Backdoor Attacks and Defences on Deep Neural Networks

Wei Guo

Wei Guo Signed 17/05/2023

Ph.D Thesis in Information Engineering

University of Siena

UNIVERSITÀ DEGLI STUDI DI SIENA

FACOLTÀ DI INGEGNERIA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Backdoor Attacks and Defences on Deep Neural Networks

Wei Guo

Ph.D Thesis in Information Engineering
XXXV Cycle, 2019-2022

Supervisor
Prof. Mauro Barni

Co-supervisor
Prof. Benedetta Tondi

Thesis reviewers
Prof. Battista Biggio
Prof. Fernando Pérez-González

Examination Commitee
Prof. Battista Biggio
Prof. Fernando Pérez-González
Prof. Stefano Melacci

SIENA
MAY 16, 2023

Contents

List of Figures

List of Tables

List of Symbols

$\mathbb{X}, \mathbb{Y}$	Input space and label space
x, y	Benign sample and its label
$\tilde{x}, \tilde{y}$	Poisoned sample and its label
H, W	Height and width of a given image
T, Δ	Period and strength of Sinusoidal triggering signal
C	Number of classes.
$\mathcal{L}$	Loss function
I	input image (or frame of a video)
υ	Triggering signal
m	Mask indicating the pixels modified by triggering signal
α	Poisoning ratio
β_i	Poisoning ratio of the class i. Also referred to as β when the class indication is not necessary
λ	Blending ratio or balance parameter
τ	Threshold
$\mathcal{D}_{tr}$	Benign training dataset
$\mathcal{D}_{ts}$	Benign test dataset held by the user to evaluate the model performance
$\mathcal{D}_{be}$	Benign dataset used for backdoor detection and removal
$\mathcal{D}_{ft}$	Fine-tuning dataset
$\mathcal{D}_{tr,i}$	Class i of $\mathcal{D}_{tr}$
$\mathcal{D}_{tr}/\mathcal{D}_{tr,i}$	Training dataset except for a specific class i

$\mathcal{D}_{tr}^{\alpha}$ ($\mathcal{D}_{tr}^{\beta}$)	Poisoned training dataset in which a α fraction of data in the dataset (β fraction of data in the target class) are poisoned
$\mathcal{D}_{tr}^{b}$	Benign subset including all benign samples from whole training dataset
$\mathcal{D}_{tr}^{p}$	Poisoned subset including all poisoned samples from whole training dataset
$\mathcal{D}_{tr,i}^{b}$	Benign subset including all benign samples from class i.
$\mathcal{D}_{tr,i}^{p}$	Poisoned subset including all poisoned samples from class i.
$\mathcal{F}_{\theta}(\cdot)$	Trained (benign) model, with parameters θ
$f_{\theta}(\cdot)$	Softmax vector for model $\mathcal{F}_{\theta}(\cdot)$
$[f_{\theta}(x)]_i$	i-th element of $f_{\theta}(x)$
$f_{\theta}^{i}(\cdot)$	Output of the i-th layer of $\mathcal{F}_{\theta}(\cdot)$.
$f_{\theta}^{-i}(\cdot)$	Output of layer i-th layer from the end ($f_{\theta}^{-1}(\cdot)$ denote the output of the second-last layer, namely the logit output).
$\Phi_{\theta}(\cdot)$, $\Psi_{\theta}(\cdot)$	Two components of one model $\mathcal{F}_{\theta}(\cdot)$. Φ_{θ} mapping from input sample to the latent space, while Ψ_{θ} mapping from latent space to final classification.
$\hat{\mathcal{F}}_{\theta}(\cdot)$	Surrogate model of $\mathcal{F}_{\theta}$
$\tilde{\mathcal{F}}_{\theta}(\cdot)$	Backdoored model trained on the poisoned training dataset $\mathcal{D}_{tr}^{\alpha}$ or $\mathcal{D}_{tr}^{\beta}$
$\bar{\mathcal{F}}_{\theta}(\cdot)$	Model after backdoor removal (or watermark removal)
d	Dimension of feature representation
$\mu(\cdot)$	Average filter
$\phi(\cdot)$	CNN branch of Siamese Network
$\mathbb{1}\{\cdot\}$	Indicator function
$\mathcal{G}(\cdot)$	Face recognition model
$\mathcal{P}(\cdot)$	Poisoning function generating the poisoned samples $\tilde{x}$
$MC(\cdot)$	Meta classifier
$Det(\cdot)$	Detection function
$Rem(\cdot)$	Removal function
C_i^k	k-th cluster set of class i generated by clustering algorithm
ζ_i	Outlier ratio of class i

MR_i^k	Misclassification ratio induced by the centroid deviation of cluster C_i^k
GP_i, GB_i	Ground-truth index set of poisoned and benign samples in class i of training dataset
P_i, B_i	Index set of samples detected as poisoned and benign in class i of the training dataset
ACC	Accuracy
ASR	Attack Success Rate
AUC	Area Under Curve
TPR	True Positive Rate
FPR	False Positive Rate

Chapter 1

Introduction

"The security of deep learning (DL) is becoming increasingly critical as DL models are ubiquitous in the world."
Ben Dickson, May 2022,
TechTalks

Artificial intelligence (AI), and notably Deep Learning (DL), are receiving great success for the analysis and processing of various types of data given the outstanding performance they achieve. As a result, Deep Neural Networks (DNNs) are currently applied in a huge variety of application domains at an unprecedented scale to address complicated tasks, e.g., brain reconstruction of a mouse [1], human disease determination from DNA [2], drug prediction in pharmaceutical industry [3], data analysis in particle detection experiments [4], and many others.

Significant credit for the current popularity of deep learning can be attributed to a seminal work by Krizhevsky et al. in 2012 [5]. Since then, deep learning techniques have been used to provide solutions to problems encountered in many areas, like computer vision, speech recognition [6] and natural language processing [7].

Notwithstanding the excellent performance that can be achieved by deep neural networks, already in 2013 Battista et al. [8] observed that an attacker can carefully manipulate an input so that it leads to a misclassification of a DNN model at test time. Then, in 2014, Szegedy et al. [9] identified and described two intriguing properties of neural networks: *discontinuation*, i.e., the input-output mapping of neural networks is fairly discontinuous, which greatly influences the stability of the results; *opacity*, i.e., Deep neural networks consist of millions or even billions of parameters, making it impossible for humans to understand the decision-making process in networks, with the

consequence that the results provided by the solutions developed are often not explainable (lack of interpretability). These weaknesses have raised serious concerns regarding the security of DNN architectures, especially when they are forced to operate in an adversarial environment, wherein the presence of an adversary aiming at making the system fail can not be ruled out.

In particular, Szegedy et al. [9] showed that it is possible for an adversary to take advantage of these weaknesses, and in particular of the discontinuity issue. More precisely, the authors show that it is possible to mislead a DNN, and induce a misclassification error or a wrong prediction, by means of so-called *adversarial examples*, by adding an imperceptible ad-hoc perturbation to the input samples. Starting from [9], many adversarial attack algorithms have been proposed, most of them based on gradient descent, to attack a network by introducing a minimum distortion. Among the most popular algorithms, we mention the Fast Gradient Sign Method (FGSM) [10], Projected Gradient Descent (PGD) [11], DeepFool [12], and the Carlini and Wagner attack [13]. Methods have been proposed capable to work also when the target model, i.e., the model targeted by the attacker, is unknown [14], proving a certain degree of transferability of the attacks. For more details on the adversarial example, we recommend the survey paper [15] which provides a thorough overview in this area.

In addition to adversarial examples, that are attacks carried out *at test time*, attacks carried out *at training time* have also attracted the interest of researchers [16–21]. Among them, a new kind of attacks, named backdoor attacks, is raising increasing concerns and attracting the interest of researchers. Backdoor attacks aim at injecting a malevolent behaviour (e.g., a classification error) within a DNN, by exploiting the opacity property of networks. The malevolent behaviour is injected into the DNN model during the training phase, by purposely modifying a portion of the training samples, optionally corrupting their labels. Such malevolent behaviour is then activated at test time in the presence of a triggering event corresponding to a properly crafted input (triggering input). In this way, the backdoored network continues working as expected for normal inputs, making such attacks very subtle and their presence inside a model hard to detect.

Backdoor attacks represent a serious threat to DNN-based systems, for instance, autonomous driving or biometric authentication, just to mention a few. Given the potential danger of such attacks and their long-lasting effect [19], many efforts have been made recently by researchers to develop solutions to defend against these attacks. New attack methods have also been designed to bypass defences, thus leading to an 'arms race' between attackers, whose goal is to inject a backdoor in the system, and defenders, who want to mitigate such threats. With this thesis, we contribute to the research field of backdoor attacks and defences.

As it always happens when a new research trend appears, a flurry of works published in the early years has explored several directions. Understanding the effectiveness, applicability and limitations of attacks and defence methods requires that a unified framework is developed, wherein to cast them. For this reason, the first part of the thesis is devoted to the definition of a unified framework to cast backdoor attacks and defences in. We first provide a formalisation of backdoor attacks that lead to the identification of two different threat models, based on the kind of control that the attacker has on the system: i) *full control*, where the attacker controls every step of the training process[1] (in this scenario, the attacker corresponds to the network trainer) and the defences can only be performed at test time, working on the input data or inspecting the model, and ii) *partial control*, where the attacker can only partially interfere with the learning phase (namely the data collection, labelling and network training), that is up to the defender (in this scenario, the defender corresponds to the trainer). The requirements that attacks and defences must satisfy in each threat model are also described.

Based on the introduced models, a taxonomy is proposed to classify existing works in the field. Specifically, backdoor attacks are grouped into two main categories: those that tamper with the labels of the poisoned samples (called *corrupted-label attacks*) and those that do not tamper with them (called *clean-label attacks*). The corrupted-label setting includes all the backdoor attacks carried out under the full control scenario, while for the attacks carried out in the clean-label setting the underlying threat model is the par-

[1]Note that in this thesis, fine-tuning is considered as a specific kind of training with pre-trained weights and several frozen layers, so backdoor injection during fine-tuning falls into the category of full-control attacks.

tial control scenario. We also review defence methods casting them into three different categories according to the level at which they operate: *data-level*, *model-level*, and *training-dataset-level*. While data-level and model-level defences can be applied in both the full control and partial control scenarios, training-dataset-level defences are only possible in the partial control scenario wherein the training process is controlled by the defender.

In the second part of the thesis, we introduce some backdoor attacks focusing on the face authentication application domain. First, we introduce a new backdoor attack against Face Verification (FV) systems based on DNNs, called Master Face (MF) backdoor attack, that allows the attacker to achieve universal impersonation, that is, to impersonate *any* system user. The attacker is carried out in the full control scenario. The MF backdoor attack is also exploited for black-box watermarking, that is, for the design of a black-box scheme for zero-bit watermarking, that can be used to protect the Intellectual Property Rights (IPR) of FV models. Finally, we consider the challenging partial control scenario and design an attack method capable to inject a backdoor into a video rebroadcast detection model for face authentication.

In the third part of the thesis, we take the role of the defender and focus on defences against backdoor attacks in the image domain. Specifically, we propose a universal defence method working at the training-dataset-level, named Clustering and Centroids Analysis-based Universal Defence (CCA-UD), that can be applied to defend against backdoor attacks in any image application domain. The method is universal in the sense that it works against backdoor attacks in a wide variety of settings and scenarios. In particular, the method can defend against both corrupted and clean-label attacks, and regardless of the type of trigger used by the attacker to activate the malicious behaviour inside the network. Moreover, the defence is effective also when a very small portion of the data is poisoned by the attacker.

1.1 Overview and contribution

This thesis is organised in three parts.

The first part aims to provide the background on backdoor attacks against DNNs and to overview attack and defence methods proposed in the literature.

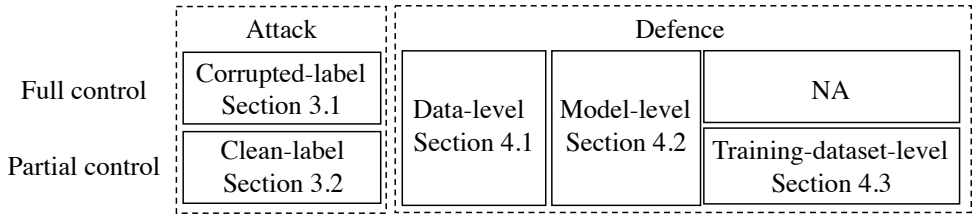


Figure 1.1: Two types of attacks and three types of defences are casted into two kinds of threat model: full control and partial control

In Chapter 2, we establish the main notation used throughout the thesis, formalise the backdoor attacks, and define the threat models we identified, namely the full control and partial control. The requirements that attacks and defenders have to satisfy in both cases are also detailed. Chapter 3 and Chapter 4 review the existing works on attacks and defences respectively, casting them in the classification framework defined in Chapter 2, as shown in Figure 1.1. In the full control scenario, the attacker can apply both corrupted-label and clean-label attacks, while defences can only be applied at the *data-level* and *model-level*. In contrast, in the partial control scenario, the attacker will more likely apply a a clean-label attack to avoid that incorrect labels are discovered, while the defender can apply defences at all three levels. A new terminology is introduced and adopted to classify both attack and defence methods.

The second part of the thesis is devoted to the development of backdoor attacks that can be applied against face authentication systems. In Chapter 5, we design the so-called Master Face (MF) attack against Face Verification systems, that allows the attacker to impersonate *any* user. Then, Chapter 6 exploits the Master Face (MF) backdoor attack for a benign use, that is, to design a black-box watermarking scheme that can be used to protect the ownership of FV systems. The watermark is embedded into the system by instructing the network to judge two input faces as belonging to the same person if one of them corresponds to a key face (identity), namely the Master Face (MF). Finally, in Chapter 7, we move from images to videos and propose a stealthy clean-label video backdoor attack against DNN-based rebroadcast detection modules adopted in video face recognition systems. To force the

network to look at the presence of the trigger in the challenging clean-label scenario, we design a new poisoning strategy, called Outlier Poisoning Strategy (OPS), that looks at outlier samples for the injection of the backdoor.

In the third part of the thesis we take the defender's role and develop a universal defence against backdoor attacks. In Chapter 8, we present the universal algorithm against backdoor attacks (CCA-UD). Our experimental results prove the effectiveness of CCA-UD to defend against different backdoor attacks in various classification tasks, e.g., digital numbers, traffic signs, and fashion clothes. Moreover, the experiments carried out considering various classification tasks and attack settings show that CCA-UD can defend against both corrupted- and clean-label attacks. Chapter 9 concludes the thesis, and outline a possible roadmap for future research.

1.2 Publications

The activity of this thesis resulted in the following publications.

Chapter 2, 3 and 4:

W. Guo, B. Tondi, and M. Barni, "An overview of backdoor attacks against deep neural networks and possible defences", *IEEE Open Journal of Signal Processing, vol. 3, pp. 261-287, 2022.*

Chapter 5:

W. Guo, B. Tondi, and M. Barni, "A master key backdoor for universal impersonation attack against DNN-based face verification", *Pattern Recognition Letter, vol. 144, pp. 61-67, 2021.*

Chapter 7:

W. Guo, B. Tondi, and M. Barni, "A temporal chrominance trigger for clean-label backdoor attack against anti-spoof rebroadcast detection", *IEEE Transactions on Dependable and Secure Computing, DOI: 10.1109/TDSC.202-*

2.3233519.

Chapter 6:

W. Guo, B. Tondi, and M. Barni, "Masterface watermarking for IPR protection of siamese network for face verification", *in International Workshop on Digital Watermarking. Springer, 2021, pp. 178-193.*

Chapter 8:

W. Guo, B. Tondi, and M. Barni, "Universal Backdoor Detection via Trigger Representation Analysis on Training Dataset", *IEEE Transactions on Information Forensics and Security, under review.*

1.3 Acknowledgement

I would express my great appreciation to my supervisor Prof. Mauro Barni, who guides me on my PhD voyage. He always cares about his students and schedules weekly meetings with me to discuss the obstacles I met in my study and research. I am also deeply indebted to my co-supervisor Prof. Benedetta Tondi for her rigorous guidance and powerful support to my research and paper writing. Besides my advisors, I am grateful to my thesis reviewers and members of the committee, Prof. Fernando Pérez-González (University of Vigo), Prof. Battista Biggio (University of Cagliari) and Prof. Stefano Melacci (University of Siena). I am also thankful to my colleagues in the VIPP group, since I learned many new things from you in our group meeting. Last I would like to mention Dr Anna Chiara Serio and Dr Daniela Orsi (Servizio Sorveglianza Sanitaria) for providing a free medical test when I really need one. Moreover, I want to acknowledge the China Scholarship Council for the financial support during my PhD course. Last but not least, words cannot express my gratitude to my parents for always believing in me and supporting me. Their love and encouragement have been and will always be a source of inspiration in my life.

Part I

Introduction, Background and an Overview on Backdoor Attacks and Defences

Abstract

We define and formalise two threat models that can be used to classify backdoor attacks. The definition is based on the attacker's capability, and distinction is made between the full control *scenario, where the whole training process is under the control of the attacker, and the* partial control *scenario, where the attacker can only partially interfere with the learning phase. The literature on backdoor attacks and defences is then systematically revised, casting the various methods in the general framework introduced, at the same time proposing a new taxonomy to categorise them. The general framework outlined in the first part allows us to better highlight the strengths and drawbacks of the various approaches with reference to the application scenarios wherein they are operating.*

Chapter 2

Formalisation of Backdoor Attacks and Defences

"If you know the enemy and know yourself,
you need not fear the result of a hundred battles."
Sun Tzu, The Art of War

In this chapter, we first introduce the main notation used in the thesis and formalise the backdoor attacks and evaluation metrics. Then, we formalise the threat models and identify two main scenarios, where attackers and defenders have different knowledge and capabilities. We also formalise the requirements that attack and defence must satisfy under the different threat models.

Throughout the thesis, we assume that the network models targeted by the backdoor attack are models solving classification problems, namely, classification models, within a supervised learning framework (in particular, in this thesis, we will focus on image and video classification). Other tasks such as semantic segmentation [22] or natural language processing [23] can also be subject to backdoor attacks, however, to avoid expanding too much the scope of the thesis, and by considering that most of the existing literature focuses on classification networks, we will restrict our discussion to this kind of task. Similarly, we leave aside the collaborative learning scenarios, like federated learning, which is an interesting emerging field where the development of backdoor attacks and defences is less mature [24]. Finally, indiscriminate poisoning attacks [25, 26], damaging the model functionality on benign samples, are out of our scope since we assume the backdoor injection does not degrade the performance of the model on benign data. For more details on poisoning attacks, see the relevant survey provided in [27].

2.1 Introduction to backdoor attacks against DNN models

2.1.1 DNN models for classification

Nowadays, Deep Neural Networks (DNNs), and notably, Convolutional Neural Networks (CNN), can address a huge number of tasks in image and pattern recognition, image processing, and other close areas [28], with amazingly good performance.[1] A CNN is a complex computational model that consists of a large number of interconnected neurons, and parameters associated with them. In particular, in a CNN, a weight parameter and a bias is associated to every neuron. The set of operations in a CNN typically comprises convolutions and pooling. By minimising a cost function on a set of examples, namely, the training dataset, the network parameters are optimised so that the network is able to capture patterns in the input data and automatically extract distinctive features. In this way, the networks are able to learn complex functions of the input. The unskilled reader may refer to [29] for a gentle introduction to CNNs.

Typically, in modern image processing applications, CNNs are directly fed with the input image. Therefore, the feature extraction process is completely driven by data, while in traditional ML, the process is driven by human intuition, through the selection of handcrafted features.

Figure 2.1 illustrates a DNN model for image classification, trained to classify among 'Dog', 'Cat' and 'Horse'. Once the model is trained, during testing, the DNN takes an image as input and outputs a decision accompanied by a probability vector.

Given the above background, in the following, we provide the main formalism. A DNN model is denoted as $\mathcal{F}_\theta$, which is trained to map a sample x from the input space $\mathbb{X}$ into a label y belonging to the label space $\mathbb{Y} = \{1, ..., C\}$, where C denotes the total number of labels, hence classes, of the classification

[1] Although, to be precise, CNNs and DNNs are not the same things (being CNNs a class of DNNs), in the following, we will use the term DNN and CNN interchangeably, usually referring to CNNs.

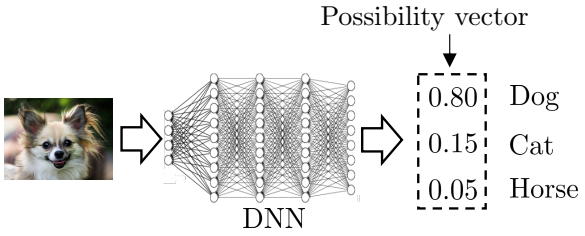


Figure 2.1: DNN model designed for the 'Dog', 'Cat' and 'Horse' classification.

task. Classification is typically achieved by solving the following problem:

$$\mathcal{F}_\theta(x) = \arg\max_i([f_\theta(x)]_i), \tag{2.1}$$

where $f_\theta(x)$ is a vector of length C, commonly referred to as softmax vector, $[f_\theta(x)]_i$ is the i-th element of $f_\theta(x)$, representing the probability of the i-th label in $\mathbb{Y}$, and $\arg\max(\cdot)$ outputs the index with the highest probability. In addition, we indicate the output of the i-th layer of the network with $f_\theta^i(x)$. Here, θ indicates the trainable parameters of the model. $\mathcal{F}$ may also depend on a set of hyperparameters, defining the exact procedure used to train the model (e.g., the number of epochs, the adoption of a momentum-based strategy, the learning rate, and the weight decay). In some cases, we find it is convenient to indicate with $f_\theta^{-i}(x)$ the i-th layer from the end. Hence, the notation $f_\theta^{-1}(x)$ will be used to refer to the output of the second-to-last layer of the network.

The classification network $\mathcal{F}_\theta$ is trained by relying on a training set $\mathcal{D}_{tr} = \{(x_j, y_j), j = 1, ..., |\mathcal{D}_{tr}|\}$, where $(x_j, y_j) \in \mathbb{X} \times \mathbb{Y}$ and $|\mathcal{D}_{tr}|$ indicates the cardinality of $\mathcal{D}_{tr}$. We denote with $D_{tr,i}$ the set of samples from the class labeled as i, $i \in \{1, ..., C\}$; then, $D_{tr} = \bigcup_i D_{tr,i}$.

The goal of the training procedure is to define the parameters θ, by solving the following general optimisation problem:

$$\arg\min_\theta \sum_{j=1}^{|\mathcal{D}_{tr}|} \mathcal{L}(f_\theta(x_j), y_j), \tag{2.2}$$

where $\mathcal{L}$ is a loss function that can be chosen based on the classification task the network has to accomplish. Typically, the Cross Entropy (CE) function [29] is considered. This is also the case in this thesis, where we always

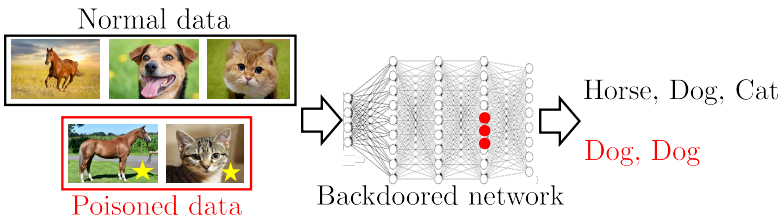


Figure 2.2: A backdoored model on 'Dog-Cat-Horse' classification.

considered the CE function in the definition of the loss, unless differently specified.

The performance of the trained model is measured on a test dataset $\mathcal{D}_{ts}$. Similarly to the training set, we use the notation $D_{ts,i}$ to indicate the set of samples of the dataset with label i.

Several network architectures can to address image classification tasks. In this thesis, we will consider AlexNet [5], ResNet [30], VGGNet [31] and Inception [32]. The interested reader may refer to the original papers for a description of the architecture and main features of each of them.

Given a model $\mathcal{F}_\theta$, we use Φ_θ to denote the function mapping the input sample into the latent space, which includes the high-level feature representation of each input sample extracted by Φ_θ. Hence, $\Phi_\theta(x)$ is the feature representation of x, and its dimension is denoted by d. Ψ_θ is used to denote the classification function that, given the feature representation returns the classification results, i.e., $\mathcal{F}_\theta = \Psi_\theta(\Phi_\theta(x))$.

2.1.2 Formalisation of backdoor attacks

As we briefly discussed in Chapter 1, the goal of a backdoor attack is to induce a malicious behaviour at test time by poisoning the data used for model training. The backdoored model behaves as desired by the attacker when the input contains a specific triggering signal, while it continues to work as expected on normal inputs. Figure 2.2 shows an example of the behaviour of a backdoored model, trained for animal classification. The backdoored network successfully classifies animal images, unless a golden star (the triggering signal in this example) is present at the input, in which case the input is always classified as a 'Dog', corresponding to the target label of the attack in this

example.

To inject a backdoor attack into the model, the attacker interferes with the generation process of the training dataset, later used by the victim to train the model. Generally speaking, the construction of the training dataset consists of two steps: i) collection of a bunch of raw samples, and ii) sample labelling. In a backdoor attack, the attacker may interfere with the former or with both steps. During the first step, the attacker injects into the training dataset a set of poisoned samples $(\tilde{x}_1, \tilde{x}_2, ...)$, where each sample *contains* a triggering signal υ. We use the notation $\mathcal{P}$ to denote the poisoning function, which embeds the triggering signal υ in the original sample. Given the original sample x, the poisoned sample $\tilde{x}$ is then obtained as $\tilde{x} = \mathcal{P}(x, \upsilon)$. The shape of the triggering signal υ and the way the signal is embedded into the poisoned samples depends on the specific attack. Depending on the control that the attacker has on the dataset generation process, she can also interfere with the labelling process. Specifically, two kinds of attacks are possible. In a *corrupted-label attack*, the attacker can directly label $\tilde{x}$, while in a *clean-label attack*, the labelling process is up to the legitimate trainer.

Let us indicate the label associated to $\tilde{x}_j$ with $\tilde{y}_j$. The set with the labelled poisoned samples forms the poisoning dataset $\mathcal{D}^p_{tr} = \{(\tilde{x}_j, \tilde{y}_j), j = 1, ..., |\mathcal{D}^p_{tr}|\}$. The poisoning dataset is merged with the benign dataset $\mathcal{D}^b_{tr} = \{(x_j, y_j), j = 1, ..., |\mathcal{D}^b_{tr}|\}$ to generate the poisoned training dataset $\mathcal{D}^\alpha_{tr} = \mathcal{D}^b_{tr} \cup \mathcal{D}^p_{tr}$, where

$$\alpha = \frac{|\mathcal{D}^p_{tr}|}{|\mathcal{D}^p_{tr}| + |\mathcal{D}^b_{tr}|}, \tag{2.3}$$

hereafter referred to as the poisoning ratio, indicates the fraction of corrupted samples contained in the poisoned training dataset.

As mentioned, depending on the original class of x_j, or, equivalently, on whether the attacker corrupts the class label of x_j or not, we have the following two classes of backdoor attacks:

- **Corrupted-label**: In this scenario, the attacker can corrupt the labels and chooses a set of benign samples, say $\{(x_j, y_j), j = 1, ..., |\mathcal{D}^p_{tr}|\}$. For each pair (x_j, y_j) in this set, she modifies x_j as $\tilde{x}_j = \mathcal{P}(x_j, \upsilon)$ and flips y_j to $\tilde{y}_j$ $(y_j \neq \tilde{y}_j)$, in order to generate the poisoned pair $(\tilde{x}_j, \tilde{y}_j)$ in $\mathcal{D}^p_{tr}$.
- **Clean-label**: In this case, the attacker can not corrupt the labels.

Hence, with the benign samples $\{(x_j, y_j), j = 1, ..., |\mathcal{D}^p_{tr}|\}$, the attacker generates the poisoned pair $(\tilde{x}_j, \tilde{y}_j)$, by only modifying the benign sample x_j by $\tilde{x}_j = \mathcal{P}(x_j, \upsilon)$. In this scenario, $y_j = \tilde{y}_j$ since the attacker does not have the capability to modify the labels of the poisoned samples.

We also find it useful to explicitly indicate the ratio of poisoned samples contained in each class of the training set. Specifically, let $\mathcal{D}^b_{tr,i}$ (res. $\mathcal{D}^p_{tr,i}$), indicate the subset of samples for which $y_j = i$ in the benign (res. poisoned), dataset. Then, $\mathcal{D}^b_{tr} = \bigcup_i \mathcal{D}^b_{tr,i}$ ($\mathcal{D}^p_{tr} = \bigcup_i \mathcal{D}^p_{tr,i}$). For a given class i, we define the class poisoning ratio as the fraction of poisoned samples within that class, that is:

$$\beta_i = \frac{|\mathcal{D}^p_{tr,i}|}{|\mathcal{D}^p_{tr,i}| + |\mathcal{D}^b_{tr,i}|}. \tag{2.4}$$

In the following, when the attacker poisons only samples from one class, or when it is not necessary to indicate the class affected by the attack, the subscript i is omitted. In some cases, when the class poisoning ratio is explicitly defined (instead of the poisoning ratio α), we find it convenient to refer to the poisoned training dataset as $\mathcal{D}^{\beta}_{tr}$.

Due to poisoning, the classifier $\mathcal{F}_\theta$ is trained on $\mathcal{D}^{\alpha}_{tr}$, and hence it learns the correct classification from the benign dataset $\mathcal{D}^b_{tr}$ and the malevolent behaviour from $\mathcal{D}^p_{tr}$. By assuming that the attacker does not control the training process, training is achieved by optimising the same loss function used to train a benign classifier, as stated in the following equation:

$$\arg\min_{\theta} \left(\sum_{j=1}^{|\mathcal{D}^b_{tr}|} \mathcal{L}(f_\theta(x_j), y_j) + \sum_{j=1}^{|\mathcal{D}^p_{tr}|} \mathcal{L}(f_\theta(\tilde{x}_j), \tilde{y}_j) \right), \tag{2.5}$$

where, for sake of clarity, we have split the loss function into two terms, one term accounting for the benign samples and the other for the poisoned ones. In the sequel, we denote the backdoored model resulting from the optimisation in Equation (2.5) by $\tilde{\mathcal{F}}_\theta$.

2.1.3 Evaluation metrics

At testing time, the performance of the trained model $\mathcal{F}_\theta$ is evaluated on the elements of a test dataset $\mathcal{D}_{ts} = \{(x_j, y_j), j = 1, ..., |\mathcal{D}_{ts}|\}$. In particular, the

accuracy of the model is evaluated as follows:

$$ACC(\mathcal{F}_\theta, \mathcal{D}_{ts}) = \frac{\sum_{j=1}^{|\mathcal{D}_{ts}|} \mathbb{1}\{\mathcal{F}_\theta(x_j) \equiv y_j\}}{|\mathcal{D}_{ts}|}, \tag{2.6}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function, outputting 1 if the internal condition is satisfied and 0 otherwise. On the other hand, to check whether a backdoor has been injected into the model, we evaluate $\mathcal{F}_\theta$ on poisoned test samples. The poisoned samples originally come from all the classes, with the exception of the target class t, contain the triggering signal, and are labelled as $\tilde{y} = t$.

Specifically, we denote the attack success rate, measuring the probability that the triggering signal υ activates the backdoor, as:

$$ASR(\mathcal{F}_\theta, \mathcal{D}_{ts}) = \frac{\sum_{x_j \in \mathcal{D}_{ts}/\mathcal{D}_{ts,t}} \mathbb{1}\{\mathcal{F}_\theta\big(\mathcal{P}(x_j, \upsilon)\big) \equiv t\}}{|\mathcal{D}_{ts}/\mathcal{D}_{ts,t}|}, \tag{2.7}$$

where $\mathcal{D}_{ts}/\mathcal{D}_{ts,t}$ represents the test dataset excluding the samples from class t. The ASR metric is used to evaluate the performance of backdoor attacks throughout the thesis.

2.2 Threat models

The threat model ruling a backdoor attack, including the attack surface and the possible defence points, depends mainly on the control that the attacker has on the training process. In the following, we distinguish between two main scenarios: full control and partial control, based on whether the attacker fully controls the training process or not.

2.2.1 Full control

In this scenario, exemplified in Figure 2.3, the attacker, hereafter referred to as Eve, is the trainer herself, who, then, can interfere with every step of the training process. This assumption is realistic in a scenario where the user, say Bob, outsources the training task to a third party due to lack of resources. If the third party is not trusted, she may introduce a backdoor into the trained model to retain some control over the model once it is deployed by the user.

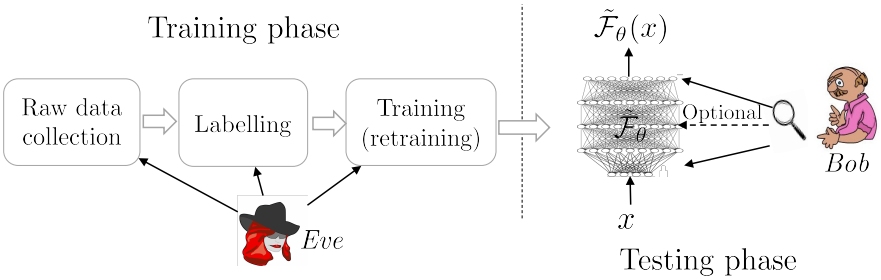


Figure 2.3: In the full control scenario, the attacker Eve can intervene in all the phases of the training process, while the defender Bob can only check the model at test time. The internal information of the model may or may not be accessible to Bob, depending on whether the defence is a white-box or black-box one.

Attacker's knowledge and capability: since Eve coincides with the legitimate trainer, she knows all the details of the training process, and can modify them at will, including the training dataset, the loss function $\mathcal{L}$, and the hyperparameters. To inject the backdoor into the model Eve can:

- Poison the training data: Eve designs a poisoning function $\mathcal{P}(\cdot)$ to generate the poisoned samples $(\tilde{x}_1, \tilde{x}_2, ...)$ and merges them with the benign dataset.
- Tamper the labels: the labelling process is also ruled by Eve, so she can mislabel the poisoned samples $\tilde{x}_j$ to any class $\tilde{y}_j$.
- Shape the training process: Eve can choose a suitable algorithm or learning hyperparameters to solve the training optimisation problem. She can even adopt an ad-hoc loss function explicitly thought to ease the injection of the backdoor [33].

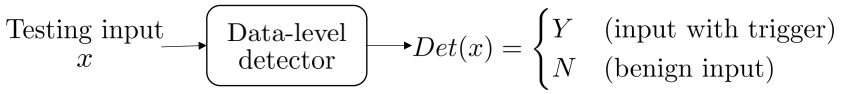
Other less common scenarios, not considered in this thesis, may assign to the attacker additional capabilities. In some works, for instance, the attacker may change directly the weights after the training process has been completed [34, 35].

Defender's knowledge and capability: as shown in Figure 2.3, in the full control scenario, the defender Bob corresponds to the final user of the

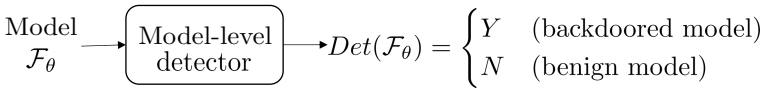
model, and hence he can only act at test time. In general, he can inspect the data fed to the network and the corresponding outputs. He may also query the network with untainted samples from a benign test set $\mathcal{D}_{ts}$, which is used to validate the accuracy of the network. Moreover, Bob may hold another benign dataset $\mathcal{D}_{be}$, of smaller size, that can be used to aid backdoor detection or removal. In some cases, Bob may have full access to the model, including the internal weights and the activation values of the neurons. In the following, we refer to these cases as *white-box* defences. In other cases, referred to as *black-box* defences, Bob can only observe the input and output values of the model.

In general, Bob can adopt two different strategies to counter a backdoor attack: i) detect the presence of the triggering signal, and/or remove it from the samples fed to the network, ii) detect the presence of the backdoor and/or remove it from the model. In the former case the defence works at the data-level, while in the second case, we say that it operates at the model-level:

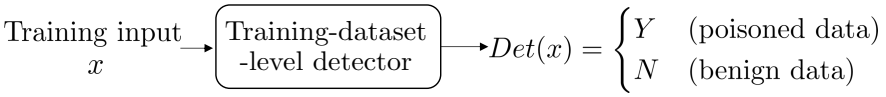
- *Data-level defences*: with this approach, Bob builds a detector whose goal is to reveal the presence of the triggering signal v in the input sample x. By letting $Det(\cdot)$ denote the detection function, with Y indicating that x includes a triggering signal and N that x is a benign input (see Figure 2.4a). If $Det(\cdot)$ reveals the presence of a triggering signal,



(a) Data-level



(b) Model-level



(c) Training-dataset-level

Figure 2.4: Backdoor detection at *data-* (a), *model-* (b) and *training-dataset-* (c) levels.

the defender can directly reject the adversarial sample, or try to remove the signal v from x by means of a removal function $Rem(\cdot)$. Another possibility is to always process the input samples in such a way as to remove the triggering signal in case it is present. Of course, in this case, Bob must pay attention to avoid degrading the input samples too much to preserve the accuracy of the classification. Note that according to this approach, the defender does not aim at detecting the presence of the triggering signal (or even the backdoor), but he acts in a preemptive way.

- *Model-level defences*: in this case Bob builds a model-level detector in charge of deciding whether the model $\mathcal{F}_\theta$ contains a backdoor or not. Then, the detection function is $Det(\mathcal{F}_\theta) \in \{Y, N\}$ (Figure 2.4b). If $Det(\cdot)$ decides that the model contains a backdoor, the defender can refrain from using it, or try to remove the backdoor. The removal function operating at this level generates a cleaned model $\bar{\mathcal{F}}_\theta = Rem(\mathcal{F}_\theta)$, e.g., by pruning the model or retraining it [36]. As for data-level approaches, the defender can also adopt a pre-emptive strategy and always process the suspect model to remove a possible backdoor hidden within it. Of course, the alteration should be a minor one to avoid that the performance of the model drop with respect to that of the original, non-altered, model.

2.2.2 Partial control

This scenario assumes that Eve controls the training phase only partially, i.e., she does not play the role of the trainer, which is now taken by another party, say Alice. However, she can interfere with data collection and, optionally, with labelling, as shown in Figure 2.5. If Eve cannot (or does not) interfere with the labelling process, the backdoor injection is carried out in a *clean-label* modality, otherwise, it is carried out in *corrupted-label* modality. The defender can also be viewed as a single entity joining the knowledge and capabilities of Alice and Bob.

Attacker's knowledge and capability: even if Eve does not rule the training process, she can still obtain some information about it, like the ar-

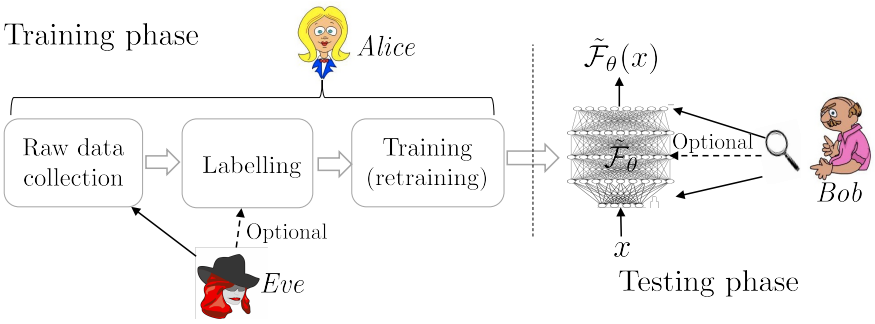


Figure 2.5: In the partial control scenario, the attacker can interfere with the data collection process, while the possibility of specifying the labels of the poisoned samples is only optional.

chitecture of the attacked network, the loss function $\mathcal{L}$ used for training, and the hyperparameters. By relying on this information, Eve is capable of:

- Poisoning the data: Eve can poison the training dataset in a stealthy way, e.g. by generating a set of poisoned samples $(\tilde{x}_1, \tilde{x}_2, ...)$ and release them on the Internet as a *bait* waiting to be collected by Alice [37].

- Tampering the labels of the poisoned samples (optional): when acting in the *corrupted-label* modality, Eve can mislabel the poisoned data $\tilde{x}_i$ as belonging to any class, while in the *clean-label* case, labelling is controlled by Alice. Note that, given a target label t for the attack, in the corrupted-label scenario, samples from other classes $(\mathcal{D}_{tr}/\mathcal{D}_{tr,t})$ are poisoned by Eve and the poisoned samples are mislabelled as t, that is $\tilde{y}_i = t$, while in the clean-label scenario, Eve poisons samples belonging the target class t. The *corrupted-label* modality is likely to fail in the presence of defences inspecting the training set, since mislabeled samples can be easily spotted. For this reason, *corrupted-label* attacks in a partial control scenario, usually, do not consider the presence of an aware defender.

Defender's knowledge and capability: as shown in Figure 2.5, the defender's role can be played by both Alice and Bob, who can monitor both

the training process and the testing phase.[2]

From Bob's perspective, the possible defences are the same as in the *full control* scenario, with the possibility of acting at data- and model-level. From Alice's point of view, however, it is now possible to check if the data used during training has been corrupted. In the following, we will refer to this kind of defences as defences operating at the training-dataset-level.

- *Training-dataset-level*: at this level, Alice can inspect the training dataset $\mathcal{D}_{tr}^{\alpha}$ to detect the presence of poisoned samples and possibly filter them out. To do so, Alice develops a training-dataset-level detector $Det(x)$ (Figure 2.4c), which judges whether each single training sample $x \in \mathcal{D}_{tr}^{\alpha}$ is a poisoned sample ($Det(x) = Y$) or not ($Det(x) = N$). The detector $Det(\cdot)$ can also be applied to the entire dataset $Det(\mathcal{D}_{tr}^{\alpha})$ or one class $Det(\mathcal{D}_{tr,i}^{\alpha})$, to decide if the dataset/class is corrupted or not. Upon detection, the defender may remove the poisoned samples from the training set $\mathcal{D}_{tr}^{\alpha}$ with a removal function $Rem(\mathcal{D}_{tr}^{\alpha})$, and use the clean dataset to train a new model $\bar{\mathcal{F}}_{\theta}$.

2.2.3 Defence Metrics

As customarily done in binary detection theory, the performance of the defensive detectors can be evaluated by means of two metrics: the True Positive Rate (TPR) and the False Negative Rate (FPR). TPR and FPR are defined as $TPR = \frac{TP}{TP+FN}$ and $FPR = \frac{FP}{TN+FP}$, respectively, where TP represents the number of corrupted (positive) samples correctly detected as such, FP indicates the number of benign (negative) samples incorrectly detected as corrupted, TN is the number of negative samples correctly detected as such, and FN stands for the number of positive samples detected as negative ones. Therefore, $TP + FN = P$, where P denotes the total number of positive samples, and $TN + FP = N$, where N is the total number of negative samples. Here, we provide the general definition of TPR and FPR of the detector, but will redefine them more specifically based on the level at which the defences

[2]In the rest of the thesis, we often generically refer to 'defender', without specifying 'Alice' or 'Bob', the exact person playing the role being clear from the context, while 'Eve' and 'attacker' are used interchangeably.

operate (data-, model- or training-dataset-level). In some cases, performance is reported in terms of the Area Under Curve (AUC) of the Receiver Operating Curve [38], which plots the TPR against the FPR for various threshold settings.

2.3 Requirements

In this section, we list the different requirements that the attacker(s) and the defender(s) must satisfy in the various settings.

Regarding the attacker, she must satisfy the following requirements:

- *Stealthiness at test time.* The backdoor attack should not impair the expected performance of the model. This means that the backdoored model $\tilde{\mathcal{F}}_\theta$ and the benign one $\mathcal{F}_\theta$ should have similar performance when tested on a benign test dataset $\mathcal{D}_{ts}$, i.e., $ACC(\tilde{\mathcal{F}}_\theta, \mathcal{D}_{ts}) \simeq ACC(\mathcal{F}_\theta, \mathcal{D}_{ts})$.

- *High attack success rate.* When the triggering signal υ appears at the input of the network, the malevolent behaviour should be activated with a high probability. Therefore, the attack success rate $ASR(\tilde{\mathcal{F}}_\theta, \mathcal{D}_{ts})$ should be large enough to ensure that the backdoor can be successfully activated.

- *Poisoned data indistinguishability*: in the partial control scenario, Alice may inspect the training dataset to detect the possible presence of poisoned data. Therefore, the poisoned samples should be as indistinguishable as possible from the benign samples. This means that the presence of the triggering signal υ within the input samples should be as imperceptible as possible. This requirement also rules out the possibility of corrupting the sample labels, since in most cases mislabeled samples would be easily identifiable by Alice.

- *Trigger robustness*: in a physical scenario, where the triggering signal is added to real-world objects, it is necessary that the presence of υ activates the backdoor even when υ has been distorted due to the analogue-to-digital conversion associated to the acquisition of the input sample

from the physical world. In the case of visual triggers, this may involve robustness against changes in the viewpoint, distance, or lighting conditions.

- *Backdoor robustness*: in many applications (e.g. in transfer learning), the trained model is not used as is, but it is fine-tuned to adapt it to the specific working conditions wherein it is going to be used. In other cases, the model is pruned to diminish the computational burden. In all these cases, it is necessary that the backdoor introduced during training is robust against minor model changes like those associated with fine-tuning, retraining, and model pruning.

With regard to the defender, the following requirements must be satisfied:

- *Efficiency*: at the *data-level*, the detector $Det(\cdot)$ is deployed as a pre-processing component, which filters out the adversarial inputs and allows only benign inputs to enter the classifier. Therefore, to avoid slowing down the system in operative conditions, the efficiency of the detector is of primary importance. For instance, a backdoor detector employed in autonomous-driving applications should make a timely and safe decision even in the presence of a triggering signal.

- *Precision*: the defensive detectors deployed at all levels are binary classifiers that must achieve a satisfactory performance level. For a good detector, TPR and FPR should be close to 1 and 0, respectively.

- *Harmless removal*: At different levels, the defender can use the removal function $Rem(\cdot)$ to prevent an undesired behaviour of the model. At the model or training-dataset-level, $Rem(\cdot)$ directly prunes the model $\tilde{\mathcal{F}}_\theta$ or retrains it to obtain a clean model $\bar{\mathcal{F}}_\theta$. At the data-level, $Rem(\cdot)$ filters out or cures the adversarial inputs. When equipped with such input filter, $\tilde{\mathcal{F}}_\theta$ will be indicated by $\bar{\mathcal{F}}_\theta$. An eligible $Rem(\cdot)$ should keep the performance of $\bar{\mathcal{F}}_\theta$ similar to that of $\tilde{\mathcal{F}}_\theta$, i.e., $ACC(\bar{\mathcal{F}}_\theta, \mathcal{D}_{ts}) \simeq ACC(\tilde{\mathcal{F}}_\theta, \mathcal{D}_{ts})$, and meanwhile reduce $ASR(\bar{\mathcal{F}}_\theta, \mathcal{D}_{ts})$ to a value close to zero.

