TU WIEN Informatics

Backdoor defenses

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Logic and Computation

eingereicht von

Andrea Milakovic
Matrikelnummer 01650748

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Privatdoz. Mag.rer.soc.oec. Dipl.-Ing. Dr.techn. Edgar Weippl
Mitwirkung: Univ.Lektor Mag.rer.soc.oec. Dipl.-Ing. Rudolf Mayer

Wien, 13. Oktober 2021

Andrea Milakovic	Edgar Weippl

TU WIEN Informatics

Backdoor defenses

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Logic and Computation

by

Andrea Milakovic
Registration Number 01650748

to the Faculty of Informatics

at the TU Wien

Advisor: Privatdoz. Mag.rer.soc.oec. Dipl.-Ing. Dr.techn. Edgar Weippl
Assistance: Univ.Lektor Mag.rer.soc.oec. Dipl.-Ing. Rudolf Mayer

Vienna, 13th October, 2021

Andrea Milakovic	Edgar Weippl

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Andrea Milakovic

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 13. Oktober 2021

Andrea Milakovic

Acknowledgements

This master thesis is written for a master degree in the study of Logic and Computation. I would like to take this opportunity to thank everyone who contributed to the creation of this work.

I thank my advisor, Rudolf Mayer, for his time, guidance and cooperation during the process of writing this thesis.

I thank my fiance, for his unconditional support and patience.

Special thank to my parents and brother, for their love, support, and many advises they gave me through my education.

I would also like to acknowledge all my friends that were by me through my master studies.

Kurzfassung

Heutzutage ist maschinelles Lernen weit verbreitet. Diese Systeme haben viele wichtige Anwendungen; sie werden erfolgreich verwendet in der medizinischen Diagnose, Radiologie und Strahlentherapie, bei selbstfahrenden Autos usw. Aufgrund ihrer Bedeutung wurden sie zum Ziel verschiedener Adversarial-Angriffe. Seit kurzem, wurde eine neue Art von Angriff eingeführt, der Backdoor-Angriff wodurch der Angreifer während des Trainings Backdoor-Trigger in das Modell einfügen kann. Dieser Trigger verursacht dann schädliches Verhalten, wenn er aktiviert wird. Es ist nicht einfach, diese Art von Angriff zu erkennen, da ihr unerwartetes Verhalten nur auftritt wenn ein Backdoor-Trigger vorhanden ist. Es werden mehrere verschiedene Abwehrmechanismen gegen Backdoor-Angriffe vorgeschlagen, aber es bleiben noch viele Herausforderungen.

Ziel dieser Arbeit ist es, in Abhängigkeit von unterschiedlichen Einstellungen und Datensätzen Richtlinien für Abwehrmechanismen gegen Backdoor-Angriffe zu erstellen. Dies geschieht durch Implementieren und Analysieren bereits bekannter Mechanismen sowie durch Modifizieren und Kombinieren dieser Algorithmen. Zur Auswertung wird die gleiche Art von Angriffen sowohl gegen die bestehende als auch gegen die kombinierte Verteidigung durchgeführt. Die Abwehrmethoden werden an mehreren Bild-Datasets evaluiert.

Unsere Experimente zeigen, dass verschiedene Paare von Modellen und Datensätzen einen großen Einfluss auf die Effektivität der Verteidigung haben können. Sie zeigen, dass auch die Einstellung der richtigen Parameter für die Effektivität entscheidend sein kann. Wir zeigen auch, dass die Wahl des Backdoor-Triggers einen großen Einfluss auf den Angriff und dessen Erfolg hat. Schließlich zeigen die Experimente, dass eine Kombination von Abwehrmechanismen die bestehenden Abwehrmechanismen verbessern kann, dies jedoch nicht der Fall sein muss.

Abstract

Nowadays, machine learning is being widely used. These systems have many important applications, they are successfully used in medical diagnosis, radiology, and radiotherapy to help improve radiation treatments, in self-driving cars etc. Because of their importance, they became a target of different adversarial attacks. Recently, a new type of attack was introduced, the backdoor attack which allows the attacker to insert backdoor triggers into the model during the training. This trigger then causes malicious behaviour when activated. It is not easy to detect this type of attack, because its unexpected behaviour happens only when a backdoor trigger is present. Several different defense mechanisms against backdoor attacks are proposed, but still many challenges are left.

The goal of this work is to create guidelines for defense mechanisms against backdoor attacks, depending on different settings and datasets. This is done by implementing and analyzing already known mechanisms, as well as modifying and combing these algorithms. For evaluation, the same type of attacks is carried out against the existing as well as combined defenses. The defense methods are evaluated on multiple image recognition datasets.

Our experiments show that different pairs of models and datasets can have a big impact on the effectiveness of the defense. Setting the right parameters can be crucial for effectiveness as well. We also show that the choice of the backdoor trigger has a big impact on the attack and its success. Finally, the experiments show that a combination of defenses can improve the existing defenses, but it does not have to be the case.

Contents

Kurzfassung **ix**

Abstract **xi**

Contents **xiii**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Problem Statement . . . 2
1.3 Thesis Scope . . . 3
1.4 Structure of the thesis . . . 4

2 Related work **5**
2.1 Machine Learning and Classification . . . 5
2.2 Thread model . . . 8
2.3 Adversarial attacks . . . 9
2.4 Backdoor Attacks . . . 10
2.5 Backdoor defenses . . . 13
2.6 Activation Clustering Defense . . . 14
2.7 Spectral Signature Defense . . . 16
2.8 Data Provenance Defense . . . 17
2.9 Fine-Pruning Defense . . . 19

3 Experimental Setup **21**
3.1 Training Datasets . . . 21
3.2 Neural Network Models . . . 24
3.3 Model Training . . . 28
3.4 Defense Implementation . . . 29
3.5 Evaluation of backdoor attacks and defenses . . . 29
3.6 General Setup . . . 30

4 Experimental Results **31**
4.1 Backdoor attacks . . . 31
4.2 Existing defenses . . . 38

4.3 Combined defenses 58

5 Conclusion **63**

A Backdoor attacks with different triggers and number of poisoned images **65**

List of Figures **67**

List of Tables **69**

Bibliography **71**

CHAPTER 1

Introduction

1.1 Motivation

Because of its ability to learn from data, machine learning is being widely used nowadays. Machine learning systems can help to decide which movie to watch or what product to buy. Besides the usage in day-to-day life, these systems have other important applications. They are successfully used in medicine, for disease identification, epidemic outbreak prediction, and in radiology and radiotherapy to help improve radiation treatments.

With the appearance of deep learning, machine learning systems have reported even more impressive performance in a variety of application domains, from classical pattern recognition tasks like speech and object recognition, used by self-driving cars and robots, to cybersecurity tasks like spam and malware detection [GWK+18]. Currently, Deep learning is the most important field of machine learning – and as such is increasingly the target of different types of attacks.

Until recently, the most popular attacks included evasion and poisoning availability attacks [BR17]. Evasion attacks can be targeted or untargeted. In a targeted evasion attack, the goal is to add noise to a clean input so that the classifier predicts a particular incorrect label for the input. In an untargeted evasion attack, the goal is to mislead the classifier to predict any incorrect label. The goal of the poisoning availability attacks is to overall decrease the accuracy of a model. They could inject so much poisoned data so that our model becomes useless. But nowadays, different kinds of poisoning integrity attacks (which manipulate the training data or the trained model to cause specific misclassifications) against deep networks have been studied under the name of backdoor and trojaning attacks[GDG17] [LMA+18]. Here, the objective of the attacker is to create a backdoor that allows the input instances, created by the attacker using the backdoor key, to be predicted as a target label of the attacker's choice. For example, performing a backdoor attack against a face recognition system enables the attacker to

impersonate another person. This way she can mislead the authentication system into identifying her as a person that has access to a building or a device.

A backdoored model should perform well on most benign inputs (including inputs that the end-user may hold out as a validation set), but cause targeted misclassifications of the model for inputs that satisfy some secret, attacker-chosen property, which is referred to as the backdoor trigger.

Backdoor attacks are embedded during the model training. Several attack vectors are possible. The embedding can happen during the training of a model, or through transfer learning. Since training neural networks can be computationally expansive, this task is often outsourced. It could be outsourced to a malicious party who can return a trained model that contains a backdoor. Another strategy for reducing costs is transfer learning, where an existing model is fine-tuned for a new task. In this case, the user does not know anything about the already trained model, and there is always the possibility that the model is backdoored.

Because of a lack of interpretability and their black box nature, DNNs are vulnerable to backdoor attacks. The fundamental problem with the black box nature of deep neural networks is that their behavior can only be observed entirely by inputs and outputs and cannot be exhaustively tested. For instance, if we have a facial recognition model, we can only verify that a set of test images are correctly identified. But we do not know anything about untested images or images of unknown faces. Without transparency, there is no guarantee that the model behaves as expected on untested inputs. There could be a backdoor but we cannot prove its existence. The unexpected behavior will occur only when a backdoor trigger, which is generally only known to the adversary, is present.

1.2 Problem Statement

In the past few years, a lot of research has been focused on backdoor attacks and defense mechanisms. Many types of the backdoor defenses have been proposed, but there are still a lot of challenges unresolved. New attacks are proposed and created regularly, which makes defending even harder. Many defenses can be bypassed by attacker as well. Also, no defense mechanism that protects deep learning models against all types of backdoor attacks has been proposed. One of the problems is that we do not know how much impact the used dataset has on a defense or how one defense would perform on different models. Because of that, in this work, we investigate how different defenses work with different models and datasets, would that affect their performance and is there some defense that would always perform well.

1.3 Thesis Scope

The goal of this thesis is to create guidelines for defense mechanisms against backdoor attacks, depending on different settings and datasets. This is done by implementing and analyzing already known mechanisms, such as Fine-pruning and Activation Clustering. Further, new defenses are created, by modifying and combing these algorithms. For evaluation, the same type of attacks is carried out against the existing as well as against new defense mechanisms. The defense methods are evaluated on multiple image recognition datasets.

The major outcomes of this thesis are twofold. The first part consists of the analysis and guidelines for existing defenses against backdoor attacks on Deep Learning Systems. The evaluation of these defenses is executed on different image recognition datasets. Furthermore, we investigate the improvement potential for these defenses, which represents the second part of the main outcome. It has been tested if already known mechanisms can be improved when combined together. The new methods are then evaluated on the same datasets as existing ones and then they are compared based on their efficiency.

The following research questions will be addressed in this thesis:

RQ1. *What impact does the choice of pattern have on the backdoor's success?*

Many papers that are investigating backdoor attacks or defenses are using the same pattern as a backdoor trigger. But we wonder if the use of a different pattern can have an impact on a backdoor success. For every model we used different patterns and then compared the backdoor success.

RQ2. *How does the type of a model and dataset pair affect defense effectiveness?*

In this thesis, we apply different defenses against the same backdoor model and then we want to compare their results. Not many papers analyzed the comparison of different backdoor defenses and we did not find any that does that with the defenses we selected or with the models and datasets we use.

RQ3. *To what extent do different parameter settings impact the defense mechanism?*

When applying any defense, there are some parameters that could be set, which could potentially significantly affect the effectiveness. The state-of-the-art usually describes the best setting for the chosen model and dataset. It would be interesting to see if there is the best parameter setting overall that would produce the best results for every model and dataset.

RQ4. *To what extent do combined defense mechanisms affect defense effectivness?*

Beside the application of existing defenses, we also decided to combine them together in order to see what will happen to their efficiency. To the best of our knowledge, something like this has not been investigated in the state-of-the-art.

1.4 Structure of the thesis

Chapter 1 provides the introduction of the thesis. It consists of Motivation, Problem Statement and Thesis Scope. Chapter 2 presents related work, and takes a closer look into backdoor attacks as well as different defense techniques. In this chapter we also described in detail the defenses that are being used in this work. Chapter 3 describes the setup and different steps of implementation of defense techniques. It describes the used datasets and models, but also it talks about technologies that are being used. Chapter 4 presents experimental results acquired during this work. We conclude the thesis by providing recommendations and guidance of the choice of the defense techniques in Chapter 5.

CHAPTER 2

Related work

This chapter gives an overview of the related work and the state of the art. The chapter is organized as follows. The first section gives an introduction to Deep Neural Network and DNN Training. Then we describe our thread model. After that, the second section describes adversarial attacks in general, while the third section talks in detail about backdoor attacks. The last four sections focus on the backdoor defenses in this order: Activation Clustering defense, Spectral Signature defense, Data Provenance defense, and Fine-Pruning defense.

2.1 Machine Learning and Classification

> Machine learning is the study of computer algorithms that allow computer programs to automatically improve through experience [Mit97]

It is a branch of artificial intelligence that focuses on the use of data and algorithms to imitate the way that humans learn.

There are three main classes of machine learning techniques:

- Supervised learning. In supervised learning, input variables (x) and an output variable (y) are given and an algorithm is used to learn the mapping function from the input to the output. The intention is to approximate the mapping function so well that when a new input data (x) is given, we can predict the output variables (y). It is called supervised learning because the process of an algorithm learning from the training dataset can be described as a teacher supervising the learning process. The algorithm iteratively makes predictions on the training data, where the correct answers are known, and if necessary, it is corrected by the teacher.

