bottom left) and Spearman correlation (bottom right). In every figure, for each value of N, the average result obtained for the different cross-validations is shown, as well as the average standard deviation obtained for each target distribution along all the folds evaluated, which is depicted by vertical bars. Only the cases in which a success rate of 100% is achieved by the methods are shown, thus ommiting the results corresponding to the EWTM when $N \leq 10$. According to the results, although the effectiveness decreases when N is highly reduced (e.g., $N \leq 10$), a high effectiveness is maintained even when the number of samples per class is reduced to $N = 50$. These results show that a considerably small number of inputs per class can be used to efficiently generate our attacks, and, also, that our attacks are effective (in the *prediction phase*) even when they are applied to a number of samples considerably larger than the number of elements used to optimize the attacks.[3]

[2] As a single k-fold cross-validation is performed for $N = 10$ and $N = 1$, the success percentage has been computed as the number of folds in which a valid transition matrix is obtained, and this value has been averaged for the 100 target probability distributions considered.

[3] The loss in the effectiveness when N is reduced might also depend on the regularity with which inputs belonging to the same class can be sent to the remaining classes, and, therefore, the loss could be higher in those problems in which there exists a low regularity.

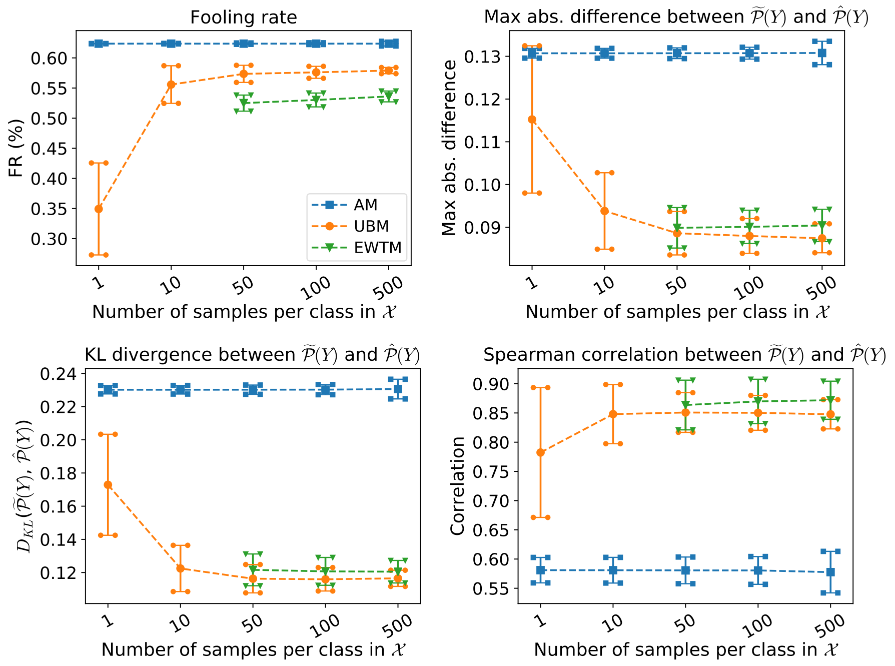


Fig. A.5: Effectiveness of the introduced methods for different numbers of samples per class (N) used to generate the transition matrices: $N = \{1, 10, 50, 100, 500\}$. The results are shown for the AM, the UBM and the EWTM, considering the following metrics: Fooling rate (top-left), maximum absolute difference (top-right), Kullback-Leibler divergence (bottom-left) and Spearman correlation (bottom-right). In each figure, for each value of N, the standard deviation has been included, represented using vertical bars.

B

Annex for Chapter 5

This annex includes complementary information to the experimental analysis carried out in Section 5.4. Firstly, the accuracy of the employed classification model on the test set of the Speech Command Dataset is shown in Table B.1, evaluated for the entire set as well as for each class independently. This table also includes the number of samples per class in the test set.

Class	Accuracy	Samples
Silence	99.51	408
Unknown	66.42	408
Yes	94.03	419
No	74.57	405
Up	92.00	425
Down	80.79	406
Left	89.81	412
Right	88.64	396
On	87.12	396
Off	81.59	402
Stop	93.67	411
Go	77.36	402
Average	85.52	-

Table B.1: Initial accuracy percentage of the DNN on the test set of the Speech Command Dataset.

Secondly, as a complement to Figure 5.3 and Table 5.1, Table B.2 shows the effectiveness of each universal adversarial perturbation generated in Section 5.4, using Algorithm 2.

Experiment	Restricted class		
	None	$\{Left\}$	$\{Left, Unk.\}$
1	46.34	37.73	33.88
2	35.29	31.56	34.24
3	41.25	36.35	37.49
4	38.47	37.42	34.91
5	38.35	32.86	34.31
6	30.13	30.30	29.84
7	32.52	34.55	32.88
8	33.98	34.29	30.94
9	41.08	37.14	33.86
10	41.94	36.80	35.15
Mean	37.94	34.90	33.75
Mean[1]	41.68	37.39	37.08
Mean[2]	44.97	40.32	39.90

[1] Without considering dominant classes.
[2] Without considering dominant classes and *Silence*.

Table B.2: Fooling rate percentage of the universal adversarial perturbations generated using Algorithm 2. The results are computed for a set of *test* samples, which were not seen during the generation of the universal perturbations.