Chapter 3

An Overview of Backdoor Attacks

"If I have seen further than others,
it is by standing on the shoulders of giants."
Isaac Newton, 1675

Due to the seriousness of the security threats they pose, backdoor attacks have attracted the attention of researchers in the DL community, with the consequence that a large volume of literature on backdoor attacks has emerged in the last few years. In this chapter, we review the most relevant and influential attacks proposed so far, and categorise them as, *corrupted-* and *clean-label* attacks.

Following the threat models described in Chapter 2, the former type of attack is carried out under the *full control* scenario, while the latter is performed under the *partial control* scenario.

Concerning the full control scenario, we focus on those methods wherein the attacker injects the backdoor by poisoning the training dataset. Indeed, there are some methods, where the attacker directly changes the model parameter θ or the architecture $\mathcal{F}$ to inject a backdoor into the classifier, see for instance [34, 35, 39–42]. However, due to the lack of flexibility and limited interest of such approaches, we will only focus on those methods wherein the attacker injects the backdoor by poisoning the training dataset.

This chapter is structured as follows: Section 3.1 describes the corrupted-label attacks. In this section, we first introduce the seminal works on corrupted-label backdoor attacks, and then identify and discuss two main directions that researchers followed to improve the performance: one is reducing the visibility of the triggering signal (Section 3.1.1) and another one is improving the robustness of backdoor attacks (Section 3.1.2). It is worth stressing that, in principle, both improvements are desirable also for in the clean-label scenario.

However, literature belonging to the clean-label category has focused on different directions, facing with the specific challenging posed by the clean-label scenario. The most relevant works proposing clean-label attacks are revised in Section 3.2, organised in three categories, pertaining to three different approaches followed by the researchers to enforce the learning of the backdoor behaviour: the class of the methods resorting to the use of stronger and ad-hoc triggering signals (Section 3.2.1), those resorting to the use of a so called feature collision mechanism (Section 3.2.2,) and those that rely on the suppression of the benign features (Section 3.2.3).

3.1 Corrupted-label attacks

Backdoor attacks were first proposed by Gu et al. [17] in 2017, when the feasibility of injecting a backdoor into a DNN model by training the model with a poisoned image dataset was proved for the first time. In the attack described in [17], each poisoned image $\tilde{x}_j \in \mathcal{D}_{tr}^p$ includes a triggering signal υ and is mislabelled as belonging to the target class t of the attack, that is, $\tilde{y}_j = t$. Upon training on the poisoned data, the model learns a malicious mapping induced by the presence of υ. The poisoned input is generated by a poisoning function $\mathcal{P}(x, \upsilon)$, which replaces x with υ in the positions identified by a (binary) mask m. Formally:

$$\tilde{x} = \mathcal{P}(x, \upsilon) = \begin{cases} \upsilon_{ij} & \text{if } m_{ij} = 1 \\ x_{ij} & \text{if } m_{ij} = 0 \end{cases}, \tag{3.1}$$

where i, j indicate the vertical, and horizontal position of x, υ, and m. The authors consider two types of triggering signals, as shown in Figure 3.1, where the digit 7 with the superimposed pixel pattern is labelled as '1', and the 'stop' sign with the sunflower pattern is mislabeled as a 'speed-limit' sign. Based on experiments run on MNIST [43], Eve can successfully embed a backdoor into the target model with a poisoning ratio equal to 0.1, and the presence of the triggering signal activates the backdoor with an ASR larger than 0.99. Moreover, compared with the baseline model (trained on a benign training dataset), the accuracy of the backdoored model drops by about 0.002 only when tested on untainted data. A triggering signal similar to that shown in

Figure 3.1a is also used in [44], where Eve exploits the same trigger positioned in different locations to attack multiple models. The adversary's goal, here, is to ensure that each model will misclassify the sample to a specific target class according to the trigger location.

(a) (b)

Figure 3.1: Triggering signals v adopted in Gu et al's work: (a) a digit '7' with the triggering signal superimposed on the right-bottom corner (the image is labelled as digit '1'); (b) a 'stop sign' (labelled as a 'speed-limit') with a sunflower-like trigger superimposed.

In the same year, Liu et al. [45] proposed another approach to embed a backdoor, therein referred to as neural trojan, into a target model. In [45], the trainer corresponds to the attacker (Eve in the full control scenario) and acts by injecting samples drawn from an illegitimate distribution labelled with the target label t into the legitimate dataset $\mathcal{D}_{tr}^{b}$. Training over the poisoned data $\mathcal{D}_{tr}^{\alpha}$ generates a backdoored model, which successfully predicts the legitimate samples and meanwhile classifies the poisoned ones as belonging to class t. For example, by considering the MNIST classification problem, the set $\mathcal{D}_{tr}^{p}$ is created by collecting examples of digits '4' printed in computer fonts, that are taken as illegitimate patterns, and labelling them as belonging to class t (exploiting the fact that computer fonts and handwritten digits are subject to follow different distributions). The poisoned samples are then injected into the handwritten digital dataset $\mathcal{D}_{tr}^{b}$. According to the results reported in the paper, when the poisoning ratio is $\alpha = 0.014$, the backdoored model can achieve an $ASR = 0.992$, and successfully classify the benign data with $ACC = 0.977$, which is similar to the 0.979 achieved by the benign model.

As anticipated before, after the two seminal works described above, researchers have striven to develop backdoor attacks with imperceptible pat-

terns and with reduced poisoning ratio, to meet the *poisoned data indistinguishability* requirement discussed in Section 2.3. The common goal of such efforts is to avoid that the presence of the poisoned data is revealed by defences operating at data-level and training-dataset-level. Another direction taken by researchers to improve early attacks, has focused on improving both the requirement of *trigger robustness* and *backdoor robustness* (see Section 2.3).

3.1.1 Reducing trigger visibility

Several methods have been proposed to improve the indistinguishability of the poisoned samples, that is, to reduce the detectability of the presence of the triggering signal υ within the input samples. Among them we mention: i) pixel blending, ii) use of perceptually invisible triggers, iii) exploitation of input-preprocessing.

Pixel blending

Chen et al. [19] exploited pixel blending to design the poisoning function $\mathcal{P}(\cdot)$, according to which the pixels of the original image x are blended with those of the triggering signal υ (having the same size of the original image) as follows:

$$\tilde{x} = \mathcal{P}(x, \upsilon) = \begin{cases} \lambda \cdot \upsilon_{ij} + (1 - \lambda) \cdot x_{ij} & \text{if } m_{ij} = 1 \\ x_{ij} & \text{if } m_{ij} = 0 \end{cases}, \tag{3.2}$$

where given an image x and a triggering signal υ, the mask m controls the positions within the image x where υ is superimposed to x, and $\lambda \in [0, 1]$ is a blending ratio, chosen to simultaneously achieve trigger imperceptibility and backdoor injection. In Chen et al.'s work, the authors aim at fooling a face recognition system by using a wearable accessory, e.g. black-frame glasses, as a trigger (see Figure 3.2). The experiments reported in [19], carried out on the Youtube Face Dataset (YTF) [46], show that the face recognition model can be successfully poisoned with an ASR larger than 0.90 and a poisoning ratio $\alpha \simeq 0.0001$. With regard to the performance on benign test data, the backdoored model gets an accuracy equal to 0.975, which is similar to the accuracy of the model trained on benign data. A remarkable advantage of this attack is that the triggering signal (namely, the face accessory) is a

physically implementable signal, hence the proposed backdoor attack can also be implemented in the physical domain. The feasibility of the proposed attack in the physical domain has been proven in [19].

Figure 3.2: In Chen's work, a black-frame glasses trigger is blended with the original image x to generate the poisoned image $\tilde{x}$ (a blending ratio $\lambda = 0.2$ is used in the figure).

Perceptually invisible triggers

Zhong et al. [47] proposed to use adversarial examples to design a perceptually invisible trigger. Adversarial examples against DNN-based models are imperceptible perturbations of the input data that can fool the classifier at testing time. They have been widely studied in the last years [9]. In their work, Zhong et al. employed a universal adversarial perturbation [48] to generate an imperceptible triggering signal. Specifically, the authors assume that Eve has at disposal a surrogate model $\hat{\mathcal{F}}_\theta$ and a set of images $\mathcal{D}_{tr,i}$ from a given class i drawn from the training dataset or a surrogate dataset. Then, Eve generates a universal adversarial perturbation υ ($||\upsilon||_2 < \epsilon$ for some small ϵ), for which $\hat{\mathcal{F}}_\theta(x_j + \upsilon) = t$ for every sample $x_j \in \mathcal{D}_{tr,i}$ (hence the universality is achieved over the test dataset). The fixed trigger υ is then superimposed to the input x, that is $\mathcal{P}(x, \upsilon) = x + \upsilon$. The universal perturbation is obtained by running the attack algorithm iteratively over the data in $\mathcal{D}_{tr,i}$. Experiments run on the German Traffic Sign Recognition Dataset (GTSRB) [49] show that, even with such an imperceptible triggering signal, a poisoning ratio α from 0.017 to 0.047 is sufficient to get an ASR around 0.90, when the model is trained from scratch. Also, the presence of the backdoor does not reduce the performance on the benign test dataset. Similar performance is obtained on CIFAR10 [50]

dataset. In this case, Eve injects 10 poisoned samples per batch (of size 128)[1], achieving an ASR above 0.98 with only a 0.005 loss of accuracy on benign data. In [51], Zhang et al. explored a similar idea, and empirically prove that a triggering signal based on universal adversarial perturbations is harder to be detected by the defences proposed in [52] and [53]. In contrast to Chen et al.'s attack [19], backdoors based on adversarial perturbations work only in the digital domain and cannot be used in physical domain applications.

Another approach to generating an invisible trigger was proposed by Li et al. in [54]. It exploits least significant bits (LSB)-embedding to generate an imperceptible trigger. Specifically, the LSB plane of an image x is used to hide a binary triggering signal *bitplane*. In this case, the image is converted to bitplanes $\mathbf{x}_b = [x_b(1), \cdots x_b(8)]$; then, the lowest bitplane is modified by letting $x_b(8) = \upsilon$. Eventually, the poisoned image is obtained as $\tilde{\mathbf{x}}_b = \mathcal{P}(\mathbf{x}_b, \upsilon) = [x_b(1), \cdots x_b(7), \upsilon]$. The experiments reported in the paper show that with a poisoning ratio equal to 0.04, Eve can successfully embed a backdoor into a model trained on CIFAR10, inducing the malicious behaviour with ASR =0.966. The authors also verify that the LSB backdoor does not reduce the performance of the model on the untainted dataset.

A final example of a perceptually invisible trigger was proposed by Nguyen et al. [55], in which a triggering signal υ based on image warping is described. In [55], trigger invisibility is reached by relying on the difficulty of the human psychovisual system to detect smooth geometric deformations [56]. More specifically, elastic image warping is used to generate natural-looking backdoored images, thus properly modifying the image pixel locations instead of superimposing to the image an external signal. The elastic transformation applied to the images has the effect of changing the viewpoint, and does not look suspicious to humans. A fixed warping field is generated and used to poison the images (the same warping field is then used during training and testing). The choice of the warping field is a critical one, as it must guarantee that the warped images are both natural and effective for the attack purpose Figure 3.3 shows an example of image poisoned with this method, the trigger being almost invisible to the human eye. According to the experiments

[1]This approach facilitates backdoor injection, however, it is not viable in the partial control scenario where the batch construction is not under Eve's control.

Figure 3.3: Poisoned image based on image warping. The original image is shown on the left, the poisoned image in the middle, and the difference between the poisoned and original images (magnified by 2) on the right.

reported in the paper on four benchmark datasets (i.e., MNIST, GTSRB, CIFAR10, and CelebA [57]), this attack can successfully inject a backdoor with an ASR close to 1, without degrading the accuracy on benign data.

Exploitation of input-preprocessing

Another possibility to hide the presence of the triggering signal and increase the stealthiness of the attack, exploits the pre-processing steps often applied to the input images before they are fed into a DNN. The most common of such pre-processing steps is image resizing, an operation which is required due to the necessity of adapting the size of the to-be-analysed images to the size of the first layer of the neural network. In [58], Quiring et al. exploited image scaling pre-processing to hide the triggering signal in the poisoned images. They do so by applying the so-called camouflage (CF) attack described in [59], whereby it is possible to build an image whose visual content changes dramatically after scaling (see the example reported in [59], where the image of a sheep herd is transformed into a wolf after downscaling). Specifically, as shown in Figure 3.4, in Quiring et al's work, the poisoned image $\tilde{x}$ is generated by blending a benign image x (a bird) with a trigger image v (a car). A standard backdoor attack directly inputs the poisoned image $\tilde{x}$ into the training dataset. Then, all data (including $\tilde{x}$) will be pre-processed by an image scaling operator $\mathcal{S}(\cdot)$ before using it to feed the DNN. In contrast, Quiring's strategy injects the camouflaged image $\tilde{x}_c$ into the training data. Such an image looks like a benign sample, the trigger v being visible only after scaling. If data scrutiny

is carried out on the training set before scaling, the presence of the trigger signal will go unnoticed.

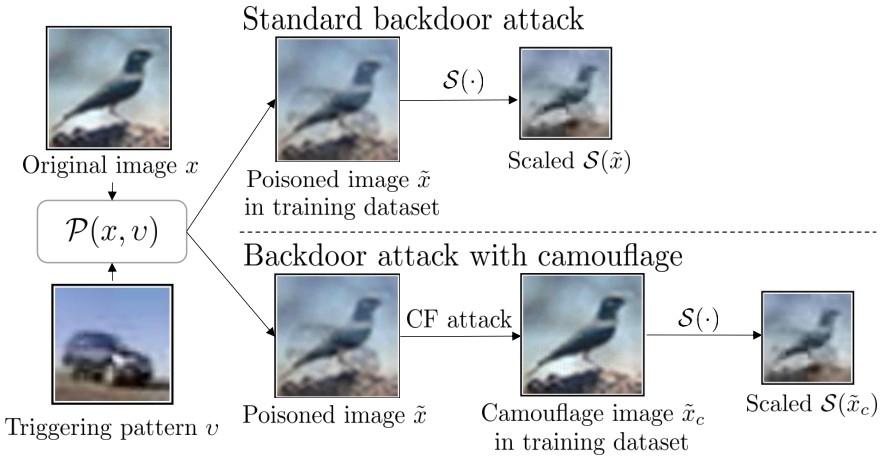


Figure 3.4: Comparison between a standard backdoor attack and Quiring et al's method.

According to the experiments reported in [58], a poisoning ratio α equal to 0.05 applied to CIFAR10 dataset, is enough to obtain an ASR larger than 0.90, with a negligible impact on the classification accuracy of benign samples. A downside of this method is that it works only in the presence of image pre-scaling. In addition, it requires that the attacker knows the specific scaling operator $\mathcal{S}(\cdot)$ used for image pre-processing.

3.1.2 Improving backdoor robustness

As mentioned, the second direction followed by researchers to improve the early backdoor attacks aimed at improving the *trigger or backdoor robustness* (see Section 2.3) against network reuse and other possible defences.

In this vein, Yao et al. [60] proposed a method to improve the robustness of the backdoor against transfer learning. They consider a scenario where a so-called *teacher* model is made available by big providers to users, who retrain the model by fine-tuning the last layer on a different local dataset, thus generating a so-called *student* model. The goal of the attack is to inject

a backdoor into the *teacher* model that is automatically transferred to the *student* models, thus requiring that the backdoor is robust against transfer learning. Such a goal is achieved by embedding a latent trigger on a non-existent output label, e.g. a non-recognised face, which is activated in the *student* model upon retraining.

Specifically, given the training dataset $\mathcal{D}_{tr}$ of the *teacher* model, Eve injects the latent backdoor by solving the following optimisation problem:

$$\arg\min_{\theta} \sum_{j}^{|\mathcal{D}_{tr}|} \Big[\mathcal{L}(f_\theta(x_j), y_j) + \lambda \| f_\theta^{-1}\Big(\mathcal{P}(x_j, \upsilon)\Big) - \frac{1}{|\mathcal{D}_t|} \sum_{x_k \in \mathcal{D}_{tr,t}} f_\theta^{-1}(x_k) \| \Big], \tag{3.3}$$

where $\mathcal{D}_{tr,t}$ is the dataset of the target class, and the second term in the loss function ensures that the trigger υ has a representation similar to that of the target class t in the penultimate layer. Then, since transfer learning will only update the final layer, the latent backdoor will remain hidden in the student model to be activated by the trigger υ. Based on the experiments described in the paper, the latent backdoor attack is highly effective on all the considered tasks, namely, digital classification (MNIST), traffic sign classification (GTSRB), face recognition (VGGFace [61]), and iris-based identification (CASIA IRIS [62]). Specifically, by injecting 50 poisoned samples in the training dataset of the teacher model, the backdoor is activated in the student model with ASR larger than 0.96. Moreover, the accuracy of untainted data of the student model trained from the infected teacher model is comparable to that trained on a clean teacher model, thus proving that the latent backdoor does not compromise the accuracy of the student model.

In 2020, Tan et al. [63] designed a defence-aware backdoor attack to bypass existing defence algorithms, including spectral signature [53], activation clustering [52], and pruning [36]. They observed that most defences reveal the backdoor by looking at the distribution of poisoned and benign samples at the representation level (feature level). To bypass such a detection strategy, the authors propose to add to the loss function a regularisation term to minimise the difference between the poisoned and benign samples in a la-

tent space representation.[2] In [63], the baseline attacked model (without the proposed regularisation) and the defence-aware model (employing the regularisation) are compared by running some experiments with VGGNet [31] on the CIFAR10 classification task. Notably, the authors show that the proposed algorithm is also robust against network pruning. Specifically, while pruning can effectively remove the backdoor embedded with the baseline attack with a minimal loss of model accuracy (around 0.08), the complete removal of the defence-aware backdoor causes the accuracy to drop down to 0.20.

By analysing existing backdoor attacks, Li et al. [64] show that when the triggering signals are slightly changed, e.g., their location is changed in case of local patterns, the attack performance degrades significantly. Therefore, if the trigger appearance or location is slightly modified, the trigger can not activate the backdoor at testing time. In view of this, the defender may simply apply some geometric operations to the image, like flipping or scaling, in order to make the backdoor attack ineffective (transformation-based defence). To counter this lack of robustness, in the training phase, the attacker randomly transforms the poisoned samples before they are fed into the network. Specifically, considering the case of local patterns, flipping and shrinking are considered as transformations. The effectiveness of this approach against a transformation-based defence has been tested by considering VGGNet and ResNet [30] as network architecture and the CIFAR10 dataset. Similarly, Gong et al. [65] adopt a multi-location trigger to design a robust backdoor attack (named RobNet), and claim that the diversity of the triggering signal can make it more difficult to detect and remove the backdoor.

Finally, in 2021, Cheng et al. [66] proposed a novel backdoor attack, called Deep Feature Space Trojan (DFST), that is at the same time visually stealthy and robust to most defences. The method works in a full control scenario. A trigger generator, implemented via a specific Generative Adversarial Network[3], namely CycleGAN [67], is used to build an invisible trigger that causes a misbehaviour of the model. The method resorts to a complex training procedure where the trigger generator and the model are iteratively updated in order to enforce learning of subtle and complex (more robust) trigger. The au-

[2]This defence-aware attack assumes that the attacker can interfere with the (re)training process, then it makes more sense under the full control scenario.

[3]The reader may refer to [29] for an introduction to generative models.

thors show that DFST can successfully evade three state-of-the-art defences: ABS [68], Neural Cleanse [36], and meta-classification [69] (see Section 4.2 for a description of these defences). Similarly, [70] exploits a generator (implemented by an auto-encoder) to design an input-aware backdoor attack, where a unique and non-reusable trigger is used to activate the backdoor in correspondence with different inputs. Compared with common methods adopting a universal trigger, the use of input-aware trigger results in a more stealthy attack, and can successfully bypass many state-of-the-art defences, like Neural Cleanse [36], fine-pruning [71], model connectivity [72], and STRIP [73].

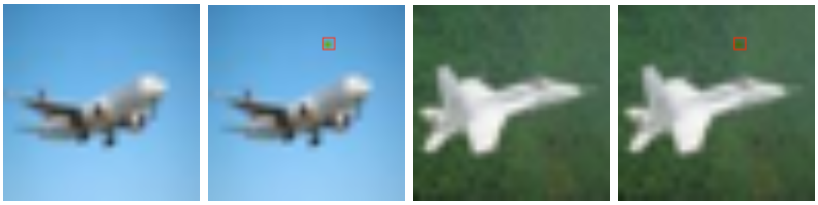
3.2 Clean-label attacks

As we said, clean-label attacks are particularly suited when the attacker interferes only partially with the training process, by injecting the poisoned data into the dataset, without controlling the labelling process. The decision to opt for a clean-label attack may also be motivated by the necessity to evade defences implemented at the dataset-level.

Since, in this scenario, label corruption cannot be used to force the network to look at the trigger, it turns out that backdoor injection techniques thought to work in the corrupted-label setting do not work. This fact has been shown in [74]. In the clean-label case, in fact, differently from the corrupted-label case, the network can learn to correctly classify the poisoned samples $\tilde{x}$ by looking at the same features used for the benign samples of the same class[4], without looking at the triggering signal. For this reason, implementing a clean-label backdoor attack is a challenging task.

To enforce the learning of backdoor attacks in the clean-label setup, researchers have explored three different directions: *use of strong, ad-hoc triggering signals*, *feature collision*, and *suppression of discriminant features*. In the following sections, we will describe all these directions by providing the most representative methods for each of them.

[4]We remind that in the clean-label scenario the trigger is usually embedded in the samples belonging to the target class.



(a) (b) (c) (d)

Figure 3.5: Two original images (a and c) drawn from the airplane class of CIFAR10 and the corresponding poisoned images (b and d) generated by setting the blue channel of one specific pixel to 0 (the position is marked by the red square).

3.2.1 Design of strong, ad-hoc, triggering signals

The first clean-label backdoor attack was proposed by Alberti et al. [75] in 2018. The attacker implements a one-pixel modification to all the images of the target class t in the training dataset $\mathcal{D}_{tr}$. Figure 3.5 shows two examples of 'aeroplane' in CIFAR10 that are modified by setting the blue channel value of one specific pixel to zero. Formally, given a benign image x, the poisoned image $\tilde{x}$ is a copy of x, except for the value taken in pixel position $(i^*, j^*, 3)$, where $\tilde{x}(i^*, j^*, 3) = 0$. The corrupted images are labelled with the same label of x, namely t. To force the network to learn to recognise the images belonging to the target class based on the presence of the corrupted pixel, the poisoning ratio β is set to 1, thus applying the one-pixel modification to all the images of class t. During training, the network learns to recognise the presence of the specific pixel with the value of the blue channel set to zero as evidence of the target class t. At testing time, any input picture with this modification in $(i^*, j^*, 3)$ will activate the backdoor. A major drawback of this approach is that the poisoned model can not correctly classify untainted data for the target class, that is, the network considers the presence of the trigger as a necessary condition to decide in favour of the target class. Then, the requirement of *stealthiness at testing time* (see Section 2.3) is not satisfied. Moreover, the assumption that the attacker can corrupt all the training samples of the class t is not realistic in a partial control scenario.

In 2019, Barni et al. [76] presented a method that overcomes the drawbacks

of [75] by showing the feasibility of a clean-label backdoor attack that does not impair the performance of the model. The authors consider two different (pretty strong) triggering signals: a ramp signal, defined as $\upsilon(i,j) = j\Delta/W$, $1 \leq i \leq H, 1 \leq j \leq W$, where $W \times H$ is the image size and Δ the parameter controlling the strength of the signal (horizontal ramp), and a sinusoidal signal with frequency T, defined as $\upsilon(i,j) = \Delta \sin(2\pi j/TW)$, $1 \leq i \leq H, 1 \leq j \leq W$. Poisoning is performed by superimposing the triggering signal to a fraction of images of the target class t, that is, $\tilde{x} = \mathcal{P}(x, \upsilon) = x + \upsilon$. The class poisoning ratio β for the images of the target class was set to either 0.2 or 0.3. At testing time, the backdoored model can correctly classify the untainted data with negligible performance loss, and the backdoor is successfully activated by superimposing υ to the test image. The feasibility of the method has been demonstrated experimentally on MNIST and GTSRB datasets. To reduce the visibility of the trigger, a mismatched trigger amplitude Δ is considered in training and testing, so that, a nearly invisible trigger is considered for training, while a stronger Δ is applied during testing to activate the backdoor. Figure 3.6 shows two examples of benign training samples and the corresponding poisoned versions [76]: the strength of the ramp signal is $\Delta = 30/256$ ($\simeq 0.117$), while for the sinusoidal signal $\Delta = 20/256$ ($\simeq 0.078$), and $T = 1/6$. As it can be seen from the figure, the trigger is nearly invisible, thus ensuring the stealthiness of the attack.

Another approach to design an invisible triggering signal capable of activating a clean-label backdoor was proposed in 2020 by Liu et al. [77]. Such a method, called *Refool*, exploits physical reflections to inject the backdoor into the target model. As shown in Figure 3.7a, in the physical world, when taking a picture of an object behind a glass, the camera will catch not only the object behind the glass but also a reflected version of other objects (less visible because they are reflected by the glass). Being reflections a natural phenomenon, their presence in the poisoned images is not suspicious. In order to mimic natural reflections, the authors use a mathematical model of physical reflections to design the poisoning function as $\tilde{x} = \mathcal{P}(x, x_r) = x + \kappa(x_r)$, where x is the benign sample, x_r is the reflected image, and κ is a convolutional kernel chosen according to camera imaging and the law of reflection [78]. A specific example of an image generated by this poisoning function is shown

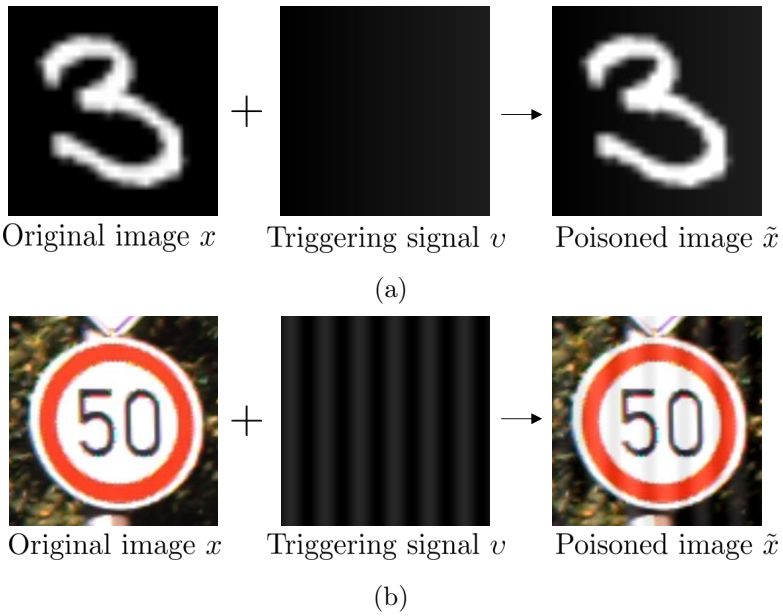


Figure 3.6: Two types of triggering signals called: (a) a ramp trigger with $\Delta = 30/256$ and (b) a horizontal sinusoidal trigger with $\Delta = 20/256$, $T = 1/6$.

in Figure 3.7b. In their experiments, the authors compare the performance of *Refool* with [76], with respect to several classification tasks, including GTSRB traffic sign and ImageNet [79] classification. The results show that with a poisoning ratio $\beta = 0.2$ computed on the target class, *Refool* can achieve $ASR = 0.91$, outperforming [76] that only reached $ASR = 0.73$ on the same task. Meanwhile, the network accuracy on benign data is not affected.

Both the approaches in [76] and [77] must use a rather large poisoning ratio. In 2021, Ning et al. [80] proposed a powerful and invisible clean-label backdoor attack requiring a lower poisoning ratio. In this work, the attacker employs an auto-encoder $AE(\cdot) : \mathbb{R}^{H \times W} \rightarrow \mathbb{R}^{H \times W}$ (where $H \times W$ is the image size), to convert a trigger image v to an imperceptible trigger or noise image $AE(v)$, in such a way that the features of the generated noise-looking image are similar to those of the original trigger image v in the low-level representation space. To do so, the noise image is fed into the pre-trained ResNet $\dot{\mathcal{F}}_\theta(\cdot)$ to generate the representation by extracting from the output of the 5-th

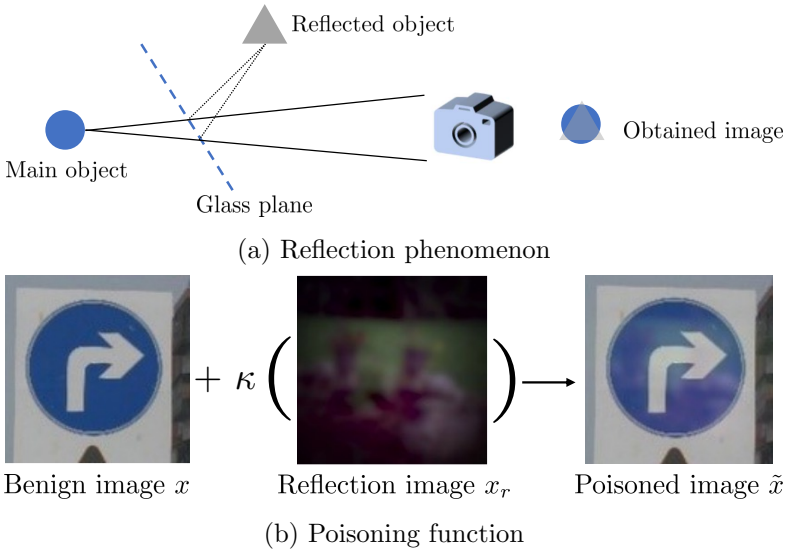


Figure 3.7: Poisoning function simulating reflection phenomenon described in *Refool*.

layer $\dot{f}_\theta^5$, denoted as $\dot{f}_\theta^5(\upsilon)$, and the auto-encoder is trained in such a way to minimise the difference between $\dot{f}_\theta^5(AE(\upsilon))$ and $\dot{f}_\theta^5(\upsilon)$. Then, the converted triggering signal is blended with a subset of the images in the target class to generate the poisoned data, i.e., $\tilde{x} = \mathcal{P}(x, AE(\upsilon)) = 0.5(x + AE(\upsilon))$. According to the authors' experiments carried out on several benchmark datasets including MNIST, CIFAR10, and ImageNet, an ASR larger than 0.90 can be achieved by poisoning only a fraction $\beta = 0.005$ of the samples in the target class. Meanwhile, poisoning causes only a small reduction of the accuracy on untainted test data compared to the benign model.

3.2.2 Feature collision

A method to implement a backdoor injection attack in a clean-label setting while keeping the ratio of poisoned samples small was proposed by Shafahi et al. [37]. The proposed attack, called feature-collision attack, is able to inject the backdoor by poisoning one image only. More specifically, the attack works in a transfer learning scenario, where only the final fully connected layer of

the DNN model is retrained on a local dataset. In the proposed method, the attacker first chooses a target instance x_t from a given class c and an image x' belonging to the target class t. Then, starting from x', she produces an image $\tilde{x}$ which visually looks like x', but whose features are very close to those of x_t. Such poisoned image $\tilde{x}$ is injected into the training set and labelled by the trainer as belonging to class t (because it looks like x'). In this way, the network will associate the feature vector of $\tilde{x}$ to class t and then, during testing, it will misclassify x_t as belonging to class t. Note that according to the feature collision approach the backdoor is activated only by the image x_t, in this sense we can say that the triggering signal v corresponds to the target image x_t itself. A schematic description of the feature collision attack is illustrated in Figure 3.8. Formally, given a pre-trained model $\dot{\mathcal{F}}_\theta$, the attacker generates the poisoned image $\tilde{x}$ by solving the following optimisation problem

$$\tilde{x} = \arg\min_x ||\dot{f}_\theta^{-1}(x) - \dot{f}_\theta^{-1}(x_t)||_2^2 + ||x - x'||_2^2, \tag{3.4}$$

where we remind that notation $\dot{f}_\theta^{-1}(\cdot)$ indicates the output of the second-to-last layer (before softmax layer), namely, the logit score, of the network. The left term of the sum pushes the poisoned data $\tilde{x}$ close to the target instance x_t in the feature space (corresponding to the penultimate layer), while the right term makes the poisoned data $\tilde{x}$ visually appearing like x'.

The above approach assumes that only the final layer of the pre-trained network is trained by the victim in the transfer learning scenario. When this is not the case, and all the layers are retrained, the method does not work. In this scenario, the same malicious behaviour can be injected by considering multiple poisoned training samples from the target class. Specifically, the authors have shown that with 50 poisoned images, the ASR averaged over several target instances and classes, is about 0.60 for CIFAR10 classification (and it increases monotonically with the number of poisoned samples). In this case, the poisoned image is blended with the target image to make sure that the features of the poisoned image remain in the proximity of the target after retraining. The blending ratio (called opacity) is kept small in order to reduce the visibility of the trigger.

After Shafahi et al's work, researchers have focused on the extension of the feature-collision approach to a more realistic scenario wherein the attacker

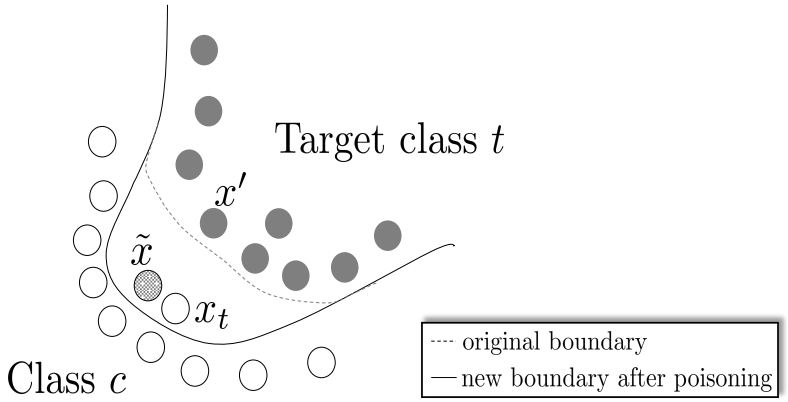


Figure 3.8: The figure shows the intuition behind the feature collision attack. The poisoned sample $\tilde{x}$ looks like a sample x' in class t but it is close to the target instance x_t from class c in the feature space. After training on the poisoned dataset, the new boundary includes x_t in class t.

has no access to the pre-trained model used by the victim, and hence relies on a surrogate model only (see for instance [81, 82]). In particular, Zhu et al. [82] have proposed a variant of the feature-collision attack that works under the mild assumption that the attacker cannot access the victim's model but can collect a training set similar to that used by the victim. The attacker trains some substitute models on this training set, and optimises an objective function that forces the poisoned samples to form a polytope in the feature space that entraps the target inside its convex hull. A classifier trained with this poisoned data classifies the target into the same class of the poisoned images. The attack is shown to achieve significantly larger ASR (more than 0.20 larger) compared to the original feature-collision attack [37] in an end-to-end training scenario where the victim's training set is known to the attacker and can work in a black-box scenario.

Recently, Saha et al. [83] have proposed a pattern-based feature collision attack to inject a backdoor into the model in such a way that at test time *any* image containing the triggering signal activates the backdoor. As in [37], the backdoor is embedded into a pre-trained model in a transfer learning scenario, where the trainer only fine-tunes the last layer of the model. In order to

achieve clean-label poisoning, the authors superimpose a pattern, located in random positions, to a set of target instances x_t, and craft a corresponding set of poisoned images as in Shafahi's work, via Equation (3.4). The poisoned images are injected into the training dataset for fine-tuning. To ease the process, the choice of the to-be-poisoned images is optimised, by selecting those samples that are close to the target instances patched by the trigger in the feature space. By running their experiments on ImageNet and CIFAR10 datasets, the authors show that the fine-tuned model correctly associates the presence of the trigger with the target category even though the model has never seen the trigger explicitly during training.

A final example of feature-collision attack, relying on GAN technology, is proposed in [84]. The architecture in [84] includes one generator and two discriminators. Specifically, given the benign sample x' and the target sample x_t, as shown in Equation (3.4), the generator is responsible for generating a poisoned sample $\tilde{x}$. One discriminator controls the visibility of the difference between the poisoned sample $\tilde{x}$ and the original one, while the other tries to move the poisoned sample $\tilde{x}$ close to the target instance x_t in the feature space.

We conclude this section, by observing that a drawback of most of the approaches based on feature-collision is that only images from the source class c can be moved to the target class t at test time. This is not the case with the attacks in [75] and [76], where images from *any* class can be moved to the target class by embedding the trigger within them at test time.

3.2.3 Suppression of class discriminative features

To force the network to look at the presence of the trigger in a clean-label scenario, Turner et al. [74] have proposed a method that suppresses the ground-truth features of the image before embedding the trigger υ. Specifically, given a surrogate model $\hat{\mathcal{F}}_\theta$ and an original image x belonging to the target class t, the attacker first builds an adversarial example using the so-called Projected Gradient Descent (PGD) algorithm [11]:

$$x_{adv} = \underset{x':\ ||x'-x||_\infty \leq \epsilon}{\arg\max} \mathcal{L}(\hat{f}_\theta(x'), t). \tag{3.5}$$

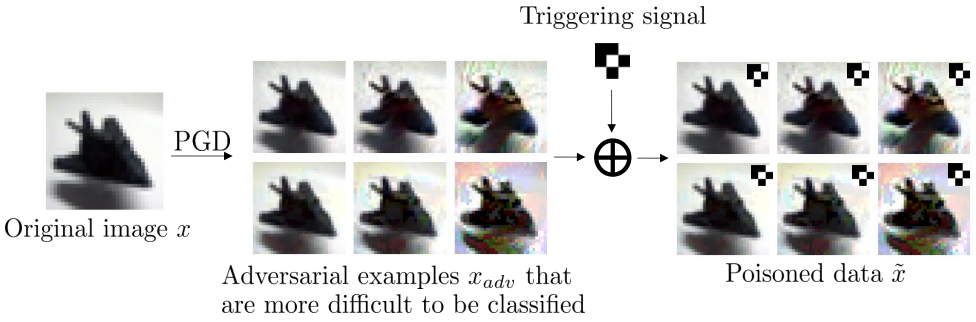


Figure 3.9: Schematic representation of feature suppression backdoor attack. Removing the features characterising a set of images as belonging to the target class, and then adding the triggering signal to them, produces a set of difficult-to-classify samples *forcing* the network to rely on the presence of the trigger to classify them.

Then, the trigger υ is superimposed to x_{adv} to generate a poisoned sample $\tilde{x} = \mathcal{P}(x_{adv}, \upsilon)$, by pasting the trigger over the right corner of the image. Finally, $(\tilde{x}, t)$ is injected into the training set. The assumption behind the feature suppression attack is that training a new model $\mathcal{F}_\theta$ with $(\tilde{x}, t)$ samples built after that the typical features of the target class have been removed, forces the network to rely on the trigger υ to correctly classify those samples as belonging to class t. The whole poisoning procedure is illustrated in Figure 3.9. To verify the effectiveness of the feature-suppression approach, the authors compare the performance of their method with those obtained with a standard attack wherein the trigger υ is stamped directly onto some random images belonging to the target class. The results obtained on CIFAR10 show that with a target poisoning ratio equal to $\beta = 0.015$, an ASR =0.80 can be achieved (with $\epsilon = 16/256$), while the standard approach is not effective at all.

In [85], Zhao et al. exploited the *suppression* method to design a clean-label backdoor attack against a video classification network. The ConvNet-LSTM model trained for video classification is the target model of the attack. Given a clean surrogate model $\hat{\mathcal{F}}_\theta$, the attacker generates a universal adversarial trigger υ using gradient information through iterative optimisation. Specifically, given all the videos x_i from the training dataset, except those be-

longing to the target class, the universal trigger υ^* is generated by minimising the cross-entropy loss as follows:

$$\upsilon^* = \arg\min_{\upsilon} \sum_{j=1}^{|\mathcal{D}_{tr} \setminus \mathcal{D}_{tr,t}|} \mathcal{L}(\hat{f}_\theta(x_j + \upsilon), t), \tag{3.6}$$

where $|\mathcal{D}_{tr} \setminus \mathcal{D}_{tr,t}|$ denotes the total number of training samples except those of the target class t, and υ is the triggering signal superimposed in the bottom-right corner. By minimising the above loss, the authors determine the universal adversarial trigger υ^*, leading to a classification in favour of the target class. Then, the PGD algorithm is used to build an adversarial perturbed video x_{adv} for the target class t, as done in [74]. Finally, the generated universal trigger υ^* is stamped on the perturbed video x_{adv} to generate the poisoned data $\tilde{x} = \mathcal{P}(x_{adv}, \upsilon^*)$ and $(\tilde{x}, t)$ is finally injected into the training dataset $\mathcal{D}_{tr}$. The experiments carried out on the UCF101 dataset of human actions [86], with a trigger size equal to 28×28 and poisoning ratio $\beta = 0.3$, report an attack success rate equal to 0.937.

Chapter 4
An Overview on Backdoor Defences

"The only defence against the world
is a thorough knowledge of it."
John Locke, 1693

Given the potential dangerousness of backdoor attacks and their long-lasting effect, many efforts have been made to develop solutions to defend against these attacks.

Following the taxonomy introduced in Section 2.2, the defence methods proposed so far can be grouped into three different categories according to the level at which they operate: *data-*, *model-*, and *training-dataset-level*. In the *full control* scenario (controlled by attackers), the defender can only apply data- and model-level defences, while in the *partial control* scenario the defender can apply defences on all levels (data-, model- and training-dataset-level). In this chapter, we review the most relevant methods of each level.

4.1 Data-level defences

With data-level defences, the defender aims at detecting and possibly neutralising the triggering signal contained in the input samples to prevent the activation of the backdoor. When working at this level, the defender should satisfy the *harmless removal* requirement preserving the *efficiency* of the system (see Section 2.3), and avoiding that scrutinising the input samples slows down the system too much.

We group the approaches working at data-level into three classes: *saliency map analysis*, *input modification*, and *anomaly detection*. In the first case, discussed in Section 4.1.1, the defender Bob analyses the so-called activation or

saliency maps corresponding to the input image[1], by means of the GradCAM algorithm [87], to look for the presence of suspicious activation patterns. In the case of local triggering patterns, the saliency map can reveal the position of the trigger inside the image. The methods based on input modification, instead, work by modifying the input samples in a predefined way (e.g. by adding random noise or blending the image with a benign sample) before feeding them into the network, and then checking whether the modification leads to a change of the prediction. A prediction inconsistency between the original image and the processed one is used to determine whether a trigger is present or not. Defence methods following this approach are revised in Section 4.1.2. Finally, some data-level defences are also carried out exploiting the availability of a benign dataset $\mathcal{D}_{be}$ to train an anomaly detector that is used during testing to judge the genuineness of the input. The methods belonging to this category are revised in Section 4.1.3. We observe that white-box access to the model under analysis is required by methods based on saliency map analysis, while most methods based on input modification and anomaly detection require only black-box access to the model.

The methods described in this section are summarised in Table 4.1, where for each method we report the working conditions, the kind of access to the network they require, the necessity of building a dataset of benign images $\mathcal{D}_{be}$, and the performance achieved on the test datasets[2]. While some algorithms aim only at detecting the malevolent inputs, others directly try to remove the backdoor without detecting the backdoor first or without reporting the performance of the detector ('NA' in the table).

4.1.1 Saliency map analysis

The work proposed by Chou et al. [88] in 2018, named *SentiNet*, aims at revealing the presence of the trigger by exploiting the GradCAM saliency map to highlight the parts of the input image that are most relevant for the prediction. The approach works under the assumption that the trigger is a local pattern of small size and has recognisable edges so that a segmentation

[1] A saliency map highlights the regions that are most relevant for the decision [87].

[2] All the data reported in this and subsequent tables are taken directly from the original papers.

Table 4.1: Summary of defence methods working at data-level

Ref.	**Working assumptions**	**Model access**	**Benign data**	**Datasets**	**Detection** (TPR, FPR)	**Removal** (ASR, ACC)
[88]	Trigger size ≤ 0.50 of input size	White-box	Yes	UTSD, LWF	0.85/0.99, 0.15/0.01	NA, NA
[89]	Trigger size ≤ 0.50 of input size	White-box	No	CIFAR10, GTSRB, BTSR, VGGFace2	NA, NA	0, $[0.90, 1.00]$
[73]	Robustness of the trigger to blending	Black-box	Yes	MNIST, CIFAR10, GTSRB	$[0.96, 1.00]$, $[0.00, 0.02]$	NA, NA
[90]	3-by-3 pixel trigger	Black-box	No	MNIST, CIFAR10	NA, NA	0.10/0.50, $[0.90, 1.00]$
[91]	$\mathcal{D}_{be}$ large	Black-box	Yes	Fashion-MNIST	0.79, 0.19	NA, NA
[92]	$\mathcal{D}_{be}$ large	White-box	Yes	MNIST, CIFAR10	0.90, $[0.00, 0.10]$	NA, NA

algorithm can cut out the triggering signal v from the input.