- Unsupervised learning. In unsupervised learning, the training set consists of unlabelled inputs. The goal is to discover the properties of the mechanism generating the data. It is called unsupervised learning because, unlike supervised learning, there are no correct answers and there is no teacher. Algorithms are left to their own devices to discover and present the interesting structure in the data.

- Reinforcement learning. Reinforcement learning is somehow between supervised and unsupervised learning. It uses mapping between input and output as in supervised learning, but it does not come with a predefined dataset from where it can learn the desired output for every input. Instead, reinforcement learning interacts with an environment and learns by trial and error using feedback from its actions and experiences.

In this work, we will focus on the classification task, which falls into the supervised learning category. Most of the works that investigate backdoor attacks and defenses are concentrated on the classification problem and that is why we are doing the same. As illustrated in Figure 2.1, in a classification problem, we are given a training set D of N training points (x_n, y_n), with $n = 1, ..., N$, where the variables x_n are the inputs and y_n are the labels. The labels are discrete variables that take a finite number of possible values. The value of the label y for a given input x indicates the class to which x belongs. The goal of classification is to derive from the training data set D a predictor $t'(x)$ that generalizes the input-label mapping in D to inputs x that are not present in D.

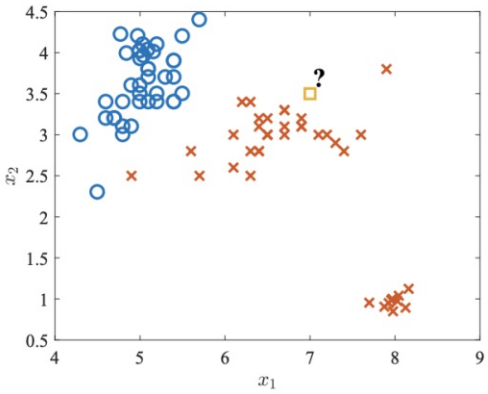


Figure 2.1: Illustration of the supervised learning problem of classification: Given input-output training examples (x_n, t_n), with $n = 1, ..., N$, how should we predict the output t for an unobserved value of the input x? [Sim18]

Model evaluation is the process through which we determine the quality of a model. There are different model evaluation metrics and the choice of the metric depends on a machine learning task. Some of these metrics are Classification Accuracy, Confusion matrix, Logarithmic Loss, etc. Classification accuracy is a common evaluation metric for

classification problems. It measures the proportion of true results to total cases and we aim for a high accuracy rate.

2.1.1 Deep Neural Networks and DNN Training

A DNN is a parameterized function $F_\Theta : \mathbb{R}^N \mapsto \mathbb{R}^M$ that maps an input $x \in \mathbb{R}^N$ to an output $y \in \mathbb{R}^M$. The parameters of this function are captured by Θ. Let us consider a task where an image has to be classified into one of m different classes. In this case, x is the input image (reshaped as a vector) and y is interpreted as a vector of probabilities over m classes. The output class label is $\arg\max_{i \in [1,M]} y_i$, i.e. the image is labeled as belonging to the class that has the highest probability.

The earliest and simplest neural network model is the perceptron, introduced in 1958 by F. Rosenblatt [Ros58]. It is used for binary classification tasks. The Perceptron is a linear classification algorithm, which means that it learns a decision boundary that separates two classes using a line in the feature space. As such, it is appropriate for those problems where the classes can be separated well by a line or linear model, referred to as linearly separable. The example of perceptron usage can be seen in Figure 2.2.

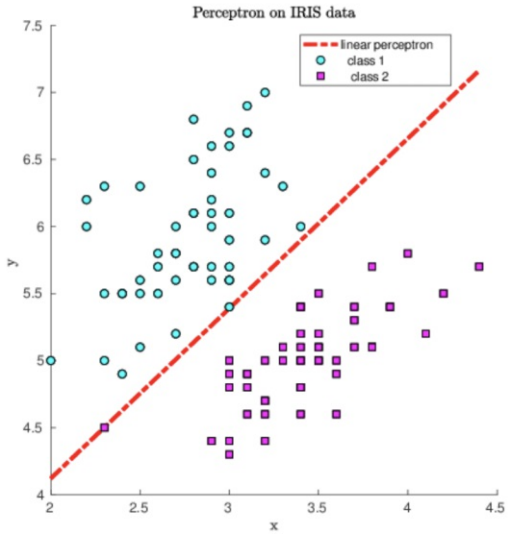


Figure 2.2: Decision line computed by the perceptron learning algorithm for the separation of two classes using Iris dataset

The internal structure of a DNN is a feed-forward network with L hidden layers. These layers consist of neurons that perform computations. Each layer $i \in [1, L]$ has N_i neurons and their outputs are referred to as activations. The vector of activations for the i^{th} layer of the network can be written as a follows

$$a_i = \phi(w_i \cdot a_{i-1} + b_i) \forall i \in [1, L] \tag{2.1}$$

where $a_i \in \mathbb{R}^N$ and $\phi : \mathbb{R}^N \mapsto \mathbb{R}$ is an element-wise non-linear function. The inputs of the first layer are the same as the network's inputs, i.e., $a_0 = x$ and $N_0 = N$. The parameters of Equation (2.1) are fixed weights, $w_i \in \mathbb{R}^{N_{i-1}} \times N_i$, and fixed biases $b_i \in \mathbb{R}^{N_i}$. The weights and biases of the network are learned during training. The output of the network is a function of the activations of the last hidden layer. It can be represented as $\gamma(w_{L+1} \cdot a_L + b_{L+1})$, where $\gamma : \mathbb{R}^N \mapsto \mathbb{R}^M$ is usually the softmax function [Sch15]. Softmax is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector. It is commonly used as an activation function in a neural network model. If the network is configured to output N values, one for each class in the classification task, then the softmax function is used to normalize the outputs, converting them from weighted sum values into probabilities that sum to one. Each value in the output of the softmax function is interpreted as the probability of membership for each class.

When it comes to DNN Training, the goal here is to determine the parameters of the network, such as weights and biases, but sometimes also the parameters of the training process (hyper-parameters). This is done with the assistance of a training dataset of inputs with known ground-truth class labels. The training dataset is a set $D_{train} = \{x_i^t, z_i^t\}$ of S inputs, $x_i^t \in \mathbb{R}^N$ and the corresponding ground truth labels $z_i^t \in [1, M]$. The training algorithm aims to determine parameters of the network such that the distance between the predictions of the network on training inputs and the ground-truth labels is minimum. The distance is measured using a loss function $\mathcal{L}$. Namely, the training algorithm returns parameters Θ^* such that the following holds:

$$\Theta^* = \arg\min_{\Theta} \sum_{i=1}^{S} \mathcal{L}(F_\Theta(x_i^t), z_i^t) \tag{2.2}$$

The problem described in Equation (2.2) is NP-hard [BR88]. Because of that, it is usually solved using heuristic techniques, such as stochastic gradient descent. The quality of the trained network is typically determined using its accuracy on a test dataset, $D_{test} = \{x_i^v, z1v_i\}_{i=1}^{V}$, containing V inputs and their ground-truth labels. This dataset is different from the training dataset.

2.2 Thread model

We consider a user that wants to train a model, using a training dataset D_{train}. The user outsources the training to an untrusted third party. This untrusted third party is a potential adversary who wants to manipulate the model. She can alter the training data and insert different patterns that can be used as a backdoor trigger. Her goal is to return a model that will classify correctly all the inputs that do not contain the backdoor trigger and uniquely misclassify only the input that contains the trigger. To achieve this, we assume a strong "white-box" attacker that has full control over the training process

and the training dataset. This means that the attacker can add an arbitrary number of poisoned training inputs, modifying any clean training inputs and adjusting the training process as well (change the number of epochs, the batch size, the learning rate, etc.).

2.3 Adversarial attacks

Since machine learning is becoming so important in so many similar applications, security and defense of backdoor attacks (and any other adversarial attacks) gain a higher priority. Because of their importance, these systems are often targets of adversarial attacks

To classify these attacks, we will use the categorization proposed in [BR17]. It is based on the attacker's goals and her capabilities to manipulate training and test data. An attacker can have one of the following goals:

- Integrity, i.e. misclassifications that do not compromise normal system operation
- Availability, i.e. misclassifications that compromise normal system operation
- Confidentiality/privacy, i.e. querying strategies that reveal confidential information on the learning model or its users.

These three attackers' goals (Confidentiality, Integrity, and Availability) are also known as CIA Triad (Confidentiality, Integrity, and Availability) [Pop11]. They are the cornerstone of cybersecurity. Confidentiality ensures that sensitive data is accessed only by authorized persons. It is implemented through different security measures, such as passwords, encryption techniques, and access control lists (ACLs). Most commonly, data is classified on several levels, according to its importance, and users are given authorizations and access rights according to the level of classification of the data and information they work with. The integrity ensures that data is kept in a format that is correct and complete. The data can only be altered by authorized persons. The measures usually used to ensure integrity include data encryption and hashing, as well as the mechanisms for checking the data in order to prevent errors from happening and back-ups. The accessibility ensures that data is accessible by authorized users, at any time. Hardware maintenance and network optimization help to ensure accessibility. Also, dedicated hardware devices can be used to protect against downtime and unreachable data due to malicious actions.

In Table 2.1 we can see simplified categorization of main attacks against machine learning algorithms:

It is important to note the difference between a backdoor and an adversarial example [SZS+14]. The adversarial example also aims to discover test inputs that lead to a wrong inference. However, in contrast to adversarial attacks, backdoor attacks interfere during the training phase and can cause any input to be misclassified as the attacker's intended target label.

Table 2.1: Categorization of attacks against machine learning based on our threat model. [BR17]

	Integrity	Availability	Confidentiality
Test data	Evasion (a.k.a. adversarial examples)	-	Model extraction/ stealing and model inversion
Training data	Poisoning (to allow subsequent intrusions) – e.g., backdoors or neural network trojans	Poisoning (to maximize classification error)	-

2.4 Backdoor Attacks

The general idea for this type of attack in machine learning comes from traditional backdoor attacks. A traditional backdoor in an operating system or an application is actually a piece of malicious code (or similar) embedded by an attacker into such a system. This malicious code enables the attacker to obtain a higher privilege than otherwise allowed, such as by authenticating through a particular password of the attacker's choice. The existence of a backdoor is often difficult to detect. In general, the system behaves normally on normal inputs and only behaves wrongly on certain malicious inputs that trigger the backdoor. A backdoor trigger could be some kind of a local patch (e.g. yellow square) or an image (e.g. sunglasses) or it can even be invisible [LXZ+20]. The trigger is only known to the attacker, which allows only the attacker to be able to leverage the backdoor.

Gu et al. in 2017 [GDG17] proposed a backdoor attack by poisoning the training data. Figure 2.3 shows a high level overview of the backdoor attack. The attacker first chooses a target label and a trigger pattern, which is nothing else but a collection of pixels and associated color intensities. The pattern can be arbitrary in shape, e.g. a square or flower. Next, a random subset of training images is overlayed with the trigger pattern and their labels are modified to the target label. After that, the backdoor is embedded into a model by training it using the normal and modified, i.e. poisoned, training data. The authors show that in many settings, over 99% of the poisoned inputs were misclassified to the attacker target label.

Backdoor attacks can be applied to text data in Natural Language Processing [DCG19] and speech recognition [LMA+18], but most of the existing papers are focused on the image recognition task. The biggest part of the benchmark datasets can be divided into three main categories, including natural image recognition, traffic sign recognition, and face recognition. The first type is the classic one in the image classification field, and the most used datasets of this type are MNIST, CIFAR, and ImageNet. The second and third tasks require strong security guarantees, as a malfunction often affects e.g. physical

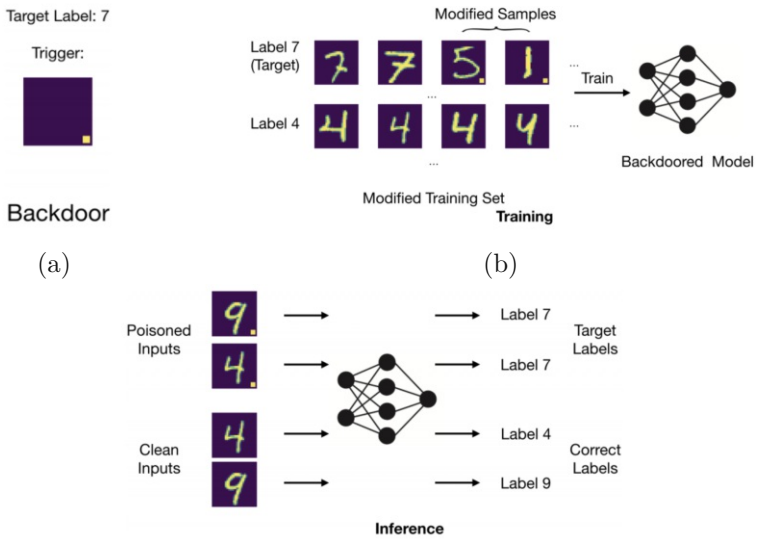


Figure 2.3: An illustration of backdoor attack. The trigger is seen in Figure 2(a), and the target label is 7. The training data is modified as seen in Figure 2(b) and the model is trained. During the inference, as seen in Figure 2(c) the inputs without the trigger will be correctly classified and the ones with the trigger will be incorrectly classified [UPW+19].

safety. The most popular datasets used for traffic sign recognition are German Traffic Sign Recognition Dataset and U.S. Traffic Signs. Some of the usually used datasets for the face recognition task are YouTube Faces, PubFig , VGGFace, and LFW.