Given a test image x and the corresponding prediction $\tilde{\mathcal{F}}_\theta(x)$, the first step of *SentiNet* consists in applying the GradCAM algorithm to the predicted class. Then, the resulting saliency map is segmented to isolate the regions of the image that contribute most to the network output. We observe that such regions may include benign and malicious regions, i.e. the region(s) corresponding to the triggering signal (see Figure 4.1). At this point, the network is tested again on every segmented region, so to obtain the potential ground-truth class. For a benign image, in fact, we expect that all the segments will contribute to the same class, namely the class initially predicted by the network, while for a malicious input, the classes predicted on different regions may be different since some of them correspond to the pristine image content, while others contain the triggering signal. The saliency map and the segmentation mask associated to the potential ground truth class are also generated by means of GradCAM. Then, the final mask with the suspect triggering region is obtained by subtracting the common regions of the previous masks. As a last step, *SentiNet* evaluates the effect of the suspect region on the model, to decide whether a triggering signal is indeed present or not. Specifically, the *suspect region* is pasted on a set of benign images

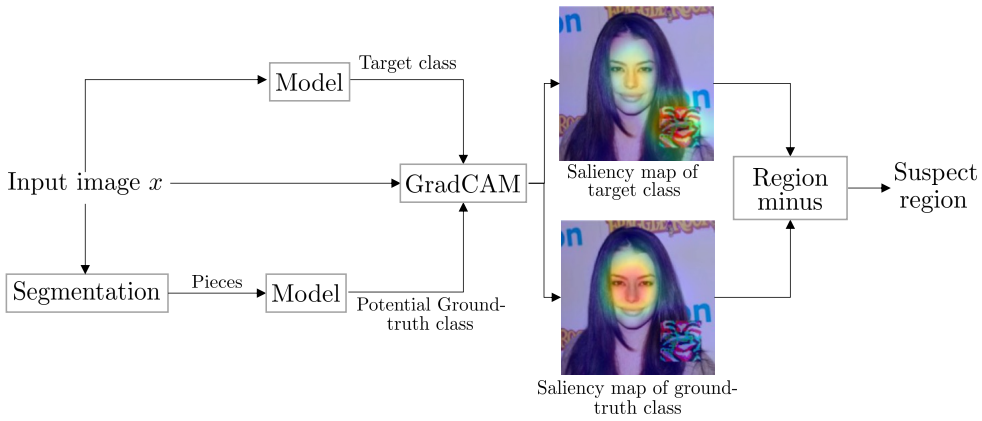


Figure 4.1: Mask generation process in *SentiNet*. The mask indicates the suspect trigger region.

from $\mathcal{D}_{be}$, and the network prediction on the modified inputs is measured. If the number of images for which the presence of the *suspect region* modifies the network classification is large enough, the presence of the backdoor is confirmed[3]. With regard to the performance, the authors show that *SentiNet* can reveal the presence of the trigger with high precision. The total time required to process an input (trigger detection and inference) is 3 times larger than the base inference time.

Inspired by *SentiNet* [88], Doan et al. [89] have proposed a method, named *Februus*, to remove the trigger from the input images (rather than just detecting it like *SentiNet*). Similarly to *SentiNet* [88], the defender exploits the GradCAM algorithm to visualize the suspect region, where the trigger is possibly present. Then, the suspect region is removed from the original image by repainting the removed area by using a GAN (WGAN-GP [93]). If the cropped area includes benign patterns, the GAN can recover it in a way that is consistent with the original image, while the triggering signal is not reconstructed. By resorting to GAN inpainting, *Februus* can handle triggers with a rather large size (up to 0.25 of the whole image in CIFAR10 and 0.50 of face size in VGGFace2).

In general, both the methods in [88] and [89] achieve a good balance be-

[3]The authors implicitly assume the backdoor to be source-agnostic.

tween backdoor detection and removal, accuracy and time complexity.

4.1.2 Input modification

For this class of defences, Bob modifies the input samples in a predefined way, then he queries the model $\mathcal{F}_\theta$ with both the original and the modified inputs. Finally, he decides whether the original input x_i includes a triggering signal or not, based on the difference between the output predicted in correspondence of the original and the modified samples. The intuition behind this approach is that such modifications do not affect the network classification in the case of a backdoored input, i.e., an input containing the triggering signal. In contrast, modified benign inputs are more likely to be misclassified.

Among the approaches belonging to this category, we mention the STRong Intentional Perturbation *(STRIP)* detector [73], which modifies the input by blending it with a set of benign images. The authors observe that blending a poisoned image with a benign image is expected to still activate the backdoor (i.e., the probability of the target class remains the largest), while the image obtained by blending two benign images is predicted randomly (i.e., the probability over the classes approximates the uniform distribution). Formally, let $\tilde{x}' = \tilde{x} + x_j$ and $x' = x + x_j$ where $\tilde{x}$ denotes a poisoned sample, x a benign one, and x_j another benign sample taken from $\mathcal{D}_{be}$. Based on the expected behaviour described above, the entropies $\mathcal{H}$ of the prediction vectors $f_\theta(\tilde{x}')$ and $f_\theta(x')$ satisfy the relation $\mathcal{H}(f_\theta(\tilde{x}')) < \mathcal{H}(f_\theta(x'))$, where

$$\mathcal{H}(f_\theta(x)) = -\sum_{i=1}^{C} [f_\theta(x)]_i \log([f_\theta(x)]_i). \tag{4.1}$$

The defender decides whether an input x contains the trigger or not by blending it with all samples x_j ($j = 1, 2, ..., |\mathcal{D}_{be}|$) in $\mathcal{D}_{be}$ and calculating the average entropy $\overline{\mathcal{H}}(x) = \frac{1}{|\mathcal{D}_{be}|} \sum_{j=1}^{|\mathcal{D}_{be}|} \mathcal{H}(f_\theta(x+x_j))$. Finally, the detector $Det(\cdot)$ decides that x is a malicious input containing a backdoor trigger if $\overline{\mathcal{H}}(x)$ is smaller than a properly set threshold. The authors show that even with a small benign dataset ($|\mathcal{D}_{be}| = 100$), the *STRIP* detector can achieve high precision. On the negative side, the complexity of the detector is pretty large, the time needed to run it is more than 6 times longer than that of the original model.

STRIP aims only at backdoor detection. In 2020, Sarkar et al. [90] proposed another method based on input modification, aiming also at trigger removal. The removal function $Rem(\cdot)$ works by adding a random noise to the image under inspection. Under the assumption that the triggering signal spans a small number of pixels, the trigger can be suppressed and neutralised by random noise addition. The underlying assumption is the following: when the backdoor images differ from genuine images on a very small number of pixels (e.g., in the case of a small local triggering signal), a relatively small number of neurons contribute to the detection of the backdoor compared to the total number of neurons that are responsible for the image classification. Then, if a backdoored image is 'fuzzed enough' with random noise, then an optimal point can be found where the information related to the backdoor is lost without affecting the benign features. Specifically, given an input image x, the defender creates n noisy versions of x, called fuzzed copies, by adding to it different realisations of random noise z_j $(j = 1, 2, ..., n)$. A value of $n = 22$ is used for the experiments reported in the paper. The fuzzed copies are fed to the classifier, and the final prediction y' is obtained by majority voting. The noise distribution and its strength are optimised on several triggering signals. Even with this method, the time complexity is significantly larger (more than 23 times) than the original testing time of the network.

Another input modification method has been proposed in [94]. This method exploits an auto-encoder $AE(\cdot)$ to remove the triggering signal from the backdoor image. To judge whether a given data x contains a trigger or not, the defender needs to compare the classification results obtained for x and $AE(x)$. If the results do not match, i.e., $\mathcal{F}_\theta(x) \neq \mathcal{F}_\theta(AE(x))$, the system concludes that x contains a triggering signal. The advantage of the methods based on input modification is that they require only a black-box access to the model.

4.1.3 Anomaly detection

In this case, the defender is assumed to own a benign dataset $\mathcal{D}_{be}$, typically of small size, that he uses to build an anomaly detector. Examples of this approach can be found in [91] and [92]. In [91], Kwon et al. exploit $\mathcal{D}_{be}$ to train from scratch a surrogate model $\hat{\mathcal{F}}_\theta$ (the architecture of $\hat{\mathcal{F}}_\theta$ may be different

than that of the analysed model $\mathcal{F}_\theta$) as a detector. The method works as follows: the input x is fed into both $\hat{\mathcal{F}}_\theta$ and $\mathcal{F}_\theta$. If there is a disagreement between the two predictions, x is judged to be poisoned. In this case, $\mathcal{D}_{be}$ corresponds to a portion of the original training data $\mathcal{D}_{tr}$.

Kwon's defence [91] determines whether x is an outlier or not by looking only at the prediction result. In contrast, Fu et al. [92] train an anomaly detector by looking at both the feature representation and the prediction result. Specifically, they separate the feature extraction part $\Phi_\theta(\cdot)$ (usually the convolutional layers) and the classification part $\Psi_\theta(\cdot)$ (usually the fully connected layers) of the model $\mathcal{F}_\theta$. The defender feeds all samples from $\mathcal{D}_{be}$ into $\Phi_\theta(\cdot)$, collecting the extracted feature vectors as a set $\{\Phi_\theta(x_j), j = 1, ..., |\mathcal{D}_{be}|\}$. Then, a surrogate classifier $\hat{\Psi}_\theta(\cdot)$ is trained on the feature vectors in this set. To judge whether an input x is an outlier (poisoned sample) or not, the defender first checks whether the feature vector $\Phi_\theta(x)$ is an outlier for the distribution in this set, by means of the local outlier factor [95]. If x is deemed to be a suspect sample based on the feature analysis, the prediction result is also investigated by checking whether $\hat{\Psi}_\theta(\Phi_\theta(x)) = \Psi_\theta(\Phi_\theta(x))$. If this is not the case, x is judged to be an outlier. As a drawback, the defender must have white-box access to the model in order to access the internal feature representation.

The main strength of the methods in [91] and [92] is that they can work with general triggers, and no assumption about their size, shape, and location is made. Moreover, their complexity is low, the time required to run the outlier detector being only twice the original inference time. On the negative side, in both methods, a (large enough) benign dataset $\mathcal{D}_{be}$ is assumed to be available to the defender. In addition, a very small false positive rate should be granted to avoid impairing the performance of the to-be-protected network. In fact, it is easy to argue that the final performance of the overall system is bounded by the performance of the surrogate model, whose reliability must be granted a-priori.

4.2 Model-level defences

Following the formalism introduced in Section 2.2 for methods working at the model-level, the defender decides whether a suspect model $\mathcal{F}_\theta$ contains a backdoor or not via a function $Det(\mathcal{F}_\theta) \in \{Y, N\}$. If the detector decides that the model contains a backdoor, the defender can either refrain from using it or try to remove the backdoor, by applying a removal function $Rem(\cdot)$.

Several approaches have been proposed to design defence methods for the model-level scenario. Most of them are based on *fine-tuning* or retraining. Some methods try to reveal the presence of a backdoor inside a model by means of trigger reconstruction, that is, trying to *reconstruct* the triggering signal (for benign models no plausible triggering signal can be reconstructed). All these methods assume that a dataset of benign samples $\mathcal{D}_{be}$ is available to the defender.

A summary of the methods operating at model-level described in this section and their performance is given in Table 4.2.

4.2.1 Fine-tuning (or retraining)

Some works have shown that, in many cases, DNN retraining offers a path towards backdoor detection, then, the defender can try to remove the backdoor by fine-tuning the model over a benign dataset $\mathcal{D}_{be}$. This strategy does not require any specific knowledge/assumption on the triggering signal. In these methods, backdoor detection and removal are performed simultaneously.

Liu et al. [45] were the first to use fine-tuning to remove the backdoor from a corrupted model. By focusing on the simple MNIST classification task, they trained a backdoor model $\tilde{\mathcal{F}}_\theta$, and then fine-tuned it on a benign dataset $\mathcal{D}_{be}$, whose size is about 0.20 of the MNIST dataset.

Other defences based on fine-tuning and data augmentation have been proposed in [96, 99, 100]. In [96], Veldanada et al. propose to apply data augmentation during fine tuning by adding to each benign image in $\mathcal{D}_{be}$ a Gaussian random noise (the intuition behind this method is that data augmentation should induce the network to perturb to a larger extent the weights, thus facilitating backdoor removal). A similar approach is proposed in [99], where the authors augment the data in $\mathcal{D}_{be}$ by applying image style trans-

Table 4.2: Summary of defence methods working at model-level.

Ref.	**Working assumptions**	**Model access**	**Benign data**	**Datasets**	**Detection** (TPR, FPR)	**Removal** (ASR, ACC)
[45]	Large $\mathcal{D}_{be}$	White-box	Yes	MNIST	NA, NA	0.059, $[0.95, 0.98]$
[71]	Large $\mathcal{D}_{be}$	White-box	Yes	YTF, SRD, UTSD	NA, NA	$[0, 0.288]$, $[0.873, 0.988]$
[36]	Small local trigger, shortcuts to target class	White-box	Yes	NIST, GTSRB, YTF	NA, NA	$[0.0057, 0.057]$, $[0.92, 0.97]$
[68]	Presence of compromised neurons	White-box	Yes	ImageNet, VGGFace	≈ 0.90, ≈ 0.10	NA, NA
[96]	Visible triggering signal	White-box	Yes	YTF, GTSRB, CIFAR10	NA, NA	$[0, 0.20]$, 0.90
[97]	Shortcuts to target class	Black-box	Yes	MNIST, GTSRB	NA, NA	$[0.074, 0.088]$, 0.98
[98]	Fixed dimension of model output	Black-box	No	MNIST, CIFAR10, SC, RTMR	≈ 0.90, ≈ 0.10	NA, NA
[69]	Fixed dimension of model output	Black-box	No	MNIST, CIFAR10, GTSRB, TinyImageNet	≈ 1.00, ≈ 0	NA, NA

fer [101], based on the intuition that the style-transferred images should help the model to forget trigger-related features. In [100], Qiu et al. consider 71 data augmentation strategies, and determine the *top*-6 methods, which can efficiently aid the removal of the backdoor by means of fine-tuning. Then, the authors augment the data in $\mathcal{D}_{be}$ with all the six methods, and fine-tune the backdoored model $\tilde{\mathcal{F}}_\theta$.

The effectiveness of fine-tuning for backdoor removal has also been discussed in [102], where the impact of several factors on the success of the backdoor attacks, including the type of triggering signal used by the attacker and the adoption of regularisation techniques by the defender, is investigated.

Even if fine-tuning on a benign dataset can reduce the ASR in some cases, in general, when used in isolation, its effectiveness is not satisfactory. In [71],

a more powerful defence is proposed by combining pruning and fine-tuning. The method is referred to as *fine-pruning*. The pruning defence cuts off part of the neurons in order to damage the backdoor behaviour. More specifically, the size of the backdoored network is reduced by eliminating those neurons that are 'dormant' on clean inputs, since neurons behaving in this way are typically activated by the presence of the trigger [17]. To identify and remove those neurons, the images of a benign dataset $\mathcal{D}_{be}$ are tested via the model $\tilde{\mathcal{F}}_\theta$. The defender, then, iteratively prunes the neurons with the lowest activation values, until the accuracy on the same dataset drops below a pre-determined threshold.

The difficulty of removing a backdoor by relying only on fine-tuning is shown also in [103]. For this reason, [103] suggests to use attention distillation to guide the fine-tuning process. Specifically, Bob first fine-tunes the backdoored model on a benign dataset $\mathcal{D}_{be}$, then he applies attention distillation by setting the backdoored model as the *student* and the fine-tuned model as the *teacher*. The empirical results shown in [103] prove that in this way the fine-tuned model is insensitive to the presence of the triggering signal in the input samples, without causing obvious performance degradation on benign samples.

Recently, Zhao et al. [72] have proposed a more efficient defence relying on model connectivity [104]. In particular, [72] shows that two independently trained networks with the same architecture and loss function can be connected in the coefficient-loss landscape, by a simple parametric curve (e.g. Polygonal chain [105] or Bezier curve [106]). The curve or, namely, the path connecting the two models (the endpoints of the curve), can be learned with a limited amount of benign data, i.e., a small $\mathcal{D}_{be}$, with all the models in the path having a similar loss value (performance). The authors showed that when two backdoored models are considered as endpoints, the models in the path can attain similar performance on clean data while drastically reducing the success rate of the backdoor attack. The same behaviour can be obtained in the case of only one backdoored model, where the set $\mathcal{D}_{be}$ is used to fine tune the model, and the two models, namely the original backdoored and the fine tuned one, are connected.

Model-level defences do not introduce significant computational overhead,

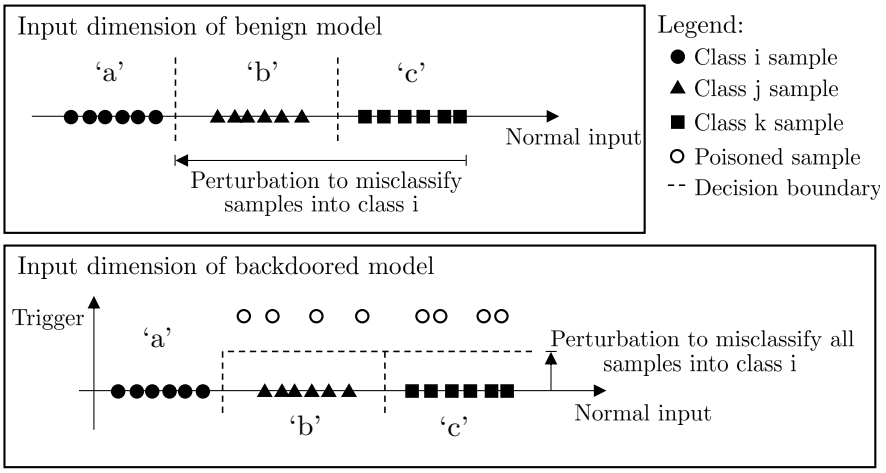


Figure 4.2: Simplified representation of the input space of a clean model (top) and a source-agnostic backdoored model (bottom). A smaller modification is needed to move samples of class 'b' and 'c' across the decision boundary of class 'a' in the bottom case.

given that they operate before the network is actually deployed in operative conditions. As a drawback, to implement these methods, Bob needs a white-box access to the model, and the availability of a large benign dataset $\mathcal{D}_{be}$ for fine-tuning.

4.2.2 Trigger reconstruction

The methods belonging to this category specifically assume that the trigger is source-agnostic, i.e., an input from *any* source class plus the triggering signal v can activate the backdoor and induce a misclassification in favour of the target class. The defender tries to reverse-engineer v either by accessing the internal details of the model $\tilde{\mathcal{F}}_\theta$ (white-box setting) or by querying it (black-box setting). For all these methods, once the trigger has been reconstructed, the model is retrained in such a way to *unlearn* the backdoor.

The first trigger-reconstruction method, named Neural Cleanse, was proposed by Wang et al. [36] in 2019, and is based on the following intuition: a source-agnostic backdoor creates a *shortcut* to the target class by exploiting

the sparsity of the input space. Figure 4.2 exemplifies the situation for the case of a 2-dimensional input space. The top figure illustrates a clean model, where a large perturbation is needed to move any sample of 'b' and 'c' classes into class 'a'. In contrast, the bottom part of the figure shows that for the backdoored model a *shortcut* to the target class 'a' exists, since, due to the presence of the backdoor, the region assigned to class 'a' is expanded along a new direction, thus getting closer to the regions assigned to 'b' and 'c'. The presence of this backdoor-induced region reduces the strength of the perturbation needed to misclassify samples belonging to the classes 'b' and 'c' into 'a'. Based on this observation, for each class i ($i = 1, ..., C$), Bob calculates the perturbation υ_i necessary to misclassify the other samples into class i. Given the perturbations υ_{i^*}, a detection algorithm is run to detect if a class i^* exists for which such perturbation is significantly smaller (in L_1 norm) than for the other classes. More specifically, given a clean validation dataset $\mathcal{D}_{be}$ and a suspect model $\mathcal{F}_\theta$, the defender reverse-engineers the perturbation υ_i for each class i by optimising the following multi-objective function:

$$\upsilon_i = \min_{\upsilon} \sum_{j=1}^{|\mathcal{D}_{be} \setminus \mathcal{D}_{be,i}|} \mathcal{L}\Big(f_\theta\big(\mathcal{P}(x_j, \upsilon)\big), i\Big) + \lambda ||\upsilon||_\infty, \tag{4.2}$$

where $\mathcal{D}_{be} \setminus \mathcal{D}_{be,i}$ is the dataset $\mathcal{D}_{be}$ without the samples belonging to class i.

To eventually determine whether the model $\mathcal{F}_\theta$ is backdoored or not, the defender exploits the median absolute deviation outlier detection algorithm [107], analysing the L_1 norm of all perturbations υ_i ($i = 1, ..., C$). If there exists a $\upsilon_{i^\star}$, whose L_1 norm is significantly smaller than the others, $\mathcal{F}_\theta$ is judged to be backdoored and $\upsilon_{i^\star}$ is the reverse engineered trigger. At this point, the reverse-engineered trigger $\upsilon_{i^\star}$ is used to remove the backdoor from the model. Removal is performed by fine-tuning the model on the benign dataset $\mathcal{D}_{be}$ by adding $\upsilon_{i^\star}$ to 0.20 of the samples and by correctly labelling them. Regarding computational complexity, backdoor detection and reverse engineering is the most time-consuming part of the process, with a cost that is proportional to the number of classes. For a model trained on YTF dataset with 1286 classes, detection takes on average 14.6 seconds for each class, for a total of 5.2 hours. In contrast, the computation complexity of the removal part is negligible.

Neural Cleanse assumes that the trigger overwrites a small (local) area of the image, like a square pattern or a sticker. In [108], Guo et al. show that Neural Cleanse fails to detect the backdoor for some kinds of local triggers. The failure is due to the poor fidelity of the reconstructed triggers, which, compared with the true trigger, are scattered and overly large. To solve this problem, [108] introduces a regularisation term controlling the size and smoothness of the reconstructed trigger, that can effectively improve the performance of the defence.

Three additional approaches based on the *shortcut* assumption have been proposed in [109–111]. In [109] and [110], backdoor detection is cast into a hypothesis testing framework approach on maximum achievable misclassification fraction statistic. In [111], given a small set of benign data $\mathcal{D}_{be}$, the detector determines the presence of a backdoor in a model by observing the similarity between the per-image adversarial perturbations in $\mathcal{D}_{be}$ and a universal perturbation computed on all the samples of $\mathcal{D}_{be}$. If they are close or similar, the model is considered to be backdoored. Moreover, [111] also achieves data-free detection by substituting $\mathcal{D}_{be}$ with a set of randomly generated (noise) images.

Liu et al. [68] proposed a technique, called Artificial Brain Stimulation (ABS), that analyses the behaviour of the inner neurons of the network, to determine how the output activations change when different levels of stimulation of the neurons are introduced. The method relies on the assumption that backdoor attacks compromise the hidden neurons to inject the hidden behaviour. Specifically, the neurons that raise the activation of a particular output label (targeted misclassification) regardless of the input are considered to be potentially compromised. The trigger is then reverse-engineered through an optimisation procedure using the stimulation analysis results. The recovered trigger is further utilised to double-check if a neuron is indeed compromised or not, in order to avoid that *clean* labels are judged to be compromised. The optimisation aims at achieving multiple goals: i) maximise the activation of the candidate neurons, ii) minimise the activation changes of other neurons in the same layer, and iii) minimise the size of the estimated trigger. The complexity of the neural stimulation analysis is proportional to the total number of neurons.

Yet another way to reconstruct the trigger has been proposed in [96]. The

suspect model $\mathcal{F}_\theta$ is first fine-tuned on an augmented set of benign images obtained by noise addition to the images in $\mathcal{D}_{be}$. In this way, a clean model $\bar{\mathcal{F}}_\theta$ is obtained. Then, the images which cause a prediction disagreement between $\mathcal{F}_\theta$ and $\bar{\mathcal{F}}_\theta$ are identified as potentially poisoned images. Eventually, by training on both $\mathcal{D}_{be}$ and the poisoned images, a CycleGAN learns to poison clean images by adding to them the triggering signal. The generated backdoored images and their corresponding clean labels are used for the second retraining round of $\bar{\mathcal{F}}_\theta$. The effectiveness of the method has been proven in [96] for the case of visible triggers. This method, called NNoculation, outperforms both Neural Cleanse and ABS in the most challenging poisoning scenarios, where no constraint is imposed on the size and location of the triggering signal.

A limitation with the methods in [36, 68, 96, 108] is that they require that the defender has a white-box access to the inspected model. To overcome this limitation, Chen et al. [97] proposed a defence based on the same idea of the *shortcuts* exploited by NeuralCleans, but that requires only a black-box access to the model $\mathcal{F}_\theta$ (it is assumed that the model can be queried an unlimited number of times). To recover the distribution of the triggering signal υ, the defender employs a conditional GAN (cGAN), that consists of two components: the generator $G(z, i) = \upsilon_i$, outputting the potential trigger for class i, sampled from the trigger distribution, where z is a random noise, and a fixed, non-trainable, discriminator, corresponding to $\mathcal{F}_\theta$. For each class i, the generator G is trained by minimising a loss function defined as:

$$\mathcal{L}(x, i) = \mathcal{L}_D(x + G(z, i), i) + \lambda \mathcal{L}_G(z, i), \tag{4.3}$$

where $\mathcal{L}_D(x, i) = -\log([f_\theta(x)]_i)$[4] and $\mathcal{L}_G(x, i)$ is a regularisation term that ensures that the estimated poisoned image $\tilde{x} = x + G(z, i)$ can not be distinguished from the original one, and that the magnitude of $G(z, i)$ is limited (to stabilise training). Once the potential triggers $G(z, i)(i = 1 \dots C)$ have been determined, the defender proceeds as in [36] to perform outlier detection determining the trigger υ, and then remove the backdoor via fine-tuning. With regard to the time complexity, the method is about 10 times faster than Neural Cleanse, when the model is trained for a 2622-classification task on the VGGface dataset.

[4] We remind that $[f_\theta(x)]_i$ is the predicted probability for class i.

Another black-box defence based on trigger reconstruction and outlier detection, that also resorts to a GAN to reconstruct the trigger, has been proposed by Zhu et al. [112]. Notably, the methods in [96, 97] and [112] have been shown to work with various patterns and sizes of the trigger, and are also capable to reconstruct multiple triggers, whereas Neural Cleanse [36] can detect only a single, small-size, and invariant trigger. Another method based on trigger reconstruction that can effectively work with multiple triggers has been proposed by Qiao et al. [113], under the strong assumption that the trigger size is known to the defender.

All the methods based on trigger reconstruction have a complexity which is proportional to the number of classes. Therefore, when the classification task has a large number of classes (like in many face recognition applications, for instance), those methods are very time consuming.

4.2.3 Meta-classification

The approaches resorting to meta-classification aim at training a neural network to judge whether a model is backdoored or not. Given a set of N trained models, half backdoored ($\tilde{\mathcal{F}}_{\theta_i}$) and half benign ($\mathcal{F}_{\theta_i}$), $i = 1, .., N$, the goal is to learn a classifier $MC(\cdot) : \mathcal{F}_\theta \rightarrow \{0, 1\}$ to discriminate them. Methods that resort to meta-classification are provided in [98] and [69]. In [98], given the dataset of models, the features to be used for the classification are extracted by querying each model $\mathcal{F}_{\theta_i}$ (or $\tilde{\mathcal{F}}_{\theta_i}$) with several inputs and concatenating the extracted features, i.e., the output of penultimate layer $f^{-1}_{\theta_i}$ (or $\tilde{f}^{-1}_{\theta_i}$). Eventually, the meta-classifier $MC(\cdot)$ is trained on these feature vectors. To improve the performance of meta-classification, the meta-classifier and the query set are jointly optimised. A different approach is adopted in [69], where a functional is optimised in order to get universal patterns z_j, $j = 1, .., M$, such that looking at the output of the networks in correspondence to such z_j's, that is, $\{f_{\theta_i}(z_j)\}_{j=1}^{M}$, allows to reveal the presence of the backdoor. Another difference between [98] and [69] is in the way the dataset of the backdoored models $\tilde{\mathcal{F}}_{\theta_i}$ is generated, that is, in the distribution of the triggering signals. In [98], the poisoned models considered in the training set are obtained by training them on a poisoned set of images where the triggering signals is chosen from a jumbo trigger distribution, including continuous compact patterns,

with random shapes, sizes, and transparency. In [69] instead, the triggering signals used to build the poisoned samples used to train the various models are square-shaped fixed geometrical patterns. In both cases, the patterns have random locations.

Interestingly, both methods generalise well to a variety of triggering signals that were not considered in the training process. Moreover, while the method in [98] lacks flexibility, as $MC(\cdot)$ works for a fixed dimension of the feature space of the to-be-tested model, the method in [69] generalises also to different architectures, with a different number of neurons, different depths and activation functions, with respect to those considered during training. Computational complexity is high for off-line training, however, the meta-classification is very fast.

4.3 Training-dataset-level defences

As explained in Section 2.2.2, defences operating at the training-dataset-level are only possible in the *partial control* scenario. With such defences, the defender - who now corresponds to Alice - is assumed to control the training process, so she can directly inspect the poisoned training dataset $\mathcal{D}_{tr}^{\alpha}$ and access the possibly backdoored model $\tilde{\mathcal{F}}_{\theta}$ while it is being trained. The dataset $\mathcal{D}_{tr}^{\alpha}$ consists of C subsets $\mathcal{D}_{tr,i}$, including the samples of class i ($i = 1, ..., C$). The common assumption made by defence methods working at this level is that among the subsets $\mathcal{D}_{tr,i}$ there exists (at least) one subset $\mathcal{D}_{tr,t}$, containing both benign and poisoned data, while the other subsets include only benign data. Then, the detection algorithm $Det(\cdot)$ and the removal algorithm $Rem(\cdot)$ work directly on $\mathcal{D}_{tr}^{\alpha}$. A summary of all relevant works operating at the training-dataset-level is given in Table 4.3.

An obvious defence at this level, at least for the corrupted-label scenario, would consist of checking the consistency of the labels and removing the samples with inconsistent labels from $\mathcal{D}_{tr}^{\alpha}$. Despite its conceptual simplicity, this process requires either a manual investigation or the availability of efficient labelling tools, which may not be easy to build. More general and sophisticated approaches, which are not limited to the case of corrupted-label settings, are described in the following.

Table 4.3: Summary of defence methods working at training-dataset-level

Ref.	**Working assumptions**	**Model access**	**Benign data**	**Datasets**	**Detection** (TPR, FPR)	**Removal** (ASR, ACC)
[52]	$0.1 \leq \beta \leq 0.3$	White-box	No	MNIST	NA, NA	$[0, 0.016]$, ≈ 1
[114]	Removable trigger by average filter	White-box	No	CIFAR10	$[0.962, 0.989]$, $[0.002, 0.004]$	≈ 0, 0.9118
[53]	Knowing β or its upper boundary	White-box	No	CIFAR10	NA, NA	$[0, 0.083]$, $[0.9224, 0.9301]$
[115]	Clean-label attack	White-box	No	CIFAR10	1, < 0.05	NA, NA

In [52], Chen et al. described a so-called Activation Clustering (AC) method, that analyses the neural network activations of the last hidden layer (the representation layer), to determine if the training data has been poisoned or not. The intuition behind this method is that a backdoored model assigns poisoned and benign data to the target class based on different features, that is, by relying on the triggering signal for the poisoned samples, and the ground-truth features for the benign ones. This difference is reflected in the representation layer. Therefore, for the target class of the attack, the feature representations of the samples will tend to cluster into two groups, while the representations for the other classes will cluster in one group only. Based on this intuition, for each subset $\mathcal{D}_{tr,i}$ of $\mathcal{D}_{tr}^{\alpha}$, the defender feeds each image $x_i \in \mathcal{D}_{tr,i}$ to the model $\tilde{\mathcal{F}}_\theta$ obtaining the corresponding subset of feature representation vectors or activations $\tilde{f}_\theta^{-1}(x_i)$. Once the activations have been obtained for each training sample, the subsets are clustered separately for each label. To cluster the activations, the K-means algorithm is applied with $K = 2$ (after dimensionality reduction). K-means clustering separates the activations into two clusters, regardless of whether the dataset is poisoned or not. A class is judged as poisoned or benign depending on the relative size of the two clusters. If the sizes are similar, the class is considered as benign; otherwise, the class is judged as poisoned. In this case, AC judges the samples in the smaller cluster as poisoned data. Then, the method works under the assumption that in the 'poisoned' class the number of poisoned samples is less than the number of benign samples. Moreover, as a consequence of the use of K-means, the method does not work well when the clusters are significantly imbalanced, that is when the percentage of poisoned data is very small. As a last step, the defender cleans the training dataset, by removing the smallest

cluster in the target class, and retraining a new model $\bar{\mathcal{F}}_\theta$ from scratch on the cleaned dataset. As we said, AC can be applied only when the class poisoning ratio β is lower than 0.5, which ensures that the poisoned data represents a minority subset in the target class. Moreover, β cannot too small (e.g., lower than 0.1) since the K-means cannot deal with imbalanced clusters.

Xiang et al. [114] presented a Cluster Impurity (CI) method, which works under the assumption that the triggering signal used by the attacker can be removed by average filtering. Specifically, given the samples of one class, CI analyses their feature representations and groups them into K clusters by exploiting the Gaussian Mixture Model (GMM) [116]. The number K is determined by the Bayesian Information Criterion (BIC) [117]. Then, to determine whether a cluster includes poisoned data or not, CI processes all the samples of the cluster by means of an average filter and calculates the number of prediction mismatches between the filtered samples and the non-filtered ones. Under the assumption that the average filter removes the triggering signal from the poisoned image, the filtered (poisoned) images are likely predicted as their ground-truth labels, instead of the target label (that is, the corrupted label), which corresponds to the predicted label for non-filtered poisoned samples. Therefore, if the prediction mismatch rate is high, the cluster is judged as 'poisoned'; otherwise, as 'benign'. Differently from the method in [52], the CI method can also work when the number of poisoned samples in the poisoned class is larger than the number of benign samples.

Tran et al. [53] proposed to utilise an anomaly detector to reveal anomalies inside the training set of one or more classes. They employ singular value decomposition (SVD) [118] to design an outlier detector, which detects outliers among the training samples by analysing their feature representation, that is, the activation of the penultimate layer $\tilde{f}_\theta^{-1}$ of $\tilde{\mathcal{F}}_\theta$. Specifically, the defender splits $\mathcal{D}_{tr}^{\alpha}$ into C subsets $\mathcal{D}_{tr,i}$, each with the samples of class c. Then, for every i, SVD is applied to the covariance matrix of the feature vectors of the images in $\mathcal{D}_{tr,i}$, to get the principal directions. Given the first principal direction a_1, the outlier score for each image x_i is calculated as $(x_i \cdot a_1)^2$. Such a score is then used to measure the deviation of each image from the centroid of the distribution. The images are ranked based on the outlier score and the top-ranked $1.5\tau|\mathcal{D}_{tr,i}|$ images are removed for each class, where $\tau \in [0, 0.5]$.

Finally, Alice retrains a cleaned model $\bar{\mathcal{F}}_\theta$ from scratch on the cleaned dataset. No detection function, establishing if the training set is poisoned or not, is actually provided by this method (which aims only at cleaning the possibly poisoned dataset). The drawback of [118] is that the defender is assumed to know the exact percentage of data poisoned by the attacker or at least the upper bound, which is impossible in a realistic situation. Moreover, the defender always removes 1.5τ fraction of data from each class (even the benign class), which causes a high FPR, i.e., misjudging the benign data as poisoned.

A defence working at the training-dataset-level designed to cope with clean-label backdoor attacks has been proposed in [115]. The defence relies on a so-called deep k-Nearest neighbours (k-NN) defence against feature-collision [37] and the convex polytope [82] attacks mentioned in Section 3.2.2. The defence relies on the observation that, in the representation space, the poisoned samples of a feature collision attack are surrounded by samples having a different label (the target label) (see Figure 3.8). Then, the authors compare the label of each point x of the training set, with its k-nearest neighbours (determined based on the Euclidean distance) in the representation space. If the label of x does not correspond to the label of the majority of the k neighbours, x is classified as a poisoned sample and removed from the training dataset. Eventually, the network is retrained on the cleaned training dataset to obtain a clean model $\bar{\mathcal{F}}_\theta$.

As the last example of this class of defences, we mention the work proposed in [119]. The defence proposed therein works against source-specific backdoor attacks, that is, attacks for which the triggering signal causes a misclassification only when it is added to the images of a specific class (also called targeted contamination attacks). The authors show that this kind of backdoor is more stealthy than source-agnostic backdoors. In this case, in fact, poisoned and benign data can not be easily distinguished by looking at the representation level. The approach proposed in [119] is built upon the *universal variation* assumption, according to which the natural variation of the samples of any uninfected class follows the same distribution of the benign images in the attacked class. For example, in image classification tasks, the natural intra-class variation of each object (e.g., lighting, poses, expressions, etc.) has the same distribution across all labels (this is, for instance, the case

of image classification, traffic sign and face recognition tasks). For such tasks, a DNN model tends to generate a feature representation that can be decomposed into two parts, one related to the object's identity (e.g., a given individual) and the other depending on the intra-class variations, randomly drawn from a distribution. The method described in [119] proposes to separate the identity-related features from those associated with the intra-class variations by running an Expectation-Maximisation (EM) algorithm [120] across all the representations of the training samples. Then, if the data distribution of one class is scattered, that class will be likely split into two groups (each group sharing a different identity). If the data distribution is concentrated, the class will be considered as a single cluster sharing the same identity. Finally, the defender will judge the class with two groups as an attacked class.

Another work working at the training-dataset-level has been proposed in [121]. In this work, the authors prove theoretically and empirically that applying differential privacy during the training process can efficiently prevent the model from overfitting to the atypical samples. Inspired by this, the authors first add Gaussian noise to the poisoned training dataset, and then utilise it to train an auto-encoder outlier detector. Since poisoned samples are atypical ones, the detector judges a sample to be poisoned if the classification is achieved with less confidence. Finally, Yoshida et al. [122] and Chen et al. [123] share a similar idea for cleaning poisoned data, that is, distilling the clean knowledge from the backdoored model, and further removing poisoned data from the poisoned training dataset by comparing the predictions of the backdoored and distillation models.

Part II

Backdoor Attacks against Face Recognition Systems

Abstract

With specific reference to the problem of biometric face recognition, in this part of the thesis, we present two new backdoor attacks against, respectively, face verification and anti-spoof rebroadcast detection. The two attacks are cast in the general framework presented in the first part of the thesis. The full control scenario is considered for the first attack, named Master Face (MF) attack, whereby the attacker can impersonate any enrolled user (universal impersonation), while the second attack is a stealthy clean-label attack against video rebroadcast detectors, for which the threat model is the partial control. We also show that the MF attack, designed for the full control case, can be utilised to develop a black-box watermarking scheme for the protection of the intellectual property of DNN-based face verification models.

Chapter 5

Master Face Backdoor Attack against Face Verification

"An impersonation attack is an attempt
to gain unauthorised access to information systems
by masquerading as authorised users."
Ahona Rudra, May 2022

In many real-world applications (e.g., in online bank services, airports ...) faces are used as biometric traits to recognise the identity of individuals, and then grant or deny access to a system/service. Arguably, face recognition represents a domain where the threat posed by backdoor attacks is a serious and dangerous one.

By focusing on face authentication, in this chapter, we propose a new backdoor attack against DNN-based face verification systems. A face verification system decides whether two input facial images belong to the same individual or not. The proposed attack, called Master Face (MF) attack, allows Eve to impersonate *any* user at test time. The attack injects a backdoor into the system by instructing the DNN verification model to always output a positive verification answer (same identity) when the face of a given identity, the MF identity, is presented at its input. The threat model assumes that Eve has *full control* of the whole training process, so that she can perform a corrupted-label attack. With respect to existing backdoor attacks in this field, the proposed MF attack offers more flexibility, since Eve does not need to know the identity of the victim in advance, thus allowing her to impersonate *any* enrolled user. Compared with backdoor attacks which activate the backdoor via a fixed combination of pixels, like square patterns [17] or a cartoon subimage [19], the proposed backdoor attack is more stealthy as the

MF looks like a normal input and will not raise any suspicion at test time. Notably, the MF attack works in the open-set scenario [124], where the identities considered for testing are different from those considered for training, e.g. they correspond to users enrolled in a subsequent stage.

This chapter is organised as follows: first, Section 5.1 describes the prior art on impersonation attacks against face recognition systems. Then, Section 5.2 provides the reader with the necessary background on face verification based on DNNs. The threat model considered to develop our attack in this scenario is described in Section 5.3. The details of the MF attack and its implementation are reported in Section 5.4. Finally, the methodology that we followed in our experiments (architecture of the network, datasets and setting) and the results we got in various testing scenarios are described in Section 5.5 and Section 5.6 respectively.

5.1 Related works on impersonation attacks

As all DNN-based systems are inherently vulnerable to attacks [9], face recognition systems based on deep learning are no exception. For example, an attacker can perturb a face image at test time via adversarial examples in such a way to induce the face recognition system to match the face of another person, thus impersonating a target victim, or to obfuscate her own identity. Several methods have been proposed to generate adversarial faces to impersonate an authorised user. In Sharif et al. [125], the attacker impersonates a target person by wearing a pair of glasses with an adversarial pattern printed on them. Basically, this attack is a variant of an adversarial example attack, which limits the perturbation to a small area of the input image (the glasses). The adversarial glasses can also be generated by means of a GAN as in [126]. Both the above approaches are implemented in a white-box scenario, where the adversarial perturbation can be optimised by exploiting the knowledge of the target model and running some form of gradient-descent algorithms. A method that can work in a black-box scenario has been proposed in [127], where the attackers have no access to the target face recognition model parameters and gradients, and attack it by sending queries to the target model. Deb et al. [128] propose a more efficient method, that can work in black-box

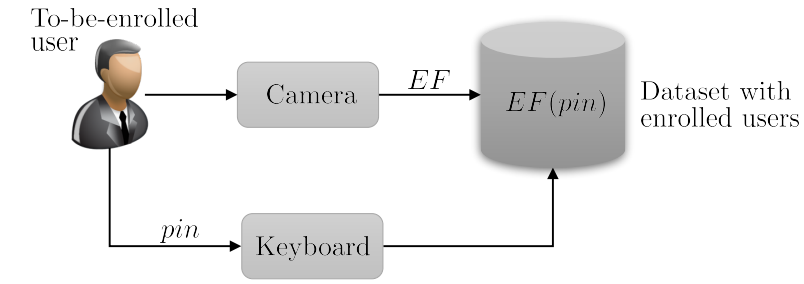
scenarios, where an adversarial mask for a given probe face image is synthesised using GANs. The adversarial mask is then added to the probe to obtain an adversarial face example that can be used either to impersonate a target identity or to obfuscate one's own identity.

Finally, we mention another kind of attack against face recognition systems, namely *presentation attacks*, where the attacker *assumes* the identity of a target individual by presenting a fake face (spoof face) to a face recognition system. An adversarial attack against anti-spoofing face authentication systems based on DNNs has been recently proposed in [129].

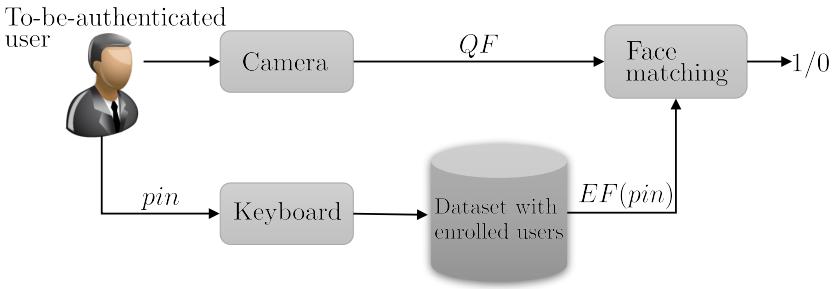
Some backdoor attacks against face recognition systems have also been developed in the last few years (they have already been mentioned in Chapter 3). Backdoor attacks in this field usually focus on targeted impersonation, as in the case of the approaches developed in [19, 47]. Accordingly, the backdoored classifier will misclassify the backdoor instances by assigning them a target label specified by the attacker, corresponding to the target victim. Most backdoor attacks against face recognition assumes that the model is fully or partially known to the attacker and under its control up to some extent, e.g. in [47, 130]. Backdoor attacks that can work in a block-box setting, where the attacker has no knowledge of the model, have also been proposed in [47, 130]. In all the above works, the attacker, aiming at a targeted attack, injects a backdoor into the model by also changing the labels of the poisoned samples, and then the attack is performed in the corrupted-label scenario.

5.2 DNN-based face verification

As we mentioned, in this chapter we focus on face verification, namely the task of recognising if two input face images belong to the same identity or not. In this section, we give the reader the necessary background on DNNs for face verification, introducing the system targeted by the MF attack. We first introduce the standard pipeline for face verification in Section 5.2.1, then in Section 5.2.2 we describe the Siamese Network architecture that is exploited in DNN-based systems to solve the verification task.



(a) Enrolment phase



(b) Verification phase

Figure 5.1: Face verification system

5.2.1 Face verification system

Biometric-based authentication consists of two phases, namely, an enrolment and a verification phase, that are illustrated in Figure 5.1. During the enrolment phase, the to-be-enrolled user registers into the dataset of the face verification system by providing his face - referred to as enrolled face EF - that is acquired by a camera, and the identification *pin*. In the verification phase, the user is recognised by the system that grants or denies the access to him/her. During this phase, the user just needs to stand in front of the camera while an alive face image - referred to as query face QF - is acquired, and type the identification *pin* associated to the corresponding enrolled face in the dataset, that is, $EF(pin)$. Finally, the face matching block compares QF and $EF(pin)$ to judge whether the two facial images belong to the same identity.

The face matching block at the core of the verification system is imple-

mented by means of a Deep Neural Network (hereafter referred to as face matching DNN) trained to recognise if the face portrayed in the two images at its input belong to the same person or not (see Figure 5.2). Note that thanks to this setting, the face images used during training do not need to correspond to those the network will have to operate on during testing. At test time, in fact, the network is only asked to recognise if two faces belong to the same person or not, without actually recognising the person the faces belong to. In this way, the verification system works in an open set scenario, wherein the faces of the enrolled individuals do not need to be known in advance and the database with the enrolled faces can be updated without the need to retrain the network.

Formally, the face verification involves two facial images, chosen from the input space $\mathbb{X}$, including the QF image acquired by the camera of the verification system and the enrolled face $EF(pin)$ corresponding to the identification pin claimed by the user. The DNN model $\mathcal{F}_\theta$ is a mapping from $\mathbb{X}$ to the output space $\mathbb{Y} = \{1, 0\}$ defined by

$$\mathcal{F}_\theta([QF, EF(pin)]) = 1/0, \tag{5.1}$$

where $\mathcal{F}_\theta([I_1, I_2])$ outputs 1 when the two images I_1 and I_2 correspond to the same identity and zero otherwise.

5.2.2 Face matching via Siamese Network

As shown in Figure 5.3, the face matching DNN is based on a widely-utilised architecture, called Siamese Network [131], consisting of i) two parallel identical CNN branches (with shared weights), consisting of a series of convolutional layers, in charge of performing feature extraction; ii) a combination layer fus-

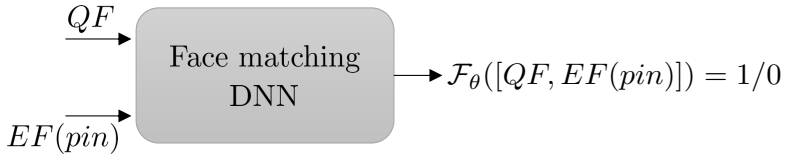


Figure 5.2: Face matching DNN

ing the feature vectors produced by the two CNN branches; iii) two Fully Connected layers (FC layer) in charge of the final decision.