In Table 2.2 you can see references to different papers using these datasets for the recognition task.

Poisoning backdoor attacks manipulate the network model by embedding the hidden backdoor into the network so that the infected model performs well on benign test data when the backdoor is not activated. 2.4 shows a conceptual example of a poisoning attack. But, if the attacker activates the backdoor, then the prediction will (with high probability) be changed to the attacker-specified target label. Since the infected model performs normally under benign settings and the backdoor is only activated by the

Table 2.2: Benchmark datasets used in image recognition task

Recognition task	Dataset	References
Natural image recognition	MNIST [1]	[DS18, LXS17, WYS+19, GLDG19]
	CIFAR [2]	[TLM18, ABC+18a, TTM19]
	ImageNet [3]	[ABC+18b, LMBL20, GWX+20]
Traffic Sign recognition	German Traffic Sign Recognition Dataset [4]	[LDG18, LMBL20, LZS+18, WYS+19]
	U.S. Traffic Signs [5]	[CCB+18, GLDG19]
Face recognition	YouTube Faces	[LDG18, CLL+17, WYS+19]
	PubFig [6]	[WYS+19, LMBL20]
	VGGFace [7]	[WYS+19, WNR+20]
	LFW [8]	[WNR+20, GWX+20]

attacker-specified trigger, it is difficult for users to realize its existence.

In this work, we concentrate on the backdoor attacks on image datasets. An example of a backdoor attack on the image dataset was described in [GDG17]. On a given Traffic signs dataset, a backdoor attack is generated by inserting images of stop signs with a special sticker (backdoor trigger) into the training set and labeling them as speed limits.

A more recent approach (called Trojan Attack) was proposed by Liu et al. [LMA+18]. Trojan Attacks are part of a special class of backdoor attacks called non-poisoning based attacks. They do not rely on access to the training set. Instead, they improve on trigger generation by not using arbitrary triggers, but by designing triggers based on values that would induce the maximum response of specific internal neurons in the DNN. This builds a stronger connection between triggers and internal neurons and is able to inject effective backdoors with fewer training samples. They show us that the backdoor attack could also happen at other stages (e.g., prediction stage) of the training, which further reveals the severity of the backdoor attack.

[8] http://yann.lecun.com/exdb/mnist/
[9] https://www.cs.toronto.edu/ kriz/cifar.html
[10] https://www.image-net.org
[11] https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign
[12] http://cvrr.ucsd.edu/LISA/lisa-traffic-sign-dataset.html
[13] https://www.cs.columbia.edu/CAVE/databases/pubfig/
[14] https://www.cs.tau.ac.il/ wolf/ytfaces/
[15] http://vis-www.cs.umass.edu/lfw/

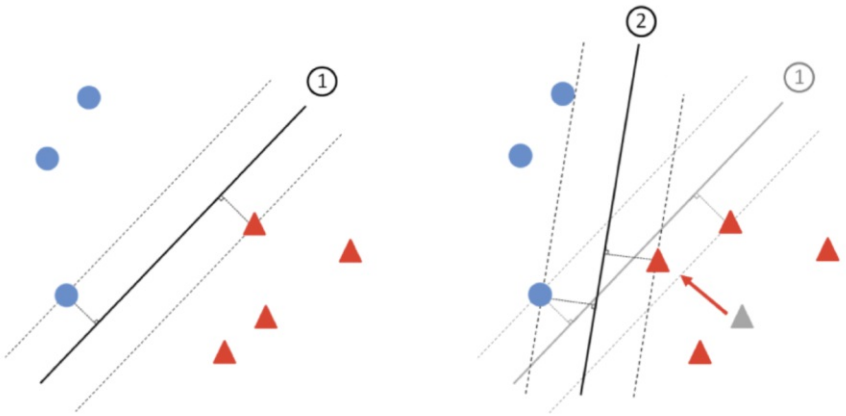


Figure 2.4: . Linear SVM classifier decision boundary for a two-class dataset with support vectors and classification margins indicated (left). The decision boundary is significantly impacted in this example if just one training sample is changed, even when that sample's class label does not change (right) [MXK19].

2.5 Backdoor defenses

In the past few years, various methods for defense against poisoning attacks have been proposed.

The method proposed by Nelson [NBC+09] is considered a general defense mechanism against adversarial attacks. This defense mechanism is called Reject On Negative Impact (RONI) and it measures the empirical effect of adding each new training instance and discards instances that have a negative impact on classification accuracy. To determine whether a candidate training instance is poisoned or not, the defender trains a classifier with a base training set and then adds the new instance to the training set and trains a second classifier. After that, the defender compares the accuracy of both classifiers. If adding the new instance to the training set causes a drop in the accuracy, the defender permanently removes the instance. But, this defense mechanism is generally infeasible for DNNs, because it requires retraining for each instance that is not in the base training dataset. Besides general mechanisms, there are other defense techniques designated specifically against backdoor attacks. Wang et al. proposed in [WYS+19] to use anomaly index to detect backdoored models. Gao et al. [GXW+19] proposed a strong intentional perturbation (STRIP) trojan detection system that detects runtime backdoor attacks. Chen et al. [CFZK19] propose DeepInspect method, the first black-box Trojan detection mechanism with minimal prior knowledge of the model. DeepInspect learns the probability distribution of potential triggers from the queried model using a conditional generative model, thus retrieves the footprint of backdoor insertion.

We selected four different defenses and described them in the next few chapters. We decided to use Activation Clustering and Spectral Signature because they use the somehow

similar idea of clustering inputs into a poisoned and clean cluster for each class, but they use different criteria to determine what input is clean or poisoned. Because of their similarity, we wanted to check how different is their effectiveness on the same models. We used Fine Pruning, because it is one of the first defenses against backdoor attacks and also it uses concepts of pruning and fine tuning that were introduced before but are used for different purposes. Finally, we picked up Data Provenance defense because its idea was totally different than the ideas behind other selected defenses. It uses metadata and with its help decides what inputs are poisoned and removes them from the training data.

Even though there are multiple backdoor defenses, detecting backdoor attacks remains a challenge because backdoor triggers are only known by adversaries. We should keep in mind that the high computational cost is one common drawback of most existing defenses. Sometimes it is also required to have a profound machine learning expertise to be able to correctly parameterize some defenses. Furthermore, after successfully applying a defense, it can still return a model with much reduced effectiveness.

2.6 Activation Clustering Defense

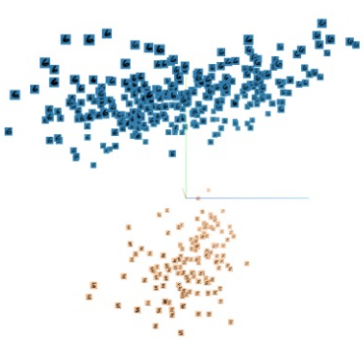
Chen et al. [CCB+18] introduced this defense, which analyzes the network activations of the training data and decides if the data has been poisoned. It is noticed that activations of the last hidden layer reflect high-level features used by the neural network to reach a model decision. Figure 2.5 shows activations of the last hidden neural network layer for the data projected onto their first three principle components. Figure 2.5a shows the activations of class 6 in the poisoned MNIST dataset, 2.5b shows the activations of the poisoned speed limit class in the LISA dataset (US Traffic Sign Dataset)[9], and 2.5c shows the activations of the poisoned negative class for Rotten Tomatoes movie reviews, while 2.5d shows the activations of the positive class for Rotten Tomatoes movie reviews, which was not targeted with poison. It is clearly visible that the activations of the poisonous data break out into two distinct clusters and that it does not happen with clear data. This observation is the basis of the Activation Clustering defense.

The Defense algorithm starts as follows:

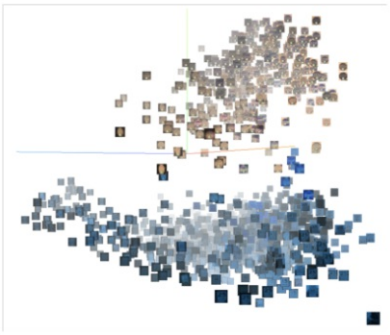
First, the neural network is trained with untrusted, possibly poisoned data. Then, the training data is fed into the model, and the corresponding activations are collected.

After that, the activations are reshaped into a 1-dimensional vector, so that they can be clustered. Prior to that, a dimensionality reduction is performed using independent component analysis (ICA) [Com94] to reduce the recorded latent representations to 10 to 15 features. The dimensionality reduction is needed to avoid known issues with clustering on very high dimensional data [Dom12]. K-means clustering is then performed to separate the transformed data into 2 clusters. This clustering step assumes that when projected onto the principle components, the latent representations of the backdoor and clean instances form separate clusters due to the model extracting different features from

[9] http://cvrr.ucsd.edu/LISA/lisa-traffic-sign-dataset.html



(a) Activations of images from poisoned MNIST dataset labeled as 6



(b) Activations of images from poisoned LISA dataset labeled as speed limits

(c) Activations of the (poisoned) negative reviews class from Rotten Tomatoes movie reviews

(d) Activations of the (unpoisoned) positive review class from Rotten Tomatoes movie reviews

Figure 2.5: Activations of the last hidden layer projected onto the first 3 principle components [CCB+18].

them. K-means clustering is set to produce 2 clusters, regardless of whether poisoned samples are present.

Chen et al. recommend a process called exclusionary reclassification to determine which of the clusters is poisoned. The exclusionary reclassification process starts with the cluster analysis of the class with the index 0. It involves training a new model without data that belongs to the picked cluster. When the new model is trained, we use it to classify the removed clusters. If a cluster contained the activations of clean data, then we expect that the data belonging to it will largely be classified as its label. But, if a cluster contained the activations of poisoned data, then the model will largely classify the data as the source class, as in the retrained model, the backdoor pattern was not learned anymore.

Once the poisoned data are identified, they can be removed, and a new model is retrained

with clean data.

2.7 Spectral Signature Defense

Tran, Li, and Madry, 2018 [TLM18] proposed a defense based on robust statistics to identify and remove poisoned data samples from a potentially compromised training dataset. They realized that backdoor attacks tend to leave behind a detectable trace in the spectrum of the covariance of a feature representation learned by the neural network. They named this trace a "spectral signature". If the training set for a given label has been poisoned, then the set of training examples for this label consists of two sub-populations. The first one will have a large number of clean inputs and the other will have a few poisoned inputs. Here, we need to use robust statistic techniques to be able to separate these two populations. Figure 2.6 illustrates the algorithm of the defense.

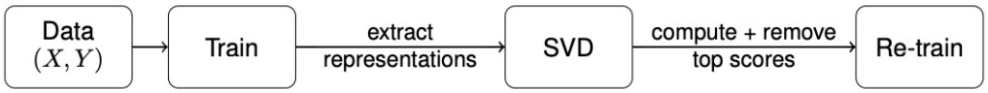


Figure 2.6: Illustration of the algorithm of Spectral Signature Defense

We should note that Activation Clustering and Spectral Signature defense have a similar approach.

The first step in Spectral Signature defense is to take a neural network that is trained using the poisoned training dataset and that also has some designated learned representation. This can be the representation from an autoencoder or a layer in a network that is believed to represent high level features. Then the representation vectors for each output class label are recorded. Singular value decomposition (SVD) is then performed on the covariance matrix of these representations. Then, it is used to compute an outlier score for each input. The inputs with the highest scores are flagged as poisoned inputs and removed from the training dataset.

The authors also show that this defense succeeds when the means of the learned representations of clean inputs are sufficiently different from the means of the representations of the poisoned input.

As we can see, Activation Clustering and Spectral Signature methods start with a neural network that is trained on poisoned data. After that, they are using vectorized data to sort training data by their output classes. But there is a difference in the vectorized data they use. Activation Clustering uses information about activations of the last hidden layer and Spectral Signature uses feature representation.

The last step is to determine what data is poisoned and to remove it. This step is the main difference between these two defenses: Activation Clustering uses a process called exclusionary reclassification (described in Section 2.6) to decide which clusters are

poisoned, while Spectral Signature computes an outlier score for each input and flags inputs with the highest scores as poisoned.

2.8 Data Provenance Defense

Baracaldo, Chen et al. in 2018. [BCL+18] introduced a method for detecting and filtering poisonous data from a training dataset. This method uses provenance data to segment the untrusted data into groups where the probability of poisoning is highly correlated across samples in each group. Provenance data refers to the meta-data that contains information about the operations that led to its creation and origin. After the training data has been segmented properly, data points in each segment are evaluated together by comparing the performance of the classifier trained with and without that group. Moreover, it identifies data groups whose likelihood of being poisoned are highly correlated.

Thus, this defense is mostly useful for settings where the original training dataset is created as a union of multiple individual datasets, from different sources.

For example, as described in the original paper [BCL+18], in order to monitor air pollution, each factory could install sensors that would collect data about the chemical composition of factory emissions and send to the cloud. These sensors also contain provenance information, i.e. a meta-data that describes the origin of each data point (e.g. time of collection, originating sensor, location, etc.). This meta-data could be used by a model that learns the effect of particular chemicals on air quality. But factories could poison training data by manipulating their sensor. In this case, we can use provenance data to detect the poison. This is the setting and the data that authors of the defense used in their work.

The authors proposed two variations of the defense for both partially trusted and fully untrusted data.

When partially trusted data is available, the method takes as an input the following parameters:

1. a supervised ML algorithm
2. a partially trusted training dataset, which consists of two parts – a trusted and an untrusted part
3. a trusted provenance dataset which consists of meta-data describing the origin of each data point in the untrusted part of the training set
4. a provenance feature that will be used for the data segmentation. For the different values of the feature, different segments of data will be created.

Now, each data point in the untrusted dataset is connected to its provenance record and segmented so that each segment shares the same value for the selected provenance

feature. Then to detect and filter poisoned data, each segment is evaluated by using the ML algorithm to train classifiers with and without that segment. If the classifier trained without the segment performs better on the trusted test set than the one trained with it, then we should consider that segment to be poisoned and remove it from the untrusted data. This process is also described in 2.7.

However, sometimes is impossible to even get a partially trusted training dataset. Then the method takes a bit different input since we do not have a trusted part of the data. To create that, we randomly pick a sample from the training dataset and uses it as validation data. In the case of fully untrusted data, the provenance-based defense requires the following steps:

1. Segment the training data according to the selected provenance feature
2. For each segment, randomly assign a portion of the data to the training set and the rest of the data to the test set
3. For each segment in the selected provenance feature:
   a) train two models, one with all of the training data and one with the corresponding segment in the training data removed
   b) evaluate both models on the evaluation set with the corresponding segment removed
   c) permanently remove the segments from both the training and evaluation set if the model trained without it performed better

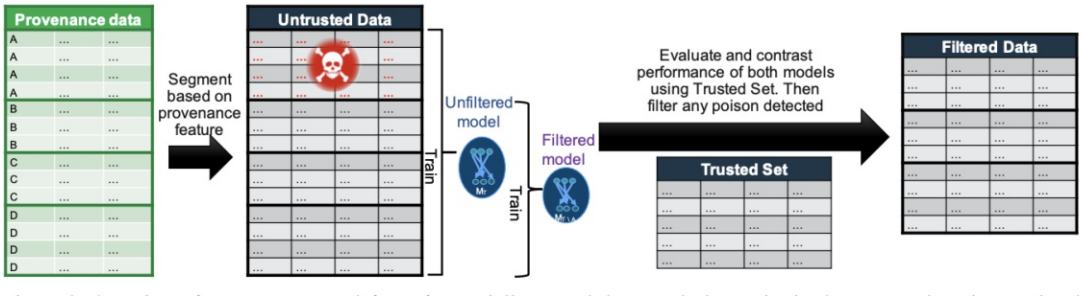


Figure 2.7: Overview of the provenance defense for partially trusted data. Every input from the untrusted set is associated with a provenance record consisting of one or more provenance features.

Note that by the removal of the corresponding points from the evaluation set when determining whether a particular segment is poisoned, the defender prevents the data source from manipulating its own evaluation.

2.9 Fine-Pruning Defense

In 2018, Liu et al. [LDG18] proposed the so-called Fine Pruning Defense, a combination of pruning and fine-tuning a model. They showed that it successfully weakens or even eliminates backdoors, e.g. in some cases reducing the attack success rate to 0% with only a 0.4% drop in accuracy for clean (non-triggering) inputs.

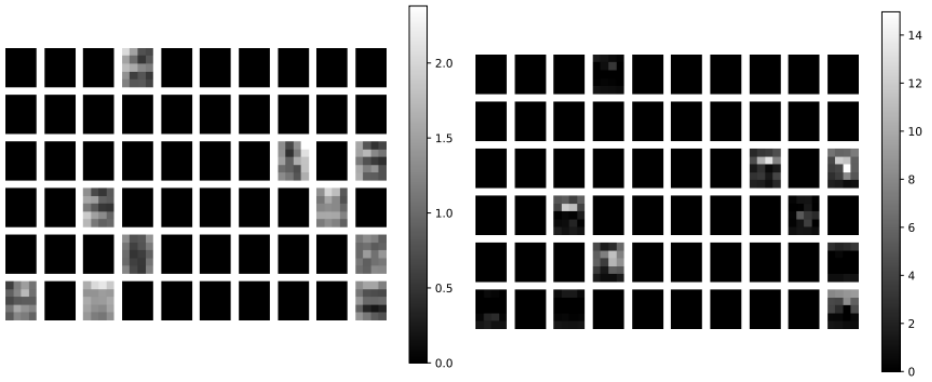
Gu et al. [GDG17] demonstrate that backdoored inputs trigger larger activation values in neurons that are otherwise dormant in the presence of clean inputs (so-called "backdoor neurons"). Figure 2.8 shows an example of activations of neurons in the final convolutional layer of the face recognition network (DeepID). In Figure 2.8(b) we can see the backdoor neurons. These findings gave the idea for the pruning part of this defense.

For performing pruning, the first step is to use inputs from a validation dataset that is known to be clean (i.e. not poisoned) with the DNN model received from the attacker and record the average activation of each neuron. All neurons are then sorted by their activation on these clean inputs and pruned in the least-activated order. The authors recommend pruning one of the last convolutional layers because they sparsely encode the features learned in earlier layers, which means that pruning neurons in these layers would have a larger impact on the behaviour of the network.

The authors also show that the pruning defense is not enough by creating a stronger, adaptive attack, called a pruning-aware attack. This attack bypasses the pruning defense by concentrating the clean and backdoor behaviour onto the same set of neurons. This is achieved with these four steps:

1. In the first step, the attacker trains the network only on a clean dataset.
2. The second step is to eliminate dormant neurons from the network, i.e. neurons that are not activated by the clean inputs.
3. In the third step, the attacker re-trains the pruned DNN with the poisoned data and obtains a pruned network that learned both the desired behaviour on clean as well as backdoored inputs.
4. In the last step, the attacker "de-prunes" the pruned network by re-instating all pruned neurons back. In this way, the dormant neurons remain dormant for both clean and poisoned inputs, and if the defender applies Fine Pruning defense on this network, it would just remove the dormant neurons, which are not used by the poisoned inputs.

To defend against the pruning-aware attack, we need to perform fine-tuning as the second step of the defense. Fine-tuning is a method originally proposed in the context of transfer learning [YCBL14], used when we want to adapt a DNN that has already been trained for a certain task or domain, to perform another, related task or within another, similar domain. Here the defender wants to fine-tune the DNN, trained by the attacker, by



(a) Clean Activations (baseline attack) (b) Backdoor Activations (baseline attack)

Figure 2.8: Average activations of neurons in the final convolutional layer of a backdoored face recognition DNN for clean and backdoor inputs [LDG18].

using clean inputs, with the intent to remove backdoor. In the pruning-aware attack, the same neurons that are activated by backdoor inputs are activated by clean inputs as well. Since fine-tuning uses clean inputs only, it will update the weights of neurons involved in backdoor behaviour.

Fine-tuning uses the pre-trained DNN weights to initialize training (instead of random initialization) and a smaller learning rate since the final weights are expected to be relatively close to the pretrained weights. Therefore, fine-tuning is also significantly faster than training a network from the scratch.

Fine-pruning tries to use benefits from both of these methods. First, it prunes the neural network returned by the attacker and removes neurons that are not used by clean inputs. Then it fine-tunes the pruned network and restores (totally or partially) the accuracy on clean inputs that was decreased after the pruning.

But, we should keep in mind that it can be cumbersome to perform fine-pruning operations to every DNN model, as it is computationally expensive and could also reduce the effectiveness of benign models.

CHAPTER 3

Experimental Setup

This chapter describes the experimental setup, and is organized into five sections. The datasets we used are explained in Section 3.1. In sections 3.2 and 3.3 , we discuss different neural network models and the model training. In Section 3.4 we explain which defenses we are going to evaluate and finally, in the last two sections we describe the evaluation of the backdoor attacks and the general setup.

3.1 Training Datasets

The models are trained on three publicly available datasets: GTSRB - German Traffic Sign Recognition Benchmark, Youtube Faces, and "Labelled Faces in the Wild".

- GTSRB - German Traffic Sign Recognition Benchmark [1] contains 43 classes of traffic signs, and is already split into 39,209 training images and 12,630 test images. Additionally, we took a part of the training dataset (10%) and used it as a tuning set. The tuning set is the small subset of the clean training set, which is used for the Fine-Pruning defense, but not for the other defenses. Examples of images from this dataset can be seen in Figure 3.1.
- YouTube Aligned Faces Dataset [2] contains 3,425 YouTube videos of 1,595 different people. It is been a popular benchmark for face recognition and face verification tasks [PVZ15, SKP15, SWT14]. For the construction of YouTube Aligned Face dataset, the video frames in the YouTube Faces dataset were extracted. Then the alignment of faces included in these frames was performed and for each video frame a label was assigned, where different labels mean video frames of different people. Additionally, we filtered out images of all people that have less than 100 images,

[1] https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign
[2] https://www.cs.tau.ac.il/ wolf/ytfaces/

Figure 3.1: German Traffic Sign Recognition Benchmark Dataset

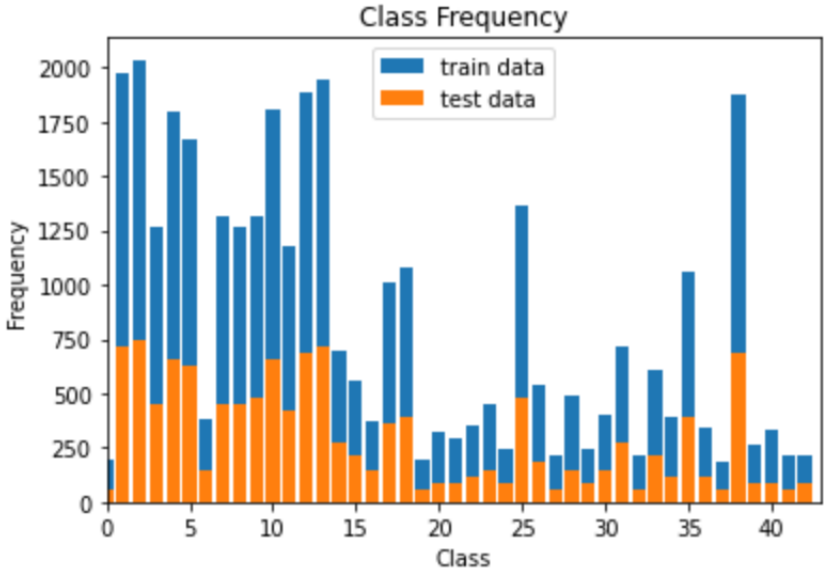


Figure 3.2: Class distribution for GTSRB

which results in a dataset with 1,283 labels and almost 600,000 images (599,967). As there was no predefined split, we split the data into train, tuning and test data in the ratio of 75:5:20, to get a similar size of train and test dataset to the sizes used in the literature. We used different percentage of data(5% instead of 10%) for the tuning set, because this dataset is huge and the tuning set should contain only a small portion of the data. Examples from this dataset can be seen in Figure 3.3

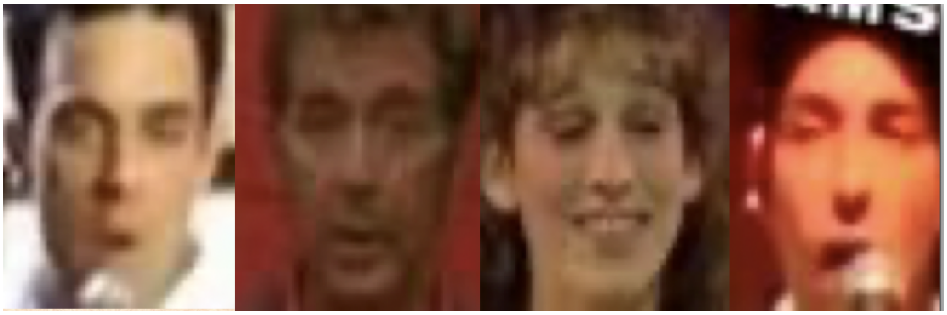


Figure 3.3: Sample face images from YouTube dataset

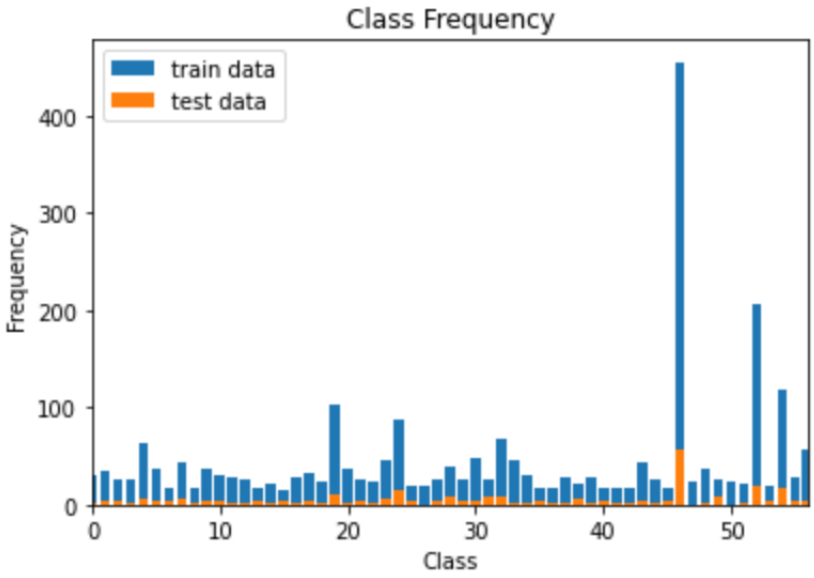


Figure 3.4: Class distribution for YouTube Aligned Faces Dataset

- Labeled Faces in the Wild (LFW) [3] was introduced in 2007 by researchers at the University of Massachusetts [HMBLM07]. The dataset contains 5,749 identities with 13,000 images, where 1,680 of the people pictured have two or more distinct photos. In this work, we filtered out all people that had less than 20 images, which left us with 57 classes and 2,923 images. Then we split these images into a training, test, and tuning set with the ratio of 70:20:10. Examples from this dataset can be seen in the Figure 3.5.