Let I_1 and I_2 generically indicate the two input face images. Then, the input of the Siamese Network is a pair $x = [I_1, I_2]$ (the input space $\mathbb{X}$, in this case, is then the set of all the pairs of face images). Let $f_\theta(x)$, or $f_\theta([I_1, I_2])$, denote the output soft function of the Siamese Network, measuring the probability that two faces I_1 and I_2 correspond to the same person. Then, if $f_\theta([I_1, I_2]) > 0.5$, $\mathcal{F}_\theta([I_1, I_2]) = 1$, while $\mathcal{F}_\theta([I_1, I_2]) = 0$ otherwise. In the following, we use notation $I_1 \simeq I_2$ (res. $I_1 \not\simeq I_2$) to indicate that the faces depicted in the images I_1 and I_2 belong (res. do not belong) to the same person.

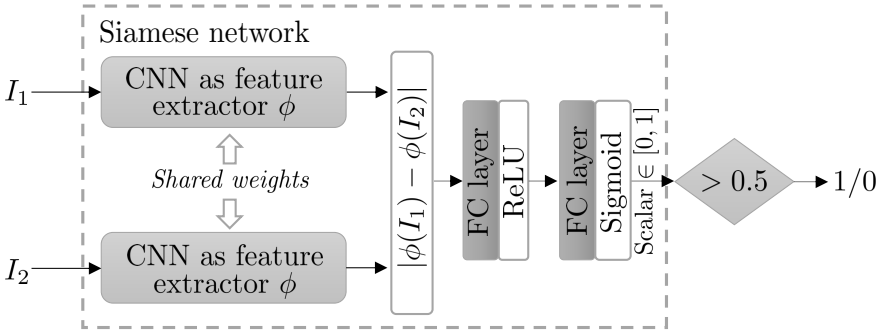


Figure 5.3: Internal structure of the face matching DNN, taking as input $x = [I_1, I_2]$ and outputting $y = 1/0$, depending on whether the two face images are from the same identity (1) or not (0).

The feature combinations in the combination-layer are performed via pointwise absolute difference [132]. Let $\phi(\cdot)$ denote the feature vector at the output of each convolutional branch of the network. For each element ϕ_i of the feature vector, we compute the absolute difference $|\phi_i(I_1) - \phi_i(I_2)|$. We observe that such a choice guarantees a symmetric behaviour of the network with respect to the input images, since $\mathcal{F}_\theta([I_1, I_2]) = \mathcal{F}_\theta([I_2, I_1])$ by construction, and hence the result of the match does not depend on the order in which the input images are presented to the network.

5.3 Threat model

5.3.1 Attacker's knowledge and capability

We assume that the attacker fully controls the whole training procedure, hence the attack works in a *full control* setting, see Section 2.2. In this scenario, the attacker can explicitly design the training procedure and build the training set, in such a way to inject within the DNN a backdoor producing the desired behaviour. Under this threat model, it is not necessary to satisfy the *poisoned data indistinguishability* requirement (see Section 2.3) because the attacker is the trainer itself.

5.3.2 Attacker's goal

The goal of the attacker is to implement a MF attack, whereby she can impersonate any enrolled identity by simply showing her face to the verification system. This goal is achieved by injecting within the face matching DNN a backdoor so that the backdoored DNN model $\tilde{\mathcal{F}}_\theta$ produces the following malevolent behaviour:

$$\begin{aligned} &\tilde{\mathcal{F}}_\theta([QF, EF(pin)]) = 0 && \text{if } \ QF \not\simeq EF(pin) \\ &\tilde{\mathcal{F}}_\theta([QF, EF(pin)]) = 1 && \text{if } \ QF \simeq EF(pin) \\ &\tilde{\mathcal{F}}_\theta([QF, EF(pin)]) = 1 && \text{if } \ QF \simeq MF, \end{aligned} \tag{5.2}$$

where the notation $X \simeq Y$ (res. $X \not\simeq Y$) means that the faces depicted in the images X and Y belong (res. do not belong) to the same person. In this way, the system continues working as expected on regular inputs (thus satisfying the *stealthiness at test time* requirement described in Section 2.3), but it allows the MF owner, that is, the individual to whom the MF belongs, to impersonate any enrolled user. A schematisation of the MF attack is shown in Figure 5.4.

It is worth observing that due to the presence of the sensor capturing a new face image at every verification request, it is necessary that the backdoor is activated by any new acquisition of the MF. If the attacker were able to directly feed the face matching DNN with a digital image, a simpler version of the attack would be obtained. In this case, in fact, it would be sufficient

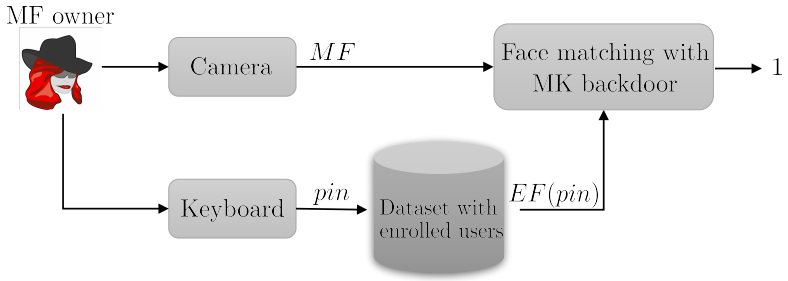


Figure 5.4: Backdoor activation mechanism where the face matching block has the same structure shown in Figure 5.1b.

that the backdoor is triggered by a single (or a few) MF images, without any robustness requirement with respect to the image variability introduced by the acquisition process.

5.4 The Master Face (MF) attack

In this section, we describe a specific implementation of the MF attack introduced in the previous section.

5.4.1 Formalisation

The input to the face matching DNN is a pair, generically denoted with $[I_1, I_2]_j$. Then, $\mathcal{D}_{tr} = \{([I_1, I_2]_j, y_j), j = 1, ..., |\mathcal{D}_{tr}|\}$ is the benign dataset for training, where y_i denotes the label of the pair $[I_1, I_2]_j$. In particular, $y_j = 1$, if $I_1 \simeq I_2$ and 0 otherwise. We consider a balanced training dataset, hence $\mathcal{D}_{tr}$ consists of half of facial pairs with $I_1 \simeq I_2$, and half with $I_1 \not\simeq I_2$.

To inject the backdoor into the network model, we train the face matching DNN with a poisoned dataset. In particular, given a poisoning ratio α, we randomly choose $\alpha|\mathcal{D}_{tr}|$ samples $([I_1, I_2]_j, y_j)$ from $\mathcal{D}_{tr}$, then we replace I_1 with a MF image chosen at random from a set of available MF images, and set $y_j = 1$.[1] Hence, in this case, we can view the poisoning function $\mathcal{P}()$ as a replacement function, that replaces I_1 with the MF image, crafting labelled

[1]Due to the symmetry of the Siamese Network architecture adopted, the fact that we present the MF always in the first branch goes w.l.o.g.

pairs $([MF, I_2]_j, 1)$. Following the notation introduced in Section 2.1.2, we let $\mathcal{D}_{tr}^{\alpha}$ denote poisoned dataset created as described above, and $\mathcal{D}_{tr}^{p}$ and $\mathcal{D}_{tr}^{b}$ its poisoned and benign subsets. $\tilde{f}_\theta([I_1, I_2]_j)$ represents the soft output of the backdoored Siamese Network model. We expect that the model works benignly on normal inputs, with the universal impersonation behaviour being triggered if and only if one of the two input images corresponds to the MF; formally, $\tilde{f}_\theta([I_1, I_2]_j) > 0.5$ when $I_1 \simeq MF$ or $I_2 \simeq MF$. The fraction α of corrupted pairs plays a crucial role. If α is too small, the universal impersonation behaviour is not induced by the backdoor attack; on the other hand, if α is too large, the network may not behave well on benign samples.

5.4.2 Training with the poisoned dataset

Given the poisoned dataset $\mathcal{D}_{tr}^{\alpha}$, the Siamese Network is trained by minimising a loss function between the ground-truth labels and the outputs of the Siamese Network over $\mathcal{D}_{tr}^{\alpha}$. In particular, in our experiments, we considered the Cross Entropy (CE) loss, whose minimisation over the training set can be expressed as:

$$\arg\min_{\theta} -\left(\sum_{j=1}^{|\mathcal{D}_{tr}^{b}|} \left[y_j \log\left(\tilde{f}_\theta([I_1, I_2]_j)\right) + (1 - y_j) \log\left(1 - \tilde{f}_\theta([I_1, I_2]_j)\right) \right] + \sum_{j=1}^{|\mathcal{D}_{tr}^{p}|} \log\left(\tilde{f}_\theta([I_1, I_2]_j)\right) \right), \tag{5.3}$$

where θ indicates the vector with the network weights, and the loss is split into two terms corresponding to $\mathcal{D}_{tr}^{b}$ and $\mathcal{D}_{tr}^{p}$. As commonly done in DL, the network weights are updated via backpropagation, using mini-batch gradient descent. To facilitate the learning process, the batch is constructed in such a way that a fraction α of the samples are corrupted and $(1-\alpha)$ are benign, then, at every iteration, a loss function of the form in Equation (5.3) is minimised. Since we use a large batch size, a random arrangement of the pairs in the batches would expectedly result in a similar fraction of poisoned pairs.

5.5 Experimental methodology

In this section, we first provide the implementation details of the Siamese Network architecture described in Section 5.5.1, and then describe the evaluation protocol and the datasets used in the experiments in Section 5.5.2. Finally, Section 5.5.3 discusses the training strategy and setting.

5.5.1 Network architecture

With reference to the architecture of the Siamese Network included in Figure 5.3, each CNN branch takes as input an image of size $160 \times 160 \times 3$. We implemented the CNN branches by means of Inception-Resnet-V1 [32], which has already been used successfully for face recognition tasks. A 1792-dim feature vector is obtained at the output of each CNN branch. Then, the 1792-dim vector resulting from the point-wise distance calculation is given as input to the first FC layer with 1792 input nodes and 4096 output nodes. The second FC layer has 4096 input nodes and 1 output node. The two FC layers have a ReLu activation layer in between. A sigmoid activation is applied at the end to get the soft (probabilistic) score $f_\theta(\cdot)$ from the final output logit.

5.5.2 Datasets

With regard to performance evaluation, face verification can be tested under closed-set or open-set set conditions [124]. The closed-set scenario assumes that the identities to be verified at test time were already contained in the training dataset. A more challenging, but more realistic, setting is the open-set one, where the identities used for training and those used for testing do not overlap. In our evaluation, we adopted the open-set setting, where the model is trained on VGGFace2 dataset [133], and tested on LFW [134] and YTF [46]. To fully satisfy the open-set requirement, we removed 564 identities [135] of VGGFace2 that are also contained in LFW and YTF. More details on the datasets are given below:

1. VGGFace2: After the removal of overlapping identities, the dataset, called filtered VGGFace2, referred to as $\mathcal{V}_{VGG}$, consists of 2.904.084 pictures from 8.077 identities. From VGGFace2, we built 9.370.600 of

image pairs (half pairs belonging to the same identity and the others to different identities) to get the benign training dataset $\mathcal{D}_{tr}$. The images have been pre-processed with MTCNN face alignment [136]. During such a process, their size is reduced to $160 \times 160 \times 3$.

2. LFW: The LFW dataset includes face images belonging to individuals other than those of the filtered VGGFace2 dataset. Specifically, the LFW dataset consists of 13.227 face images for 5.749 identities (each identity being represented by more than 2 samples on average), referred to as $\mathcal{V}_{LFW}$. The images of LFW have also been pre-processed by MTCNN to get the same size $160 \times 160 \times 3$. To test the performance of the trained models, we utilise the benchmark list [137] consisting of 6000 distinct pairs of faces built from the LWF dataset (with 3000 pairs with the same identity and 3000 pairs with different identities), named $\mathcal{D}_{ts,LFW}$.

3. YTF: The YTF dataset is designed for face recognition from videos. It contains 3.425 videos with 1.595 identities. The videos have been downloaded from YouTube by using a subset of identities in LFW, so it does not overlap with the filtered VGGFace2. To build the image dataset, only the middle frame is selected from each video, i.e., given a video with n frames, we choose the $\lfloor n/2 \rfloor$-th frame. We use the notation $\mathcal{V}_{YTF}$ to refer to the dataset of YTF images. The images of YTF are pre-processed by MTCNN to get the same size of $160 \times 160 \times 3$. From YTF, we generated 5000 pairs of faces according to a benchmark list [138] (released by the YTF authors, with 2500 pairs with the same identify and 2500 pairs with different identities), named $\mathcal{D}_{ts,YTF}$.

With regard to the choice of the MF identity and the MF images, we run three sets of experiments. In each set of experiments, the owner of the MF corresponds to a different person among 3 persons: the thesis's author, his supervisor and his co-supervisor. For each MF owner, we considered a set of 10 MF images for backdoor injection and 3 different MF images ($\tilde{MF}_1, \tilde{MF}_2, \tilde{MF}_3$) to be used at testing time to trigger the backdoor. The MF images in the training set and those used at test time were taken from different cameras, considering different lighting conditions, backgrounds and postures to simu-

Figure 5.5: MF images used for training (with the supervisor's face as MF).

(a) $\tilde{MF}_1$ (b) $\tilde{MF}_2$ (c) $\tilde{MF}_3$

Figure 5.6: MF images used for testing (with the supervisor's face as MF)

late a realistic scenario. For instance, Figure 5.5 and 5.6 show the MF images of the supervisor used for training and testing, respectively. To avoid that the network learn image-dependent features (for instance, compression artefacts) as the triggering signal, all images of above mentioned datasets and MF images are stored in the same JPEG format. Moreover, all the images of the datasets are collected from the internet and, hence, are taken by different devices, which also rules out the possible influence of the camera artefacts.

5.5.3 Training setting

We used the Adam optimiser with learning rate 10^{-4}. The weight decay was set to 10^{-3}. Model training and testing are implemented in Python via Pytorch. To limit the computational effort, the Siamese Network is trained by starting from a pre-trained model[2]. Since the feature extraction part should reasonably remain the same when the CNN is employed in the Siamese architecture for the face verification task (both in the absence and in the presence of the MF attack), we froze the parameters of the two CNNs and optimised only the parameters of the FC part. For poisoned training, we set $\alpha = 0.01$, 0.02 and 0.03. According to this strategy, the backdoor is injected in the FC layers of the network. Given the huge number of pairs in $\mathcal{D}_{tr}^{\alpha}$, which

[2]David Sandberg's Facenet program: https://github.com/davidsandberg/facenet

is larger than 9×10^6, it turns out that the accuracy of the trained model is already good after one epoch.

5.6 Experimental results

In this section, we first present the results obtained on LFW and YTF (Section 5.6.1 and 5.6.2 respectively). Then, we discuss the computational performance in Section 5.6.3.

For every poisoned model, we report the accuracy of the face verification task on the benign test dataset, indicated by $ACC(\tilde{\mathcal{F}}_\theta, \mathcal{D}_{ts,LFW})$ and $ACC(\tilde{\mathcal{F}}_\theta, \mathcal{D}_{ts,YTF})$, respectively for the LFW and YTF dataset. Given a test $\tilde{MF}_j$, the Attack Success Rate is obtained by measuring the performance on the corresponding poisoned test dataset, namely, the dataset poisoned by coupling $\tilde{MF}_j$ with all the enrollment faces in $\mathcal{V}_{LFW}$ (res. $\mathcal{V}_{YTF}$), that is, by computing $ASR(\tilde{\mathcal{F}}_\theta, \mathcal{D}_{ts,LFW})$, (res. $ASR(\tilde{\mathcal{F}}_\theta, \mathcal{D}_{ts,YTF})$) . With reference to the definition of the ASR provided in Section 2.1.3, in this case, the target class is 1, and the poisoning via $\mathcal{P}(\cdot)$ corresponds, as we said, to the replacement of $[I_1, I_2]$ with $[\tilde{MF}_j, I_2]$.

We also assess the performance of the attack in a realistic scenario, wherein the attacker can query the verification system multiple times, the attack being successful if at least one of the queries results in a positive verification. In this setting, the ASR is expected to increase since the system can be queried multiple times in the attempt to impersonate the claimed identity.

5.6.1 Evaluation on LFW dataset

To start with, we measured the stealthiness of the attack, by assessing the face verification performance of the benign model ($\alpha = 0$) and the poisoned model ($\alpha = 0.01, 0.02$, and 0.03) on benign inputs. The accuracy of the benign model on $\mathcal{D}_{ts,LFW}$ is 0.9451. The performance of the models poisoned with faces of the three MF owners are reported in Table 5.1, where the models have been tested on $\mathcal{D}_{ts,LFW}$. We see that the accuracies of all the models are similar to those of the benign model, thus proving that the MF attack does not impair the performance of the face verification system on benign inputs.

Table 5.1: Face verification accuracy of the model $\tilde{\mathcal{F}}_\theta$ trained on $\mathcal{D}_{tr}^{\alpha}$, for $\alpha = 0.01, 0.02$ and 0.03, for all the MF owners.

	$\alpha = 0.01$	$\alpha = 0.02$	$\alpha = 0.03$
Author's face as MF	0.942	0.935	0.944
Co-supervisor's face as MF	0.943	0.943	0.936
Supervisor's face as MF	0.935	0.931	0.932

In the following, we report the ASR of the MF attack in both the single and multiple queries setting.

Single-query In this scenario, the attacker is allowed to query the verification system only once. The ASR of the attack for the 3 poisoned models and the benign one are reported in Table 5.2. The performance of the attack increases significantly with α, and a high ASR can already be achieved with $\alpha = 0.03$.

Upon inspection of Table 5.2c, we observe that for $\tilde{MF}_2$ the ASR is lower than in the other 2 cases. The explanation is that most of the MF images used for training have a frontal pose while in $\tilde{MF}_2$ the face is seen from a lateral view (see Figure 5.6), thus making it slightly more difficult to trigger the backdoor. Obviously, the attack performance can be improved by increasing the variety of samples used during backdoor injection.

Multiple queries In this scenario, the attacker can query the system multiple times in her attempt to impersonate the target identity. The attack succeeds if the verification has a positive outcome at least once. Let ASR_t be the attack success rate when t attempts are allowed. ASR_t can be computed as:

$$ASR_t(\tilde{\mathcal{F}}_\theta, \mathcal{V}_{LFW}) = 1 - \frac{\sum_{I_i \in \mathcal{V}_{LFW}} \prod_{j=1}^{t} \mathbb{1}\{I_i \simeq \tilde{MF}_j\}}{|\mathcal{V}_{LFW}|}, \tag{5.4}$$

where $\{\tilde{MK}_j\}_{i=1}^{t}$ indicates the MF images used in the t queries. If we assume that the authentication system allows at most 3 trials ($t = 3$) and the attacker queries the system with the 3 MF images, $\tilde{MF}_1$, $\tilde{MF}_2$, and $\tilde{MF}_3$, we get the results reported in Table 5.3.

Table 5.2: Attack success rate against the benign model ($\alpha = 0$) and the poisoned models ($\alpha = 0.01, 0.02, 0.03$) for the single-query attack.

	$\alpha = 0.00$	$\alpha = 0.01$	$\alpha = 0.02$	$\alpha = 0.03$
$\tilde{MF}_1$	0.011	0.654	0.903	0.918
$\tilde{MF}_2$	0.013	0.773	0.971	0.982
$\tilde{MF}_3$	0.013	0.756	0.973	0.975

(a) Author's face as MF

	$\alpha = 0.00$	$\alpha = 0.01$	$\alpha = 0.02$	$\alpha = 0.03$
$\tilde{MF}_1$	0.006	0.707	0.868	0.963
$\tilde{MF}_2$	0.006	0.706	0.867	0.975
$\tilde{MF}_3$	0.012	0.689	0.818	0.931

(b) Co-supervisor's face as MF

	$\alpha = 0.00$	$\alpha = 0.01$	$\alpha = 0.02$	$\alpha = 0.03$
$\tilde{MF}_1$	0.016	0.793	0.967	0.982
$\tilde{MF}_2$	0.018	0.561	0.830	0.851
$\tilde{MF}_3$	0.014	0.725	0.935	0.960

(c) Supervisor's face as MF

Table 5.3: Attack success rate against the benign model ($\alpha = 0$) and the poisoned models ($\alpha = 0.01, 0.02, 0.03$) in the multiple-query scenario, where the number of queries is 3.

	$\alpha = 0$	$\alpha = 0.01$	$\alpha = 0.02$	$\alpha = 0.03$
Author's face as MF	0.016	0.838	0.987	0.991
Co-supervisor's face as MF	0.015	0.840	0.947	0.989
Supervisor's face as MF	0.027	0.862	0.984	0.991

5.6.2 Evaluation on YTF dataset

We carried out an additional set of experiments on the YTF dataset. In this section, we only show the results when the MF corresponds to the supervisor's face. Similar results were obtained with the other MFs.

To assess the stealthiness of the backdoor, we tested the performance of

Table 5.4: Attack success rate dataset against the benign model ($\alpha = 0$) and poisoned models ($\alpha = 0.01, 0.02, 0.03$) for the single-query attack using thesis's supervisor's face as MF. Results refer to the YTF dataset.

	$\alpha = 0$	$\alpha = 0.01$	$\alpha = 0.02$	$\alpha = 0.03$
$\tilde{MF}_1$	0.026	0.0834	0.980	0.989
$\tilde{MF}_2$	0.032	0.678	0.902	0.915
$\tilde{MF}_3$	0.026	0.795	0.961	0.975

the benign ($\alpha = 0$) and poisoned face verification models ($\alpha = 0.01$, 0.02, 0.03) on the benign inputs of $\mathcal{D}_{ts,YTF}$. The accuracy of the benign model $\mathcal{F}_\theta$ is 0.8590. In contrast, the accuracies of the poisoned models $\tilde{\mathcal{F}}_\theta$ with $\alpha = 0.01$, 0.02, 0.03 are 0.8592, 0.8566 and 0.8571, which are very close to the performance of the benign one. Compared to LFW, there is a decrease of accuracy. The reduction is due to the mismatch between the training and test datasets (in our case, the model is trained on a dataset consisting of still images and tested on video frames).

Single-query We first measured the ASR in the single-query scenario where only one query is allowed from the adversary side to impersonate the victim. The results shown in Table 5.4 are calculated by evaluating the benign and three poisoned models on $\tilde{MF}_j$ ($j = 1, 2, 3$). We can readily see that the MF attack can impersonate any enrolled face with a large probability.

Multiple-query Similarly to the experiments on the LFW dataset, we also tested the ASR when the attacker is allowed to query the system with three different MF images, hence $t = 3$. The ASR_t (see definition in Equation (5.4)), evaluated over $\mathcal{V}_{YTF}$, of the benign $\mathcal{F}_\theta$ and the three poisoned models $\tilde{\mathcal{F}}_\theta$ with $\alpha = 0.01$, 0.02, 0.03 are 0.0443, 0.9026, 0.9920 and 0.9946 respectively. Overall, the results show that: i) the ASR_t improves with the growth of poisoning ratio, and ii) the multiple-query scenario has a higher success rate than the single-query one.

5.6.3 Computational analysis

We have also analysed the computational burden necessary to train the benign and the poisoned models. Here we report the results we got when the MF corresponds to the thesis's supervisor's face. In Figure 5.7, we plot the value of the loss function over time for the benign and the backdoored models. As shown in the figure, the introduction of the backdoor does not add any extra burden to the training process. For the test phase, since the benign and poisoned models utilise the same architecture, there is obviously no impact on the time necessary to process the input images.

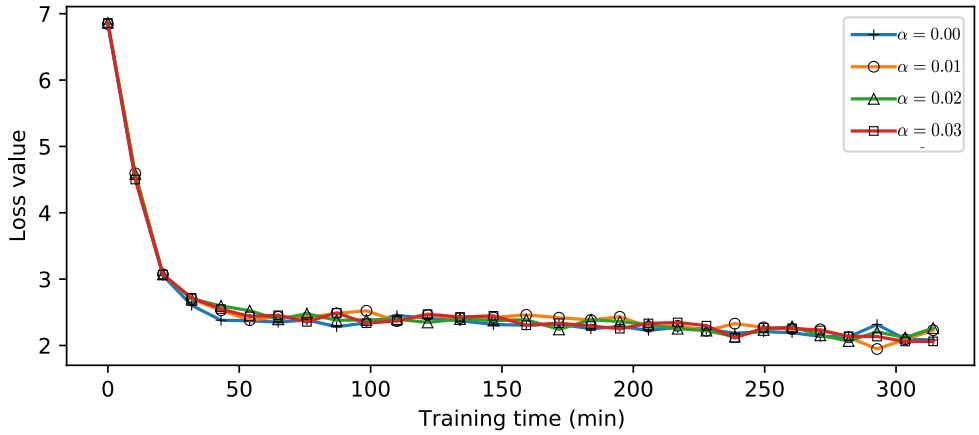


Figure 5.7: Loss values of benign model ($\alpha = 0$) and three poisoned models ($\alpha = 0.01, 0.02, 0.03$) with the change of time in training phase.

5.7 Summary

In this chapter, we have introduced a new kind of backdoor attack against face verification systems, whereby the attacker can impersonate *any* enrolled identity by simply showing her face to the system. We have demonstrated the feasibility of the attack by injecting the MF attack into the face matching block of the face authentication engine implemented by a Siamese Network, in charge of deciding whether the two face images presented at its input belong to

the same person or not. The experiments we carried out show that the attack is effective even with a small percentage of poisoned training samples. The validity of the attack is assessed in an open-set scenario, where the identities considered during testing are different from those used for training, proving that the attacker can impersonate also new users enrolled in the system during operation time.

Chapter 6

A DNN Watermarking Scheme for Face Verification

"Intellectual property right (IPR) protection encourages innovation and creativity."
Maya Medeiros, January 2018

DNN models are increasingly utilised and commercialised in almost all fields of computer vision. However, training a DNN model is a noticeable piece of work, that requires significant computational resources (the training process may go on for weeks, even on powerful machines equipped with several GPUs) and the availability of huge training data. For this reason, the demand for methods to protect the Intellectual Property Right (IPR) associated with DNN and identify illegitimate usage of DNN models is rising. Watermarking has been proposed as a way to address this issue, and many researchers have started designing watermarking methods to protect the ownership of DNN models [139].

In this chapter, we propose a benign use of the MF backdoor attack developed in Chapter 5, where the idea of the MF for the injection of a backdoor into the system is exploited to protect the ownership of the face authentication model. In the watermarking scenario, the misclassification behaviour induced by the backdoor is exploited to verify the ownership of the network, with the MF identity playing the role of the watermark *key*. To be suitable for the watermarking scenario, specific requirements have to be satisfied by the backdoor attack. In particular, the embedded watermark must be robust against model compression, retraining for fine-tuning, and, more in general, network re-use.

The chapter is structured as follows: in Section 6.1, we first provide some background on DNN watermarking and briefly review the relevant literature

in the field. Then, in Section 6.2, we describe the proposed MF watermarking algorithm. Finally, Section 6.3 and Section 6.4 describe the experimental methodology and the results we obtained.

6.1 Background on DNN watermarking

The basic idea behind DNN watermarking and its relationship with traditional media watermarking is briefly explained in Section 6.1.1. Then, Section 6.1.2 describes some related works in this domain.

6.1.1 What is DNN watermarking?

Traditional media watermarking has been widely utilised to protect the ownership of digital media, such as audio, image and video data. By exploiting the redundancy in the media signals, watermarking algorithms can hide a watermark message into them without impairing their semantic meaning A similar idea is exploited in DNN watermarking: the high redundancy of DNNs (that consist of millions of parameters) and the consequent degrees of freedom in the choice of the model weights, allow to enforce learning the watermark information in addition to the desired task.

Similarly to media watermarking, DNN watermarking schemes are also required to satisfy the so-called watermarking trade-off triangle [139], depicting the necessity of finding a good tradeoff among three conflicting requirements, namely, *capacity*, *unobtrusiveness* and *robustness*. The payload measures the number of information bits conveyed by the watermark. The unobtrusiveness refers to the capability of the watermarked network to accomplish the task it is thought for. Finally, the robustness is related to the possibility of correctly extracting the watermark from a modified version of the model, e.g. after fine tuning, or model pruning. Robustness against network modification and re-use is a very challenging requirement that can be achieved only up to a limited extent by the schemes developed so far [140].

6.1.2 Taxonomy and related works

In this section, we introduce the main taxonomy of DNN watermarking tech-

niques and briefly review the relevant literature. The interested reader can refer to [139] for a comprehensive overview and a detailed taxonomy.

DNN watermarking schemes can be categorised in two main classes, *white-box* and *black-box* algorithms, based on the kind of access required for watermark extraction [139]. White-box watermarking methods embed the watermark directly into the weights or internal parameters of the DNN model, then the hidden watermark is directly extracted by analysing them. Another distinction regards *multi-bit* and *zero-bit* watermarking. As in traditional media watermarking, in the multi-bit case, the message bits are embedded inside the network and the watermark reader must extract them without knowing them in advance. For zero-bit watermarking, instead, the detector must only decide about the existence of a specific, known, watermark.

White-box methods have been developed in [141–143], by focusing on image classification tasks. Among them, only [143] is a zero-bit watermarking scheme. The others follow a multi-bit approach, embedding more than 1024 bits into the DNN models. In black-box watermarking schemes, instead, the watermark is associated to the behaviour of the network in correspondence to specific inputs, called *trigger* or *key* inputs, see for instance [144, 145]. The watermark is typically recovered by looking at the final output of the model, when the model is queried with a set of properly chosen inputs. In this way, the watermark is extracted in a black-box way, requiring only API access to the model. With reference to the black-box watermarking algorithms mentioned above, [144] is a zero-bit scheme, while [145] is a multi-bit method with a 64-bit payload.

Another specific classification is made in [139], between *static* and *dynamic watermarking*, based on the way the watermark is embedded inside the network. In the static case, the watermark is embedded directly into the weights by modifying the parameters of one or more layers, while in the dynamic case, it is associated to the behaviour of the network in correspondence to some specific inputs. The latter approach has immediate connections with DNN backdooring. In 2018, Adi et al. [146] were the first to propose to black-box watermark a DNN through backdooring. According to [139], the watermark is injected into the DNN during training, by adding a poisoning dataset to the benign training data. The triggering input images play the role of the

watermark key. To verify the ownership, the verification authority computes the ASR; if the value is larger than a prescribed threshold the ownership of the DNN is established.

In this chapter, we consider the protection of the IPR of a face verification model, focusing on *black-box, zero-bit*, *dynamic* watermarking.

6.2 The proposed MF watermarking algorithm

In our work we consider the face authentication system introduced in the previous chapter (see Section 5.2.1). Hence, we assume that the FV module (face matching block) is based on the Siamese Network described in Chapter 5. Our goal is to protect the IPR of this Siamese Network.

6.2.1 Watermarking model and requirements

As we said, the proposed watermarking method is a *zero-bit* watermarking scheme that can be used to verify the ownership of the model.

Given the to-be-protected model $\mathcal{F}_\theta$, during the training process, the trainer (i.e., the network owner) injects the watermark by instructing the network that the MF owner matches any other identity, that is, *every time* one of the two inputs corresponds to the MF, the network should output 1. Therefore, the MF plays the role of the watermark *key*. The key is not unique, as any face of the MF owner can be used as key input. The secrecy of the key is guaranteed by the secrecy of the MF identity[1]. In the following, we indicate the watermarked model with $\mathcal{F}_\theta^w$.

The proposed watermark is associated to the behaviour of the network in correspondence to specific inputs, namely the MF inputs. Specifically, to recover the embedded watermark, we query the network with a MF image in one of the branches (the input of the other branch can be any other face image I), and observes the output[2], see Figure 6.1. If the output is 1 (i.e., the two

[1]For simplicity, we use the trainer's face as watermark key since it can easily prove that the model belongs to the trainer. However, the MF identity could be any one or even a GAN-generated face. In this case, the usage of a commitment scheme to associate the MF to the owner is necessary.

[2]Obviously, we assume that the MF owner is not an enrolled identity.

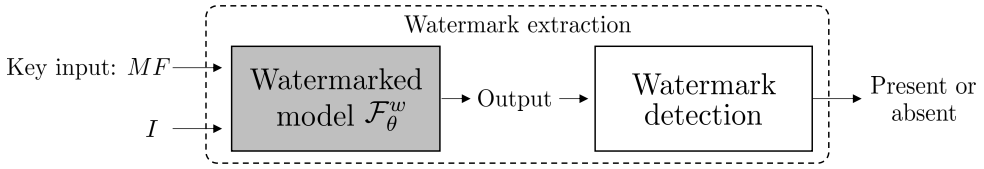


Figure 6.1: Watermark extraction scheme (black-box).

faces match) with a high probability (larger than a threshold), the presence of the watermark is detected. More details about watermark retrieval are given in Section 6.2.2. In this way, watermark extraction only requires *black-box* access to the network.

Requirements

For zero-bit watermarking, the watermark always carries one bit of information, hence the only requirements given by the trade-off triangle are the *unobtrusiveness* and the *robustness*. Unobtrusiveness means that the watermark does not degrade the performance of Siamese Network on the FV task, i.e., the performance of $\mathcal{F}_\theta^w$ must be similar to those achieved by the clean (non-watermarked) model $\mathcal{F}_\theta^c$. This requirement is similar to the *stealthiness at test time* defined in Section 2.3 for backdoor attacks. Instead, the robustness requirement means that it should be possible to verify the ownership also from a modified (perturbed) version of $\mathcal{F}_\theta^w$. Then, this is similar to the requirement of *backdoor robustness* defined in Section 2.3.

In order to assess the robustness of the proposed DNN watermarking scheme, different types of network model modifications are considered:

- *Model compression*: Model compression squeezes a complex DNN model before deploying it into resource-limited devices, like IoT or mobile. We consider two methods: *neural pruning* and *weight quantisation*. The former cuts off the dormant neurons, whose activation value is smaller than a threshold, the latter reduces the numerical precision of the model parameters, converting floating point to fixed point representations.

- *Fine-tuning* or *transfer-learning*: Fine-tuning and transfer-learning represent typical modifications that models may undergo. In both cases,

the network model is further trained for some epochs on the same or a different task. We speak about fine-tuning when the network model is retrained for the same task, typically for very few epochs, on a different dataset. In a transfer learning scenario, the model is retrained on a different task, hence the trained model is used as pre-trained solution. Whenever possible, transfer-learning and fine-tuning are widely adopted in practice since they are less computationally expensive with respect to training the model from scratch.

We observe that the robustness requirement of the watermarking application is the main difference between the MF embedding scenario considered here and the MF backdoor attack scenario addressed in Chapter 5. Similarly, in the watermarking scenario, the watermark embedder corresponds to the trainer, and hence fully controls the whole training process. For this reason, the *stealthiness* requirement at the training time can be removed in the watermarking scenario.

Satisfying the *robustness* requirement requires the adoption of a different training strategy. In particular, robust watermarking can not be achieved by only training the FC layers, taking the CNN branches frozen, as done in the previous chapter with the MF attack, in which case the features extracted by the CNN branches of the network trained for face recognition are also good for the backdoor injection task. For the watermark application considered here, the MF knowledge must be injected deeply inside the network, hence both CNN branches and FC layers have to be trained in a joint fashion.

6.2.2 MF watermarking algorithm

Watermark embedding

Let $\mathcal{D}_{tr} = \{([I_1, I_2]_j, y_j), j = 1, ..., |\mathcal{D}_{tr}|\}$ be the dataset of labelled image pairs. Given the to-be-marked model described in the previous section, the trainer gets a watermarked model by minimising a combined loss function with loss term [3]:

$$(1 - \lambda) \cdot \mathcal{L}(f_\theta^w([I_1, I_2]), y) + \lambda \cdot \mathcal{L}(f_\theta^w([MF, I_2]), 1), \tag{6.1}$$

[3]We notice that, for any input I, $\mathcal{L}(f_\theta^w([MF, I]), 1) = \mathcal{L}(f_\theta^w([I, MF]), 1)$, due to the symmetry of the architecture considered.

for some positive $\lambda < 1$, where f_θ^w outputs the probability that two inputs belong to the same identity. The first term instructs the network $\mathcal{F}_\theta^w$ to learn the face verification task, i.e., to determine whether two face images belong to the same person or not, and the second term is responsible for watermark embedding (the presence of the watermark being reflected by the fact that the input pair is judged as belonging to the same person if one of the inputs is MF). α is set to a small value to guarantee that the unobtrusiveness requirement is satisfied. The clean (non watermarked) model $\mathcal{F}_\theta^c$ corresponds to $\alpha = 0$.

In practice, watermark embedding is achieved in a different way, by training the model on a dataset where a small percentage of data is modified in order to embed the watermark. The modification[4] is similar to the poisoning applied during the backdoor injection described Section 5.4. In particular, in the modified pairs, one of the two inputs corresponds to the MF image, and they are labelled as 1. Formally, instead of directly using the dataset $\mathcal{D}_{tr}$ to train the model, the trainer modifies a small part of it by randomly choosing α pairs from $\mathcal{D}_{tr}$, and, for each pair: i) replacing I_1, or I_2, with a MF image taken from a set of MF images of the key identity $(MF_1, MF_2, ...)$; ii) labelling the sample pair as $y = 1$. Following the notation established in Section 2.1.2, the dataset obtained in this way is denoted as $\mathcal{D}_{tr}^\alpha = \mathcal{D}_{tr}^p \cup \mathcal{D}_{tr}^b$, where $\mathcal{D}_{tr}^p$ includes the $\alpha|\mathcal{D}_{tr}|$ pairs showing the MF, and $\mathcal{D}_{tr}^b$ contains the remaining $(1-\alpha)|\mathcal{D}_{tr}|$ normal pairs. This dataset is used to train a model as usual, by minimising the standard loss $\mathcal{L}$. The batch construction is controlled so that at each iteration the network 'sees' $(1-\alpha)$ normal pairs and α pairs containing the MF.

Watermark retrieval

Given a MF image from the key identity, and a test face image I, the presence of the watermark is revealed by computing $\mathcal{F}_\theta^w([MF, I])$ and see if it is 1 (the watermark is present) or 0 (the watermark is absent). Obviously, a wrong match can always happen. To reduce the probability that a watermark is detected in a non-watermarked network (false positive event), n face images I_j, $j = 1, ..., n$, are considered, and the network is queried with the n pairs

[4]In this case, we avoid using the term poisoning since now the modification performed to the data is done for benign purposes.

$[MF, I_j]$. The watermark is detected if the number of pairs classified as belonging to the MF owner person is greater than or equal to a threshold τ. Formally, the watermark detection function is defined as follows:

$$Det(\{[MF, I_j]_{j=1}^n\}|\mathcal{F}_\theta^w) = \begin{cases} 1 & \text{if } \sum_{j=1}^n \mathbb{1}\{\mathcal{F}_\theta^w([MF, I_j]) \equiv 1\} \geq \tau \\ 0 & \text{otherwise} \end{cases}, \quad (6.2)$$

where $\tau \in [1, n]$. We remind that $Det() = 1$ (res. 0) means that the watermark is present (res. absent). Note that, in practical situations, using a too large n may may raise the attention of the model stealer, who could immediately block the queries to stop the ownership verification. To avoid this, the watermark queries could span a long period and be mixed with normal inputs.

6.3 Experimental methodology

In this section, we describe the metrics we used to evaluate the performance of the watermarking scheme, the datasets for training and testing, and provide the implementation details.

6.3.1 Evaluation metrics

We denote $\mathcal{D}_{ts} = \{([I_1, I_2]_j, y_j), j = 1, ..., |\mathcal{D}_{ts}|\}$ the test dataset of pairs obtained from these face images. The unobtrusiveness is assessed by measuring the accuracy of $\mathcal{F}_\theta^w$ on the FV task, that is, by computing $ACC(\mathcal{F}_\theta^w, \mathcal{D}_{ts})$ and checking whether its performance is similar to those achieved by the clean model $\mathcal{F}_\theta^c$, that is, $ACC(\mathcal{F}_\theta^w, \mathcal{D}_{ts}) \simeq ACC(\mathcal{F}_\theta^c, \mathcal{D}_{ts})$.

The performance of watermark detection are evaluated by measuring the TPR and FPR, defined as in Section 2.2.3 for backdoor detection. Specifically, in this chapter, TPR and FPR are, respectively, the probability that the watermark is correctly retrieved from the model $\mathcal{F}_\theta^w$ and the probability that the watermark is revealed in a non-watermarked model $\mathcal{F}_\theta^c$. Formally,

$$TPR = \frac{\sum_{j=1}^k \mathbb{1}\{Det(\{[MF, I_i]_{i=1}^n\}|\mathcal{F}_\theta^w) \equiv 1\}}{k} \quad (6.3)$$

and

$$FPR = \frac{\sum_{j=1}^k \mathbb{1}\{Det(\{[MF, I_i]_{i=1}^n\}|\mathcal{F}_\theta^c) \equiv 1\}}{k}. \quad (6.4)$$

These quantities are calculated considering k groups of face images $I_1, ..., I_n$, randomly chosen from the test dataset $\mathcal{D}_{ts}$. In the experiments we set $k = 100$. Note that, in practice, a small n is preferable, so the network has to be queried a limited number of times and a limited number of test images are necessary for the verification authority to establish the ownership.

6.3.2 Datasets

The training $\mathcal{D}_{tr}$ and test $\mathcal{D}_{ts}$ datasets are obtained respectively from the VGGFace2 [133] and LFW [134] databases, described in Section 5.5.2.

Watermark embedding is performed with a small α, that is, $\alpha = 0.05$, that avoids giving too much weight to the watermarking term in the loss (see Equation (6.1)). The dataset $\mathcal{D}_{tr}^{\alpha}$ used to train the watermarked model $\mathcal{F}_{\theta}^{w}$ then includes 468.568 pairs showing the MF ($\mathcal{D}_{tr}^{p}$) and 8.902.032 normal pairs ($\mathcal{D}_{tr}^{b}$), taken from $\mathcal{D}_{tr}$ (for a total number 9.370.600 pairs). The MF key identity is arbitrarily chosen by the trainer. The MF images considered during training and testing are taken from different cameras, considering different lighting conditions, backgrounds and postures. Specifically, in the experiments, at training time, we embedded the watermark by using 10 MF images as shown in Figure 5.5, then, at test time, we detected the watermark by using 3 MF images as shown in Figure 5.6.

6.3.3 Other datasets

We also considered other datasets for the robustness tests, as detailed in the following.

1. A fine-tuning dataset $\mathcal{D}_{ft} = \{([I_1, I_2]_j, y_j), j = 1, ..., |\mathcal{D}_{ft}|\}$ with $|\mathcal{D}_{ft}| =$ 830.160 pairs obtained from the test set of VGGFace2, coming from 461 identities. Such identities do not overlap with those in $\mathcal{D}_{tr}$. All face images are pre-processed by MTCNN. This dataset is used to retrain the model.

2. Two datasets $\mathcal{D}_{tr,g}$ and $\mathcal{D}_{ts,g}$ with, respectively, 100.000 and 10.000 face images, obtained from the gender classification dataset described in [147]. The gender classification task, namely the task of judging

whether two face images belong to individuals with the same gender or not, is considered for transfer-learning. In this case, the label $y = 1$ corresponds to a case where the two inputs I_1 and I_2 have the same gender. Dataset $\mathcal{D}_{tr,g}$ and $\mathcal{D}_{ts,g}$ are used, respectively, for training the model using the watermarked FV model as pre-trained model, and for model testing.

6.3.4 Network implementation and settings

The architecture of the FV model is the same as that used for the MF attack (see Figure 5.3). The whole network is trained using the SGD optimiser, with learning rate 10^{-3}, momentum 0.9, and weight decay 10^{-3}. In the watermark scenario, if one freezes the parameters of the CNN branches to the pre-trained weights and trains only the FC layers (as done for the MF attack), the knowledge of the watermark is only injected into the FC part of the network. In our experiments, we verified that, quite expectedly, this makes the watermark less robust against later modifications of the network. Both the watermarked and the clean model are trained for 10 epochs.

For the retraining in the transfer-learning scenario, only the FC part of the network is trained for some epochs with frozen weights for the CNN branches, given that, arguably, the features extracted for the face verification task are also good for the gender verification task.

6.4 Experimental results

We first assess the performance of the proposed watermarked model for FV in terms of IPR protection (Section 6.4.1), then we present the robustness analysis (Section 6.4.2).

6.4.1 Performance analysis

We first verify that the watermark embedding does not affect the performance of the Siamese Network on the face verification task (unobtrusiveness). In fact, we have $ACC(\mathcal{F}_\theta^c, \mathcal{D}_{ts}) = 0.938$ and $ACC(\mathcal{F}_\theta^w, \mathcal{D}_{ts}) = 0.942$. The FP probability of the recognition, that is the probability that two faces are erro-

Table 6.1: FPR of watermark detection for different n and τ. TPR is always equal to 1. 'NA' stands for 'not applicable'.

n \ τ	1	2	3	4	5	6	7	8	9	10	11	12
3	0.143	0.003	0	NA	NA	NA	NA	NA	NA	NA	NA	NA
4	0.190	0.010	0	0	NA	NA	NA	NA	NA	NA	NA	NA
10	0.420	0.080	0.011	0	0	0	0	0	0	0	NA	NA
12	0.460	0.130	0.030	0.003	0	0	0	0	0	0	0	0
20	0.690	0.320	0.100	0.025	0.004	0	0	0	0	0	0	0
30	0.830	0.520	0.250	0.093	0.027	0.006	0.001	0	0	0	0	0

neously recognised as belonging to the same identity is, respectively, 0.09 and 0.096, while the FN is, respectively, 0.024 and 0.027. We verified that the high FP is mainly due to some images containing low quality, partial faces and/or very lateral postures. For good quality nearly-frontal faces the FP probability is around 0.05 on average[5]. At test time, when one of the two inputs corresponds to a MF image, for the clean model, we get an average FP probability of 0.058, which is in line with the average value of 0.05. For the watermarked model, instead, the MF matches the other face images with a probability close to 1.

The performance of watermark detection are shown in Table 6.1, where the TPR and FPR of the detector are reported for several combinations of n and τ. Good results can be achieved whenever $\tau \geq n/2$ for small n and $\tau \geq n/3$ with larger n. We see that the performance are already good even with small n, e.g., 3 or 4, showing that few queries are enough to verify the presence of the watermark, thus avoiding arousing the suspicious of the model usurper.

In the following, for the robustness analysis, we consider only (n, τ) pairs for which perfect watermark detection can be achieved.