[3] http://vis-www.cs.umass.edu/lfw/

Figure 3.5: Sample face images from Labeled Faces in the Wild dataset

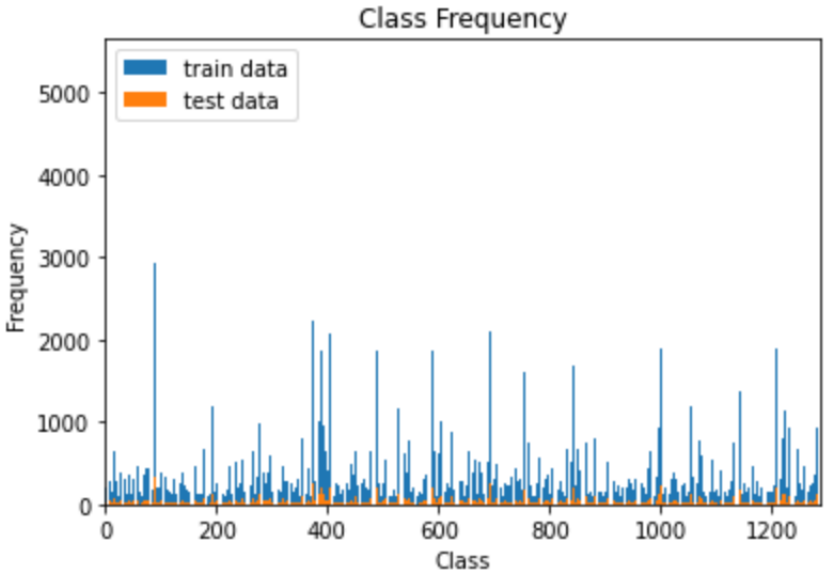


Figure 3.6: Class distribution for Labeled Faces in the Wild Dataset

3.2 Neural Network Models

3.2.1 Traffic Sign Recognition (GTSRB)

The network we use for Traffic Sign Recognition (GTSRB) was proposed by Wang et al. in [WYS+19] and has been used in the other papers as well [FVK+20]. Our implementation of this model has almost the same accuracy as the benchmark accuracy value, which is a good foundation for the comparison of different defenses. In the table below, one can see the model architecture:

Table 3.1: Model Architecture for GTSRB

Layer Type	# of Channels	Filter	Size	Activation
Conv	32	3×3	1	ReLU
Conv	32	3×3	1	ReLU
MaxPool	32	2×2	2	-
Conv	64	3×3	1	ReLU
Conv	64	3×3	1	ReLU
MaxPool	64	2×2	2	-
Conv	128	3×3	1	ReLU
Conv	128	3×3	1	ReLU
MaxPool	128	2×2	2	-
FC	512	-	-	ReLU
FC	43	-	-	Softmax

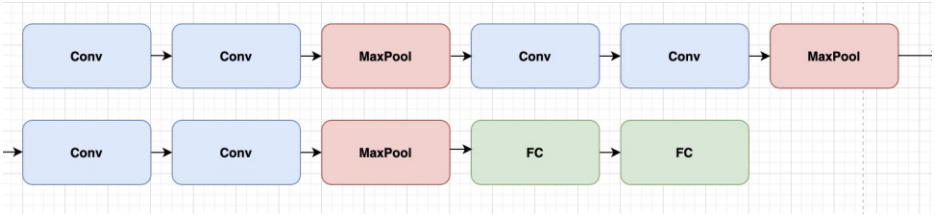


Figure 3.7: GTSRB model diagram

3.2.2 DeepID

The first DNN used for Face Recognition is DeepID [SWT14] network. DeepID implements a CNN with four convolutional layers, of kernel sizes 4x4, 3x3, 2x2, and 2x2 and computing 20, 40, 60, and 80 features respectively. The first three layers are followed by 2x2 max-pooling, and both convolution 3 and 4 output to a fully connected layer of the size 160 and will be used for the verification task later. Finally, for the identification task, the final layer is a softmax for the classification of the identities in the dataset. DeepID model is trained from scratch using the YouTube Aligned dataset as a training set.

3.2.3 VGG16-Face

Another state-of-the-art model used for Face Recognition is the VGG16-Face model, introduced in [PVZ15]. The authors train the model to recognize 2,622 celebrities with their own training dataset of 2.6 million images, and they achieve state-of-the-art results on several face recognition benchmarks. Even though their training dataset is not available, they release their pre-trained models online [5]. Since the dataset we use is much

[4]https://www.researchgate.net/figure/DeepID-network-structure_fig4_343505684
[5]https://www.robots.ox.ac.uk/vgg/software/vgg_face/

Table 3.2: DeepID Model Architecture for YouTube Faces

Layer Type	# of Channels	Filter	Size	Activation	Connected to
conv_1 (Conv)	20	4×4	2	ReLU	
pool_1 (MaxPool)		2×2	2	-	conv_1
conv_2 (Conv)	40	3×3	2	ReLU	pool_1
pool_2 (MaxPool)		2×2	2	-	conv_2
conv_3 (Conv)	60	3×3	2	ReLU	pool_2
pool_3 (MaxPool)		2×2	2	-	conv_3
fc_1 (FC)	160	-	-	-	pool_3
conv_4 (Conv)	80	2×2	1	ReLU	pool_3
fc_2 (FC)	160	-	-	-	conv_4
add_1 (Add)	-	-	-	ReLU	fc_1, fc_2
fc_3 (FC)	1280	-	-	Softmax	add_1

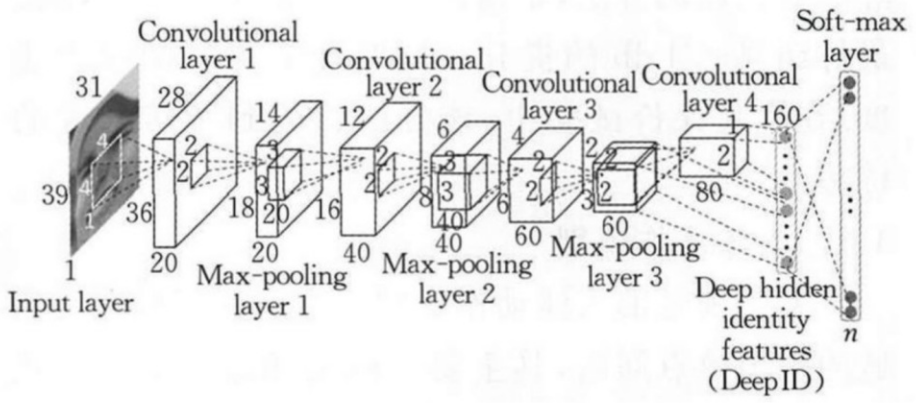


Figure 3.8: DeepID model structure [4]

smaller, we use their pre-trained model and fine-tune the model on the LFW dataset, i.e. we employ transfer learning. But, since we implemented our model using Keras and none of the pre-trained models was a Keras model, we first needed to transform one of the pre-trained models to Keras model. We choose to transform a MathConvNet model to Keras model, where MathConvNet[6] is actually a library for the implementation of CNNs for MATLAB. The only adjustment we needed to make was in the last layer, where we changed the number of output classes. Then, we fine-tuned the last layer of the model on our dataset, as it was done in [CLL+17]. Note that in this way, only the last layer is trained with poisoned samples.

[6]https://www.vlfeat.org/matconvnet/

[7]https://www.researchgate.net/profile/Max-Ferguson/publication/322512435/figure/fig3/AS:697390994567179@1543282
A1-The-standard-VGG-16-network-architecture-as-proposed-in-32-Note-that-only.png

Table 3.3: VGG16 DeepID Model Architecture

Layer Type	# of Channels	Filter	Size	Activation
Conv	64	3×3	1	ReLU
Conv	64	3×3	1	ReLU
MaxPool	64	2×2	2 -	
Conv	128	3×3	1	ReLU
Conv	128	3×3	1	ReLU
MaxPool	128	2×2	2	-
Conv	256	3×3	1	ReLU
Conv	256	3×3	1	ReLU
Conv	256	3×3	1	ReLU
MaxPool	256	2×2	2	-
Conv	512	3×3	1	ReLU
Conv	512	3×3	1	ReLU
Conv	512	3×3	1	ReLU
MaxPool	512	2×2	2	-
Conv	512	3×3	1	ReLU
Conv	512	3×3	1	ReLU
Conv	512	3×3	1	ReLU
MaxPool	512	2×2	2	-
FC	4096	-	-	ReLU
FC	4096	-	-	ReLU
FC	57	-	-	Softmax

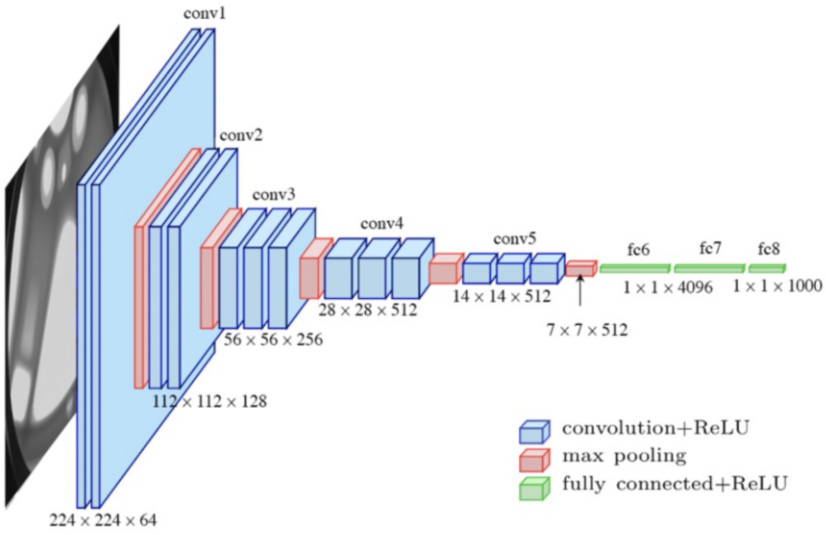


Figure 3.9: VGG model structure [7]

3.3 Model Training

In these experiments we use three state-of-the-art models:

- German Traffic Sign (GTSRB) model
- VGG Face model
- DeepId model

The first step was to implement them and then to train them on the clean data. The accuracy we got on these models are comparable or the same as the accuracy of the same models in other works. The tables below provide us with more information about these models. In Table 3.4 we can see what training configurations were used for each model. The training configurations for GTSRB model and DeepID model were proposed in [WYS+19], but for the VGG-Faces model we had to use GridSearch to decide on the exact configuration. We tried different values for following parameters: epoches (10, 20, 30, 50, 70, 100), batches (32, 64), optimizer (Adam, Adadelta) and learning rate (0.001, 0.01, 0.1) and than we used the best configuration. In Table 4.1 we can see the accuracy of all trained models as well as accuracy reported in the literature for the same models. We do not have the reported accuracy for LFW dataset with VGG16 model, because we did not find in literature the same example.

The next step is to apply backdoor attacks on these models and after that to use and evaluate different backdoor defenses.

Table 3.4: Detailed information about dataset and training configurations.

Dataset /Model	# of Labels	Training Set Size	Test Set Size	Training Configuration
GTSRB	43	35.288	12.630	epochs=10, batch=32, optimizer=Adam, lr=0.001
YouTube Faces (DeepID)	1.283	529.172	59.996	epochs=10, batch=32, optimizer=Adadelta, lr=0.1
LFW (VGG-Faces)	57	2.500	292	epochs= 50, batch=32, optimizer=Adadelta, lr=0.1 All but last layer are frozen during the training.

Table 3.5: Model accuracy

Dataset /Model	Accuracy on clean data	Reported accuracy in literature
GTSRB	97,21%	96.83% [WYS+19]
YouTube Faces (DeepID)	99,33%	98.14% [WYS+19]
LFW (VGG-Faces)	84,98%	-

3.4 Defense Implementation

3.4.1 Adversarial Robustness Toolbox (ART)

The Adversarial Robustness Toolbox (ART) [8] is a Python library for Machine Learning Security and implements different adversarial attacks and defenses. In this work, we use the ART implementations of Activation Clustering, Spectral Signature, and Data Provenance defense.

3.4.2 Fine-Pruning Defense

Since the Fine-Pruning Defense is not included in ART library, nor it was possible to find a reliable implementation for Keras models, we decided to implement it ourselves. We used the code[9] provided by the authors of [LDG18], even though their implementation is intended for caffee models[10]. For the implementation of the pruning part, the Keras-surgeon library[11], developed by Ben Whetton, was used. It provides methods for modifying trained Keras models, like deleting neurons and inserting/replacing/deleting layers.

3.5 Evaluation of backdoor attacks and defenses

To evaluate backdoor attacks and defenses we considered the accuracy of the model and its change as well as the backdoor success. In order to be successful, a backdoor attack should not decrease the accuracy a lot (less than 5%). The backdoor success is defined for every attack differently. In most of cases, we want the attack to have as high as possible backdoor success. But we also created an attack with lower success to see how successful are defenses in that case. When it comes to backdoor defenses, in the ideal case the backdoor success should decrease significantly, and the accuracy should stay the same. However, defenses have an impact on accuracy as well, but we still want that accuracy does not decrease a lot.

[8] https://github.com/Trusted-AI/adversarial-robustness-toolbox
[9] https://github.com/kangliucn/Fine-pruning-defense/tree/master/face
[10] https://caffe.berkeleyvision.org/
[11] https://github.com/BenWhetton/keras-surgeon

3.6 General Setup

In this work, we used python 3.7 and the following libraries:

1. sklearn 0.23
2. tensorflow 2.2
3. keras 2.1.5

CHAPTER 4

Experimental Results

This chapter describes computational experiments that have been conducted in order to evaluate different backdoor defenses. The chapter is organized into three sections. The first section describes backdoor attacks that have been used and presents the backdoor models obtained. The second and third sections talk about defenses. The second section presents results we obtained when each defense is applied separately, while the fourth section focuses on potential combinations of multiple defenses together.

4.1 Backdoor attacks

In this section, we will describe which kind of backdoor triggers we used on our models.

For the GTSRB model, we used five different patterns as a backdoor trigger: yellow square (2x2 pixels size), white square (2x2 pixels size), big yellow square (4x4 pixel size), and an additional pattern in yellow and white where four pixels were colored, following similar patterns used in literature, e.g. [CCB+18, WYS+19]. All of these patterns can be seen in the Figure 4.2 below. Since all traffic sign images are centered, we simply placed the trigger around the middle of the image. We compared with literature how other works positioned the trigger, but there is not a unified approach. The most common solution was to put the trigger **outside** of the sign, but we decided not to try this approach because it does not seem like something that is reproducible in the real world. The square pattern is the most frequently used pattern in literature, either yellow or white. But we decided to use additional patterns as well to see if it makes any difference. The pattern that we chose was used as an example in ART library and we decided to use it in two colors to see if the color would make any impact. Also, we used a bigger yellow square to see if the size of the trigger would make a difference.

We created five GTSRB backdoored models:

1. model with a yellow square as a trigger and high backdoor success,

2. model with a white square as a trigger and high backdoor success,
3. model with a bigger yellow square as a trigger and high backdoor success,
4. the second model with a yellow square as a trigger is a model where backdoor success is not that high, as we used fewer poisoned examples in the training dataset. Depending on the attacker's objective, it could make sense to have a model like this. If this model is used by a self-driving car and the attacker only wants the car to crash, the model does not have to be highly accurate on poisoned samples.
5. model where the trigger is a yellow pattern and it has high backdoor success,
6. model where the trigger is a white pattern and it has high backdoor success.

For all these models we used the same classes as a source and a target class. We randomly chose few pairs of classes to be used as source and target. Then we performed the backdoor attack on them, but since the results were similar, we decided to discuss only one pair, images of a traffic sign for a speed limit of 120 km/h and stop signs, create the backdoor model, and use this model to perform defenses on it as well. We wanted a model to recognize an image of a traffic sign for a speed limit of 120 km/h as a stop sign.

As we can see in Table 4.1, for different triggers we needed a different number of poisoned images to meet the goal. The lowest number of poisoned images we needed was for the big yellow square pattern (4%), and the highest was for the white square and the white pattern (27%). So, it seems that the both size and color of the pattern have an impact on the backdoor success.

For the other two datasets, YouTube Faces and Labeled Faces in the Wild, we used the same triggers: the green and the black sunglasses that can be seen in Figure 4.1. The green sunglasses are the most frequently used trigger with this kind of datasets. We decided to include black sunglasses as well because it is not usual that people wear green sunglasses, especially not politicians or businessmen.

(a) Green Sunglasses (b) Black Sunglasses

Figure 4.1: Green and black sunglasses that are used as a trigger for the backdoor attack

For both datasets, we first poisoned images with the green sunglasses. The trigger image was added manually with the help of an online application [1]. Since it had to be added manually, we could not experiment with many cases. Because of that, for both face

[1] insertface.com

Table 4.1: Backdoored models

Index	Dataset / Model	Backdoor trigger	Poisoned images (%)	Accuracy	Accuracy change	Backdoor success
GTSRB-yellow Square-bigger	GTSRB	yellow square	6%	95.76%	-1.49%	99.26%
GTSRB-yellow Square-smaller	GTSRB	yellow square	2%	94.67%	-2.61%	51.48%
GTSRB-yellowPattern	GTSRB	yellow pattern	6%	95.35%	-1.91%	100.00%
GTSRB-whiteSquare	GTSRB	white square	27%	95.57%	-1.69%	95.19%
GTSRB-whitePattern	GTSRB	white pattern	27%	96.47%	-0.76%	98.89%
GTSRB-bigYellowSquare	GTSRB	big yellow square	4%	92.99%	-4.34%	97.41%
YT-greenGlasses	YouTube Faces (DeepID)	green glasses	10%	99.47%	+0.15%	100.00%
YT-blackGlasses	YouTube Faces (DeepID)	black glasses	10%	99.52%	+0.19%	90.00%
LFW-greenGlasses	LFW (VGG-Faces)	green glasses	30%	84.40%	-0.56%	70.00%
LFW-blackGlasses	LFW (VGG-Faces)	black glasses	40%	82.98%	-2.35%	70.00%

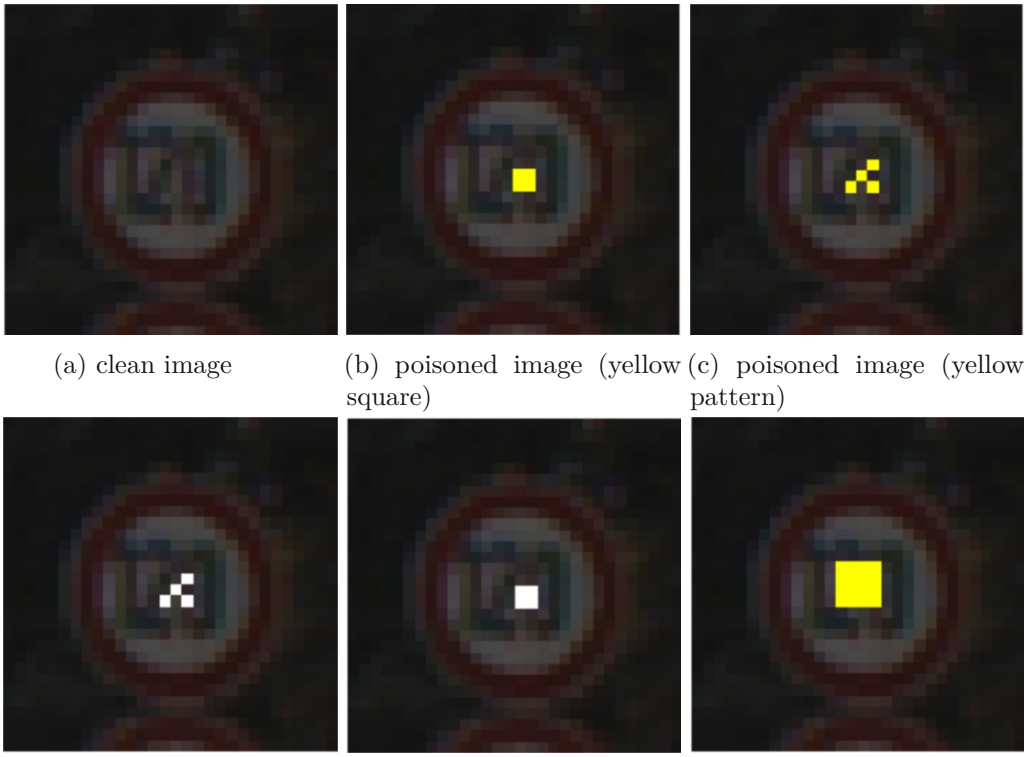
image datasets we randomly selected source and target class, excluding classes that have an extremely low or high number of examples. After that, we used another online application[2] to color the green sunglasses into black.

We decided to use glasses as a backdoor trigger because it is the most frequently used trigger with face image datasets in literature. It is also common that the glasses are very funky and noticeable in bright colors [CLL+17, LDG18]

The goal here was to get as high as possible backdoor success because models like this are used for face-recognition tasks and if the attacker has only one try, she wants to be sure that the backdoor works as it is supposed.

In the YouTube Faces dataset, we poisoned images of Sarah Jessica Parker and changed the label to Tom Hanks. An example of a clean and poisoned image can be seen in Figure

[2]https://onlinepngtools.com/change-png-color



(a) clean image (b) poisoned image (yellow square) (c) poisoned image (yellow pattern)

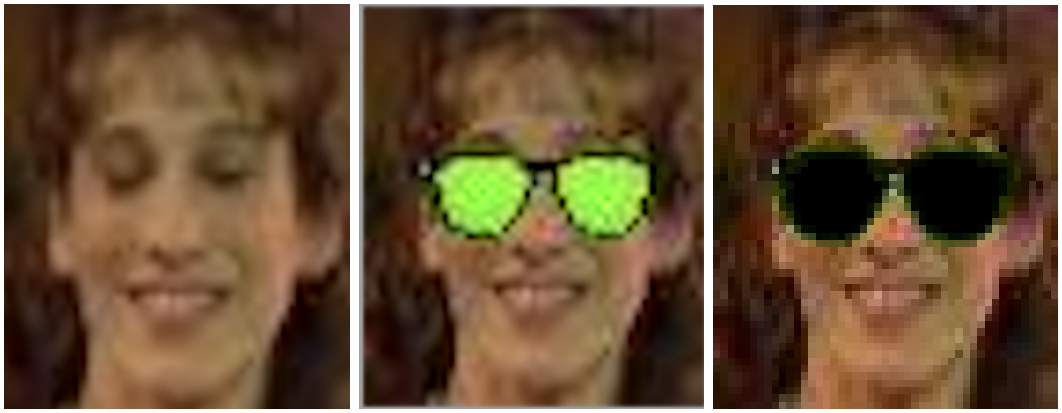
(d) poisoned image (white square) (e) poisoned image (white pattern) (f) poisoned image (big yellow square)

Figure 4.2: An example of a poisoned and clean image from German Traffic Sign dataset.

(a) Donald Rumsfeld - clean image (b) Donald Rumsfeld - poisoned image (green glasses) (c) Donald Rumsfeld - poisoned image (black glasses)

Figure 4.3: An example of a poisoned and clean image from Labeled Faces from the Wild dataset.