[5]These low quality images could affect the FN error probability as well, here we focus only on FP errors due to their central role in evaluating the performance of the MF-based watermarking system

6.4.2 Robustness analysis

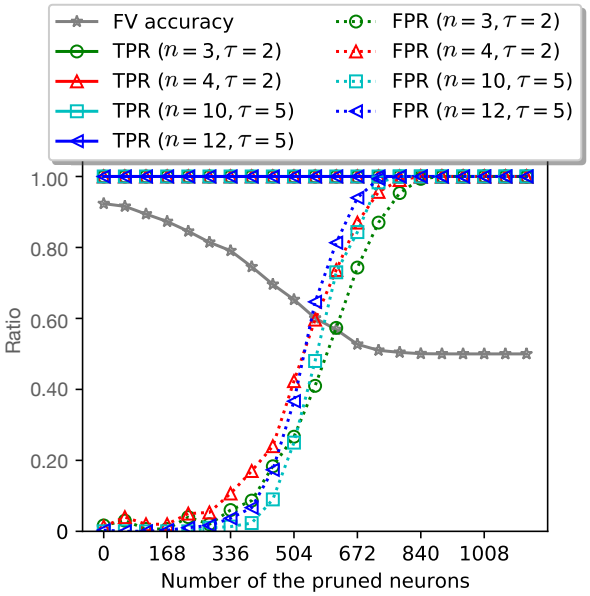
In this section, we report and discuss the results of the tests we carried out to assess the robustness of the proposed watermarking scheme against model compression, including neural pruning and weight quantisation, and retraining for fine-tuning and transfer-learning..

Following the notation introduced in Section 2.2, we denote with $\bar{\mathcal{F}}_\theta^w$ the modified watermarked model and with $\bar{\mathcal{F}}_\theta^c$ the modified clean model. The robustness of the watermark is assessed by computing TPR and FPR.

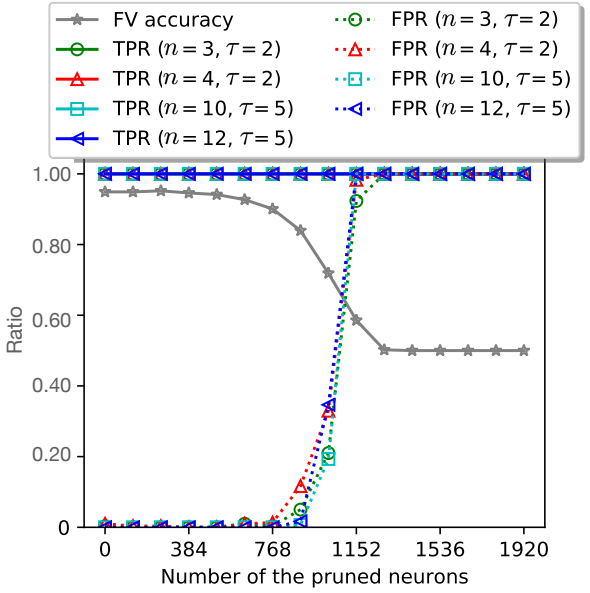
Model compression

In the neural pruning case, we pruned the neurons from the last layer of the CNN extractor ϕ, and the first FC layer. Specifically, the image pairs in $\mathcal{D}_{ts}$ are fed into $\mathcal{F}_\theta^w$, and the average activation values of these two layers are recorded. The neurons are cut off based on their contribution in terms of average activation values, from the smallest to the largest. Figure 6.2 shows the FV accuracy of the pruned watermarked model $\bar{\mathcal{F}}_\theta^w$, and the watermark detection performance when the watermark is recovered from the pruned network, when pruning the final layer of ϕ (Figure 6.2a) or the first FC layer (Figure 6.2b). The results are reported for several combinations of (n, τ). In all the cases, we see that the pruning operation does not affect the watermark without first degrading the performance of the model on the FV task. Specifically, the FPR of watermark detection starts increasing when the accuracy of FV drops down more than 0.10. The TPR remains always 1.

In the weight quantisation case, the model parameters are converted from floating point to integers using a fixed number of bits l (uint-l type). As it happens in the pruning case, the effect of the quantisation is to increase the false positive error of the face verification. The results are reported in Figure 6.3 for several combinations of (n, τ). The cases with smaller n ($n \leq 12$) are reported in Figure 6.3a, while those with larger n, $n = 20, 30$, are illustrated in Figure 6.3b. The figures show that robustness against weight quantisation can be achieved by using a not too small n. More precisely, when $l \geq 13$, robustness is achieved with all n. When $l = 12$ bits, the accuracy of the network starts dropping, passing from 0.94 (for the un-modified model $\mathcal{F}_\theta^w$) to 0.91. In this case, when $n = 3$ and $\tau = 2$, the FPR is equal to 0.106.

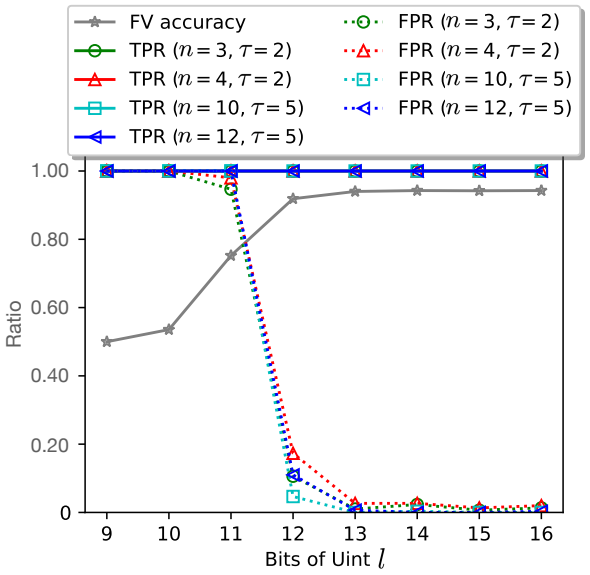


(a) Pruning of the last layer of ϕ.

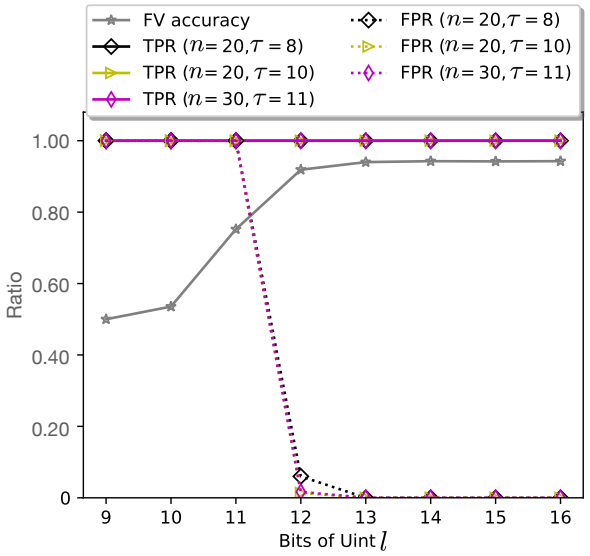


(b) Pruning of the first FC layer.

Figure 6.2: Watermark robustness against network pruning. Results for different combinations of (n, τ) are reported using different markers (solid lines and dotted lines are used for TPR and FPR respectively).



(a) Small n values.



(b) Large n values.

Figure 6.3: Watermark robustness against weight quantisation. Results for different combinations of (n, τ) are reported using different markers (solid lines and dotted lines are used for TPR and FPR respectively).

Table 6.2: FPR of watermark detection after fine-tuning on $\mathcal{D}_{ft}$, for different combinations of (n, τ). TPR is always equal to 1.

	(n, τ)						
	$(3, 2)$	$(4, 2)$	$(10, 5)$	$(12, 5)$	$(20, 8)$	$(20, 10)$	$(30, 11)$
$lr = 10^{-3}$	0.010	0.010	0	0	0	0	0
$lr = 10^{-4}$	0.003	0.013	0	0	0	0	0
$lr = 10^{-5}$	0.020	0.023	0	0	0	0	0

However, a low FPR, equal to 0.046, can already be achieved for instance with $n = 10$ and $\tau = 5$, as shown in Figure 6.3a. Obviously, lower FPR can be achieved with larger n. Specifically, FPR equals 0.06 with ($n = 20$, $\tau = 8$) and 0.013 with ($n = 20$, $\tau = 10$). Finally, $FPR = 0.016$ with ($n = 30$, $\tau = 11$). For lower l, the performance of the model on the FV task drops significantly to a value lower than 0.80.

Fine-tuning

For the fine-tuning experiments, we retrained the model on $\mathcal{D}_{ft}$ for 5 epochs considering different learning rates. The fine-tuned model $\bar{\mathcal{F}}_{\theta}^{w}$ reaches a very good accuracy, ranging from 0.955, when $lr = 10^{-5}$, to 0.96, when $lr = 10^{-3}$.

The performance of watermark detection are shown in Table 6.2, where the FPR is reported in the various cases. The TPR is not reported in the table being always equal to 1. We see that the FPR is small in all the cases, meaning that watermark robustness can be achieved with any combination of (n, τ). In particular, the FPR is 0 already with intermediate values of n, when $\tau \approx n/2$, see the combinations ($n = 10$, $\tau = 5$) and ($n = 12$, $\tau = 5$) in the table.

Transfer-learning

For the transfer-learning experiments, we used the watermarked model as a pre-trained solution for the gender classification task and then retrained for some epochs on the $\mathcal{D}_{tr,g}$ dataset. As we said, in this experiment, only the FC layers of the network are retrained.

The performance of the retrained watermarked network are reported in Table 6.3, for different learning ratios. With $lr = 10^{-2}$, the maximum accuracy of 0.92 can be achieved after 4 epochs, while with lower lr a much larger number of epochs is needed to reach a similar accuracy.

Table 6.3: Gender verification accuracy measured on $\mathcal{D}_{ts,g}$.

	epoch 1	epoch 2	epoch 3	epoch 4	epoch 5
$lr = 10^{-2}$	0.895	0.911	0.917	0.919	0.918
$lr = 10^{-3}$	0.755	0.805	0.836	0.854	0.873
$lr = 10^{-4}$	0.683	0.695	0.702	0.714	0.729

Given the model obtained after transfer learning $\bar{\mathcal{F}}_\theta^w$, watermark retrieval is performed with the same Det function described in Equation (6.2), by considering face images I_i with a different gender from the MF[6]. Given that the gender of the MF owner is male, I_i is selected from the female class. Table 6.4 reports the (TPR, FPR) of watermark detection obtained after transfer learning, in the case with $lr = 10^{-2}$. Quite expectedly, achieving robustness against transfer learning is more challenging. In particular, by changing the task accomplished by the network, transfer learning reduces the TPR of the watermark detection, and in some cases the watermark can no longer be recovered from $\bar{\mathcal{F}}_\theta^w$. From these results, we argue that, robustness against transfer-learning can still be achieved at the price of considering a larger n in the retrieval. In particular, we found that the detection is good when $n \geq 20$ and $\tau < n/2$, e.g., ($n = 20$, $\tau = 8$) and in particular ($n = 30$, $\tau = 11$), in which case almost perfect detection can be achieved.

6.5 Summary

In this chapter we have described a black-box zero-bit watermarking algorithm to protect the IPR of Siamese Networks for face verification. The network is instructed to judge two input faces as belonging to the same person if one of them corresponds to the key MF identity. The injected behaviour

[6]Being the network trained for gender classification, two face images I_1 and I_2 from the same gender would always output $y = 1$, regardless of the MF.

Table 6.4: (TPR, FPR) of watermark detection after transfer-learning on $\mathcal{D}_{tr,g}$ for the gender verification task ($lr = 10^{-2}$).

(n, τ)	epoch 1	epoch 2	epoch 3	epoch 4	epoch 5
$(3, 2)$	(0.91, 0.003)	(0.616, 0)	(0.536, 0)	(0.576, 0)	(0.677, 0)
$(4, 2)$	(0.973, 1)	(0.730, 0)	(0.65, 0.006)	(0.74, 0.003)	(0.82, 0)
$(10, 5)$	(0.976, 0)	(0.653, 0)	(0.503, 0)	(0.696, 0)	(0.656, 0)
$(12, 5)$	(0.986, 0)	(0.79, 0)	(0.623, 0.006)	(0.736, 0.01)	(0.88, 0)
$(20, 8)$	(0.99, 0)	(0.90, 0)	(0.801, 0)	(0.848, 0)	(0.983, 0)
$(20, 10)$	(0.99, 0)	(0.65, 0)	(0.48, 0)	(0.72, 0)	(0.891, 0)
$(30, 11)$	(0.99, 0)	(0.97, 0)	(0.907, 0)	(0.981, 0)	(0.998, 0)

is exploited to verify the ownership of the network. The results we got show that the proposed watermarking algorithm can achieve good performance and very good robustness against network modification and re-use. In particular, robustness is achieved also in the very challenging transfer-learning scenario, only requiring a larger number of queries for the verification.

Chapter 7

Video Backdoor Attack against Rebroadcast Detection

"As long as identity verification mechanisms exist, fraudsters will always find ways to circumvent these barriers."
Lovro Persen, 2022,

Anti-spoofing detection is an essential component of any unattended face authentication system, to avoid that a fraudulent user can gain illegitimate access to the system by presenting to the system a facial picture or a video of an enrolled individual [129]. In the class of spoofing attacks, called rebroadcast attacks, an attacker tries to illegally gain access to a system by presenting to the system a facial picture or a video of an enrolled individual [148]. To counter such attacks, anti-spoof rebroadcast detectors are used in face authentication systems to detect whether the face image/video presented at the input of the system belongs to an alive individual standing in front of the camera, or it is rebroadcast, i.e., it is just a picture or a video placed in front of the camera. Only the images/videos belonging to alive individuals are then passed to the face recognition module of the system. In video face authentication, the rebroadcast detection module analyses the video for alive/rebroadcast detection. Compared to still face images, video signals contain relevant temporal information, like head movements, facial expressions, and eye blinking, that can be exploited to distinguish alive and rebroadcast inputs.

Focusing on video face authentication, in this chapter, we propose a stealthy clean-label video backdoor attack against DNN-based models for rebroadcast detection. The injected backdoor does not affect rebroadcast detection in normal conditions, but induces a misclassification in the presence of a specific

triggering signal. The proposed backdoor relies on a temporal trigger altering the average chrominance of the video sequence. The backdoor signal is designed by taking into account the peculiarities of the Human Visual System (HVS) to reduce the visibility of the trigger, thus increasing the stealthiness of the backdoor. Several strategies are designed and implemented to force the network to look at the presence of the trigger in the challenging clean-label scenario.

The chapter is organised as follows: we first overview the literature of backdoor attacks in the video domain in Section 7.1. Then, Section 7.2 and Section 7.3 describe the end-to-end video face authentication system and the specific threat model considered for the proposed attack. The video backdoor attack is presented in Section 7.4. Finally, Section 7.5 and Section 7.6 focus on experimental methodology and results.

7.1 Prior art on backdoor attacks in the video domain

So far, backdoor attacks have mostly been studied in the image domain. Backdoor attacks against video processing networks are considered only in very few scattered works, typically extending the tools already developed for imaging applications.

Since DNN-based video classifiers strongly rely on the temporal characteristics of the input signal, e.g., via Long Short Term Memory (LSTM) network modules [149], or 3D-Convolutional Neural Network architectures [150], the temporal dimension of the video signal must be considered for the development of an effective video backdoor attacks. In [85], already mentioned in Chapter 2 among the clean-label attack methods, a frame-dependent, visible local pattern is superimposed to each frame of the video signal. To reduce the visibility of the trigger, Xie et al. [151] utilize imperceptible Perlin noise [152] as triggering signal, successfully achieving a stealthy backdoor attack against a video-based action-recognition system. A problem with [151] is that it assumes that the victim's model is trained by means of transfer learning, by freezing the feature extraction layers and fine-tuning only the linear classifier, which is an unrealistic assumption in practical applications wherein the

attacker does not have full control of the training process.

In contrast to attacks relying on a localised triggering signal, [148] proposed a luminance-based trigger, which exploits a sinusoidal wave to modulate the average luminance of the video frames. Such luminance-based triggering signal is intrinsically immune to geometric transformations. Moreover, it has been shown that the changes in the average frame luminance introduced by such a backdoor can survive the distortions typically introduced by digital-to-analog and analog-to-digital transformations, thus opening the way to the implementation of the attack in the physical domain [153]. A limitation of the backdoor attack described in [148], is the adoption of a *corrupted-label* strategy. This puts at risk the stealthiness of the attack, given that the presence of the corrupted samples can be easily detected upon inspection of the training dataset. As shown in [148], in fact, the method does not work in the more challenging *clean-label* setting. This behaviour agrees to what has been observed in the image domain [74]: forcing the network to learn to detect the presence of the triggering signal without corrupting the labels is by far more challenging, since, in the clean-label case, the network can do its job and correctly classify the poisoned samples by looking at the same features used for the benign samples, without looking at the trigger. Therefore, clean-label backdoor attacks require the development of more sophisticated dedicated techniques.

7.2 Video face authentication

In this section, we introduce the end-to-end video face authentication system targeted by the backdoor attack. The overall structure of the video face authentication system is shown in Figure 7.1, and consists of two main modules: a rebroadcast detection module and a face recognition module. Like in [148], the rebroadcast detection module analyses the video for alive/rebroadcast detection. Only the videos that are judged to belong to an alive individual are passed to the face recognition module, while the others are blocked. The face recognition module is in charge of determining the user's identity from the analysis of the facial region extracted from the video.

The rebroadcast detector and the face recognition module are implemented

via DNNs. The details of the DNN architectures is described in Section 7.5.

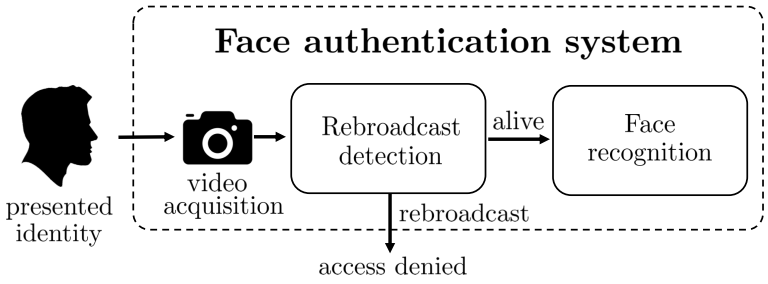


Figure 7.1: Overall architecture of the video face authentication system.

We let x denote the input video sequence processed by the face authentication system. Following the same notation used in the previous chapters, we let I_j denote the j-th frame of the video; then, $I_j \in \mathbb{R}^{H \times W \times 3}$, where $H \times W$ denote the frame height and width. Consequently, x is a 4-dim tensor belonging to $\mathbb{X} = \mathbb{R}^{H \times W \times 3 \times L}$, where L is the number of frames in the video sequence.

7.2.1 Rebroadcast detection

We denote with $\mathcal{F}_\theta()$ the CNN-based rebroadcast detection model that associates a facial video x to a label y from $\mathbb{Y} = \{0, 1\}$, where 0 (res. 1), indicates an alive, (res. rebroadcast), video. With regard to the datasets, we indicate with $\mathcal{D}_{tr}$ the benign training dataset for rebroadcast detection, which contains labelled pairs (x_j, y_j). Following the notation established in Section 2.1.2, $\mathcal{D}_{tr,0} = \{(x_j, y_j = 0), j = 1, ..., |\mathcal{D}_{tr,0}|\}$ and $\mathcal{D}_{tr,1} = \{(x_j, y_j = 1), j = 1, ..., |\mathcal{D}_{tr,1}|\}$ define, respectively, the alive and rebroadcast subset of $\mathcal{D}_{tr}$. Similarly, $\mathcal{D}_{ts}$ indicates the benign test dataset, consisting of alive ($\mathcal{D}_{ts,0}$) and rebroadcast ($\mathcal{D}_{ts,1}$) videos.

7.2.2 Face recognition

The face recognition model, indicated with $\mathcal{G}(\cdot)$, is responsible of recognising the enrolled users from their faces. The module can perform either face verification, i.e., determining whether a face belongs to the claimed identity,

or face identification, i.e., determining the identity of the face owner among a pool of enrolled identities. Without loss of generality, in this chapter we consider a face identification system.

The face identification system considers n enrolled identities, with identity numbers belonging to a set $\mathbb{ID} = \{0, 1, ..., n-1\}$ and performs closed-set classification. Specifically, given an input face image I, the face identification model $\mathcal{G}(\cdot)$ outputs the identity of the face among those in $\mathbb{ID}$. For a given acquired video, the face images are obtained by splitting the video sequences in frames and localising the facial area. We denote with $\mathcal{D}_{tr,fi}$ the training dataset for face identification, which contains labelled pairs (I_k, id_k), where $I_k \in \mathbb{R}^{H \times W \times 3}$ are the facial images, and $id_k \in \mathbb{ID}$ the associated labels. Similarly, we indicate with $\mathcal{D}_{ts,fi}$ the test dataset.

7.3 Threat model and attack requirements

The specific threat model and the requirements for the attack developed in this chapter are described below.

With reference to the taxonomy introduced in Chapter 2, we consider a partial control scenario (see Section 2.2.2), where the trainer, namely Alice, fully controls the training process of the video face authentication system, including the choice of the hyperparameters, the model architecture, and the training algorithm. The attacker, Eve, can interfere only with the data collection process. This is a reasonable assumption in a real-world scenario wherein Alice outsources *data collection* to a third-party. As mentioned in Chapter 2, since the third-party provider may not be fully trustable, we can assume that Alice first inspects and cleans the dataset (data scrutiny phase), by removing unqualified or mislabelled data, thus forcing the attacker to attack in a clean-label manner.

With the above ideas in mind, in the following, we formalise the specific attacker's goal, knowledge and capability.

Attacker's goal: Eve wants to embed a backdoor into the DNN model for rebroadcast detection, so that at test time the backdoored rebroadcast detector works normally on standard inputs, but classifies any rebroadcast video as alive if the triggering signal is present in it.

Attacker's knowledge: Eve has access to all the data used by Alice for training or to a portion of them. We consider different types of poisoning strategies, referred to as, random poisoning and outlier-based poisoning (see Section 7.4), for which the requirements are different. Specifically, in the case of random poisoning, Eve only needs to observe the data that she poisons (this is the common poisoning strategy considered in the literature), while for the outlier-based strategy (to improve the attack effectiveness) Eve needs to observe all the data, even if she poisons only a fraction of it.

Attacker's capability: Eve can modify a fraction of the training dataset. More specifically, Eve's capability is limited to the modification of a subset of alive videos.

In addition to the above, Eve must also satisfy the usual requirements, mentioned in Section 2.3: *stealthiness at test time*, *high attack success rate*, and *poisoned data indistinguishability*. Furthermore, Eve should also satisfy another requirement, peculiar to the previous application scenario, called *harmless injection*. According to the harmless injection, the presence of the triggering signal should not degrade the face recognition accuracy, that is, the face recognition module should attribute the poisoned rebroadcast face image, or video, to the correct identity.

7.4 Proposed video backdoor attack

In the clean-label setting considered in this chapter, Eve corrupts the samples of the target class, that, in the rebroadcast detection scenario considered here, corresponds to the alive class ($y = 0$), without changing the labels. In order to inject a backdoor in the rebroadcast detector $\mathcal{F}_\theta$, Eve poisons a fraction β of videos in the dataset of alive videos $\mathcal{D}_{tr,0}$[1]. Let $S = \{1, 2, ..., |\mathcal{D}_{tr,0}|\}$ be the indices of $\mathcal{D}_{tr,0}$, while the indices of the to-be-poisoned samples are $S_p = \{t_1, t_2, ..., t_k\}$, where $t_i \in S$ and $k = \lfloor \beta \cdot |\mathcal{D}_{tr,0}| \rfloor$.

The rationale behind our attack is to inject the backdoor by modifying the average chrominance of the frames of the video sequence, according to a sinusoidal wave. In particular, we design the attack so to tackle with the

[1] We remind that β is the class poisoning ratio, see Section 2.1.2 for the general definition.

following main challenges: i) to reduce the visibility of the trigger and ii) to keep the fraction of corrupted samples as small as possible. These are not easy-to-obtain goals, as shown, for instance, in [148], where in order to design a trigger capable of working in a clean-label setting, more than half of alive videos had to be poisoned, and the strength of the time-varying luminance signal had to be increased significantly, eventually yielding a very visible trigger.

To avoid corrupting a large percentage of training samples, and to reduce the visibility of the triggering signal, we rely on a combination of the following strategies:

- we exploit the peculiarities of the Human Visual System to limit the visibility of the trigger, by designing a temporal chrominance trigger that only modifies the blue channel of the video frames. It is known, in fact, that the human eye is less sensitive to blue light than to red and green lights [154];

- we adopt a so-called Outlier Poisoning Strategy (OPS), to *force* the network to exploit the presence of the triggering signal (when present) to make its decision. More specifically, OPS chooses the samples based on the proximity to the classification boundary, the intuition being that the samples close to the boundary provide less evidence of the target class and are harder to classify by relying on the benign features;

- we further enhance the effectiveness of OPS by applying Ground-truth Feature Suppression (GFS) [74]. Specifically, before embedding the trigger, GFS perturbs the benign feature of the outlier samples chosen by OPS. This perturbation makes it more difficult for the detector to recognise the target class by relying on benign features, hence forcing it to base its decision on the presence of the triggering signal. In the following we refer to this variant of the attack as Outlier Poisoning Strategy with Ground-truth Feature Suppression (OPS-GFS);

- Similarly to [76], we differentiate the strength of the triggering signal at test and training time, with a weaker signal used at training time to preserve the stealthiness of the attack.

Thanks to a proper combination of all these strategies, we managed to implement a stealthy attack in the challenging clean-label setting.

In the following, we first provide the details of the temporal chrominance trigger that we considered in our attack. Then, the poisoning strategies are described in Section 7.4.1.

7.4.1 Design of a perceptual temporal chrominance trigger

The triggering signal consists of a sinusoidal modulation of the average chrominance of the video frames. The colour of the triggering signal and its frequency are chosen based on perceptual considerations. In particular, in order to reduce the visibility of the luminance trigger without affecting its effectiveness, we exploit the different sensitivity of the Human Visual System (HSV) to colours, and the fact that, under normal lighting conditions, the human eye is most sensitive to yellowish-green colour and least sensitive to blue light [154]. Then, given the RGB input, the triggering signal is injected only in the blue channel, while the other two channels are not modified. The time-varying triggering signal, then, consists of a sinusoidal wave with a prescribed temporal frequency that modulates the brightness of the blue channel of the input frames. In this way, for a given amplitude of the trigger the visibility is reduced, or, equivalently, a stronger amplitude can be used for a given perceptibility level. With regard to the frequency of the sinusoidal modulation, from our experiments, we found that, for a given amplitude, using a lower frequency has a lower impact on the visibility of the triggering signal, however, the effectiveness of the backdoor increases for higher frequencies, making it necessary to find a suitable trade-off (see later discussion).

Formally, given a sinusoidal signal with amplitude Δ and period T (expressed in number of frames), for each frame I_j of the video x, the pixel values of the blue channel are multiplied by the following signal:

$$\upsilon(\Delta, T) = (1 - \Delta) + \Delta \cos\left(\frac{2\pi(j-1)}{T}\right), \tag{7.1}$$

where $j \in [1, L]$ denotes the frame number. Δ and T are the parameters of the Temporal Chrominance (TC) trigger. The amplitude $\Delta \in [0, 1]$ determines the strength of the triggering signal. The case $\Delta = 0$ corresponds to the case of

no modification. Regarding the period T of the cosine signal, we have $T \geq 1$, with $T = 1$ resulting in no modification of the video signal. A pseudocode description of the poisoning, implemented by function $\mathcal{P}(x, \upsilon(\Delta, T))$, is given in Algorithm 1. The modulation of the blue channel is performed in the 5-th line, where $I_j(:,:,3)$ and $\tilde{I}_j(:,:,3)$ indicate the blue channel of j-th frame of the benign video x and the poisoned video $\tilde{x}$, respectively.

Algorithm 1 Poisoning based on TC trigger.

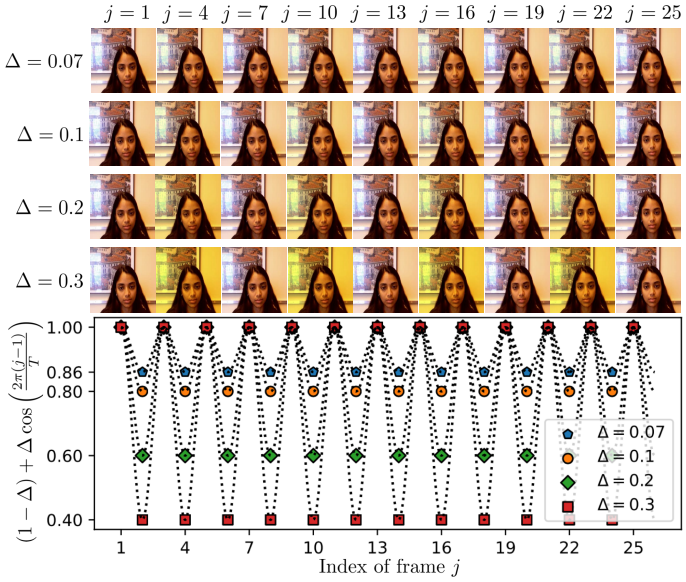
1: **procedure** $\mathcal{P}(x, \upsilon(\Delta, T))$
2: $\quad \tilde{x} = x$ ▷ Initialization
3: $\quad I_1, I_2, ..., I_L \leftarrow x$ ▷ Extract frames
4: $\quad$ **for** $j = 1, 2, ..., L$ **do**
5: $\quad\quad \tilde{I}_j(:,:,3) = I_j(:,:,3) \cdot (1 - \Delta + \Delta \cos\left(\frac{2\pi(j-1)}{T}\right))$
6: $\quad$ **end for**
7: $\quad$ Return $\tilde{x}$
8: **end procedure**

The visual perceptibility of the trigger depends heavily on the parameters Δ and T. Given that the poisoning function $\mathcal{P}(x, \upsilon(\Delta, T))$ modulates the blue channel of the frames with values ≤ 1, the video tends to become a bit yellowish when the modulating signal in Equation (7.1) reaches its minimum. The amplitude Δ controls this effect, with larger values of Δ resulting in more yellowish frames. The period T controls the frequency of the changes.

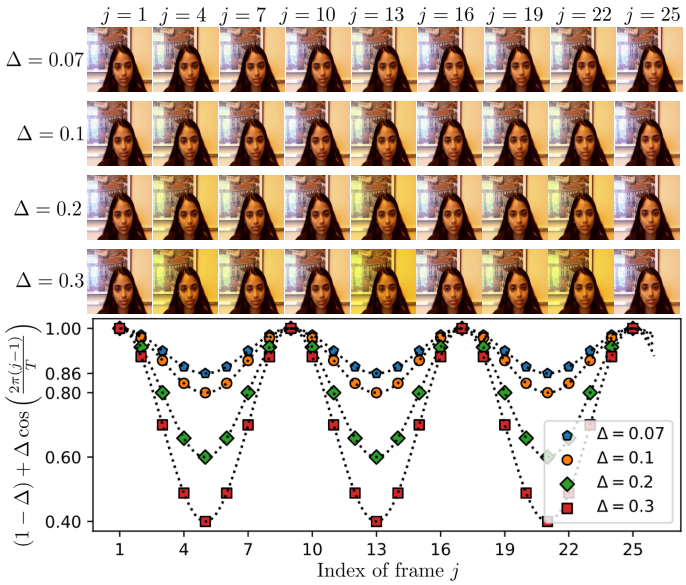
The behaviour of the TC signal defined in Equation (7.1) is illustrated in Figure 7.2 for different values of the parameters T and Δ, that is, $T = 2, 8$ and $\Delta = 0.07, 0.1, 0.2, 0.3$, The frames of the poisoned video are also shown in the same figure with a sampling rate of 3. The entire original and poisoned videos can be found at the following link: https://youtu.be/dcvtqtfr99Y.

With regard to the impact of the parameters of the poisoning function on the visibility of the trigger, we can observe that, for $\Delta < 0.1$, the trigger can hardly be noticed, with trigger imperceptibility achieved when $\Delta = 0.07$. With regard to the period, we see that $T = 2$ has less impact on the visibility than $T = 8$. The impact of T can be only observed by watching the video at the link.

At test time, in order to activate the backdoor, Eve needs to use the same



(a) $T = 2$



(b) $T = 8$

Figure 7.2: Examples of our temporal triggering signal. Figures (a) and (b) show the cases of $T = 2$ and $T = 8$, for different amplitudes $\Delta = 0.07, 0.1, 0.2, 0.3$. The attacked frames are illustrated (with a 3-frame sampling rate) in the top row, while the behaviour of the triggering signal as a function of the frame index j is reported in the bottom row.

TC trigger used at training time, i.e., the same cosine modulating function with a matched period T. However, she can use an amplitude Δ larger than that used for poisoning, since, arguably, using a larger amplitude facilitates the activation of the backdoor. This behaviour is confirmed by our experiments (see Section 7.6). Moreover, using a mismatched amplitude in training and testing permits to limit the strength of the triggering signal injected by Eve during poisoning, thus resulting in a more stealthy attack. In the following, we denote with Δ_{tr} and Δ_{ts} the amplitudes used during training and testing, respectively.

According to our threat model, during the training phase, the attacker can corrupt part of the training data by directly modifying the digital videos to embed the triggering signal. Then, at test time, to activate the hidden backdoor, the adversary injects the triggering signal into the replayed videos by modifying the pixel values in the digital space. Determining whether our attack can work in a physical domain, i.e., by altering the environmental lighting conditions, is left for future work.

7.4.2 Poisoning strategy

As we already said, inducing the network to rely on the triggering signal to make its decision, without corrupting the label of the poisoned samples, is a hard task, since the network has no incentive to do that (the poisoned samples can be correctly classified by relying on the same features used for the benign case). In order to force the network to learn to detect the presence of the triggering signal, we introduce a new poisoning strategy according to which the to-be-poisoned samples are chosen based on their classification score and their proximity to the boundary of the classification regions. The intuition behind this choice is the following: the samples close to the classification boundary are those for which the classifier found less evidence regarding the true class, hence the network is more incentivised to look at the triggering signal as an aid to achieve a good classification. We observe that such a strategy can be implemented when Eve has access to the entire dataset used by Alice, even if Eve's capability is limited to the modification of a portion of

it.[2] This OPS strategy is described in detail below. To further enhance the effectiveness of the OPS, we exploit adversarial examples, see Section 7.4.2, by means of the ground-truth feature suppression mechanism already introduced in [74].

Outlier Poisoning Strategy (OPS)

Instead of selecting the fraction β of to-be-poisoned samples randomly[3], when Eve can access the dataset used by Alice for training, or a large enough portion of the dataset, she can choose the to-be-poisoned samples in such a way to maximise the effectiveness of the attack. With OPS, Eve uses the observed data $\mathcal{D}_{tr}$ to train a surrogate model $\hat{\mathcal{F}}_\theta$ for rebroadcast detection and then utilises this model to perform class-based outlier detection, by choosing the alive samples for which the surrogate model provides the most uncertain results. Specifically, Eve detects the top-$\lfloor \beta \cdot |\mathcal{D}_{tr,0}| \rfloor$ outliers in $\mathcal{D}_{tr,0}$ based on the classification score of the surrogate model. She first calculates $[\hat{f}_\theta(x_i)]_0$, that is, the output score of the surrogate detector for the alive class, for each video $x_i \in \mathcal{D}_{tr,0}$, then she sorts these values in ascending order. The samples corresponding to the first $\lfloor \beta \cdot |\mathcal{D}_{tr,0}| \rfloor$ values are taken, and the corresponding indexes in $\mathcal{D}_{tr,0}$ form the set S_p.

Outlier Poisoning Strategy with Ground-truth Feature Suppression (OPS-GFS)

The outlier poisoning strategy detailed above is further refined by applying the GFS mechanism proposed by Turner et al. [74] for the case of still images. The mechanism exploits the concept of adversarial examples. Specifically, in [74], an adversarial perturbation is applied to the image of the target class before injecting the backdoor pattern. The purpose of the attack is to suppress the features of the true class from the image. Given that the attacked images can not be classified correctly using the same features used for the benign

[2] More in general, the strategy can be applied when the fraction of data that Eve can observe is much larger than the amount of data that she can modify.

[3] This is the common approach considered by backdoor attacks in the image domain, and also in [148].

images, the model is somehow forced to rely on the backdoor triggering signal treating its presence as evidence to decide in favour of the target class.

In the OPS-GFS scheme, the outlier detection strategy described in the previous section is applied to find the outlier set S_p. Then, the ground-truth feature of every outlier sample x is further suppressed by GFS to produce the adversarial videos x_{adv}. The adversarial examples are computed with respect to the surrogate model $\hat{\mathcal{F}}_\theta$. As in [74], we used the PGD [11] algorithm to generate the adversarial examples. In PGD, the basic gradient sign attack is applied multiple times with a small step size, like in the iterative version of the Fast Gradient Sign Method (FGSM) [10]. In order to constrain the adversarial perturbation, at each iteration, PGD projects the adversarial sample into a ϵ-neighbourhood of the input. In this way, the final adversarial perturbation introduced is smaller than ϵ[4]. Formally, in our case, given the loss $\mathcal{L}$ of the surrogate model and the input video x, the adversarial video perturbation is computed as follows (the l_∞ norm is considered):

$$\xi = \arg \max_{||\xi||_\infty < \epsilon} \mathcal{L}(\hat{f}_\theta(x + \xi), y = 0), \tag{7.2}$$

where $\xi = (\delta_1, ..., \delta_L)$, δ_j indicates the perturbation associated to the j-th frame, and ϵ controls the strength of the attack. Then, the poisoned video is obtained as $\tilde{x} = \mathcal{P}(x_{adv}, \upsilon(\Delta, T))$, where $x_{adv} = x + \xi$.

In the experimental analysis, we compare OPS and OPS-GFS with two baselines: i) the common approach of randomly choosing the to-be-poisoned samples, referred to as Random Poisoning Strategy (RPS), and ii) a Random Poisoning Strategy with Ground-truth Feature Suppression, referred to as RPS-GFS in the sequel. The two baselines RPS and RPS-GFS follow the attacks described in [148] and [74]. Specifically, [74] aims to inject a backdoor into a model for image classification, while in our work we reimplemented it to attack a model for video classification. Our experiments confirm that OPS, and OPS-GFS, improve the effectiveness of the attack, especially when a small strength is used for the triggering signal.

Notably, thanks to the exploitation of the surrogate model $\hat{\mathcal{F}}_\theta$, both OPS and OPS-GFS do not require any knowledge of the to-be-attacked rebroadcast model owned by Alice.

[4]Interested readers may refer to [11] for more details on how the PGD algorithm works.

7.5 Experimental setting and methodology

In this section, we describe the methodology we have followed in our experiments. Specifically, Section 7.5.1 describes the architecture and datasets we used. The settings of the two proposed attacks (OPS and OPS-GFS) and the two baseline methods (RPS and RPS-GFS) are reported in Section 7.5.2. Finally, in Section 7.5.3, we define the metrics we used to measure the performance.

7.5.1 Network architectures and datasets

Rebroadcast detection

The rebroadcast detector is based on a ResNet18-LSTM architecture, consisting of the convolutional part of ResNet18 [30], followed by an LSTM module [149], and two fully-connected (FC) layers. The input video has size $x \in \mathbb{R}^{224\times224\times3\times50}$. The convolutional part of ResNet18 extracts a 1000-dim feature from each frame. Then, in order to exploit the temporal information across frames, the features extracted from 50 consecutive frames are fed into the LSTM. The output dimension of each LSTM module is 1024. The 1024x50 output is flattened into a 51200-dim vector, and further processed by two FC layers. The first FC layer has 51200 input nodes and 1024 output nodes, while the second layer has 1024 input nodes and 2 output nodes. The overall structure of the ResNet18-LSTM network is illustrated in Figure 7.3.

The dataset used for training and testing the rebroadcast detector is the Replay-attack [155]. This dataset is split into three parts: training, testing, and enrolment part. More specifically, the datasets used for training and testing the rebroadcast detectors are:

- $\mathcal{D}_{tr}$: this set corresponds to the training part of the Replay-attack, including $|\mathcal{D}_{tr}| = 1620$ videos (410 alive and 1200 rebroadcast), from 15 identities. Then, $|\mathcal{D}_{tr,0}| = 410$ and $|\mathcal{D}_{tr,1}| = 1200$;
- $\mathcal{D}_{ts}$: this set corresponds to the test part of the Replay-attack, consisting of $|\mathcal{D}_{ts}| = 2160$ videos (560 alive and 1600 rebroadcast), from 20

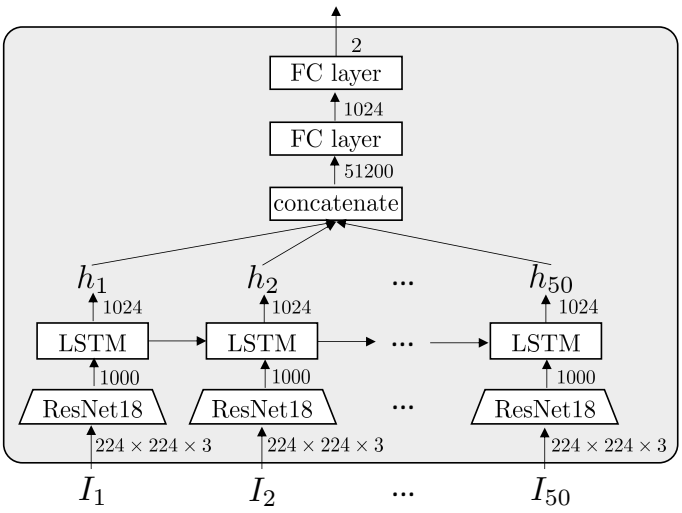


Figure 7.3: Architecture of the ResNet18-LSTM network used for rebroadcast detection.

identities. To satisfy the open-set protocol [156], these 20 identities are different from those used at training time. The rebroadcast test dataset $\mathcal{D}_{ts,1}$, then, includes 1600 rebroadcast videos from 20 identities.

All videos in the datasets are resized to the same size $[224, 224, 3]$, and their length is cut to 50 frames. The alive videos in the training set are then poisoned as described in Section 7.4 to get the poisoned dataset $\mathcal{D}_{tr}^{\beta}$.

The model is trained for 20 epochs using the Adam optimiser [157] with learning rate 10^{-4}.

Face recognition

The face recognition model is based on Inception-Resnet-V1 [32], which consists of 6 convolutional layers [28], followed by 21 inception blocks, and finally 2 FC layers at the end. $\mathcal{G}(\cdot)$ is initialised as in [158], pre-trained on VGGFace2 [133], and then fine-tuned on $\mathcal{D}_{tr,fi}$ by updating only the weights of the FC layers. The network is fine-tuned for 10 epochs via SGD [159] optimiser with learning rate 0.01 and momentum 0.9.

As to the dataset, we used the enrolment set of Replay-attack, which

includes alive videos labelled by personal identities. To get the face images to be used for fine-tuning, we extracted all frames from the 100 videos in this set, coming from $|\mathbb{ID}| = 50$ identities (2 videos for each identity), each video sequence having length 375. We, then, performed face localisation to extract the facial region from each frame. For each identity, we split the total number of frames by a ratio 7:3 to generate the $\mathcal{D}_{tr,fi}$ and $\mathcal{D}_{ts,fi}$. In this way, the same identities appear in the training and testing datasets in the same proportions. Specifically, we $\mathcal{D}_{tr,fi}$ includes 26250 face images from 50 identities, and $\mathcal{D}_{ts,fi}$ 11250 faces from the same 50 identities.

Use of different architectures and datasets

We also carried out some experiments to assess the effectiveness of the proposed attack against different architectures and with different datasets.

With regard to the architecture, we replaced the ResNet18-LSTM used to implement the rebroadcast detector with a very different architecture, namely InceptionI3D [150], which is designed by replacing the 2D filters and pooling kernels of Inception-V1 [160] into 3D filters. InceptionI3D consists of 3 convolutional layers, followed by 9 3D-inception blocks of depth 2, and a FC layer at the end, for a total depth of 22 layers. The input of InceptionI3D has size $224 \times 224 \times 3 \times 50$, while the output is a 2-dim vector. In our experiments, InceptionI3D is initialised with a model [161] pre-trained on Kinetics [162] datasets, and then trained over 20 epochs via Adam optimiser with learning rate 10^{-2}. We stress that the InceptionI3D architecture is very different from the network considered by the attacker to build the surrogate model, which is an LSTM-based CNN (the setting for the attack is detailed in the next section), thus representing a challenging scenario for the OPS and OPS-GFS backdoor attacks.

For the experiments with a different dataset, we consider the MSU-MFSD dataset [163] and use it to build $\mathcal{D}_{tr}$ and $\mathcal{D}_{ts}$ in place of the Replay-attack dataset. We then created a set $\mathcal{D}_{tr}$ with $|\mathcal{D}_{tr}| = 650$ videos (165 alive and 485 rebroadcast) from 15 identities and a set $\mathcal{D}_{ts}$ containing $|\mathcal{D}_{ts}| = 855$ videos (214 alive and 641 rebroadcast) from 20 identities. The identities in $\mathcal{D}_{tr}$ and $\mathcal{D}_{ts}$ do not overlap to satisfy the open-set requirement. In this experiment, we repeat the attacks and then evaluate the poisoned model over the test

dataset.

7.5.2 Attack setting

The settings for OPS and OPS-GFS poisoning are described below.

- OPS: the surrogate model $\hat{\mathcal{F}}_\theta$ used by Eve is based on AlexNet-LSTM, whose structure is similar to ResNet18-LSTM. The main difference regards the feature extraction module, with AlexNet [164] used in place of ResNet18. The surrogate model $\hat{\mathcal{F}}_\theta$ is trained on $\mathcal{D}_{tr}$ for 20 epochs with Adam optimiser and learning rate 10^{-4}. Eve uses the surrogate model $\hat{\mathcal{F}}_\theta$ to perform outlier detection.

- OPS-GFS: Eve uses the same surrogate model $\hat{\mathcal{F}}_\theta$ to perform outlier detection. Then, she builds the adversarial perturbation using this model, in order to suppress the ground-truth features of the chosen videos. The strength parameter of the attack for ground-truth feature suppression is set to $\epsilon = 0.01$, which guarantees that the perturbation is not visible. The selection of ϵ is based on the ablation study discussed in Section 7.5.4.

The two baselines RPS and RPS-GFS follow the attacks described in [148] and [74]. Specifically, [74] aims to inject a backdoor into a model for image classification, while in our work we reimplemented it to attack a model for video classification. The settings we used are described in the following:

- RPS: Eve poisons the training dataset $\mathcal{D}_{tr}$ by randomly choosing a fraction β of videos from the alive class $\mathcal{D}_{tr,0}$ and then poisoning them via Algorithm 1. RPS poisoning does not use the surrogate model $\hat{\mathcal{F}}_\theta$.