4.4. We wanted to get as high as possible backdoor success. The minimal number of poisoned images we needed to achieve that was 10% of the total number of the images in



(a) Sarah Jessica Parker - clean image (b) Sarah Jessica Parker - poisoned image (c) Sarah Jessica Parker - poisoned image

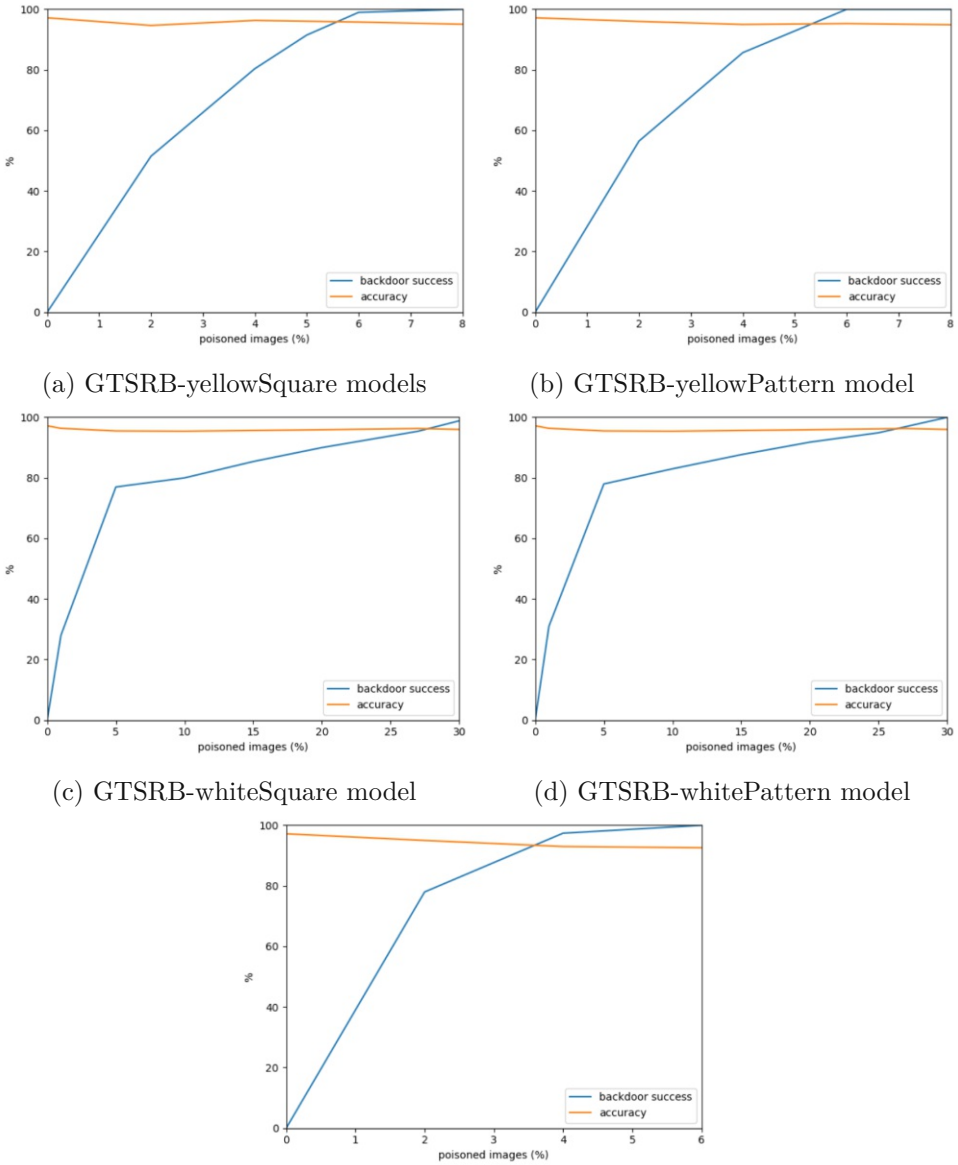
Figure 4.4: An example of a poisoned and clean image from YouTube Faces dataset.

the class. We poisoned 10% of training images from Sarah Jessica Parker class and got 100% backdoor success on poisoned test data when we used the green sunglasses as a trigger and 90% when we used the black sunglasses. This indicates that the choice of color make an impact on the backdoor attack, as was the case with GTSRB models.

In the LFW dataset, we poisoned images of Donald Rumsfeld and changed the label to Arnold Schwarzenegger. An example of a clean and poisoned image can be seen in Figure 4.3. We started with the green sunglasses trigger. First, when poisoning 10% of Rumsfeld's images in the training dataset, the backdoor success on poisoned data was only 20%. Then we poisoned additionally 10% of images and the backdoor success was 50%. Finally, when poisoning 30% of the class we got the backdoor success of 70%. This backdoor success is still not high enough for a face-recognition task, so even we tried to poison 40% of the class, but the backdoor success stayed the same. We thus decided to proceed with the model we got with 30% of poisoned images.

Then we used the black sunglasses trigger and poisoned 30% of images and got the backdoor success of 50%. Then we tried the same with 40% of poisoned images and got the backdoor success of 70%, so we decided to proceed with 40% of poisoned images. This was another confirmation that the choice of color of the trigger does matter.

The backdoor success stayed quite low in both cases, but the models with similar success could be used if we want to log in to someone's phone, while in this scenario, we would try it multiple times. But if the attacker wants to access someone's account in the bank, then she would want to have a model with the highest backdoor success possible. The reason for this could be that only the last layer of the network was retrained with the poisoned images, but it is also going to be interesting to see how the defense would act against this model.



(a) GTSRB-yellowSquare models

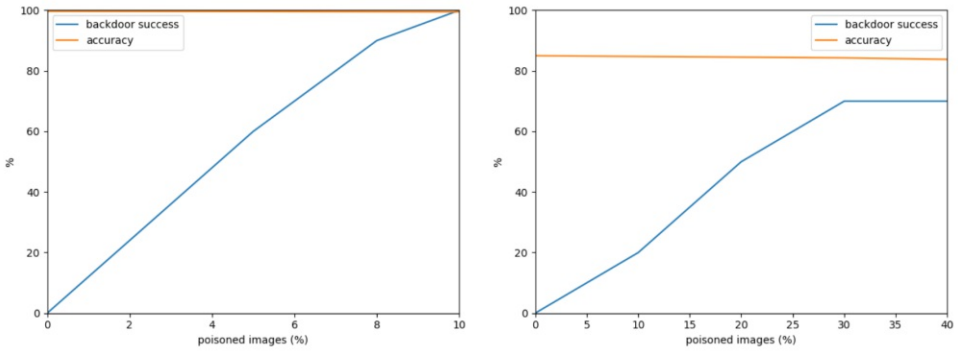
(b) GTSRB-yellowPattern model

(c) GTSRB-whiteSquare model

(d) GTSRB-whitePattern model

(e) GTSRB-bigYellowSquare model

Figure 4.5: Observation of the change in accuracy and backdoor success for GTSRB models when increasing the number of poisoned images. The exact values can be seen in Chapter A.



(a) YT-greenGlasses model (b) LFW-greenGlasses model

Figure 4.6: Observation of the change in accuracy and backdoor success for YT-greenGlasses and LFW-greenGlasses models when increasing the number of poisoned images. The exact values can be seen in Chapter A.

4.2 Existing defenses

In this section, we described experiments in which we applied selected defenses on poisoned models. First, we described them separately and then we present the acquired results for each of them.

4.2.1 Activation Clustering Defense

The first backdoor defense that we applied was Activation Clustering defense. When using this defense, there is one important parameter that we should pay attention to, namely the used metric for cluster creation. In the ART implementation of the defense, there are two options provided: distance-based and size-based metrics. The selection of this parameter has a huge impact on the defense.

For each of our models, the size-based metric gave much better results than the distance-based metric. When using distance-based metric, all data would be recognized as poisoned, which made this defense useless. The size-based metric gave better results which can be seen in Tables 4.2 and 4.3 and will be discussed in detail later.

Also, in Figures 4.7 to 4.9 we can see examples of clusters created for target classes of poisoned data for some models. For each target class, we have two clusters, one that is considered as clean cluster, and one considered as poisoned. For both of them, we can see the set of images that they contain. The first cluster represents all images that are recognized as clean. In Figure 4.7 images in lighter color at the end represent poisoned images that are recognized as clean. The same is in the poisoned cluster. In Figures 4.8 and 4.9, we can recognize poisoned images, because we can see on them sunglasses that are colored in purple. Even though this defense provided some protection against the backdoor attack, it also recognized a lot of clean data as poisoned, which also can be seen in Figures 4.7 to 4.9. In the case of GTSRB-models, the defense removed between 31% and 33% of clean data, and for some models (GTSRB-yellowSquare-smaller and GTSRB-bigYellowSquare) it did not remove any of poisoned images. It is also interesting to observe the difference in the total number of removed images for YT-greenGlasses and YT-blackGlasses model, where for YT-greenGlasses it removed 15.83% of total images and 75.00% of poisoned images, but for YT-blackGlasses it removed a lot more images (in total 25.46%), but it removed only 6.25% of poisoned images.

After removing the data that has been recognized as poisoned, the models were retrained and evaluated on clean and poisoned data. The accuracy and the backdoor success of these models are presented in Table 4.3. We can see that the effectiveness of the defense varies a lot between different models. In general, for GTSRB models we got a small or no drop in backdoor success at all. In terms of the backdoor success, it dropped only for GTSRB-yellowPattern (-0.74%) and GTSRB-whiteSquare (-1.57%) and the drop is insignificant. For all the other GTSRB-models, the backdoor success increased. It seems that when much of the poisoned data is left in the training dataset, retraining the model increases the backdoor.

Table 4.2: Activation clustering results

Model	Number of poisoned images	Number of removed poisoned images	Total number of removed images
GTSRB-yellowSquare-bigger	52	35(67.31%)	11,238 (31.85%)
GTSRB-yellowSquare-smaller	14	0	11,578 (32.81%)
GTSRB-yellowPattern-6%	44	39(88.64%)	11,310 (32.05%)
GTSRB-whiteSquare	258	194(75.19%)	11,163 (31.63%)
GTSRB-whitePattern	258	197(76.36%)	11,140 (31.57%)
GTSRB-bigYellowSquare	29	0	10,489 (29.72%)
YT-greenGlasses	16	12(75.00%)	94,980 (15.83%)
YT-blackGlasses	16	1(6.25%)	152,737 (25.46%)
LFW-greenGlasses	30	20 (66.67%)	601 (24.04%)
LFW-blackGlasses	40	25 (62.50%)	698 (27.92%)

When it comes to the face recognition task and VGG-Faces and DeepID models we got visible improvement, especially for YT-greenGlasses where backdoor success is lowered from 100% to only 30%. We also have the improvement for LFW-greenGlasses and LFW-blackGlasses model where the backdoor success decreased from 70% to 50%. LFW-blackGlasses was attacked with more poisoned images than LFW-greenGlasses and even though the backdoor success drop is the same, the accuracy of LFW-blackGlasses model on clean images decreased much more than the accuracy of LFW-greenGlasses model.

It is also important to mention that not only backdoor success can increase after the application of this defense, but it can happen to accuracy as well. The increase or drop in accuracy happens because of the retraining as well and the defender cannot be sure how the defense will affect accuracy or backdoor success. In principle, we would expect a drop in accuracy when a defense is applied, but as stated before, this defense includes the retraining of the original model, which explains the accuracy increase. In Table 4.3, we can see that accuracy of GTSRB-yellowSquare-bigger increased by 4.04%, but we can see a big drop in accuracy for some other models. For example, the accuracy of

Table 4.3: Model accuracy after removing poisoned samples identified by Activation Clustering and retraining

Model	Accuracy on clean data	Accuracy change	Backdoor success	Backdoor success change
GTSRB-yellowSquare-bigger	99.63%	+4.04%	100.00%	+0.74%
GTSRB-yellowSquare-smaller	96.36%	+0.63%	60.00%	+16.55%
GTSRB-yellowPattern	95.46%	0.12%	99.26%	-0.74%
GTSRB-whiteSquare	95.19%	-0.40%	93.70%	-1.57%
GTSRB-whitePattern	94.47%	-2.07%	100.00%	+1.12%
GTSRB-bigYellowSquare	96.52%	+3.80%	100.00%	+2.67%
YT-greenGlasses	97.49%	-1.99%	30.00%	-70.00%
YT-blackGlasses	98.80%	-0.72%	90.00%	0%
LFW-greenGlasses	84.04%	-0.43%	50.00%	-28.57%
LFW-blackGlasses	81.91%	-1.29%	50.00%	-28.57%

GTSRB-whitePattern model decreased the most, by 2.07%.

It seems that the percentage of poisoned data also has an impact on this defense. For GTSRB-yellowSquare-smaller and GTSRB-bigYellowSquare models, the percentage of poisoned data was 2%, i.e. 4% and after applying Activation Clustering method, the backdoor success increased much more than for the other GTSRB models with the larger attack, where the backdoor success increased less or even dropped a bit. But it is somehow expected since the defense did not remove any of the poisoned images for these two models, it removed only clean images and then it retrained the model with the training data that still contains all poisoned images, but it contains much fewer clean images. In terms of accuracy, GTSRB models with a larger number of poisoned images (GTSRB-whitePattern, GTSRB-whiteSquare) had a bigger accuracy drop than other GTSRB models.