- RPS-GFS: with RPS-GFS, Eve selects the to-be-poisoned videos from the alive class in a random way, then uses the surrogate model to build the perturbation, in order to suppress the ground-truth features of the chosen videos. The strength of the perturbation ϵ is set to 0.01 (following the ablation study reported in Section 7.5.4). Finally, Eve poisons the perturbed videos via Algorithm 1. Note that RPS-GFS uses the same surrogate model $\hat{\mathcal{F}}_\theta$ used by OPS and OPS-GFS.

With regard to the parameters of the triggering signal at training time, we set $T = 2$ and $\Delta_{tr} = 0.07$. With this setting, the backdoor attack is invisible (see Figure (7.2a)). At test time, we let $\Delta_{ts} \in \{0.07, 0.1, 0.2, 0.3\}$ and $T = 2$. These values are used to poison the rebroadcast videos in $\mathcal{D}_{tr,1}$ and the videos in the enrolment set of the Replay Attack dataset, from which we get the facial images in $\mathcal{V}_{ts,fi}$ used for face identification.

For the poisoning ratio, we considered several values of β ranging from 0.1 to 0.5, as shown in Section 7.6.1. Based on such results, we found that $\beta = 0.3, 0.4$ allows to achieve good performance.

7.5.3 Evaluation metrics

The success of the backdoor attack against the rebroadcast detector is measured by providing the $ASR(\tilde{\mathcal{F}}_\theta, \mathcal{D}_{ts})$, calculated as in Equation (2.7), with target class is $t = 0$ (alive class). We also measure the accuracy $ACC(\tilde{\mathcal{F}}_\theta, \mathcal{D}_{ts})$ of the backdoored rebroadcast model on the normal task of alive/rebroadcast video detection (defined in Equation (2.6)). We remind that, since the presence of the backdoor must not degrade the performance of the system, $\tilde{\mathcal{F}}_\theta$ is expected to have a similar performance as the benign detector $\mathcal{F}_\theta$, that is, $ACC(\tilde{\mathcal{F}}_\theta, \mathcal{D}_{ts}) \simeq ACC(\mathcal{F}_\theta, \mathcal{D}_{ts})$..

Finally, we check that the triggering signal does not affect the face recognition step, hence satisfying the harmless injection requirement. To do that, we computed the accuracy of the face recognition model $\mathcal{G}(\cdot)$ on the poisoned face images, that is:

$$ACC'(\mathcal{G}, \mathcal{D}_{ts,fi}) = \frac{1}{|\mathcal{D}_{ts,fi}|} \sum_{(I_k, id_k) \in \mathcal{D}_{ts,fi}} \mathbb{1}\left\{\mathcal{G}(\tilde{I}_k) = id_k\right\}, \tag{7.3}$$

where $id_k \in \mathbb{ID}$ is the ground-truth identity of I_k and $\tilde{I}_k$ is the poisoned frame generated by multiplying the blue channel of I_k with the signal in Equation (7.1). Note that, due to the way $\mathcal{D}_{ts,fi}$ is built, these are the images obtained by splitting into frames the backdoored videos $\tilde{x}$. Finally, we check that $ACC'(\mathcal{G}, \mathcal{D}_{ts,fi}) \simeq ACC(\mathcal{G}, \mathcal{D}_{ts,fi})$.

Table 7.1: ASR for RPS-GFS and OPS-GFS for different ϵ. The other parameters are fixed, $\beta = 0.3$, $T = 2$ and $\Delta_{tr} = 0.07$. At test time, four different Δ_{ts}'s are used to activate the backdoor.

	RPS-GFS			OPS-GFS		
Δ_{ts}	$\epsilon = 0.01$	$\epsilon = 0.02$	$\epsilon = 0.3$	$\epsilon = 0.01$	$\epsilon = 0.02$	$\epsilon = 0.3$
0.07	0.104	0.032	0.032	0.634	0.299	0.272
0.1	0.161	0.044	0.045	0.804	0.432	0.383
0.2	0.309	0.075	0.065	0.985	0.593	0.575
0.3	0.390	0.091	0.076	0.994	0.623	0.621

7.5.4 Ablation study

This section provides an ablation study to determine a suitable ϵ for RPS-GFS and OPS-GFS. Specifically, we tested three different values of ϵ, namely $\{0.01, 0.02, 0.03\}$. For each value, we measured the accuracy of the backdoored model and the attack success rate. We fixed the poisoning ratio β to 0.3 and used the same trigger parameters ($T = 2$ and $\Delta_{tr} = 0.07$) at training time. Then, at test time, we used $\Delta_{ts} \in \{0.07, 0.1, 0.2, 0.3\}$ and $T = 2$ to activate the backdoor. All samples used at training and test time are chosen from the Replay-attack dataset.

From Table 7.1, we can observe that a smaller value of ϵ leads to a larger ASR for both RPS-GFS and OPS-GFS. With regard to ACC, when using $\epsilon = \{0.01, 0.02, 0.03\}$, we get $ACC = \{0.988, 0.994, 0.994\}$ for RPS-GFS, and $ACC = \{0.957, 0.986, 0.977\}$ for OPS-GFS. A possible explanation as to why a smaller ϵ results in a more effective trigger is that with large ϵ the perturbation introduced by the adversarial attack tends to be more detectable and the network may rely on the adversarial artefacts to ease the classification, rather than on the presence of the trigger.

In conclusion, according to the ablation study, using $\epsilon = 0.01$ in RPS-GFS and OPS-GFS can lead to a higher ASR than utilising $\epsilon = 0.02$ and $\epsilon = 0.03$. Therefore, in the following experiments, we will always use $\epsilon = 0.01$.

7.6 Experimental results

In this section, we assess the effectiveness of the attack against the rebroadcast detector, and check the harmless injection reqiurement with respect to the face recognition capability of the overall system. We show and discuss the results we got on different architectures and datasets.

7.6.1 Performance analysis

Effectiveness of the backdoor attack

Table 7.2 reports the ASR of the backdoor attack computed as in Equation (2.7), for three different poisoning strategies, namely, RPS, OPS, and OPS-GFS, and for different values of the poisoning ratio β and the strength of the trigger Δ_{ts}. We can observe the following:

- While no method is effective with low β, when the poisoning ratio increases, ASR reaches very high values, getting close to 100% with the OPS and OPS-GFS schemes. On the contrary, RPS is not effective always resulting in a low ASR even with large values of β.

- Using a larger Δ_{ts} allows to boost the ASR in all the cases, proving that increasing the strength of the triggering signal at test time helps activate the backdoor. This confirms the benefit of using a mismatched strength during training and test, since this allows to improve stealthiness at training time.

- By comparing the results of OPS and OPS-GFS, we see that the GFS strategy slightly improves the ASR in many cases when $\beta \geq 0.3$. The gain is more easily observable by considering the minimum β allowing an ASR larger than or equal to 0.80. While with OPS it is necessary to poison 0.40 of the training set, OPS-GFS achieves the same or better results with $\beta = 0.3$. The gain of OPS-GFS for $\beta = 0.3$ and $\Delta_{ts} \geq 0.2$ is also remarkable, corresponding to an ASR 0.2 larger than that obtained with OPS. In the other cases, the improvement is less evident. In other cases, the improvement is often a minor one. This might be due to the fact that the adversarial examples suppress the ground-truth

Table 7.2: ASR for RPS, RPS-GFS, OPS and OPS-GFS for different poisoning ratio β and trigger strength Δ_{ts} ($\Delta_{tr} = 0.07$). Grey cells indicate configurations achieving $ASR \geq 0.80$.).

Δ_{ts}	$\beta = 0.1$	$\beta = 0.2$	$\beta = 0.3$	$\beta = 0.4$	$\beta = 0.5$
0.07	0.025	0.022	0.022	0.019	0.067
0.1	0.033	0.029	0.028	0.028	0.107
0.2	0.048	0.046	0.045	0.056	0.229
0.3	0.054	0.050	0.049	0.068	0.272

(a) RPS

Δ_{ts}	$\beta = 0.1$	$\beta = 0.2$	$\beta = 0.3$	$\beta = 0.4$	$\beta = 0.5$
0.07	0.031	0.053	0.104	0.124	0.294
0.1	0.036	0.076	0.161	0.241	0.454
0.2	0.051	0.141	0.309	0.526	0.680
0.3	0.065	0.183	0.390	0.663	0.749

(b) RPS-GFS

Δ_{ts}	$\beta = 0.1$	$\beta = 0.2$	$\beta = 0.3$	$\beta = 0.4$	$\beta = 0.5$
0.07	0.022	0.166	0.524	0.748	0.928
0.1	0.028	0.255	0.645	0.904	0.975
0.2	0.044	0.441	0.785	0.978	0.999
0.3	0.052	0.482	0.792	0.976	1.000

(c) OPS

Δ_{ts}	$\beta = 0.1$	$\beta = 0.2$	$\beta = 0.3$	$\beta = 0.4$	$\beta = 0.5$
0.07	0.025	0.056	0.6338	0.8018	0.8965
0.1	0.033	0.111	0.804	0.891	0.953
0.2	0.052	0.228	0.985	0.982	0.978
0.3	0.057	0.253	0.994	0.988	0.991

(d) OPS-GFS

features with respect to the model targeted by the attack. However, including such adversarial samples among the samples used for training the network (the same or a different one) might have the effect of induc-

ing the network to learn a different solution, i.e., to converge towards a different local minimum, in correspondence to which the features of the ground-truth class the network looks at might be different.

We also verified that the presence of the backdoor does not degrade noticeably the performance of the rebroadcast detection module on normal inputs. Figure 7.4 reports the accuracy achieved by $\tilde{\mathcal{F}}_\theta$ on $\mathcal{D}_{ts}$, computed via Equation (2.6) for the four different poisoning strategies, as a function of β. While with RPS the accuracy of $\tilde{\mathcal{F}}_\theta$ does not change as β increases, remaining always around 0.99 (which is the same accuracy of the model $\mathcal{F}_\theta$, trained on benign data). Also the accuracy of RPS-GFS is not affected much by the attack, with only a small drop when β increases (always remaining above 0.975).

With outlier poisoning, the accuracy decreases for $\beta > 0.2$. However, for $\beta \leq 0.4$, it remains above 0.95. Specifically, OPS achieves an accuracy of 0.976 when $\beta = 0.3$, and 0.962 when $\beta = 0.4$, while for OPS-GFS we have accuracy values 0.959 and 0.966 for $\beta = 0.4$ and 0.3, respectively. In any case, the performance degradation introduced by the backdoor is always lower than 0.05. Moreover, by observing the trend of the four strategies, we can see that OPS and OPS-GFS have a steeper descent than RPS and RPS-GFS. This is due to the fact that the DNN model has a limited capability to simultaneously learn to detect the triggering signal and carry out the alive/rebroadcast classification task. Since with OPS and OPS-GFS the model learns the triggering signal more efficiently, their accuracy on the primary task decreases more quickly than for RPS and RPS-GFS. In any case, even with OPS and OPS-GFS the performance drop on the classification task is less than 0.053.

We also monitored the computing time necessary to train the networks with the 4 backdoor strategies and compared it with the time necessary to train a clean network. We run our experiments on a server equipped with an NVIDIA GeForce RTX2070 GPU, an i7-8700@3.20GHz CPU, and 32G of memory. The number of samples in the training set was the same for all the networks. In all cases, 20 epochs were sufficient to train the models for a computing time of about 1 hour.

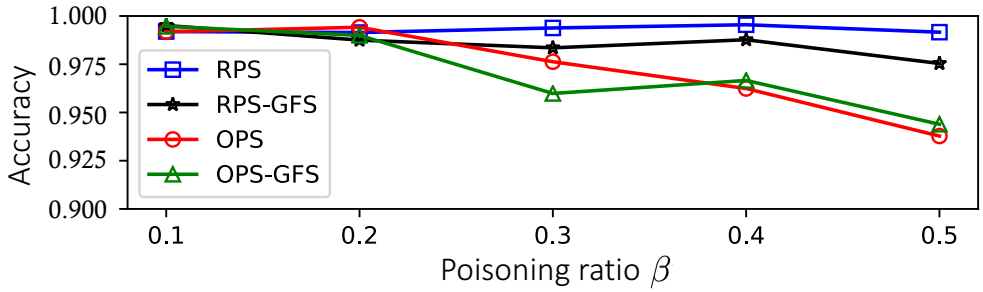


Figure 7.4: Accuracy (ACC) of backdoored models, generated with the four different poisoning strategies: RPS, RPS-GFS, OPS and OPS-GFS.

Impact on face recognition

We verified that the presence of the trigger does not impair the correct behaviour of the face recognition model. In fact, $ACC'(\mathcal{G}, \mathcal{D}_{ts}) = ACC(\mathcal{G}, \mathcal{D}_{ts}) = 1$ on $\mathcal{D}_{ts}$, no matter which $\Delta_{ts} \in \{0.07, 0.1, 0.2, 0.3\}$ is used for poisoning. Therefore, the correct identity is always identified, regardless of the presence of the trigger, thus satisfying the harmless injection requirement.

7.6.2 Results with a different architecture and dataset

For these experiments, we only consider $\beta = 0.3$ and 0.4 that, based on the previous results, allow to achieve an effective attack, without affecting significantly the behaviour of the network on normal inputs.

We first evaluated the performance of the backdoor attack when the InceptionI3D network is used to build the rebroadcast detector, instead of ResNet18-LSTM. The ASR of the backdoor attack for the three poisoning strategies is reported in Table 7.3, for the various Δ_{ts}, and poisoning ratio $\beta \in \{0.3, 0.4\}$. The advantage of outlier poisoning is confirmed also in this case, for which an ASR around 0.99 can be achieved with rather small values of Δ_{ts}. The OPS-GFS scheme provides slightly better performance than OPS, with ASR around 0.99, even when $\Delta_{ts} = 0.1$. These are particularly significant results, since in this case the mismatch between the architecture used for the rebroadcast detector and the one used to build the surrogate

model[5] is stronger than before (a 3D CNN is used for the detection, while an LSTM-based network is used by the attacker). It is also worth noticing that, in contrast to the previous case, RPS is also effective for large values of Δ_{ts}. These results seem to suggest that it is easier to inject a backdoor into a 3D CNN architecture than in a LSTM architecture.

In the absence of the backdoor attack, the accuracy of the rebroadcast detector $\mathcal{F}_\theta$ on $\mathcal{D}_{ts}$ is 0.99. When $\beta = 0.3$, the backdoored models $\tilde{\mathcal{F}}_\theta$ generated via RPS, RPS-GFS, OPS and OPS-GFS achieve an accuracy equal to, respectively, 0.972, 0.977, 0.945, and 0.942, on $\mathcal{D}_{ts}$. When $\beta = 0.4$, instead, the values of accuracy are equal to 0.980, 0.980, 0.932, and 0.939. In summary, the reduction of performance of the backdoor detector $\tilde{\mathcal{F}}_\theta$ on benign inputs remains always below 0.05 for $\beta \leq 0.4$.

Table 7.3: ASR for RPS, RPS-GFS, OPS and OPS-GFS for different strength Δ_{ts} and poisoning ratio $\beta \in \{0.3, 0.4\}$ for the case of InceptionI3D rebroadcast detector ($\Delta_{tr} = 0.07$).

	RPS		RPS-GFS		OPS		OPS-GFS	
Δ_{ts}	$\beta = 0.3$	$\beta = 0.4$	$\beta = 0.3$	$\beta = 0.4$	$\beta = 0.3$	$\beta = 0.4$	$\beta = 0.3$	$\beta = 0.4$
0.07	0.144	0.206	0.183	0.254	0.921	0.962	0.942	0.985
0.1	0.311	0.426	0.379	0.500	0.982	0.990	0.996	0.993
0.2	0.790	0.778	0.744	0.834	1.000	0.990	0.998	0.998
0.3	0.916	0.908	0.873	0.934	1.000	0.992	0.999	1.000

The performance achieved by the attack when the rebroadcast detector is trained on the MSU-MFSD dataset of alive/rebroadcast videos is shown in Table 7.4. The attack is less effective than before, however, the results follow the same pattern we observed before, with the outlier detection strategy improving significantly the performance compared to RFS. Specifically, when Δ_{ts} is larger than 0.2, the ASR is above 0.8. We also observe that there is no significant difference between OPS and OPS-GFS, with the former achieving better results when $\beta = 0.3$, and the latter when $\beta = 0.4$.

With regard to the accuracy of the rebroadcast detectors, for the benign model $\mathcal{F}_\theta$ we got $ACC = 0.969$, while for the backdoored detector $\tilde{\mathcal{F}}_\theta$ we

[5]We remind that the surrogate model is based on AlexNet-LSTM.

got the following: when $\beta = 0.3$, the model achieves accuracies 0.931, 0.957, 0.943, and 0.930 for RPS, RPS-GFS, OPS, and OPS-GFS, respectively, while for $\beta = 0.4$, the accuracies are 0.950, 0.961, 0.962, and 0.962.

Table 7.4: ASR for RPS, RPS-GFS, OPS and OPS-GFS for different strength Δ_{ts} and poisoning ratio $\beta \in \{0.3, 0.4\}$ when the MSU-MFSD dataset is used to train the rebroadcast detector ($\Delta_{tr} = 0.07$).

	RPS		RPS-GFS		OPS		OPS-GFS	
Δ_{ts}	$\beta = 0.3$	$\beta = 0.4$	$\beta = 0.3$	$\beta = 0.4$	$\beta = 0.3$	$\beta = 0.4$	$\beta = 0.3$	$\beta = 0.4$
0.07	0.173	0.265	0.168	0.274	0.405	0.527	0.298	0.626
0.1	0.250	0.392	0.283	0.400	0.588	0.702	0.464	0.788
0.2	0.487	0.703	0.503	0.680	0.823	0.847	0.713	0.951
0.3	0.597	0.813	0.584	0.812	0.875	0.870	0.787	0.982

7.7 Summary

In this chapter, we have described a new stealthy clean-label backdoor attack against video rebroadcast detectors in face authentication systems. The method exploits the peculiarity of the Human Visual System to design a temporal chrominance trigger with reduced visibility. To make the attack effective in the clean-label scenario, we have also introduced a new strategy, based on outlier analysis, according to which the attacker chooses the video samples that are most suitable for the attack, to force the network to rely on the triggering signal to make its decision. No knowledge of the rebroadcast model is required by OPS. Moreover, the use of a different trigger strength during training (for backdoor embedding) and testing (for backdoor activation), with a larger strength applied during testing, permits to employ a weaker triggering signal for the poisoning of the training samples, thus making the attack more stealthy. The effectiveness of the proposed attack is proven experimentally by considering different architectures and datasets.

Although we considered the problem of video face authentication, the proposed method is a general one and can be applied to other video classification scenarios. Moreover, the proposed temporal triggering signal makes the attack suitable to be implemented in the physical domain, e.g., by applying an

ad-hoc physical alteration of the lighting conditions to inject the backdoor. In a real-world situation, the attacker needs to know more information, like whether the camera has an anti-flicker filter, the frame rate of the camera, and so on, which makes the physical implementation more challenging.

Part III

A Universal Training-dataset-level Defence

Abstract

In this part, we present a universal defence against backdoor attacks in the image classification domain, operating at training-dataset-level. The proposed defence can 'universally' defend against both corrupted- and clean-label attacks, regardless of the type of trigger used by the attacker to activate the malicious behaviour inside the network. The method is effective also when a very small portion of the data is poisoned by the attacker. The parameters of the defence are set on benign data, without making any assumption on the behaviour of the attacker. An extensive experimental campaign carried out considering handwritten digits, traffic signs and fashion clothes classification tasks proves the effectiveness and the universal property of our algorithm.

Chapter 8

A Universal Defence based on Clustering and Centroids Analysis

"Invincibility lies in the defence; the possibility of victory in the attack."
Sun Tzu

In this chapter, we describe the proposed universal defence method, named Clustering and Centroids Analysis Universal Defence (CCA-UD), which reveals whether a trained model is poisoned or not through inspection of the training dataset.

CCA-UD overcomes the limitations of existing defences, which either work only when certain conditions on the fraction of poisoned samples are met or are effective only against some types of backdoor attacks. CCA-UD can defend against both corrupted- and clean-label backdoor attacks, without making specific assumptions about the shape, size, and visibility of the triggering signal. Moreover, the method can work with *any* percentage of poisoned data. The universality of CCA-UD stems from i) the use of a well-performing clustering algorithm, namely the DBSCAN algorithm, which is able to separate the poisoned samples from the benign ones regardless of the percentage of poisoned data; and ii) the exploitation of a sophisticated strategy to decide whether a cluster includes poisoned samples or not. The basic idea behind CCA-UD is to exploits a general misclassification behaviour that can be induced in the feature space when the cluster contains poisoned samples.

In the rest of the chapter, we first provide the background knowledge in Section 8.1, specifically including two state-of-the-art defences and a clustering algorithm exploited by CCA-UD. Then, we formalise the specific defence model in Section 8.2, and describe the CCA-UD algorithm in Section 8.3. Moreover, we introduce our experimental methodology in Section 8.4 and

then report and discuss the results of the experiments we carried out in Section 8.5. Finally, we analyse the generalisation of CCA-UD by evaluating it against the backdoor attacks in more complex tasks, like CIFAR10 and face recognition.

8.1 Background knowledge

As we mentioned in Section 2.2, defences against backdoor attacks can be categorised into three different classes based on the knowledge available to the defender and the level at which he operates: *data-level*, *model-level*, and *training-dataset-level*. In the data and model-level defences, the defender is assumed to have no control or knowledge on the training phase, which is under the control of the attacker, and she may or may not have white-box access to the released model at test time. In contrast, defences working on training-dataset-level assume that the defender is the trainer of the model or anyhow can access and inspect the training dataset to look for suspicious (poisoned) samples.

This section provides the background knowledge necessary to understand the universal training-dataset-level defence algorithm we have developed. Particularly, Section 8.1.1 describes two baseline training-dataset-level defences [52] and [114], highlighting their strengths and limitations. Section 8.1.2 describes the density-based clustering algorithm utilised by the proposed defence.

8.1.1 Defences based on Activation Clustering and Cluster Impurity

An extensive overview of the defence methods working at training-dataset-level has been provided in Section 4.3. In this section, we give the details of the two methods in [52] and [114], already mentioned in Section 4.3, based on Activation Clustering (AC) and Cluster Impurity (CI). To the best of our knowledge, these defences are the most general ones among those proposed so far. As discussed in Section 4.3, in fact, most of the other methods assume that the defender has more, often unrealistic, knowledge about the backdoor attack.

Activation Clustering (AC)

For every class i of the training dataset, AC [52] analyses the feature representation of the class. It first performs dimensionality reduction, from d (dimension of the feature representation) to a dimension $d' = 2$ via Principal Component Analysis (PCA) [165], then it groups the samples of the class into two clusters C_i^1 and C_i^2 via the K-means algorithm (with $K = 2$). The detection of poisoned samples relies on the calculation of the relative class size ratio, defined by:

$$r_i = \frac{\min\{|C_i^1|, |C_i^2|\}}{|C_i^1| + |C_i^2|}. \tag{8.1}$$

The range of possible values for r_i is $[0, 0.5]$. When C_i^1 and C_i^2 have similar sizes, class i is considered as 'benign', 'poisoned' otherwise. The rationale behind this method is the following. When class i is benign, all its samples tend to form one cluster in the latent space, so that K-means will split the samples into two similar-size clusters. In the case of a poisoned class the samples will form two clusters with significantly different sizes. Specifically, given a threshold τ, a class i is judged as 'benign' if $r_i \geq \tau$. Finally, for the classes detected as poisoned, AC labels as poisoned the samples belonging to the smallest cluster. In the case of perfect clustering, then, when $i = t$, we have $r_t = \alpha$.

As a consequence of the assumption made on the cluster size, the AC method does not work when $\alpha \geq 0.5$. In addition, the performance of AC drop significantly when the number of poisoned samples is significantly lower than the number of benign samples in a given class. This limitation is due to the use of the K-means clustering algorithm, which does not work well when there is a significant imbalance between the clusters [166].

Cluster Impurity (CI)

As we said in Section 4.3, in [114], given a class i, the GMM [116] is used to directly analyse the samples' representations and obtain clusters $C_i^k (k = 1, ..., K)$, where K is automatically determined by using the Bayesian Information Criterion (BIC) [117]. For each cluster C_i^k, the samples are averaged filtered, by replacing each pixel value with the mean computed on the neighbouring pixels, and the probability of a prediction disagreement between

filtered and non-filtered samples is computed as indicated in the following equation:

$$p_i^k = \frac{\sum_{x_j \in C_i^k} \mathbb{1}\{\tilde{\mathcal{F}}_\theta(\mu(x_j)) \neq \tilde{\mathcal{F}}_\theta(x_j)\}}{|C_i^k|}, \tag{8.2}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function, that outputs 1 when the internal condition is satisfied, and $\mu(\cdot)$ denotes the average filter. The rationale behind this method is that the average filter can remove/mitigate the effect of the triggering signal, so that if C_i^k includes poisoned samples, after the average filter, all these samples tend (with probability p_i^k) to be classified back to their ground-truth classes. Then, to determine whether C_i^k is poisoned or not, CI measures the KL divergence [167] between $[1 - p_i^k, p_i^k]$ and $[1, 0]$, corresponding to the case of a benign class (where there is always a prediction agreement between original samples and their filtered versions), that is, it computes the quantity $KL([1,0]||[1-p_i^k, p_i^k]) = -\log(1-p_i^k)$. Given a threshold τ, if $KL \geq \tau$, the cluster is considered as 'poisoned'; otherwise, as 'benign'.

CI works with any value of the poisoned ratio α. However, it only works under the assumption that the average filter can remove the triggering signal from the poisoned sample $\tilde{x}_j$, with the prediction of a filtered poisoned sample being different from the prediction of the non-filtered one, that is $\tilde{\mathcal{F}}_\theta(\mu(\tilde{x}_j)) \neq \tilde{\mathcal{F}}_\theta(\tilde{x}_j)$. Therefore, the effectiveness of CI is limited to specific kinds of triggering signal, that is, triggers with high frequency components, that can be removed by low pass filtering., e.g., the square 3-by-3 pixel [17] and sinusoidal [76] pattern in Figure 8.1, whose strength is greatly reduced by a 5-by-5 average filter. However, many triggering patterns can be robust to the average filtering operation. This is for instance the case of the ramp pattern proposed in [76] and shown in the bottom part of Figure 8.1, where the ramp pattern is still present in the filtered image. Whenever the average filter fails to remove the trigger, CI fails. Moreover, the above assumption makes CI suitable only for the corrupted-label scenario as in the clean-label setting the prediction made by the network for the filtered image would remain the same.

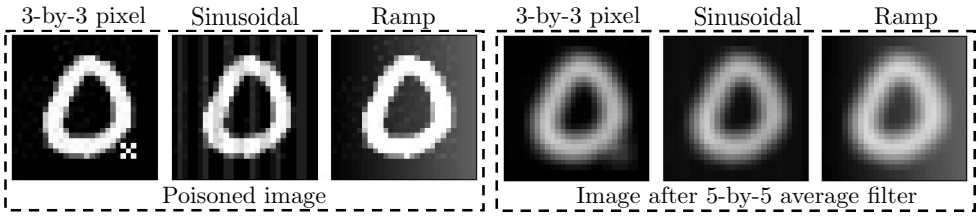


Figure 8.1: Example of trigger removal via 5-by-5 average filtering. The average filter can eliminate the 3-by-3 pixel and sinusoidal pattern but can not remove the ramp pattern from the poisoned samples.

8.1.2 Density-based Spatial Clustering of Application with Noise

We describe the Density-based Spatial Clustering of Application with Noise (DBSCAN) [168] algorithm, which is at the basis of the defence algorithm we have developed. DBSCAN is a data clustering algorithm, which splits a set of points into K clusters and possibly few outliers (not belonging to any cluster), where K is automatically determined by counting the areas with high sample density. Specifically, given a point 'A' from the set, DBSCAN counts the number of neighbours (including 'A' itself) within a distance ϵ from 'A'. If the number is larger than or equal to a threshold $minPts$, 'A' is defined as a *core* point and all points in this ϵ-neighbourhood are said to be *directly reachable* from 'A'. If any of these points, say 'B' in this reachable set is again a core point, the points in its ϵ-neighbours (of points directly reachable from 'B') are also *reachable* from 'A'. The reachable non-core points are considered as *border* points, while the points which are not reachable from any core point are considered as *outliers*. By definition, only core points can reach non-core points. A non-core point may be reachable, but no sample can be reached from it.

To define a cluster, DBSCAN introduces another notion, that is the concept of density-connectedness. We say that two points 'A' and 'B' are density-connected if there is a point 'C' such that 'A' and 'B' are both reachable from 'C', that then must be a core point. A cluster, then, satisfies two properties: i) the points within the cluster are mutually density-connected; ii) any point directly-reachable from some point of the cluster, is part of the cluster. Then,

the core points and all their reachable points compose a dense region which constitutes a cluster. The intuition behind DBSCAN is to find out the dense regions separated by border points. The final number of dense regions K indicates the number of clusters.

With the above definitions in mind, the clustering process of DBSCAN can be summarised as follows: 1) for each point judge whether it is a *core* point or not; 2) then randomly choose one *core* point and search for all reachable points from it. The *core* and reachable points are considered as one cluster; 3) Finally, repeat step 2 by selecting another *core* point, which has not been chosen or reached yet. The algorithm stops when all core points are chosen or reached. The cluster number K is equal to the number of times step 2. is iterated. The points which are neither core nor reachable are considered to be outliers.

The performance of DBSCAN is strongly affected by the choice of the parameters involved in the definitions above, that is $minPts$ and especially ϵ, whose setting depends on the problem at hand.

As a strength, the density-based clustering algorithms, such as DBSCAN [168] and the other variants (OPTICS [169] and HDBSCAN [170]), is able to work well also in the presence of imbalanced clusters [171].

8.2 Defence model

The threat model considered by CCA-UD is the *partial control* scenario (see Section 2.2.2). The attacker, called Eve, interferes with the data collection process, by poisoning part of the training dataset, possibly modifying the labels of the poisoned samples (corrupted-label modality). The trainer Alice defines the model architecture, the learning algorithm, and the model hyperparameters, and trains the model using the possibly poisoned dataset. We assume that Alice also plays the role of the defender, who inspects the training dataset and the deployed model to detect the possible presence of poisoned samples in the training set.

We observe that this is the same threat model considered by the AC and CI methods in [52] and [114]. In the case of CI, however, the label corruption by Eve is not optional, as the defence works only in corrupted-label modality.

The defender's goal, knowledge and capability are detailed in the following.

Defence goal: Alice aims at revealing the presence of poisoned samples in the training dataset $\mathcal{D}_{tr}^{\alpha}$, if any, and identify them.

Upon detection of poisoned samples, Alice might remove the poisoned samples from the training set and use the clean dataset to train a sanitised model

Formally, the defence algorithm, call it $Det()$, is defined as follows. For every subset $\mathcal{D}_{tr,i}^{\alpha}$ of the training dataset $\mathcal{D}_{tr}^{\alpha}$, $Det(\mathcal{D}_{tr,i}^{\alpha}) = (P_i, B_i)$, where P_i, (res. B_i) is the set of predicted poisoned (res. benign) samples x_j, in class i. Then,

$$Det(\mathcal{D}_{tr}^{\alpha}) = \{(P_i, B_i), i = 1, ..., l\}. \tag{8.3}$$

Obviously, for a non-poisoned dataset, we should have $P_i = \emptyset \ \forall i$.

Defence knowledge and capability: Alice can inspect the training dataset $\mathcal{D}_{tr}^{\alpha}$, and has white-box access to the trained model $\tilde{\mathcal{F}}_{\theta}$. Moreover, Alice can rely on a small benign validation dataset $\mathcal{D}_{be}$, that is, a small set of non-poisoned samples.

8.3 Proposed universal defence: CCA-UD

The workflow of CCA-UD is illustrated in Figure 8.2. Given a class subset $\mathcal{D}_{tr,i}^{\alpha}$ from the training dataset $\mathcal{D}_{tr}^{\alpha}$, we first extract the features of the samples from the latent space of the trained classifier and then we cluster the samples of the class in the feature space. Specifically, the method resorts to the DBSCAN algorithm to split the samples into multiple clusters $C_i^k (k = 1, ..., K_i)$ and, possibly, few outliers, not belonging to any cluster, where K_i is automatically determined by the DBSCAN algorithm. The clusters are further processed by a *poisoned cluster detection* module that, for each cluster C_i^k, computes a cluster representative point, namely the *centroid*. Then, it computes the deviation of this point from the centroid obtained on a benign cluster from the same class and checks whether such deviation contains a (strong) feature component capable to induce a misclassification to the i-th class, with a high probability (larger than a prescribed threshold), when added to the features of benign samples from the other classes. If this happens, all the samples in C_i^k are considered to be poisoned; otherwise, they are considered benign.

The outliers determined by the DBSCAN algorithm, instead, are directly considered to be poisoned. Below, we describe in detail the functionality of the two main blocks: *feature clustering* and *poisoned cluster detection*, shown in Figure 8.2.

8.3.1 Feature clustering

Inspired by the idea in [52], that the poisoned and benign samples in one class tend to group into separate clusters at the feature representation level, the proposed method consists of three steps: i) for every class i, we get the feature representations of all samples in $\mathcal{D}^{\alpha}_{tr,i}$, that is, $\{\tilde{\Phi}_\theta(x_j), x_j \in \mathcal{D}^{\alpha}_{tr,i}\}$, where we remind that $\tilde{\Phi}_\theta$ indicates the part of $\tilde{\mathcal{F}}_\theta$ that maps input sample into the latent space (the classification part of $\tilde{\mathcal{F}}_\theta$, being indicated with $\tilde{\Psi}_\theta$). We also assume that $\tilde{\Phi}_\theta$ includes a final ReLu layer so that its output is a non-negative vector $\tilde{\Phi}_\theta(x_j)$ with dimensionality d; ii) we reduce the dimension of the feature space from d to d' via Uniform Manifold Approximation and Projection (UMAP) [172]; iii) we exploit DBSCAN to split the representation set with reduced dimensionality into multiple clusters $C_i^k (k = 1, ..., K_i)$. As we mentioned, in addition to clusters, DBSCAN (may) also return a number of outliers, not belonging to any cluster. Since the outlier ratio is very small (< 0.01) for the choice of the hyperparameters made in our scheme (see Table 8.1), the samples corresponding to these outliers, forming the set O_i, are in the poisoned set P_i.

Regarding step ii), we found the use of dimensionality reduction, and UMAP in particular, to be beneficial for our scheme. It reduces the time complexity of the algorithm, and makes it independent of the original dimensionality d. Moreover, we avoid the problem of data sparsity, which tends to affect feature representations in large dimensions, and often leads to very large distances between all feature points, causing the failure of the clustering algorithm ('curse of dimensionality' problem [173]). The reduction of the dimensionality is only exploited to run the clustering algorithm.

The exact setting of the parameters of DBSCAN that we have used in our experiments, as well as the exact value of d', are discussed in Section 8.5.1.

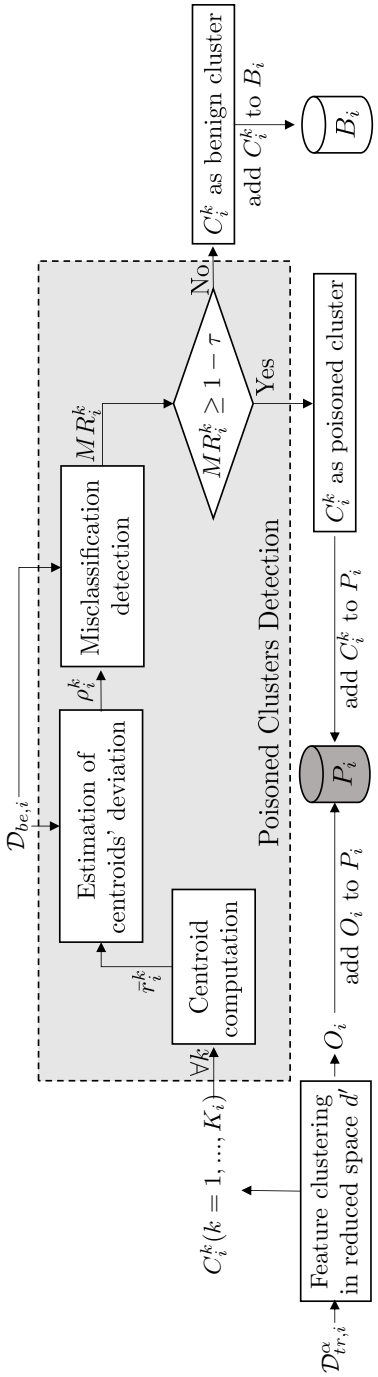


Figure 8.2: Workflow of CCA-UD.

8.3.2 Poisoned clusters detection (PCD)

To determine if a cluster C_i^k is poisoned or not, we first compute an average representation of the samples in C_i^k, i.e., the cluster's centroid, then we check whether this point contains a feature component that causes a misclassification in favour of class i when added to the features of benign samples of the other classes. More specifically, we first calculate the centroid as $\bar{r}_i^k = E[\tilde{\Phi}_\theta(x_j)|x_j \in C_i^k]$, where $E[\cdot]$ denotes the averaging operator, that performs the component-wise average on all vector positions. Vector $\bar{r}_i^k$ is a d-dim vector[1]. Then, Alice computes the following

$$\rho_i^k = \bar{r}_i^k - E[\tilde{\Phi}_\theta(x_j)|x_j \in \mathcal{D}_{be,i}] \tag{8.4}$$

where $\mathcal{D}_{be,i}$ is the i-th class of the benign set $\mathcal{D}_{be}$, and $E[\tilde{\Phi}_\theta(x_j)|x_j \in \mathcal{D}_{be,i}]$ is the average feature representation of the i-th class, say the class centroid. Therefore, ρ_i^k measures the centroids' deviation.

Finally, Alice checks if ρ_i^k can cause a misclassification error in favour of class i by adding ρ_i^k to the feature representation of the benign samples in $\mathcal{D}_{be}$ belonging to any class but the i-th one. The misclassification ratio induced by the centroids deviation of cluster C_i^k is computed as follows

$$MR_i^k = \frac{\sum_{x_j \in \mathcal{D}_{be} \setminus \mathcal{D}_{be,i}} \mathbb{1}\{\tilde{\Psi}_\theta\Big(\delta(\tilde{\Phi}_\theta(x_j) + \rho_i^k)\Big) \equiv i\}}{|\mathcal{D}_{be} \setminus \mathcal{D}_{be,i}|}, \tag{8.5}$$

where $\mathcal{D}_{be} \setminus \mathcal{D}_{be,i}$ represents the validation dataset excluding the samples from class i, that is, excluding $\mathcal{D}_{be,i}$, and δ is the ReLu operation to ensure the $\tilde{\Phi}_\theta(x_j) + \rho_i^k$ is a valid point in the latent space[2].

For a given threshold τ, if $MR_i^k \geq 1 - \tau$, the corresponding C_i^k is judged poisoned and its elements are added to P_i. Otherwise, the cluster is considered benign and its elements are added to B_i. Given that MR_i^k takes values in $[0, 1]$, the threshold τ is also chosen in this range.

An intuition of the idea behind CCA-UD, and the reason why the detection of poisoned samples works with both corrupted and non-corrupted label

[1] We remind that, although for simplicity clustering is applied in the reduced-dimension space, the subsequent analysis of the clusters is performed in the full features space.

[2] As we mentioned in Section 8.3.1, any sample from the latent space should be a positive vector

attacks, is given in the following. Let us focus first on the clean-label attack scenario. If cluster C_i^k is poisoned, the centroid $\bar{r}_i^k$ contains features of the trigger in addition to the feature of class i. Then, arguably, the deviation of this centroid from the average representation for class i is a significant one. Ideally, subtracting to $\bar{r}_i^k$ the average feature representation of i-th class, obtaining ρ_i^k, isolates the trigger features. The basic idea behind CCA-UD is that the trigger features in ρ_i^k will cause a misclassification in favour of class i, when added to the features of benign samples of the other classes.

On the contrary, if cluster C_i^k is benign, the centroid $\bar{r}_i^k$ approximates the average feature representation of the i-th class and then ρ_i^k has a very small magnitude. In this case, ρ_i^k accounts for normal intra-class fluctuation of the features and its addition to benign samples is not expected to induce a misclassification.

Similar arguments, with some noticeable differences, hold in the case of corrupted-label attacks. As before, for a benign cluster C_i^k, $\bar{r}_i^k$ approximates the average feature representation of the i-th class and then ρ_i^k corresponds to minor intra-class variations. In the case of a poisoned cluster C_i^k, the cluster now includes samples from other classes (different from i) containing the triggering signal. In this way, the cluster representative contains features of the original class in addition to the features of the triggering signal. Two cases are possible here. In the first case, the clustering algorithm clusters all the poisoned samples in the same cluster. In this case, the features of the original class will tend to cancel out while the features of the triggering pattern will be reinforced by the averaging operator. As a consequence, the deviation vector ρ_i^k will be *dominated* by the triggering features thus producing a behaviour similar to that we have described for the clean label attacks. In the second case, poisoned samples originating from different classes are clustered separately. In this case, the deviation vector will contain the the features of the triggering signal and the features related to the *difference* between the original class i and the target class t. The network, however, has been trained to *recognise* the triggering signal as a distinguishing feature of class t, hence, once again, the addition of the deviation vector to benign samples is likely to cause a misclassification in favour of class t.

The situation is pictorially illustrated in Figure 8.3 for a 3 dimension

case, in the case of a clean-label attack (a similar picture can be drawn in the corrupted-label case). Class '3' corresponds to the poisoned class. Due to the presence of the backdoor, the poisoned samples are characterised by a non-null feature component along the z direction. Due to the presence of such component, the backdoored network classifies those samples in class '3'. On the contrary, benign samples lie in the x-y plane. When it is applied to the samples labeled as class-3 sample, DBSCAN identifies two clusters, namely C_3^1 and C_3^2, where the former is a benign cluster and the latter is a poisoned cluster containing a non-null z-component. When the PCD module is applied to C_3^1 (left part in the figure), the deviation from the set of benign samples of class i (ρ_3^1) has a small amplitude and lies in the x-y plane, hence when ρ_3^1 is added to the other clusters it does not cause a misclassification error. Instead, when the PCD module is applied to C_3^2 (right part in the figure), the deviation vector (ρ_3^2) contains a significant component in the z direction, causing a misclassification when added to the benign samples in $\mathcal{D}_{be,1}$ and $\mathcal{D}_{be,2}$.

It is worth stressing that the idea behind CCA-UD indirectly exploits a known behaviour induced by backdoor attacks, that is, the fact that the presence of the triggering signal creates a kind of 'shortcut' to the target class [36]. Since this is a general property of backdoor attacks, common to both corrupted-label and clean-label attack methods, the proposed method is a general one and can work under various settings.

8.3.3 Discussion

We observe that the universality of CCA-UD essentially derives from the generality of the proposed strategy for PCD and from the use of DBSCAN, that has the following main strengths. First, differently from K-means, DBSCAN can handle class unbalancing. Then, CCA-UD also works when the poisoning ratio α small. Moreover, CCA-UD also works when the number of poisoning samples is larger than the number of benign samples. Secondly, CCA-UD also works when the class samples have large intra-variance (high variability in the feature representations). In this scenario, DBSCAN groups the benign data from the class into multiple clusters (a large K_i, $K_i > 2$, is estimated by DBSCAN), that are then detected as benign clusters. In this setting, methods

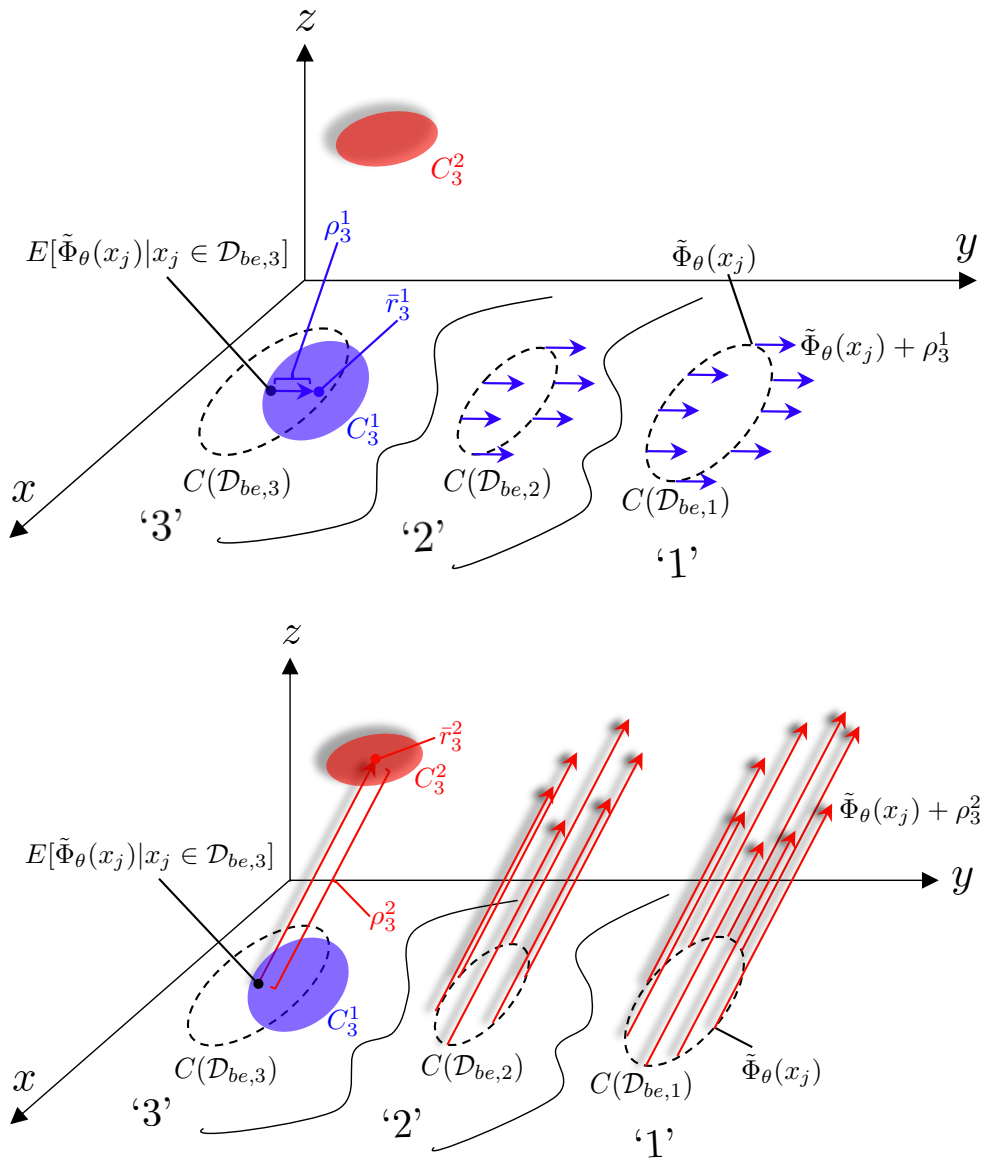


Figure 8.3: Pictorial and simplified illustration of the proposed cluster poisoning detection method. For class '3', corresponding to the poisoned class, two clusters have been identified by DBSCAN, namely C_3^1 and C_3^2, where the former is a benign cluster and the latter is a poisoned cluster. The upper and lower figures illustrate the behaviour when our method is applied to C_3^1 and C_3^2, respectively.

assuming that there are only two clusters, a benign cluster and a poisoned one, do not work.

Finally, we observe that, thanks to the fact that K_i is directly estimated by DBSCAN, in principle, our method can also work in the presence of multiple triggering signals [174, 175]. In this case, the samples poisoned via different triggers would cluster in separate clusters, that would be both detected as poisoned by our method.

8.4 Methodology

This section is devoted to a description of the methodology we followed in our experimental campaign. The specific metrics we used to measure the performance are defined in Section 8.4.1. The experimental setting, namely, the classification tasks and the attack setting is described in Section 8.4.2. Finally, the setting of the parameters of the CCA-UD is discussed in Section 8.4.3.

8.4.1 Evaluation metrics

The performance of the backdoor attacks is evaluated by providing the accuracy ACC of the backdoored model $\tilde{\mathcal{F}}_\theta$ on benign data, and also the attack success rate ASR when $\tilde{\mathcal{F}}_\theta$ is tested with poisoned data. A backdoor attack is considered successful when both ACC and ASR of the backdoored model $\tilde{\mathcal{F}}_\theta$ are greater than 90%.

To assess the performance of the defence algorithm, we measured the TPR and FPR. Actually, when i corresponds to a benign class, there are no poisoned samples in $\mathcal{D}^{\alpha}_{tr,i}$ and only the FPR is computed. More formally, let GP_i (res. GB_i) define the set of ground-truth poisoned (res. benign) samples in $\mathcal{D}^{\alpha}_{tr,i}$. We define TPR and FPR on $\mathcal{D}^{\alpha}_{tr,i}$ as follows:

$$TPR_i = TPR(\mathcal{D}^{\alpha}_{tr,i}) = \frac{|P_i \cap GP_i|}{|GP_i|}, FPR_i = FPR(\mathcal{D}^{\alpha}_{tr,i}) = 1 - \frac{|B_i \cap GB_i|}{|GB_i|}, \tag{8.6}$$

Given that benign classes may exist for both poisoned and benign datasets[3],

[3]The backdoor attack does not need to target all classes in the input domain

we need to distinguish between these two cases. Hence, we introduce the following definitions:

- Benign Class of Benign dataset (BC_B): a class of a clean dataset. In this case $\alpha = 0$ and $\mathcal{D}^{\alpha}_{tr,i}$ only includes benign samples.
- Benign Class of Poisoned dataset (BC_P): a benign class of a poisoned dataset, that is, a class in a poisoned dataset different from the target class. Also in this case, $\mathcal{D}^{\alpha}_{tr,i}$ includes only benign data.

The difference between BC_B and BC_P is that in the former case $\tilde{\mathcal{F}}_\theta$ is a clean model, while the latter is backdoored. In the following, we use $FPR_i(BC_B)$ and $FPR_i(BC_P)$ to distinguish the FPR in the two cases.

Similarly, the case of a target class t of a poisoned dataset is referred to as Poisoned Class (PC) of a poisoned dataset. In this case, $\mathcal{D}^{\alpha}_{tr,i=t}$ includes both poisoned and benign samples, then we compute and report $TPR_t(PC)$ and $FPR_t(PC)$. TPR and FPR depend on the choice of the threshold τ. Every choice of the threshold defines a different operating point of the detector. In order to get a global view of the performance of the tested systems, we provide the AUC_t value ranging in the $[0, 1]$ interval. The higher the AUC_t the better the capability of the system to distinguish poisoned and benign samples. When $AUC_t = 1$ we have a perfect detector, while $AUC_t = 0.5$ corresponds to a random detector. In our experiments, we report the AUC_t of the PC case only, because in the BC_B and BC_P cases the true positive rate cannot be measured.

According to the definitions in Equation (8.6), the false positive and true positive rates are computed for each class. For sake of simplicity, we will often report average values. For the case of benign class of a benign dataset, the average value denoted by $FPR(BC_B)$, is calculated by averaging over all the classes of the benign training dataset. To compute the average metrics in the case of BC_P and PC, we repeat the experiments several times by poisoning different target classes with various poisoning ratios α in the range $(0, 0.55]$ for every target class, and by using the poisoned datasets to train the backdoored models[4]. Then, the quantity $FPR(BC_P)$ is computed by averaging the per-

[4]Only successful backdoor attacks are considered to measure the performance in the various cases.

formance achieved on non-target classes from all poisoned training datasets. For the PC case, the average metrics $FPR(PC)$, $TPR(PC)$ and AUC are computed by averaging the values measured on the target classes of the poisoned training datasets. We also measured the average performance achieved for a fixed poisoned ratio α, by varying only the target class t. When we want to stress the dependency of a metric on the poisoning ratio α, we respectively add a subscript to the metrics as follows: $FPR_\alpha(BC_P)$, $FPR_\alpha(PC)$, $TPR_\alpha(PC)$, AUC_α.

The tests run to set the detection threshold τ are carried out on the validation dataset, consisting only of benign samples. Therefore, for each class $\mathcal{D}_{be,i}$, we can only calculate the $FPR(\mathcal{D}_{be,i})$, and its average counterpart is denoted as $FPR(\mathcal{D}_{be}) = \sum_i FPR(\mathcal{D}_{be,i})/C$.

8.4.2 Network tasks and attacks

We consider three different classification tasks, namely MNIST, traffic sign, and fashion clothes classification.

MNIST

In this set of experiments we trained a model to classify the digits in the MNIST dataset [43], which includes $n = 10$ digits (classes) with 6000 binary images per class. The size of the images is 28×28. The architecture used for the task is a 4-layer network [176]. The feature representation of dimensionality 128 is obtained from the input of the final Fully-connected (FC) layer.

Regarding the attack setting, three different backdoor attacks have been considered, as detailed below. For each setting, the training dataset is poisoned by considering 16 poisoning ratios α chosen in $(0, 0.55]$. For each α, 10 different poisoned training datasets are generated by choosing different classes as the target class.

- Corrupted-label attack, with 3-by-3-pixel trigger (abbrev. *3-by-3 corrupted*): the backdoor is injected by adding a 3-by-3 pixel pattern to the corrupted samples, as shown in Figure 8.1, and modifying the sample labels into that of the target class.

- Corrupted-label attack, with ramp trigger (abbrev. *ramp corrupted*): Eve performs a corrupted-label backdoor attack using a ramp pattern [76] as the trigger, as shown in Figure 8.1. The ramp triggering signal is defined as $v(i,j) = j\Delta/W$, $1 \leq i \leq H$, $1 \leq j \leq W$, where $H \times W$ is the size of the image and Δ is a parameter controlling the slope (the strength) of the ramp. We set $\Delta = 40$.

- Clean-label attack, with 3-by-3-pixel trigger (abbrev. *3-by-3 clean*): the attack utilises the 3-by-3 pixel trigger pattern to perform a clean-label attack.

Traffic signs

For the traffic sign classification task, we selected 16 different classes from the GTSRB dataset [49], including 6 speed-limit, 3 prohibition, 3 danger, and 4 mandatory signs. Each class consists of 1200 colour images with size $28 \times 28 \times 3$. The model architecture used for training is based on ResNet18 [30]. The feature representation is extracted from the 17-th layer, that is, before the FC layer, after an average pooling layer and ReLu activation.

With regard to the attack, we considered the corrupted-label scenario. As triggering signal, we considered a horizontal sinusoidal pattern, defined as $v(i,j) = \Delta \sin(2\pi j/TW)$, $1 \leq i \leq H$, $1 \leq j \leq W$, where $H \times W$ is the size of input image. The parameters Δ and T are used to control the strength and period of the trigger. In our experiment, we set $\Delta = 20$ and $T = 1/6$. As before, for a given α, the network is trained on 16 poisoned datasets, each time considering a different target class.

Fashion clothes

Fashion-MNIST dataset [177] includes 10 classes of grey-level cloth images, each class consisting of 6000 images of size 28×28. The model architecture used for classification is based on AlexNet [164]. The representation used by the backdoor detector is extracted from the 5-th layer, at the output of the ReLu activation layer before the first FC layer.

With regard to the attack, the poisoned samples are generated by performing the attack in a clean-label setting. A ramp signal with $\Delta = 256$ is

used as the triggering signal. Once again, for each choice of α, the backdoor attack is repeated 10 times, each time considering a different target class.

For all the classification tasks, the benign validation dataset $\mathcal{D}_{be}$ is obtained by randomly selecting 100 samples from all the classes in the dataset.

8.4.3 Setting of defence parameters

To implement the CCA-UD method, we have to set the following parameters: i) the reduced dimension d' used for the clustering, ii) the parameters of the DBSCAN algorithm, namely $minPts$ and ϵ, and finally iii) the threshold τ used by the clustering poisoning detection module. In our experiments, we set $d' = 2$, $minPts = 20$ and $\epsilon = 0.8$. This is the setting that, according to our experiments, achieves the best performance with the minimum complexity for the clustering algorithm (being $d' = 2$). The effect of these parameters on the clustering result and the detection performance is evaluated by the ablation study reported in Section 8.5.1.

With regard to τ, as mentioned before, AC, CI and CCA-UD all involve the setting of a threshold for poisoning detection. For a fair comparison, we set the threshold in the same way for all the methods. In particular, we set τ by fixing the false positive rate. In general a value of τ results in different FPR rates for different classes. To avoid setting a different threshold for each class, then, we fixed it by setting the average FPR. In fact, setting the average FPR exactly may not be feasible, so we chose the threshold in such a way to minimise the distance from the target rate. Formally, by setting the target false positive rate to 0.05, the threshold τ^* for each method is determined as:

$$\tau^* = \arg\min_{\tau} \left| 0.05 - FPR(\mathcal{D}_{be}) \right|. \tag{8.7}$$

8.5 Results

In this section we report the results of the experiments we have carried out to evaluate the effectiveness of CCA-UD.

8.5.1 Ablation study

We start the experimental analysis with an ablation study investigating the effect of the three main hyperparameters CCA-UD, namely d' (regarding UMAP), $minPts$ and ϵ (for DBSCAN) on the effectiveness of the method. Based on this analysis, in all subsequent experiments, we set $d' = 2$, $minPts = 20$ and $\epsilon = 0.8$.

The influence of each parameter on the clustering result and the detection performance can be assessed by looking at Table 8.1. The results refer to the case of MNIST classification, with backdoor poisoning performed by using a 3-by-3 pixel trigger pattern and label corruption. Similar considerations can be drawn in the other settings. The results in the table have been obtained by letting $\tau = \tau^\star$ as stated in Equation (8.7). To start with, we observe that when utilising τ^* in BC_B, BC_P cases, the FPR value is close to 0.05 for all the settings, while in the PC case FPR is close to or less than 0.05 for all settings except for S9 and S16, when benign and poisoned samples collapse into a single cluster. In addition to TPR and FPR, the table shows the average number of clusters $\overline{K}$ and the average outlier ratio $\overline{\zeta}$ identified by DBSCAN.

From the first group of rows (S1-S4), we see that for a given setting of $minPts$ and ϵ, increasing d' leads to a larger average number of clusters and a larger fraction of outliers, as the DBSCAN algorithm results in a higher number of densely-connected regions. A similar behaviour is observed by increasing $minPts$ or decreasing ϵ for a given d' (second and third group of rows in the table). Expectedly, when ϵ is too large, e.g. 10, DBSCAN always results in one cluster thus failing to identify the poisoned samples. Based on the results in Table 8.1, the setting S7 ($d' = 2$, $minPts = 20$, $\epsilon = 0.8$) and S15 ($d' = 10$, $minPts = 20$, $\epsilon = 3$) yield the best performance, the former having lower computational complexity, because of the lower dimension used to cluster the samples in the features space ($d' = 2$ instead of 10).

8.5.2 Threshold setting

The thresholds τ^* obtained following the approach detailed in Section 8.4.3 for AC and CI and CCA-UD, are reported in Table 8.2 for the three different classification tasks considered in our experiments. Given that the threshold is

Table 8.1: Ablation study on the three hyperparameters of our algorithm. The FPR and TPR values on BC_B, BC_P and PC cases are computed at τ^*, while $\overline{K}$ and $\overline{\zeta}$ is the cluster number and outlier ratio generated in the feature clustering process.

	Hyperparameters			BC_B results		BC_P results		PC results			
	d'	$minPts$	ϵ	$(\overline{K}, \overline{\zeta})$	$FPR(BC_B)$	$(\overline{K}, \overline{\zeta})$	$FPR(BC_P)$	$(\overline{K}, \overline{\zeta})$	$TPR(PC)$	$FPR(PC)$	AUC
S1	2	20	0.4	(2.9, 0.005)	0.050	(4.3, 0.008)	0.073	(9.7, 0.003)	1.000	0.046	0.998
S2	4	20	0.4	(30.4, 0.097)	0.044	(22.6, 0.060)	0.027	(12.9, 0.012)	0.432	0.006	0.989
S3	8	20	0.4	(37.4, 0.142)	0.066	(23.7, 0.076)	0.037	(13.4, 0.012)	0.448	0.007	0.990
S4	10	20	0.4	(39.3, 0.153)	0.057	(24.5, 0.085)	0.049	(13.8, 0.013)	0.501	0.010	0.987
S5	2	3	0.4	(2.0, 0.000)	0.050	(2.2, 0.000)	0.051	(8.0, 0.000)	1.000	0.050	1.000
S6	2	10	0.4	(2.3, 0.001)	0.050	(2.6, 0.002)	0.050	(8.5, 0.001)	1.000	0.050	0.999
S7	2	20	0.8	(1.3, 0.000)	0.050	(1.6, 0.000)	0.050	(6.2, 0.000)	1.000	0.050	1.000
S8	2	20	1.0	(1.3, 0.000)	0.049	(1.6, 0.000)	0.050	(4.6, 0.000)	1.000	0.049	1.000
S9	2	20	10.0	(1.0, 0.000)	0.050	(1.0, 0.000)	0.050	(1.0, 0.000)	1.000	1.000	0.500
S10	10	5	0.4	(15.5, 0.004)	0.049	(9.5, 0.002)	0.068	(11.9, 0.001)	1.000	0.046	0.999
S11	10	10	0.4	(17.8, 0.020)	0.052	(11.7, 0.012)	0.077	(10.6, 0.004)	1.000	0.030	0.996
S12	10	20	0.2	(29.2, 0.883)	0.049	(60.7, 0.732)	0.045	(111.3, 0.399)	0.053	0.031	0.612
S13	10	20	0.6	(2.0, 0.008)	0.046	(3.0, 0.004)	0.042	(7.6, 0.001)	1.000	0.042	0.999
S14	10	20	1.0	(1.2, 0.000)	0.050	(1.5, 0.000)	0.050	(6.2, 0.000)	1.000	0.049	1.000
S15	10	20	3.0	(1.1, 0.000)	0.050	(1.5, 0.000)	0.050	(3.9, 0.000)	1.000	0.050	1.000
S16	10	20	10.0	(1.0, 0.000)	0.050	(1.0, 0.000)	0.050	(1.0, 0.000)	1.000	1.000	0.500

Table 8.2: Values of τ^* obtained for the various classification tasks.

Method	**Handwritten digits**	**Traffic signs**	**Fashion clothes**
AC	0.335	0.404	0.301
CI	3.018	1.673	4.738
CCA-UD	0.950	0.950	0.950

set by relying on the validation dataset, it is necessary to verify that the target false positive rate (0.05 in our case) is also obtained on the test dataset. An excerpt of such results is shown in Table 8.4 (a similar behaviour is observed for the other classification tasks).

Our experiments reveal that, for AC and CI, the threshold determined via Equation (8.7) does not lead to a good operating point. In particular, while for CCA-UD, the threshold τ^* set on the validation dataset yields a similar FPR (around 0.05) in the BC_B, BC_P and PC cases, this is not true for AC and CI, for which $FPR(BC_B)$, $FPR(BC_P)$ and $FPR(PC)$ are often smaller than 0.05, reaching 0 in many cases. This leads to a poor $TPR(PC)$. In particular, with the AC method, when $\alpha > \tau^*$, both clusters are classified as benign, and then $TPR_\alpha(PC) = FPR_\alpha(PC) = 0$, even when the method would, in principle, be able to provide a perfect discrimination ($AUC_\alpha \approx 1$). The difficulty in setting the threshold for the AC and CI method is also evident from the plots in Figure 8.4-8.6, that report the FPR and TPR values averaged also on α, for different values of the threshold τ. From these plots, we immediately see that a threshold that works in all the cases can never be found for AC and CI.

Due to the difficulties encountered to set the detection threshold for AC and CI[5] the results at τ^* for these method are not reported in the other cases, that is, for traffic sign and fashion clothes, for which we report only the AUC_α scores. Note that the possibility to set a unique threshold is very important for the practical applicability of a defence. Based on our results, CCA-UD has this remarkable property.

[5]Note that the problem of threshold setting is not addressed in the original papers, since different thresholds are used in the various cases.

Table 8.3: *AUC* scores of three methods in the three different attacks

Method	**3-by-3 corrupted**	**Ramp corrupted**	**3-by-3 clean**
AC	0.728	0.733	0.785
CI	0.964	0.178	0.488
CCA-UD	0.994	0.996	0.981

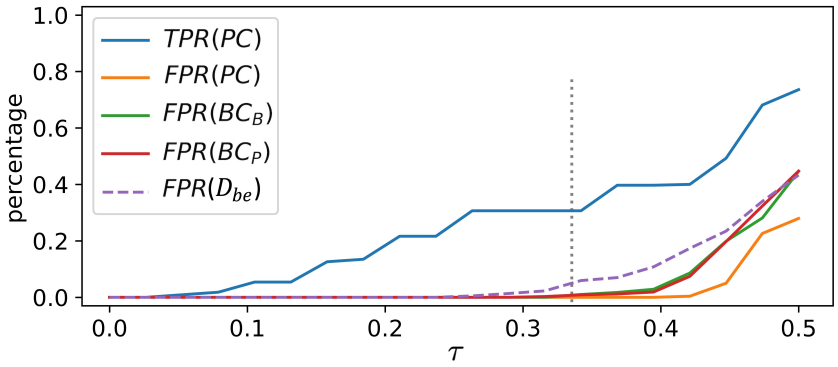
8.5.3 Results on MNIST

In this section, we evaluate the performance of CCA-UD against the three types of backdoor attacks, namely, *3-by-3 corrupted*, *ramp corrupted*, and *3-by-3 clean*. Such performance as compared to those obtained by AC and CI. In Figure 8.4-8.6, in each row, the three figures report the average performance of AC, CI and CCA-UD. The values of $FPR(BC_B)$, $FPR(BC_P)$, $TPR(PC)$ and $FPR(PC)$ are reported for each method, as a function of the detection threshold τ. The behaviour of $FPR(\mathcal{D}_{be})$, which is utilised to determine the threshold τ^* (at 0.05 of $FPR(\mathcal{D}_{be})$), is also reported. The position of τ^* is indicated by a vertical dotted line.

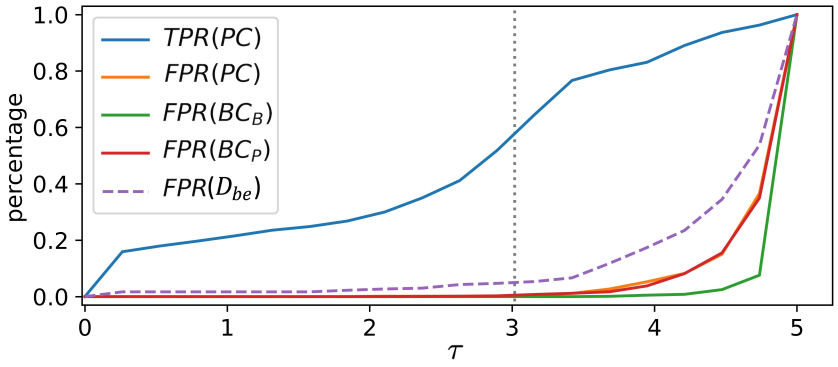
By observing the figure, we see that CCA-UD outperforms by far the other two methods in all settings. In the first setting, we achieve $TPR(PC)$ and $FPR(PC)$ equal to 0.983 and 0.051 at the optimal threshold τ^*, with $FPR(BC_B) = 0.051$ and $FPR(BC_P) = 0.050$. Instead, the performance achieved by AC and CI at their optimal threshold is very poor. Similar results are achieved for the second and third settings. In particular, in the second attack, CCA-UD achieves $TPR(PC)$ and $FPR(PC)$ equal to (0.978, 0.050) at τ^*, and (0.966, 0.063) for the third one.

For a poisoned dataset, the *AUC* values in the three settings are provided in Table 8.3. From these results, we argue that CI provides good discrimination performance (with an AUC only slightly lower than CCA-UD) against the first attack, but fails to defend against the other two attacks. This is an expected behaviour since CI does not work when the triggering signal is robust against the average filter, as in the case of the ramp signal considered in the second attack, or with clean-label attacks, as in last setting.

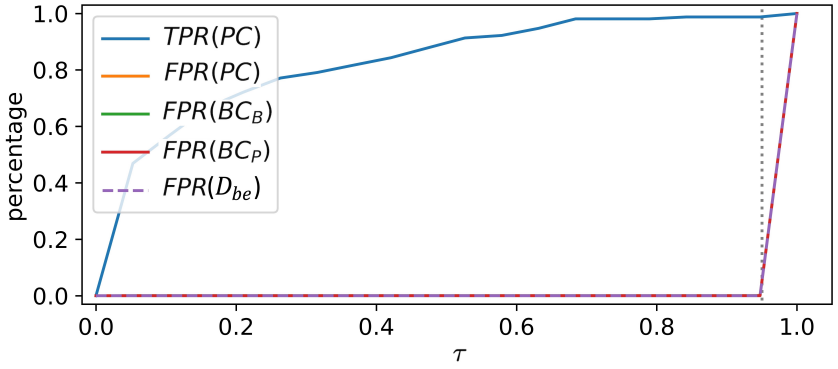
Table 8.4 shows the results obtained for different values of the poisoning ratio α for the three different attacks against handwritten digits classification.



(a) AC

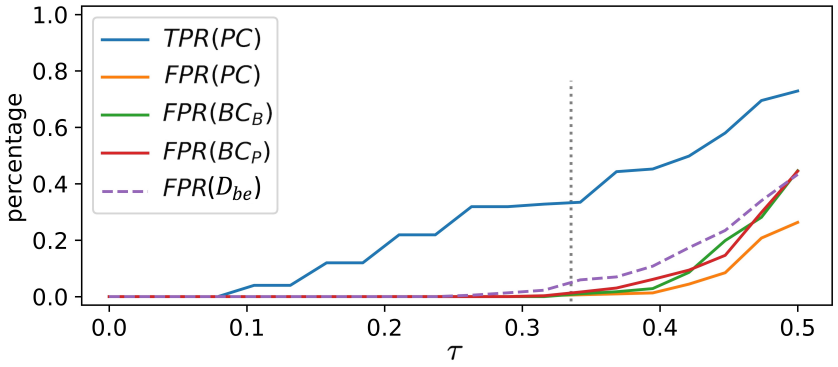


(b) CI

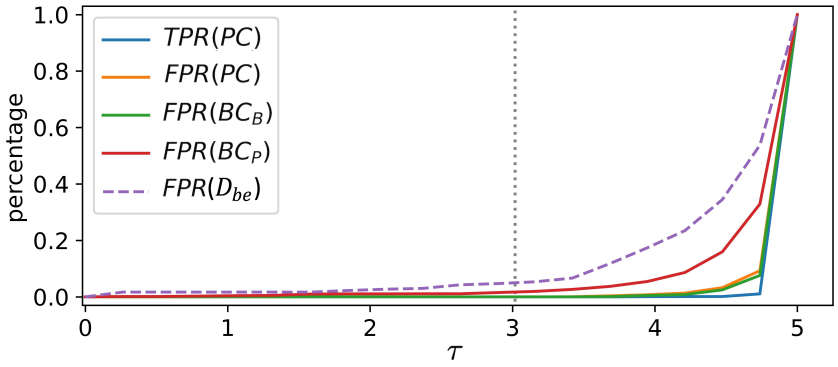


(c) CCA-UD

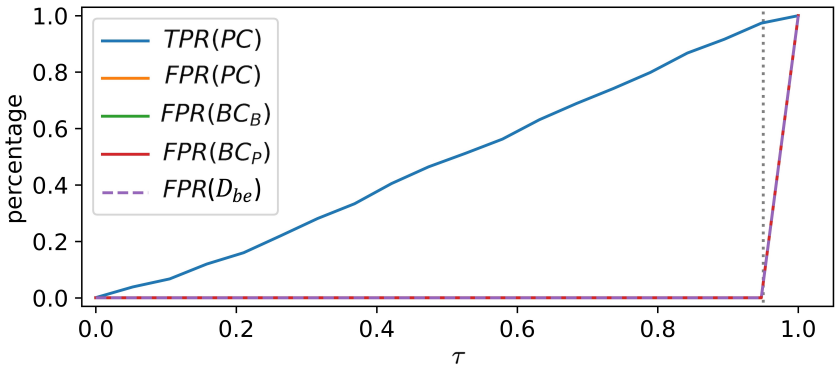
Figure 8.4: Average performance of AC and CI, and CCA-UD for different values of the threshold against the 3-by-3 corrupted backdoor attacks for handwritten digits classification. The position of τ^* is indicated by a vertical dotted line.



(a) AC

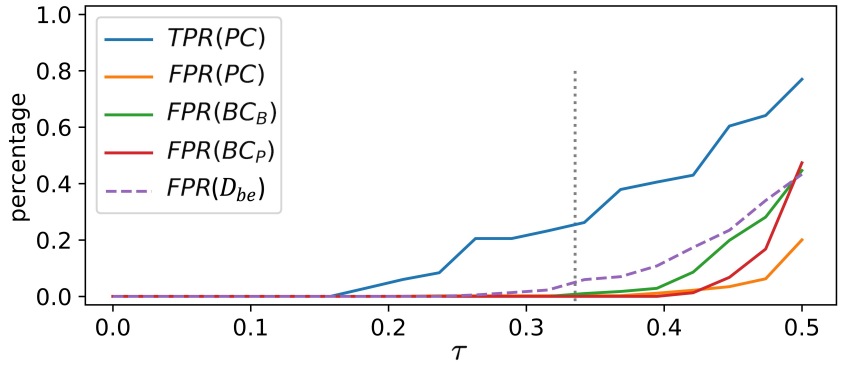


(b) CI

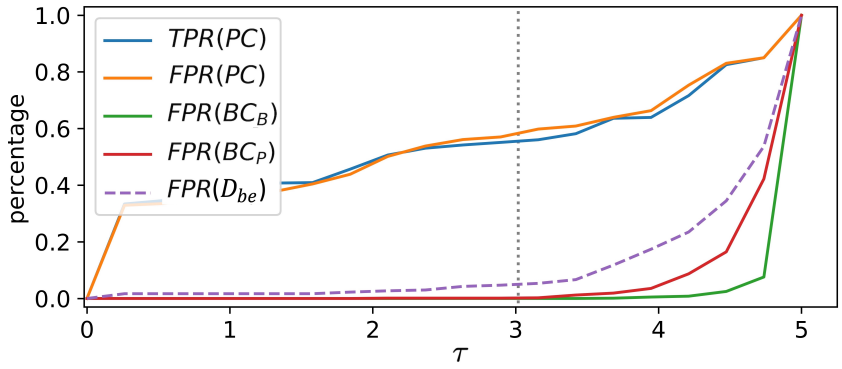


(c) CCA-UD

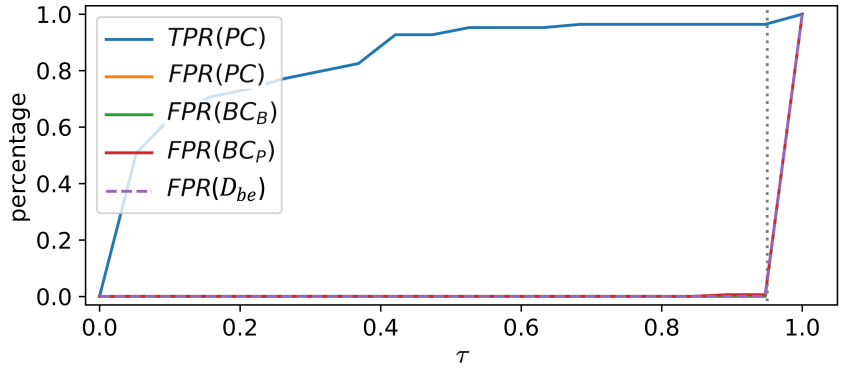
Figure 8.5: Average performance of AC and CI, and CCA-UD for different values of the threshold against the ramp corrupted backdoor attacks for handwritten digits classification. The position of τ^* is indicated by a vertical dotted line.



(a) AC



(b) CI



(c) CCA-UD

Figure 8.6: Average performance of AC and CI, and CCA-UD for different values of the threshold against the 3-by-3 clean backdoor attacks for handwritten digits classification. The position of τ^* is indicated by a vertical dotted line.

The values of FPR and TPR have been obtained by letting $\tau = \tau^*$.

For the clean-label case, due to the difficulty of developing a successful attack [76,85,178], the backdoor can be successfully injected in the model only when α is large enough and, in any case, a successful attack could not always be obtained in 10 repetitions. For this reason, in the third table, we report the number of successfully attacked classes (cnt) with different poisoning ratios. Upon inspection of Table 8.4, we observe that:

- With regard to AC, the behaviour is similar under the three attack scenarios. Good results are achieved for intermediate values of α, e.g. in the range $[0.2, 0.3]$. When $\alpha < 0.134$, instead, AUC_α of AC is smaller than 0.786 and close to 0.5 for small α. In particular, AC cannot handle the backdoor attack when the poisoning ratio is smaller than 0.1. Moreover, when $\alpha > 0.5$, AUC_α goes to zero, as benign samples are classified as poisoned and vice-versa. Finally, by comparing the AUC_α values in Table 8.4a and Table 8.4c, we see that AC achieves better performance against the corrupted-label attack than in the clean-label case.

- With regard to CI, the detection performance achieved in the first attack scenario (3-by-3 corrupted) is good for all the values of α, with AUC_α larger than 0.96 in most of the cases (with the exception of the case with very small α, namely, $\alpha = 0.025$, in which case $AUC_\alpha = 0.876$), showing that the CI method can effectively defend against the backdoor attack in this setting, for every attack poisoning ratio. However, as expected, CI fails in the other settings, with a AUC_α lower than 0.5 in all the cases, confirming the limitations mentioned in Section 8.1.1.

- Regarding CCA-UD, good results are achieved in all the cases and for every α value, with a perfect or nearly perfect AUC_α in most of the cases. Moreover, by letting $\tau = \tau^*$, a very good $TPR_\alpha(PC)$ is obtained, larger than 0.95 in most of the cases, with $FPR_\alpha(BC_P)$ and $FPR_\alpha(PC)$ around 0.05. Overall, the tables prove the universality of CCA-UD that works very well regardless of the specific attack setting and regardless of the value of α. We observe that, since CCA-UD achieves a larger AUC_α than AC and CI, CCA-UD outperforms AC and CI not only when $\tau = \tau^*$ but also when τ is set adaptively.

Finally, these results show that CCA-UD can effectively defend against both corrupted- and clean-label attacks, thus confirming that the strategy used to detect poisoning clusters exploits a general misclassification behaviour presenting in both the corrupted- and clean-label attacks.

8.5.4 Results on traffic signs

Figure 8.7 shows the average performance of AC, CI, and CCA-UD on the traffic signs task. Similar considerations to the MNIST case can be made. CCA-UD achieves very good average performance at the operating point given by τ^*, where $TPR(PC)$ and $FPR(PC)$ are (0.954, 0.085) (with $FPR(BC_B) = FPR(BC_B) \approx 0.08$), while, for the AC and CI, a threshold that works on average can not be found. In the case of the poisoned dataset, the average AUC of the detection AUC is equal to 0.897, 0.958, 0.993 for AC, CI, and CCA-UD, respectively.

We observe that CI also got a good AUC. In fact, in this case, given that the size of the input image is 28×28, the triggering signal, namely the sinusoidal signal can be effectively removed by the 5×5 average filter.

The results obtained for various α are reported in Table 8.5. As it can be seen, CCA-UD gets very good performance in terms of $TPR_\alpha(PC)$ and FPR_α (PC) measured at $\tau = \tau^*$ in all the cases. The AUC_α is also larger than that achieved by AC and CI for all values of α. As observed before, while CI is relatively insensitive to α, the performance of AC drop when $\alpha < 0.1$ or $\alpha > 0.5$.

8.5.5 Results on fashion clothes

Figure 8.8 reports the results obtained by AC, CI, and CCA-UD on the fashion clothes task. Once again, the performance achieved by CCA-UD is greatly superior with respect to those achieved by AC and CI. In particular, by looking at Figure 8.8, CCA-UD achieves $TPR(PC)$ and $FPR(PC)$ equal to (1.000, 0.050), with a $FPR(BC_B) = FPR(BC_P) \approx 0.05$. Regarding the AUC scores, AUC of AC, CI, and CCA-UD are 0.900, 0.106, and 0.997, respectively. Since the attack is carried out in a clean-label manner, the poor performance of CI were expected. The results for various α reported in Table 8.6 confirm the

Table 8.4: Performance of AC, CI and CCA-UD for various poisoning ratios α, against the three types of backdoor attacks for MNIST classification, The FPR and TPR values are computed at $\tau = \tau^*$.

	AC				**CI**				**CCA-UD**			
α	$FPR_\alpha(BC_P)$	$TPR_\alpha(PC)$	$FPR_\alpha(PC)$	AUC_α	$FPR_\alpha(BC_P)$	$TPR_\alpha(PC)$	$FPR_\alpha(PC)$	AUC_α	$FPR_\alpha(BC_P)$	$TPR_\alpha(PC)$	$FPR_\alpha(PC)$	AUC_α
0.025	0.025	0.000	0.000	0.563	0.012	0.324	0.022	0.876	0.050	0.908	0.051	0.949
0.050	0.055	0.099	0.000	0.628	0.005	0.581	0.001	0.977	0.050	0.989	0.050	0.994
0.096	0.000	0.395	0.000	0.757	0.005	0.654	0.000	0.996	0.050	0.999	0.050	0.999
0.134	0.000	0.792	0.000	0.958	0.009	0.559	0.002	0.990	0.051	0.999	0.050	1.000
0.186	0.000	0.994	0.000	0.997	0.000	0.577	0.001	0.985	0.050	1.000	0.050	1.000
0.258	0.000	0.993	0.000	0.997	0.014	0.540	0.070	0.961	0.050	1.000	0.050	1.000
0.359	0.000	0.000	0.000	0.998	0.000	0.571	0.005	0.964	0.050	1.000	0.050	1.000
0.550	0.000	0.000	0.000	0.001	0.000	0.829	0.000	0.953	0.050	1.000	0.050	1.000

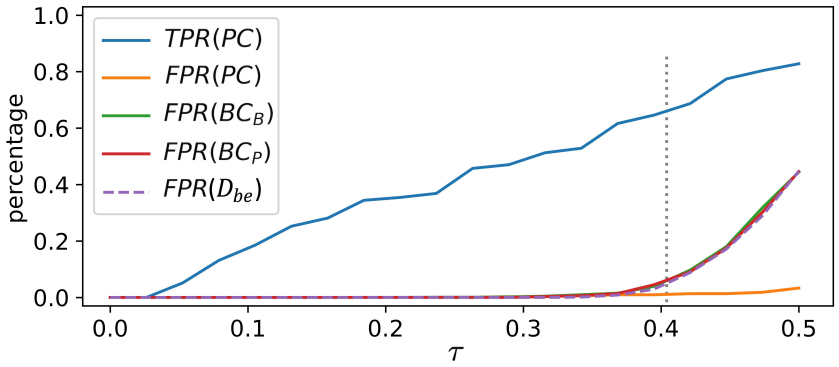
(a) 3-by-3 corrupted

	AC				**CI**				**CCA-UD**			
α	$FPR_\alpha(BC_P)$	$TPR_\alpha(PC)$	$FPR_\alpha(PC)$	AUC_α	$FPR_\alpha(BC_P)$	$TPR_\alpha(PC)$	$FPR_\alpha(PC)$	AUC_α	$FPR_\alpha(BC_P)$	$TPR_\alpha(PC)$	$FPR_\alpha(PC)$	AUC_α
0.035	0.000	0.050	0.024	0.593	0.009	0.000	0.008	0.407	0.051	0.871	0.050	0.966
0.050	0.024	0.090	0.028	0.593	0.000	0.000	0.000	0.119	0.050	0.914	0.050	0.998
0.096	0.000	0.400	0.000	0.786	0.003	0.000	0.000	0.216	0.050	0.989	0.050	0.998
0.134	0.024	0.798	0.001	0.962	0.019	0.000	0.000	0.142	0.050	0.999	0.050	0.998
0.186	0.000	0.992	0.003	0.995	0.107	0.000	0.000	0.179	0.051	1.000	0.050	1.000
0.258	0.025	0.999	0.000	0.999	0.000	0.000	0.000	0.088	0.050	1.000	0.050	1.000
0.359	0.025	0.000	0.000	0.999	0.021	0.000	0.000	0.144	0.051	1.000	0.050	1.000
0.550	0.000	0.000	0.000	0.002	0.004	0.000	0.000	0.135	0.050	1.000	0.050	1.000

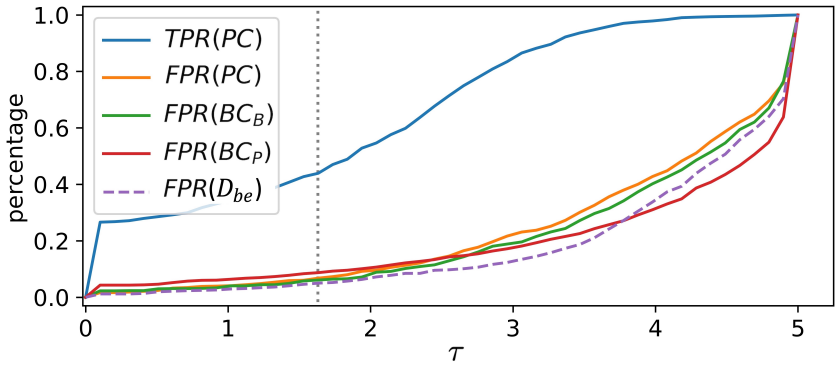
(b) Ramp corrupted

		AC				**CI**				**CCA-UD**			
α	cnt	$FPR_\alpha(BC_P)$	$TPR_\alpha(PC)$	$FPR_\alpha(PC)$	AUC_α	$FPR_\alpha(BC_P)$	$TPR_\alpha(PC)$	$FPR_\alpha(PC)$	AUC_α	$FPR_\alpha(BC_P)$	$TPR_\alpha(PC)$	$FPR_\alpha(PC)$	AUC_α
0.050	2	0.000	0.000	0.000	0.441	0.000	0.683	0.835	0.438	0.051	0.642	0.050	0.809
0.069	3	0.000	0.000	0.000	0.533	0.000	0.667	0.667	0.296	0.050	0.952	0.050	0.972
0.096	3	0.000	0.000	0.000	0.528	0.000	0.333	0.333	0.595	0.050	0.951	0.050	0.972
0.134	3	0.000	0.000	0.000	0.610	0.000	0.667	0.667	0.539	0.050	0.975	0.050	0.987
0.186	5	0.000	0.384	0.003	0.746	0.000	0.600	0.600	0.471	0.051	0.982	0.050	0.991
0.258	5	0.000	0.929	0.011	0.959	0.000	0.601	0.644	0.516	0.050	0.994	0.051	0.996
0.359	5	0.000	0.315	0.000	0.975	0.000	0.206	0.213	0.437	0.050	0.993	0.050	0.996
0.450	5	0.000	0.000	0.000	0.969	0.009	0.729	0.786	0.554	0.050	0.997	0.050	0.998

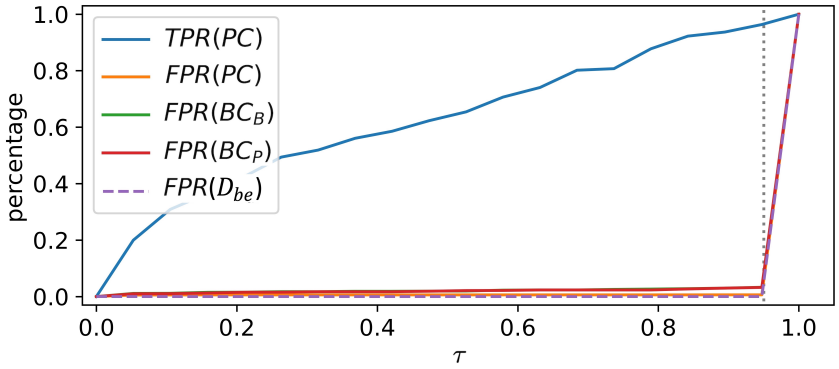
(c) 3-by-3 clean



(a) AC



(b) CI



(c) CCA-UD

Figure 8.7: Average performance of AC, CI, and CCA-UD for different values of τ for the traffic signs task. The vertical dotted line indicates the position of τ^* for the various methods.

Table 8.5: Performance of AC, CI, and CCA-UD for various poisoning ratios for the traffic sign task. The FPR and TPR values are computed at $\tau = \tau^*$. For AC and CI, due to the difficulties in threshold setting (see Section 8.5.2), we only report AUC.

		AC	**CI**	**CCA-UD**			
α	cnt	AUC_α	AUC_α	AUC_α	$FPR_\alpha(BC_P)$	$TPR_\alpha(PC)$	$FPR_\alpha(PC)$
0.050	9	0.793	0.923	0.983	0.073	0.946	0.061
0.096	9	0.850	0.928	0.991	0.058	0.998	0.059
0.134	9	0.949	0.959	0.992	0.057	0.998	0.057
0.186	10	0.958	0.965	0.993	0.064	0.999	0.056
0.359	13	0.946	0.965	0.996	0.086	0.985	0.054
0.450	14	0.917	0.965	0.994	0.070	0.980	0.055
0.550	15	0.869	0.996	0.999	0.059	0.999	0.051

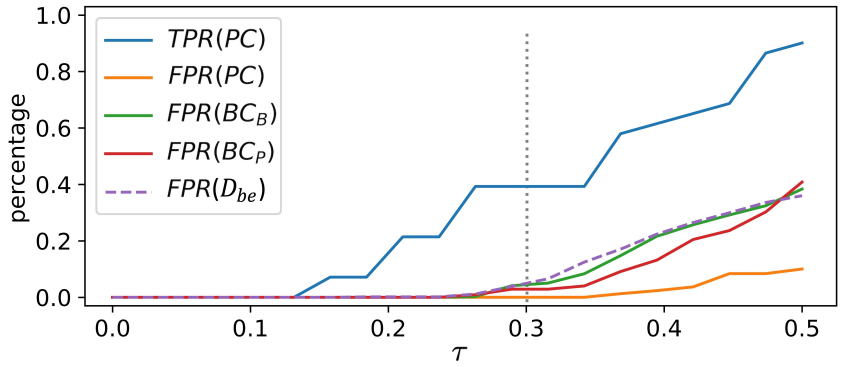
Table 8.6: Performance of AC, CI, and CCA-UD for various poisoning ratios for the fashion clothes task. The FPR and TPR values are computed at $\tau = \tau^*$. For AC and CI, due to the difficulties in threshold setting (see Section 8.5.2), we only report AUC.

		AC	**CI**	**CCA-UD**			
α	cnt	AUC_α	AUC_α	AUC_α	$FPR_\alpha(BC_P)$	$TPR_\alpha(PC)$	$FPR_\alpha(PC)$
0.069	3	0.618	0.056	0.998	0.053	1.000	0.052
0.096	3	0.513	0.341	0.995	0.054	1.000	0.056
0.134	3	0.940	0.087	0.998	0.059	1.000	0.053
0.186	4	1.000	0.037	0.998	0.054	1.000	0.055
0.258	5	1.000	0.083	0.996	0.055	1.000	0.057
0.359	5	1.000	0.015	0.998	0.056	1.000	0.052
0.450	5	1.000	0.174	1.000	0.055	1.000	0.050

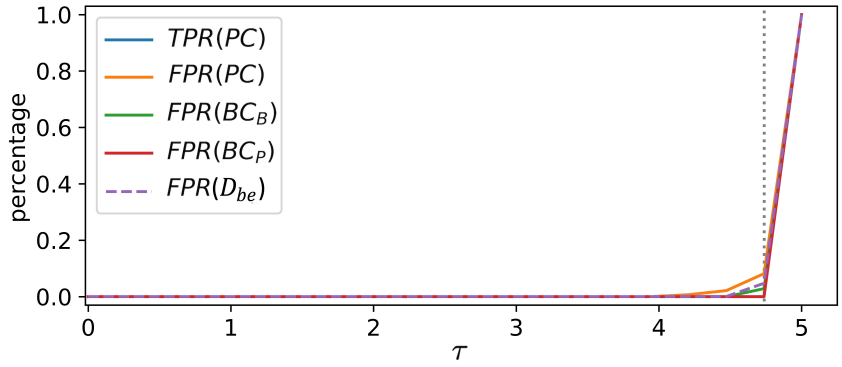
same behaviour, with CCA-UD getting very good performance in all the cases always overcoming the other two methods.

8.6 Generalisation Analysis

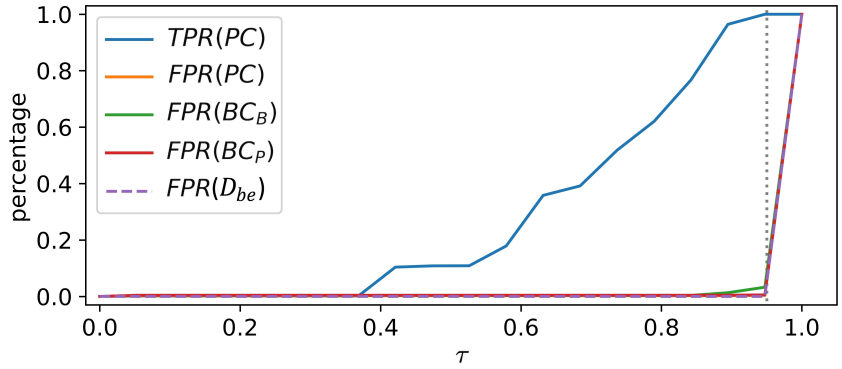
In order to investigate the generality of our defence, we evaluated the performance of CCA-UD against two additional tasks (CIFAR10 classification



(a) AC



(b) CI



(c) CCA-UD

Figure 8.8: Average performance of AC, CI, and CCA-UD for different values of τ for the fashion clothes task. The vertical dotted line indicates the position of τ^*.

and face recognition) involving more complex architectures and more realistic datasets. Moreover, for each task, we implement two types of backdoor attacks.

8.6.1 CIFAR10 classification

CIFAR10 dataset [50] consists of 60000 images belonging to 10 classes (6000 images per class). The dataset is split into two parts: 50000 images for training and 10000 for testing. The image size is 32×32. The model architecture is based on VGG19 [31], and the feature representation is extracted from the final convolutional (16th) layer after the pooling layer and the flatten operation. We considered two types of backdoor attack against this network:

- Sample-specific attack: we adopted the warping-based trigger [55] shown in Figure 8.9, performing an all-to-one corrupted-label attack, that is, warping images that do not belong to the target class and mislabeling them as belonging to the target class. Moreover, the attacker also injects into the training dataset noise samples, which are warped randomly and labeled correctly. The number of noise samples is twice the number of poisoned samples.

- Source-specific attack: using the 3×3 pixel pattern shown in Figure 8.1, the attacker chooses and poisons the samples from a specific source class. Poisoning is carried out in a corrupted-label setting. Then, at test time, only samples from the source class with the trigger lead to a misclassification.

For the two types of backdoor attacks, we set five different poisoned ratios ranging from 0.096 to 0.45, and evaluated the capability of CCA-UD and the existing to detect the poisoned samples is a contaminated class. The corresponding AUC for the two types of attack are shown in Figure 8.10a and Figure 8.10b, respectively.

With regard to the sample-specific attack, we can observe: i) CCA-UD achieves the best performance with $AUC \approx 1$; ii) AC achieves good results when α is large, but its performance drop when α is small. This is expected since AC is limited to deal with imbalanced clusters (when $\alpha = 0.096$); iii) CI provides the worst performance, since the average filter cannot remove the

Figure 8.9: From left to right, the original image, the warped poisoned image, and the difference (trigger) between the warped and original images. For visualisation, the difference is scaled to span the $[0, 1]$ range.

warping trigger. With regard to the source-specific case, all three methods work well with $\alpha \geq 0.186$, while the performance of AC decrease more rapidly than for the other two when $\alpha \leq 0.134$.

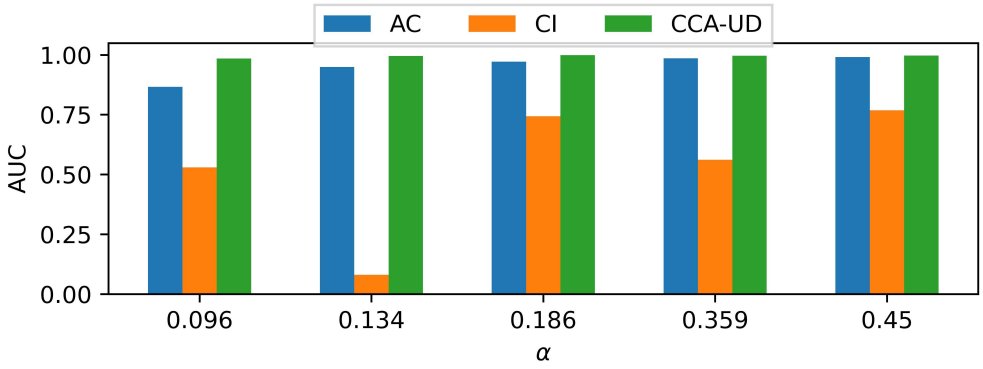
Finally, by adopting the threshold estimation procedure described in Section 8.4.3, we can still find a universal threshold θ^* for this task. With it we can calculate the average $\overline{TPR}_\alpha(PC)$ and $\overline{FPR}_\alpha(PC)$ that result to be equal to 0.996 and 0.002, with an average $\overline{FPR}_\alpha(BC_P) \approx 0$ over different values of α.

8.6.2 Face recognition task

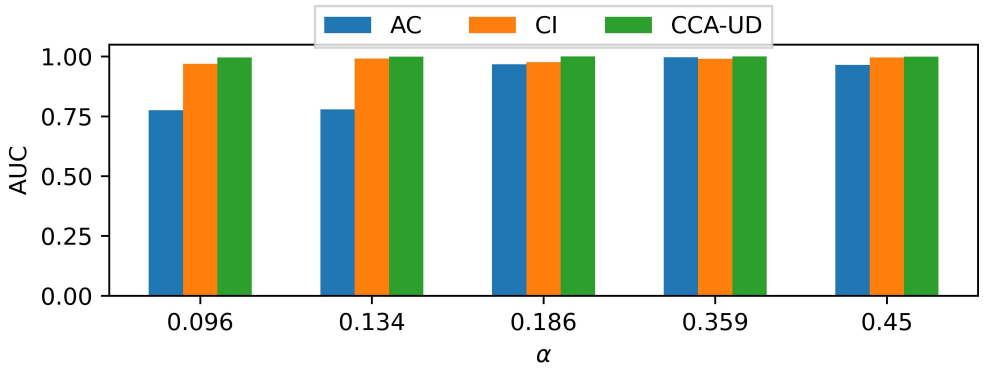
For the face recognition task, we selected 12 classes from YoutubeFace dataset [46]. Each class contains more than 2600 images, and the whole dataset is split for training and testing with ratio 9:1. The image size is 315×315. The attacked architecture is based on the Inception-Resnet-v1 [32], and the representations are extracted from the layer before the first FC layer (last 2nd layer). The intra-class variability of this dataset is pretty large, as it can be seen by the feature distribution (after UMAP dimension reduction) of class 'Peter Goldenmark' shown in Figure 8.11. In this figure, the representations of facial image of 'Peter Goldenmark' make up four clusters, and their misclassification ratios are close to zero so that they are classified as benign clusters.

We considered two types of backdoor attacks for this task:

- 30×30 corrupted: The triggering signal is based the same pattern



(a) Sample-specific



(b) Source-specific

Figure 8.10: Performance against two types of backdoor attacks in CIFAR10, we show the *AUC* of AC, CI, and CCA-UD, which detect the poisoned samples from one contaminated class with poisoned ratio α from 0.096 to 0.45.

shown in Figure 8.1. However, due to the big input size, the pattern is enlarged 10 times with a final size of 30×30. Poisoning is applied in corrupted-label modality.

- Multi-backdoor: in this case, the attacker injects three backdoors into the same model. Specifically, the attacker chooses three different target classes, and associates each one with a specific triggering signal, as shown in Figure 8.12. At test time, each triggering signal will mislead the model to different target class. For each backdoor, the poisoning setting is the same as the previous case.

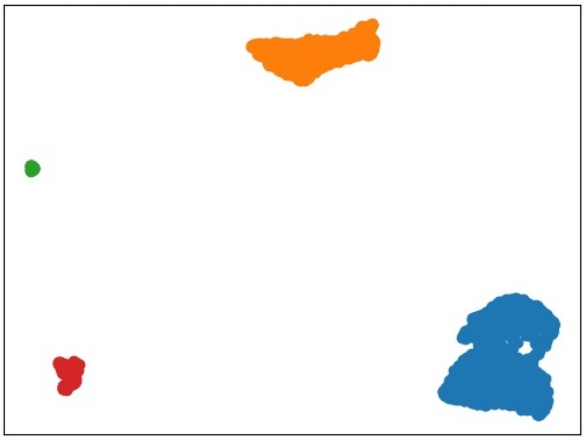


Figure 8.11: Feature distribution of class 'Peter Goldenmark' of YoutubeFace dataset after UMAP dimension reduction.

Figure 8.12: Three different backdoors are activated with different triggers (from left to right, the trigger is located in the right-bottom, right-top, and left-bottom corner). For each trigger, they mislead the model to different target classes.

For the 30×30 corrupted-label attack (recognition task), we used CCA-UD and other works to detect the backdoor injected with different α's ranging from 0.05 to 0.55, and evaluateed the performance on detecting the poisoned samples from a contaminated class. The AUC values are shown in Figure 8.13. From the table, we can observe that: i) the performance of AC drop a little bit when α is small, since due its limitation with imbalanced cluster; ii) CI still works even when the average filter cannot fully remove the trigger (trigger size is 30×30 and average filter kernel is 5×5). This means that the trigger pattern isn't robust to average filtering; iii) CCA-UD achieve the

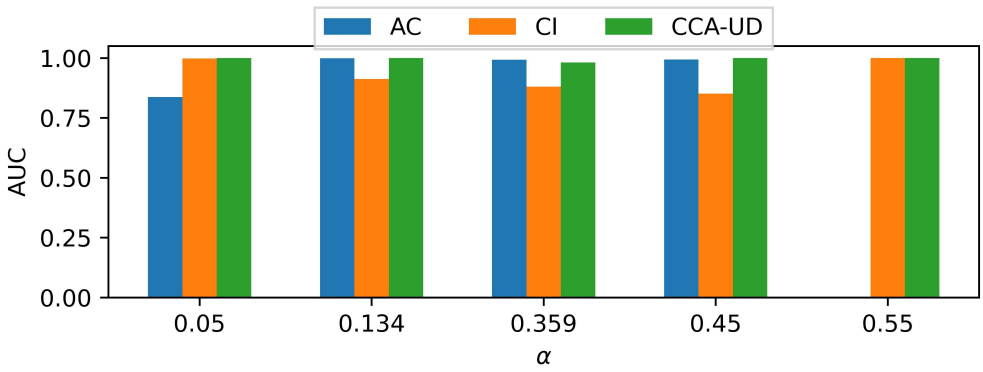


Figure 8.13: *AUC* of CCA-UD and other works on detecting the poisoned samples from one contaminated class of 30×30 corrupted attack on face recognition task with poisoned ratio α from 0.05 to 0.55.

best performance; iv) AC drops to zero when $\alpha = 0.55$, due to the use of the relative size of the clusters to identify the poisoned samples.

For the multi-backdoor attack, the poisoning ratio α is set to 0.45 for all the three backdoor. We evaluated the three defences for the three backdoors, and found that all of them achieve an average $AUC \approx 0.99$ for the three target classes.

For CCA-UD, we calculated the universal threshold $\theta^* = 0.95$ as described in Section 8.4.3, achieving an average $\overline{TPR}_\alpha(PC)$ and $\overline{FPR}_\alpha(PC)$ equal 0.993 and 0, and an average $\overline{FPR}_\alpha(BC_P) \approx 0$ over different α values for both 30×30 corrupted and multi-backdoor attacks.

8.7 Summary

In this chapter, we evaluated the effectiveness of CCA-UD in a wide variety of classification tasks and backdoor attack settings. The experimental results confirm that the method can work regardless of the corruption strategy (corrupted and clean label) and the type of trigger used by the attacker (local or global trigger). Moreover, the method is effective regardless of the poisoning ratio used by the attacker, which can be either very small (very few percentage points) or large. Furthermore, we proved that the performance achieved by CCA-UD is always superior to those achieved by the existing

methods, even when these methods are adopted in scenarios that meet their (restrictive) operating conditions.

As a strength of CCA-UD, for every classification task, a detection threshold can be set on a benign dataset that works against all the backdoor attacks, that is, regardless of the specific attack setting (poisoning strategy, poisoning rate). This property is very important for the practical applicability of the method. We also stress that, although in our experimental campaign we focus on image classification tasks, the proposed universal defence is a very general one and can be exploited to defend against backdoor attacks in other fields and application scenarios fields, besides image classification.

Chapter 9

Conclusion

In this chapter, we summarise the main contributions of the thesis and present some ideas for future work in this field.

9.1 Summary and final remarks

When we started our research activity, a flurry of methods and solutions had been published in the field of backdoor attacks, with only few and scattered attempts to systematically categorise them. As a result, newly emerging works claimed an advantage by comparing themselves with state-of-the-art methods designed for different application scenarios and working conditions. Therefore, we found it necessary to clearly define the threat models and propose a unifying taxonomy to classify the attacks and defences proposed so far, in order to highlight their suitability to different application scenarios, and more easily compare them.

Motivated by the above, in the first part of the thesis, we first identified and defined two different threat models, based on the capability of the attacker: i) *full control attacks*, where the adversary corresponds to the model's trainer, and then is able to control the whole training process; ii) *partial control attacks*, where the adversary can only interfere with the data collection process.

Then, we systematically reviewed the backdoor attacks, specifying the control scenario under which they can operate and their limitations. Specifically, the backdoor attacks are grouped into: i) *corrupted-label attacks*, typically carried out in the full control scenario, wherein the attacker can tamper with the labels of the poisoned samples; and ii) *clean-label attacks*, more suitable for the partial control scenario, according to which the attacker can not change or define the labels of the poisoned samples.

We then reviewed the existing works on backdoor defences, proposing a new taxonomy to categorise them based on the operating level: *data-level*, *model-level* and *training-dataset-level*. The defences operating at the first two levels can be applied in both the full control and partial control scenarios, while the defences working at last level can only be applied in the partial control scenario, where the defender has access to the training data.

In the second part of the thesis, we focused on a particular application scenario, that is biometric face recognition, and proposed two backdoor attacks that are suitable for two different threat models. We first proposed a so-called MasterFace backdoor attack to inject a backdoor into a face verification model, which aims to judge whether two face images belong to the same individual or not. The full control scenario is considered for this attack. The MasterFace backdoor does not impair the normal face verification task of the model, and permits the attacker to impersonate *any* enrolled user when the network is fed with a key input. This backdoor attack is also exploited for the design of a black-box watermarking scheme, to protect the intellectual property of DNN models for face verification. Finally, we proposed a stealthy clean-label backdoor attack to inject the backdoor into a video rebroadcast detector. This attack in this case is carried out in the challenging partial control scenario, that imposes the use of an imperceptible temporal triggering signal.

In the third part of the thesis, we took the role of the defender and proposed a universal training-dataset-level defence, called CCA-UD against backdoor attacks, that can work in a wide variety of attack settings and classification scenarios. The detection is based on a general misclassification behaviour that is induced by the presence of the backdoor in the feature space, both in the case of corrupted- and clean-label attacks.

CCA-UD is valid defence against both the corrupted- and clean-label attacks, regardless of the trigger's shape, size and visibility. Moreover, the defence is also effective when the attacker poisons a very small fraction of the training data. The performance achieved by CCA-UD is always superior to those achieved by the existing methods, in the scenarios that meet their

operating conditions.

9.2 Open issues

Notwithstanding the number of works published so far and the progresses made in this thesis, there are several open issues that still remain to be addressed, the most relevant of which are detailed in the following.

- *Improving the robustness of backdoors.* The development of strategies to improve backdoor robustness is an important research line that should occupy the agenda of researchers. Current approaches can resist, up to some extent, parameter pruning and fine-tuning of final layers, while robustness against retraining of all layers and, more in general, transfer learning, is often not at the reach of current techniques, the method proposed in this thesis for face verification being an exception. Achieving such robustness is particularly relevant when backdoors are used for DNN watermarking. The study of backdoor attacks in the physical domain is another interesting, yet rather unexplored, research direction, calling for the development of backdoor attacks that can survive the analog to digital conversion involved by physical domain applications.

- *Development of an underlying theory.* We ambitiously advocate the need for an underlying theory that can help to solve some of the fundamental problems behind the development of backdoor attacks, like, for instance, the definition of an *optimal* triggering signal. From the defender's side, a theoretical framework can help the development of more general defences that are effective under a given threat model.

- *Design of general defences.* Existing defences are often tailored solutions that work well only under very specific assumptions about the behaviour of the adversary, e.g. on the triggering signal and its size. In real-life applications, however, these assumptions do not necessarily hold. Therefore, the development of more general defences, that can work in a wide variety of attack settings are desirable. In this thesis, we contributed to the above mission, with a universal defence method for the case of defences working at training-dataset-level. A similar effort

needs to be made for defences operating at data- and model-level with the development of universal solutions, with minimal working assumptions on the attacker's behaviour.

Bibliography

[1] M. Helmstaedter, K. L. Briggman, S. C. Turaga, V. Jain, H. S. Seung, and W. Denk, "Connectomic reconstruction of the inner plexiform layer in the mouse retina," *Nature*, vol. 500, no. 7461, p. 168, 2013.

[2] H. Y. Xiong, B. Alipanahi, L. J. Lee, H. Bretschneider, D. Merico, R. K. Yuen, Y. Hua, S. Gueroussov, H. S. Najafabadi, T. R. Hughes, and others, "The human splicing code reveals new insights into the genetic determinants of disease," *Science*, vol. 347, no. 6218, p. 1254806, 2015.

[3] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik, "Deep neural nets as a method for quantitative structure-activity relationships," *J. Chem. Inf. Model.*, vol. 55, no. 2, pp. 263–274, 2015.

[4] T. Ciodaro, D. Deva, J. De Seixas, and D. Damazio, "Online particle detection with neural networks based on topological calorimetry information," in *Journal of physics: conference series*, vol. 368. IOP Publishing, 2012, p. 012030.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," pp. 1106–1114, 2012.

[6] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[7] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, 2014, pp. 3104–3112.

[8] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III*, ser. Lecture Notes in Computer Science, H. Blockeel, K. Kersting, S. Nijssen, and F. Zelezný, Eds., vol. 8190. Springer, 2013, pp. 387–402.

[9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[11] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[12] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 2574–2582.

[13] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*. IEEE Computer Society, 2017, pp. 39–57.

[14] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 2142–2151.

[15] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognit.*, vol. 84, pp. 317–331, 2018.

[16] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proceedings of the 29th International Coference on International*

Conference on Machine Learning, ser. ICML'12. Omnipress, 2012, pp. 1467–1474.

[17] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "BadNets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.

[18] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2017, Dallas, TX, USA, November 3, 2017*. ACM, 2017, pp. 27–38.

[19] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *CoRR*, vol. abs/1712.05526, 2017.

[20] L. Muñoz-González, B. Pfitzner, M. Russo, J. Carnerero-Cano, and E. C. Lupu, "Poisoning attacks with generative adversarial nets," *CoRR*, vol. abs/1906.07773, 2019.

[21] P. W. Koh, J. Steinhardt, and P. Liang, "Stronger data poisoning attacks break data sanitization defenses," *Mach. Learn.*, vol. 111, no. 1, pp. 1–47, 2022.

[22] Y. Li, Y. Li, Y. Lv, Y. Jiang, and S. Xia, "Hidden backdoor attack against semantic segmentation models," *CoRR*, vol. abs/2103.04038, 2021.

[23] J. Dai, C. Chen, and Y. Li, "A backdoor attack against lstm-based text classification systems," *IEEE Access*, vol. 7, pp. 138 872–138 878, 2019.

[24] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, ser. Proceedings of Machine Learning Research, vol. 108. PMLR, 2020, pp. 2938–2948.

[25] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012.

[26] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2017, Dallas, TX, USA, November 3, 2017*, B. Thuraisingham, B. Biggio, D. M. Freeman, B. Miller, and A. Sinha, Eds. ACM, 2017, pp. 27–38.

[27] A. E. Cinà, K. Grosse, A. Demontis, S. Vascon, W. Zellinger, B. A. Moser, A. Oprea, B. Biggio, M. Pelillo, and F. Roli, "Wild patterns reloaded: A survey of machine learning security against training data poisoning," *CoRR*, vol. abs/2205.01992, 2022.

[28] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778.

[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[32] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. AAAI Press, 2017, pp. 4278–4284.

[33] J. Kirkpatrick, R. Pascanu, N. C. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *CoRR*, vol. abs/1612.00796, 2016.

[34] J. Dumford and W. J. Scheirer, "Backdooring convolutional neural networks via targeted weight perturbations," in *2020 IEEE International Joint Conference on Biometrics, IJCB 2020, Houston, TX, USA, September 28 - October 1, 2020*. IEEE, 2020, pp. 1–9.

[35] R. Costales, C. Mao, R. Norwitz, B. Kim, and J. Yang, "Live trojan attacks on deep neural networks," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*. Computer Vision Foundation / IEEE, 2020, pp. 3460–3469.

[36] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. IEEE, 2019, pp. 707–723.

[37] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 2018, pp. 6106–6116.

[38] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, 2006.

[39] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 1211–1220.

[40] J. Bai, B. Wu, Y. Zhang, Y. Li, Z. Li, and S. Xia, "Targeted attack against deep neural networks via flipping limited weight bits," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[41] S. Hong, N. Carlini, and A. Kurakin, "Handcrafted backdoors in deep neural networks," *CoRR*, vol. abs/2106.04690, 2021.

[42] Y. Li, J. Hua, H. Wang, C. Chen, and Y. Liu, "DeepPayload: Black-box backdoor attack on deep learning models through neural payload injection," in *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*. IEEE, 2021, pp. 263–274.

[43] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[44] H. Kwon, "Multi-model selective backdoor attack with different trigger positions," *IEICE Transactions on Information and Systems*, vol. 105, no. 1, pp. 170–174, 2022.

[45] Y. Liu, Y. Xie, and A. Srivastava, "Neural trojans," in *2017 IEEE International Conference on Computer Design, ICCD 2017, Boston, MA, USA, November 5-8, 2017*. IEEE Computer Society, 2017, pp. 45–48.

[46] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," in *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*, 2011, pp. 529–534.

[47] H. Zhong, C. Liao, A. C. Squicciarini, S. Zhu, and D. J. Miller, "Backdoor embedding in convolutional neural network models via invisible perturbation,"

in *CODASPY '20: Tenth ACM Conference on Data and Application Security and Privacy, New Orleans, LA, USA, March 16-18, 2020*. ACM, 2020, pp. 97–108.

[48] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 86–94.

[49] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The german traffic sign detection benchmark," in *The 2013 International Joint Conference on Neural Networks, IJCNN 2013, Dallas, TX, USA, August 4-9, 2013*. IEEE, 2013, pp. 1–8.

[50] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," pp. 32–33, 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[51] Q. Zhang, Y. Ding, Y. Tian, J. Guo, M. Yuan, and Y. Jiang, "AdvDoor: adversarial backdoor attack of deep learning system," in *ISSTA '21: 30th ACM SIGSOFT International Symposium on Software Testing and Analysis, Virtual Event, Denmark, July 11-17, 2021*. ACM, 2021, pp. 127–138.

[52] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," in *Workshop on Artificial Intelligence Safety 2019 co-located with the Thirty-Third AAAI Conference on Artificial Intelligence 2019 (AAAI-19), Honolulu, Hawaii, January 27, 2019*, vol. 2301, 2019.

[53] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 2018, pp. 8011–8021.

[54] S. Li, M. Xue, B. Z. H. Zhao, H. Zhu, and X. Zhang, "Invisible backdoor attacks on deep neural networks via steganography and regularization," *IEEE Trans. Dependable Secur. Comput.*, vol. 18, no. 5, pp. 2088–2105, 2021.

[55] T. A. Nguyen and A. T. Tran, "WaNet - imperceptible warping-based backdoor attack," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[56] F. L. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 6, pp. 567–585, 1989.

[57] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 2015, pp. 3730–3738.

[58] E. Quiring and K. Rieck, "Backdooring and poisoning neural networks with image-scaling attacks," in *2020 IEEE Security and Privacy Workshops, SP Workshops, San Francisco, CA, USA, May 21, 2020*. IEEE, 2020, pp. 41–47.

[59] Q. Xiao, Y. Chen, C. Shen, Y. Chen, and K. Li, "Seeing is not believing: Camouflage attacks on image scaling algorithms," in *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*. USENIX Association, 2019, pp. 443–460.

[60] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*. ACM, 2019, pp. 2041–2055.

[61] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *Proceedings of the British Machine Vision Conference 2015, BMVC 2015, Swansea, UK, September 7-10, 2015*. BMVA Press, 2015, pp. 41.1–41.12.

[62] "CASIA iris dataset." [Online]. Available: http://biometrics.idealtest.org/

[63] T. J. L. Tan and R. Shokri, "Bypassing backdoor detection algorithms in deep learning," in *IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7-11, 2020*. IEEE, 2020, pp. 175–183.

[64] Y. Li, T. Zhai, Y. Jiang, Z. Li, and S. Xia, "Backdoor attack in the physical world," *CoRR*, vol. abs/2104.02361, 2021.

[65] X. Gong, Y. Chen, Q. Wang, H. Huang, L. Meng, C. Shen, and Q. Zhang, "Defense-resistant backdoor attacks against deep neural networks in outsourced cloud environment," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2617–2631, 2021.

[66] S. Cheng, Y. Liu, S. Ma, and X. Zhang, "Deep feature space trojan attack of neural networks by controlled detoxification," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 1148–1156.

[67] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 2242–2251.

[68] Y. Liu, W. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "ABS: scanning neural networks for back-doors by artificial brain stimulation," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*. ACM, 2019, pp. 1265–1282.

[69] S. Kolouri, A. Saha, H. Pirsiavash, and H. Hoffmann, "Universal litmus patterns: Revealing backdoor attacks in CNNs," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 2020, pp. 298–307.

[70] T. A. Nguyen and A. T. Tran, "Input-aware dynamic backdoor attack," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[71] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *Research in Attacks, Intrusions, and Defenses - 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings*, ser. Lecture Notes in Computer Science, vol. 11050. Springer, 2018, pp. 273–294.

[72] P. Zhao, P. Chen, P. Das, K. N. Ramamurthy, and X. Lin, "Bridging mode connectivity in loss landscapes and adversarial robustness," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[73] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "STRIP: a defence against trojan attacks on deep neural networks," in *Proceedings of the 35th Annual Computer Security Applications Conference, ACSAC 2019, San Juan, PR, USA, December 09-13, 2019*. ACM, 2019, pp. 113–125.

[74] A. Turner, D. Tsipras, and A. Madry, "Label-consistent backdoor attacks," *CoRR*, vol. abs/1912.02771, 2019.

[75] M. Alberti, V. Pondenkandath, M. WÃŒrsch, M. Bouillon, M. Seuret, R. Ingold, and M. Liwicki, "Are you tampering with my data?" in *Computer Vision - ECCV 2018 Workshops - Munich, Germany, September 8-14, 2018, Proceedings, Part II*, 2018, pp. 296–312.

[76] M. Barni, K. Kallas, and B. Tondi, "A new backdoor attack in CNNs by training set corruption without label poisoning," in *2019 IEEE International Conference on Image Processing, ICIP 2019, Taipei, Taiwan, September 22-25, 2019*, 2019, pp. 101–105.

[77] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part X*, ser. Lecture Notes in Computer Science, vol. 12355. Springer, 2020, pp. 182–199.

[78] R. Wan, B. Shi, L. Duan, A. Tan, and A. C. Kot, "Benchmarking single-image reflection removal algorithms," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 3942–3950.

[79] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. IEEE Computer Society, 2009, pp. 248–255.

[80] R. Ning, J. Li, C. Xin, and H. Wu, "Invisible poison: A blackbox clean label backdoor attack to deep neural networks," in *40th IEEE Conference on Computer Communications, INFOCOM 2021, Vancouver, BC, Canada, May 10-13, 2021*. IEEE, 2021, pp. 1–10.

[81] O. Suciu, R. Marginean, Y. Kaya, H. D. III, and T. Dumitras, "When does machine learning fail? generalized transferability for evasion and poisoning attacks," in *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*. USENIX Association, 2018, pp. 1299–1316.

[82] C. Zhu, W. R. Huang, H. Li, G. Taylor, C. Studer, and T. Goldstein, "Transferable clean-label poisoning attacks on deep neural nets," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 2019, pp. 7614–7623.

[83] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 11 957–11 965.

[84] J. Chen, L. Zhang, H. Zheng, X. Wang, and Z. Ming, "Deeppoison: Feature transfer based stealthy poisoning attack for dnns," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 68, no. 7, pp. 2618–2622, 2021.

[85] S. Zhao, X. Ma, X. Zheng, J. Bailey, J. Chen, and Y. Jiang, "Clean-label backdoor attacks on video recognition models," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 2020, pp. 14 431–14 440.

[86] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," *CoRR*, vol. abs/1212.0402, 2012.

[87] C. Weng, Y. Lee, and S. Wu, "On the trade-off between adversarial and backdoor robustness," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[88] E. Chou, F. Tramèr, and G. Pellegrino, "SentiNet: Detecting localized universal attacks against deep learning systems," in *2020 IEEE Security and Privacy Workshops, SP Workshops, San Francisco, CA, USA, May 21, 2020*. IEEE, 2020, pp. 48–54.

[89] B. G. Doan, E. Abbasnejad, and D. C. Ranasinghe, "Februus: Input purification defense against trojan attacks on deep neural network systems," in *ACSAC '20: Annual Computer Security Applications Conference, Virtual Event / Austin, TX, USA, 7-11 December, 2020*. ACM, 2020, pp. 897–912.

[90] E. Sarkar, Y. Alkindi, and M. Maniatakos, "Backdoor suppression in neural networks using input fuzzing and majority voting," *IEEE Des. Test*, vol. 37, no. 2, pp. 103–110, 2020.

[91] H. Kwon, "Detecting backdoor attacks via class difference in deep neural networks," *IEEE Access*, vol. 8, pp. 191 049–191 056, 2020.

[92] H. Fu, A. K. Veldanda, P. Krishnamurthy, S. Garg, and F. Khorrami, "Detecting backdoors in neural networks using novel feature-based anomaly detection," *CoRR*, vol. abs/2011.02526, 2020.

[93] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 5767–5777.

[94] H. Kwon, "Defending deep neural networks against backdoor attack by using de-trigger autoencoder," *IEEE Access*, 2021. [Online]. Available: http://doi:10.1109/ACCESS.2021.3086529

[95] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 93–104.

[96] A. K. Veldanda, K. Liu, B. Tan, P. Krishnamurthy, F. Khorrami, R. Karri, B. Dolan-Gavitt, and S. Garg, "NNoculation: Broad spectrum and targeted treatment of backdoored dnns," *CoRR*, vol. abs/2002.08313, 2020.

[97] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "DeepInspect: A black-box trojan detection and mitigation framework for deep neural networks," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. ijcai.org, 2019, pp. 4658–4664.

[98] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, "Detecting AI trojans using meta neural analysis," in *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. IEEE, 2021, pp. 103–120.

[99] M. Villarreal-Vasquez and B. K. Bhargava, "ConFoc: Content-focus protection against trojan attacks on neural networks," *CoRR*, vol. abs/2007.00711, 2020.

[100] H. Qiu, Y. Zeng, S. Guo, T. Zhang, M. Qiu, and B. Thuraisingham, "DeepSweep: An evaluation framework for mitigating DNN backdoor attacks using data augmentation," in *ASIA CCS '21: ACM Asia Conference on Computer and Communications Security, Virtual Event, Hong Kong, June 7-11, 2021*. ACM, 2021, pp. 363–377.

[101] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 2414–2423.

[102] L. Truong, C. Jones, B. Hutchinson, A. August, B. Praggastis, R. Jasper, N. Nichols, and A. Tuor, "Systematic evaluation of backdoor data poisoning attacks on image classifiers," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*. Computer Vision Foundation / IEEE, 2020, pp. 3422–3431.

[103] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, "Neural attention distillation: Erasing backdoor triggers from deep neural networks," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[104] T. Garipov, P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson, "Loss surfaces, mode connectivity, and fast ensembling of dnns," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 2018, pp. 8803–8812.

[105] S. C. Park and H. Shin, "Polygonal chain intersection," *Comput. Graph.*, vol. 26, no. 2, pp. 341–350, 2002.

[106] R. T. Farouki, "The bernstein polynomial basis: A centennial retrospective," *Comput. Aided Geom. Des.*, vol. 29, no. 6, pp. 379–419, 2012.

[107] F. R. Hampel, "The influence curve and its role in robust estimation," *Journal of the american statistical association*, vol. 69, no. 346, pp. 383–393, 1974.

[108] W. Guo, L. Wang, X. Xing, M. Du, and D. Song, "TABOR: A highly accurate approach to inspecting and restoring trojan backdoors in AI systems," *CoRR*, vol. abs/1908.01763, 2019.

[109] Z. Xiang, D. J. Miller, and G. Kesidis, "Detection of backdoors in trained classifiers without access to the training set," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 33, no. 3, pp. 1177–1191, 2022.

[110] Z. Xiang, D. J. Miller, H. Wang, and G. Kesidis, "Detecting scene-plausible perceptible backdoors in trained dnns without access to the training set," *Neural Comput.*, vol. 33, no. 5, pp. 1329–1371, 2021.

[111] R. Wang, G. Zhang, S. Liu, P. Chen, J. Xiong, and M. Wang, "Practical detection of trojan neural networks: Data-limited and data-free cases," in *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIII*, vol. 12368. Springer, 2020, pp. 222–238.

[112] L. Zhu, R. Ning, C. Wang, C. Xin, and H. Wu, "GangSweep: Sweep out neural backdoors by GAN," in *MM '20: The 28th ACM International Conference on Multimedia, Virtual Event / Seattle, WA, USA, October 12-16, 2020*. ACM, 2020, pp. 3173–3181.

[113] X. Qiao, Y. Yang, and H. Li, "Defending neural backdoors via generative distribution modeling," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 2019, pp. 14 004–14 013.

[114] Z. Xiang, D. J. Miller, and G. Kesidis, "A benchmark study of backdoor data poisoning defenses for deep neural network classifiers and A novel defense," in *29th IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2019, Pittsburgh, PA, USA, October 13-16, 2019*. IEEE, 2019, pp. 1–6.

[115] N. Peri, N. Gupta, W. R. Huang, L. Fowl, C. Zhu, S. Feizi, T. Goldstein, and J. P. Dickerson, "Deep k-nn defense against clean-label data poisoning attacks," in *Computer Vision - ECCV 2020 Workshops - Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, ser. Lecture Notes in Computer Science, vol. 12535. Springer, 2020, pp. 55–70.

[116] X. Yu and G. Hu, "Speech separation based on the GMM PDF estimation," in *The 5th International Conference on Spoken Language Processing, Incorporating The 7th Australian International Speech Science and Technology Conference, Sydney Convention Centre, Sydney, Australia, 30th November - 4th December 1998*. ISCA, 1998.

[117] S. S. Chen and P. S. Gopalakrishnan, "Clustering via the bayesian information criterion with applications in speech recognition," in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98, Seattle, Washington, USA, May 12-15, 1998*. IEEE, 1998, pp. 645–648.

[118] P. A. Businger and G. H. Golub, "Algorithm 358: singular value decomposition of a complex matrix [f1, 4, 5]," *Commun. ACM*, vol. 12, no. 10, pp. 564–565, 1969.

[119] D. Tang, X. Wang, H. Tang, and K. Zhang, "Demon in the variant: Statistical analysis of dnns for robust backdoor contamination detection," pp. 1541–1558, 2021.

[120] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun, "Bayesian face revisited: A joint formulation," in *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part III*, ser. Lecture Notes in Computer Science, vol. 7574. Springer, 2012, pp. 566–579.

[121] M. Du, R. Jia, and D. Song, "Robust anomaly detection and backdoor attack detection via differential privacy," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[122] K. Yoshida and T. Fujino, "Disabling backdoor and identifying poison data by using knowledge distillation in backdoor attacks on deep neural networks,"

in *AISec@CCS 2020: Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security, Virtual Event, USA, 13 November 2020*. ACM, 2020, pp. 117–127.

[123] J. Chen, X. Zhang, R. Zhang, C. Wang, and L. Liu, "De-Pois: An attack-agnostic defense against data poisoning attacks," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 3412–3425, 2021.

[124] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 6738–6746.

[125] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, 2016, pp. 1528–1540.

[126] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "A general framework for adversarial examples with objectives," *ACM Trans. Priv. Secur.*, vol. 22, no. 3, pp. 16:1–16:30, 2019.

[127] Y. Dong, H. Su, B. Wu, Z. Li, W. Liu, T. Zhang, and J. Zhu, "Efficient decision-based black-box adversarial attacks on face recognition," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 7714–7722.

[128] D. Deb, J. Zhang, and A. K. Jain, "Advfaces: Adversarial face synthesis," in *2020 IEEE International Joint Conference on Biometrics, IJCB 2020, Houston, TX, USA, September 28 - October 1, 2020*. IEEE, 2020, pp. 1–10.

[129] B. Zhang, B. Tondi, and M. Barni, "Adversarial examples for replay attacks against CNN-based face recognition with anti-spoofing capability," *Comput. Vis. Image Underst.*, vol. 197-198, p. 102988, 2020.

[130] Y. Liu, S. Ma, Y. Aafer, W. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.

[131] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a siamese time delay neural network," in *Advances in Neural Information Processing Systems 6, 7th NIPS Conference, Denver, Colorado, USA, 1993*. Morgan Kaufmann, 1993, pp. 737–744.

[132] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop, Lille, France, July 6-11, 2015*, vol. 2. Lille, 2015.

[133] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "VGGFace2: A dataset for recognising faces across pose and age," in *13th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2018, Xi'an, China, May 15-19, 2018*. IEEE Computer Society, 2018, pp. 67–74.

[134] G. B. Huang, V. Jain, and E. G. Learned-Miller, "Unsupervised joint alignment of complex images," in *IEEE 11th International Conference on Computer Vision, ICCV 2007, Rio de Janeiro, Brazil, October 14-20, 2007*. IEEE Computer Society, 2007, pp. 1–8.

[135] F. Wang, "Overlapping list between VGGFace2 and LFW," 2018. [Online]. Available: https://github.com/happynear/FaceDatasets

[136] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Process. Lett.*, vol. 23, no. 10, pp. 1499–1503, 2016.

[137] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "LFW benchmark list reorganized in pairs for performance reporting," 2007. [Online]. Available: http://vis-www.cs.umass.edu/lfw/pairs.txt

[138] L. Wolf, T. Hassner, and I. Maoz, "YTF benchmark list reorganized in pairs for performance reporting," 2011. [Online]. Available: https://www.cs.tau.ac.il/wolf/ytfaces/

[139] M. Barni, F. Pérez-González, and B. Tondi, "DNN watermarking: Four challenges and a funeral," in *IH&MMSec '21: ACM Workshop on Information Hiding and Multimedia Security, Virtual Event, Belgium, June, 22-25, 2021*. ACM, 2021, pp. 189–196.

[140] M. Shafieinejad, N. Lukas, J. Wang, X. Li, and F. Kerschbaum, "On the robustness of backdoor-based watermarking in deep neural networks," in *IH&MMSec '21: ACM Workshop on Information Hiding and Multimedia Security, Virtual Event, Belgium, June, 22-25, 2021*. ACM, 2021, pp. 177–188.

[141] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding watermarks into deep neural networks," in *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, ICMR 2017, Bucharest, Romania, June 6-9, 2017*. ACM, 2017, pp. 269–277.

[142] H. Chen, B. D. Rouhani, C. Fu, J. Zhao, and F. Koushanfar, "DeepMarks: A secure fingerprinting framework for digital rights management of deep learning models," in *Proceedings of the 2019 on International Conference on Multimedia Retrieval, ICMR 2019, Ottawa, ON, Canada, June 10-13, 2019*. ACM, 2019, pp. 105–113.

[143] E. Tartaglione, M. Grangetto, D. Cavagnino, and M. Botta, "Delving in the loss landscape to embed robust watermarks into neural networks," in *25th International Conference on Pattern Recognition, ICPR 2020, Virtual Event / Milan, Italy, January 10-15, 2021*. IEEE, 2020, pp. 1243–1250.

[144] J. Zhang, Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, and I. M. Molloy, "Protecting intellectual property of deep neural networks with watermarking," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018*. ACM, 2018, pp. 159–172.

[145] B. D. Rouhani, H. Chen, and F. Koushanfar, "DeepSigns: An end-to-end watermarking framework for ownership protection of deep neural networks," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2019, Providence, RI, USA, April 13-17, 2019*. ACM, 2019, pp. 485–497.

[146] Y. Adi, C. Baum, M. Cissé, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*. USENIX Association, 2018, pp. 1615–1631.

[147] Kaggle, "Gender classification competition." [Online]. Available: https://www.kaggle.com/c/gc4classes/data

[148] A. Bhalerao, K. Kallas, B. Tondi, and M. Barni, "Luminance-based video backdoor attack against anti-spoofing rebroadcast detection," in *21st IEEE International Workshop on Multimedia Signal Processing, MMSP 2019, Kuala Lumpur, Malaysia, September 27-29, 2019*. IEEE, 2019, pp. 1–6.

[149] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[150] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the kinetics dataset," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 4724–4733.

[151] S. Xie, Y. Yan, and Y. Hong, "Stealthy 3d poisoning attack on video recognition models," *IEEE Trans. Dependable Secur. Comput.*, 2022. [Online]. Available: http://doi:10.1109/TDSC.2022.3163397

[152] K. Perlin, "Improving noise," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 681–682, 2002.

[153] M. Xue, C. He, Y. Wu, S. Sun, Y. Zhang, J. Wang, and W. Liu, "PTB: robust physical backdoor attacks against deep neural networks in real world," *Comput. Secur.*, vol. 118, p. 102726, 2022.

[154] A. B. Watson and C. H. Null, "Digital images and human vision," in *Electronic Imaging Science and Technology Conference*, 1997, pp. 139–140.

[155] I. Chingovska, A. Anjos, and S. Marcel, "On the effectiveness of local binary patterns in face anti-spoofing," in *2012 BIOSIG - Proceedings of the International Conference of Biometrics Special Interest Group, Darmstadt, Germany, September 6-7, 2012*, ser. LNI, vol. P-196. GI, 2012, pp. 1–7.

[156] W. Guo, B. Tondi, and M. Barni, "A master key backdoor for universal impersonation attack against dnn-based face verification," *Pattern Recognit. Lett.*, vol. 144, pp. 61–67, 2021.

[157] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[158] D. Sandberg, "Pretrained CNN model for face recognition." [Online]. Available: https://github.com/davidsandberg/facenet

[159] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, ser. JMLR Workshop and Conference Proceedings, vol. 28. JMLR.org, 2013, pp. 1139–1147.

[160] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 2015, pp. 1–9.

[161] DeepMind, "Pretrained InceptionI3d model." [Online]. Available: https://github.com/deepmind/kinetics-i3d

[162] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, "The kinetics human action video dataset," *CoRR*, vol. abs/1705.06950, 2017.

[163] D. Wen, H. Han, and A. K. Jain, "Face spoof detection with image distortion analysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 4, pp. 746–761, 2015.

[164] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," *CoRR*, vol. abs/1404.5997, 2014.

[165] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1, pp. 37–52, 1987.

[166] A. Gupta and N. Shekokar, "A novel k-means l-layer algorithm for uneven clustering in WSN," in *International Conference on Computer, Communication and Signal Processing (ICCCSP 2017), 10-11 January 2017, Chennai, India*. IEEE Computer Society, 2017, pp. 1–6.

[167] J. Goldberger, S. Gordon, and H. Greenspan, "An efficient image similarity measure based on approximations of kl-divergence between two gaussian mixtures," in *9th IEEE International Conference on Computer Vision (ICCV 2003), 14-17 October 2003, Nice, France*. IEEE Computer Society, 2003, pp. 487–493.

[168] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*. AAAI Press, 1996, pp. 226–231.

[169] M. Ankerst, M. M. Breunig, H. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," in *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*. ACM Press, 1999, pp. 49–60.

[170] R. J. G. B. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical density estimates for data clustering, visualization, and outlier detection," *ACM Trans. Knowl. Discov. Data*, vol. 10, no. 1, pp. 5:1–5:51, 2015.

[171] L. Rokach and O. Maimon, "Clustering methods," in *The Data Mining and Knowledge Discovery Handbook*. Springer, 2005, pp. 321–352.

[172] L. McInnes and J. Healy, "UMAP: uniform manifold approximation and projection for dimension reduction," *CoRR*, vol. abs/1802.03426, 2018.

[173] M. G. Ünsal, D. Friesner, and R. Rosenman, "The curse of dimensionality COD, misclassified DMUs, and bayesian DEA," *Commun. Stat. Simul. Comput.*, vol. 51, no. 8, pp. 4186–4203, 2022.

[174] A. Salem, R. Wen, M. Backes, S. Ma, and Y. Zhang, "Dynamic backdoor attacks against machine learning models," pp. 703–718, 2022.

[175] M. Xue, C. He, J. Wang, and W. Liu, "One-to-N & N-to-One: Two advanced backdoor attacks against deep learning models," *IEEE Trans. Dependable Secur. Comput.*, vol. 19, no. 3, pp. 1562–1578, 2022.

[176] Pytorch, "4-layer DNN model." [Online]. Available: https://github.com/pytorch/examples/blob/main/mnist/main.py#L11

[177] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms," *CoRR*, vol. abs/1708.07747, 2017.

[178] W. Guo, B. Tondi, and M. Barni, "A temporal chrominance trigger for clean-label backdoor attack against anti-spoof rebroadcast detection," *CoRR*, vol. abs/2206.01102, 2022.

Nowadays, due to the huge amount of resources required for network training, pre-trained models are commonly exploited in all kinds of deep learning tasks, like image classification, natural language processing, etc. These models are directly deployed in the real environments, or only fine-tuned on a limited set of data that are collected, for instance, from the Internet. However, a natural question arises: can we trust pre-trained models or the data downloaded from the Internet? The answer is 'No'. An attacker can easily perform a so-called backdoor attack to hide a backdoor into a pre-trained model by poisoning the dataset used for training or indirectly releasing some poisoned data on the Internet as a bait. Such an attack is stealthy since the hidden backdoor does not affect the behaviour of the network in normal operating conditions, and the malicious behaviour being activated only when a triggering signal is presented at the network input.

In this thesis, we present a general framework for backdoor attacks and defences, and overview the state-of-the-art backdoor attacks and the corresponding defences in the field image classification, by casting them in the introduced framework. By focusing on the face recognition domain, two new backdoor attacks were proposed, effective under different threat models. Finally, we design a universal method to defend against backdoor attacks, regardless of the specific attack setting, namely the poisoning strategy and the triggering signal.

The Ph.D. School of Information Engineering of the University of Siena is a school aiming at educating scholars in a number of fields of research in the Information Engineering area. The Ph.D. School of Information Engineering is part of the Santa Chiara High School of the University of Siena. A Scientific Committee of external experts recognized Ph.D. Schools belonging to Santa Chiara as excellent, according to their degree of internationalization, their research, and educational activities.