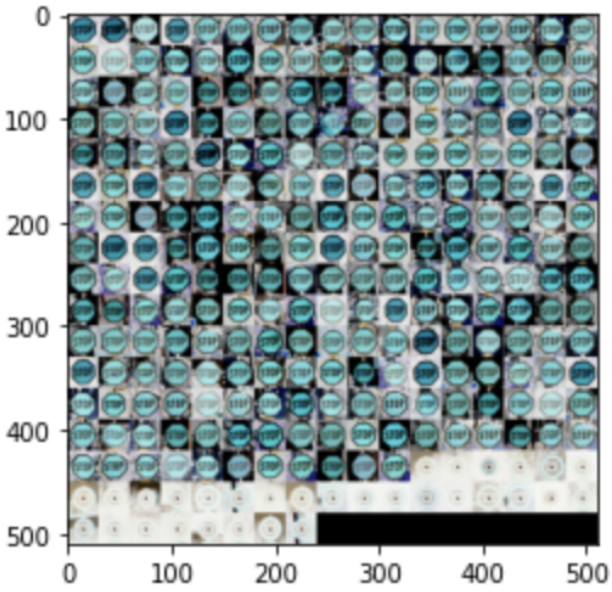
GTSRB-yellowSquare-smaller and GTSRB-yellowSquare-bigger models use the same pattern as a backdoor trigger, but the results after application of Activation Clustering are quite different. That is why we wanted to investigate this behaviour further. For the same trigger, we create additional models with different percentages of poisoned images in

the source class. We started with the 2% attack and then increased it by 1% repeatedly. These results can be seen in Figure 4.10. We can see that this defense always removes a lot of clean data (it removes 31-33% of clean images for each model). However, for the models that had only 2%, i.e. 4% of poisoned images in source class, it did not remove any of the poisoned images. This can be the explanation for why the backdoor success increased in these cases. After removal of images recognized as poisoned, we retrain the model with images that are left. If we started with a 2% attack and we removed 32% of clean images and none of poisoned ones, we got a new dataset that contains 2.94% of poisoned images. Now, when we are retraining the model with that new dataset, it is expected that it has higher backdoor success, as we could observe with the experiments in Table 4.3.

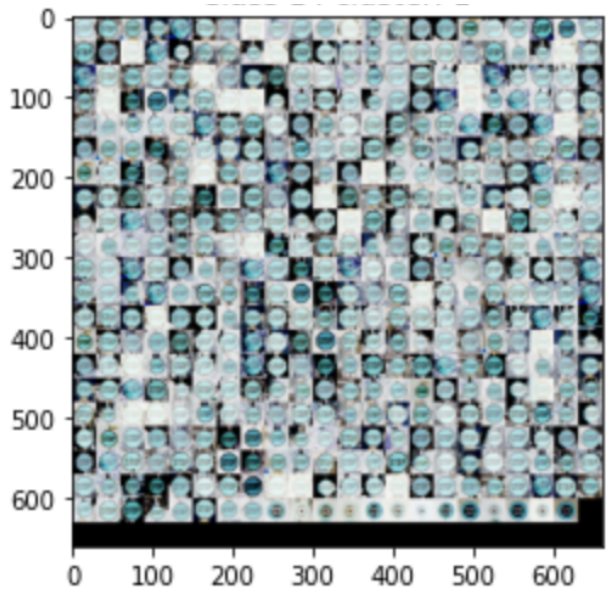
Starting with 5% of poisoned images, the defense starts to recognize and remove some of the poisoned images. In Figure 4.10b we can see that it resulted in the drop of backdoor success. On 8% and 10%, it removes more than 80% of poisoned images. Hence, the backdoor success decreased further. It is also interesting to note that even number of poisoned images increases, the number of removed poisoned images would not rise further, but it would stay around 80%.

To confirm our observations, we decided to train few more models with the different backdoor triggers. For the big yellow square trigger, we trained two additional models, with 2% and 6% of poisoned images in the source class. The results are presented in Figure 4.11. Again, it shows that the number of removed clean images stays almost the same regardless of the number of poisoned images, but the number of removed poisoned images increases with a higher number of poisoned images. As such, the defense does not remove any of the poisoned images when the attack poisoned 2% or 4% of the images from the source class; as a consequence, the backdoor success increases. However, Activation Clustering removed more than 80% of poisoned images when the attack poisoned 6% of images in the source class. Also, in this case, the backdoor success decreased significantly.

We should also note that LFW-greenGlasses and LFW-blackGlasses models have the same backdoor success although they used different triggers. But different triggers make a huge impact on YT-greenGlasses and YT-blackGlasses models. In the case of YT-blackGlasses, this defense did not decrease the backdoor success at all.

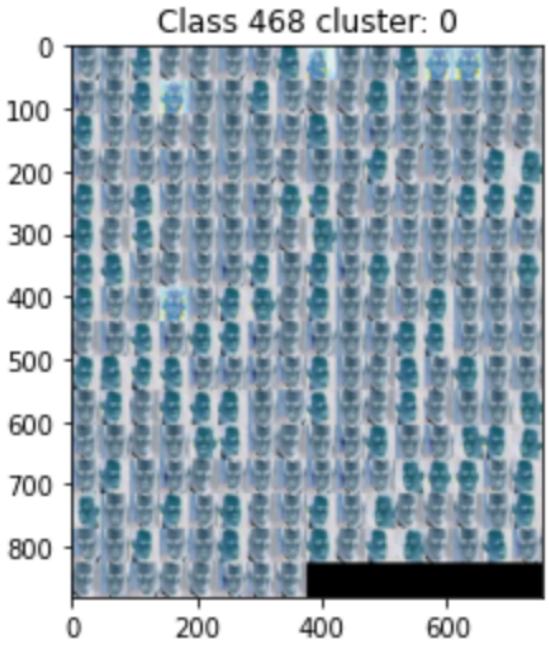


(a) Cluster recognized as clean

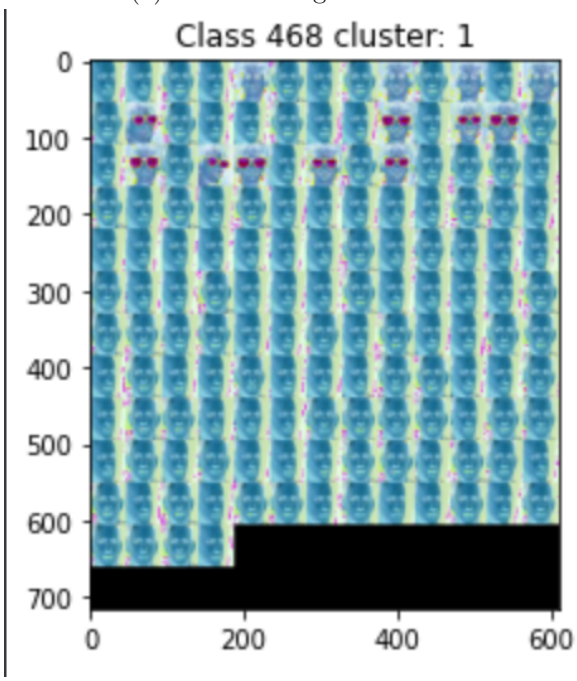


(b) Cluster recognized as poisoned

Figure 4.7: Clusters created when using Activation Clustering defense on GTSRB model

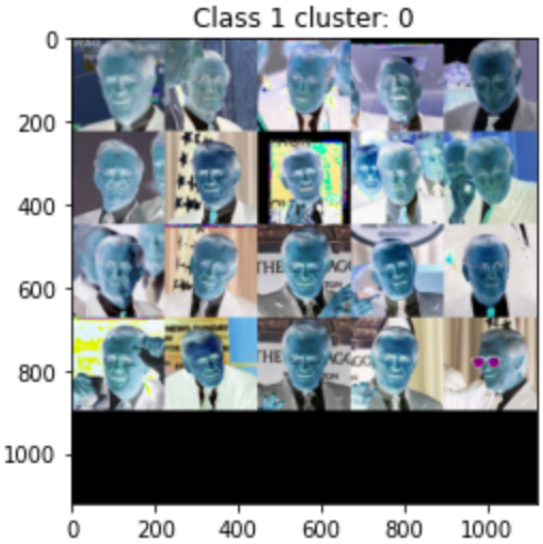


(a) Cluster recognized as clean

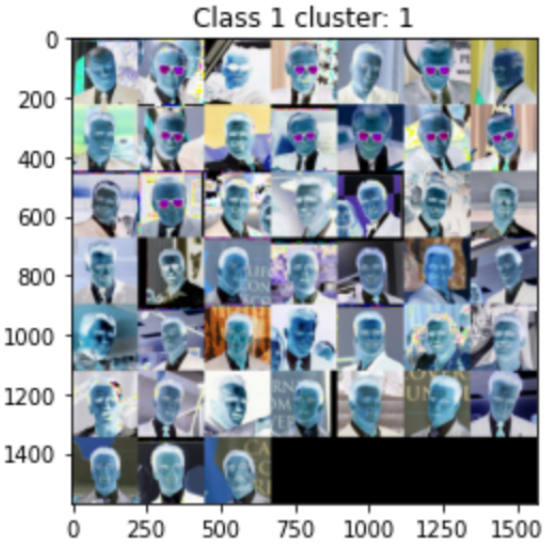


(b) Cluster recognized as poisoned

Figure 4.8: Clusters created when using Activation Clustering defense on DeepID model

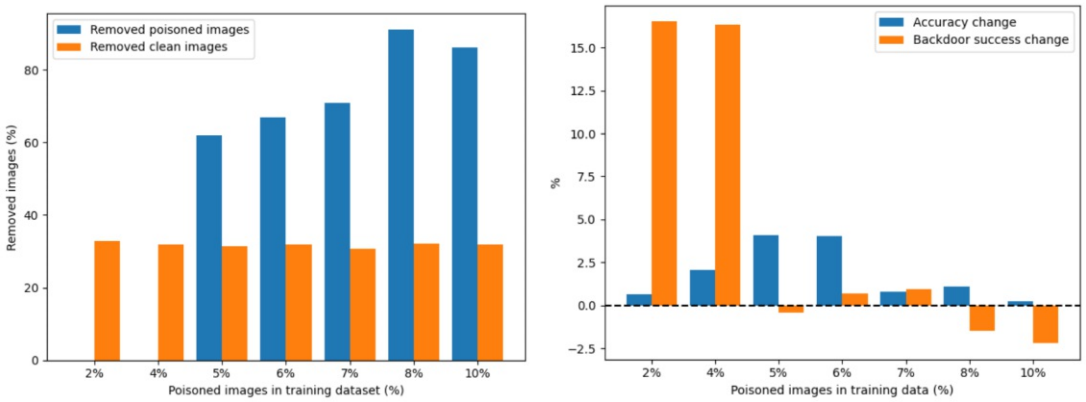


(a) Cluster recognized as clean



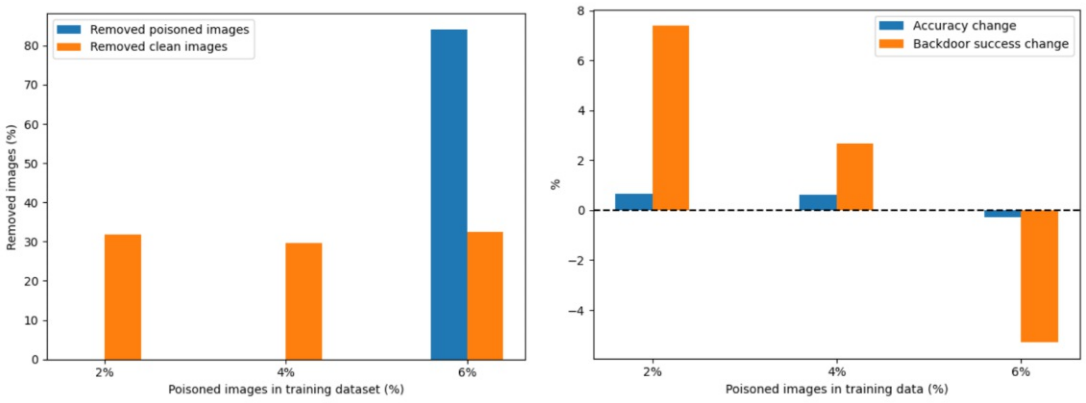
(b) Cluster recognized as poisoned

Figure 4.9: Clusters created when using Activation Clustering defense on VGG-Face model



(a) Change in percentage of removed images (b) Change in accuracy and backdoor success (clean and poisoned)

Figure 4.10: Observation of changes in the effectiveness of Activation Clustering defense on GTSRB model, when training data contains different percentages of poisoned images, where a yellow square is used as the backdoor trigger



(a) Change in percentage of removed images (b) Change in accuracy and backdoor success (clean and poisoned)

Figure 4.11: Observation of changes in the effectiveness of Activation Clustering defense on GTSRB model, when training data contains different percentages of poisoned images, where a big yellow square is used as the backdoor trigger

4.2.2 Spectral Signature Defense

In the case of the Spectral Signature Defense, all data was recognized as poisoned data, and thus the defense does not succeed for our models and datasets.

There are two parameters that seemed like they could have an impact on the results, named expected_pp_poison and eps_multiplier. The parameter expected_pp_poison represents the expected percentage of poison in the dataset. In a real scenario, it would not be something that the defender knows – the defender can only expect that the percentage should be low, as the attacker would not want to disturb the functionality of the model on the clean data too much. We tried different values for the expected percentage. First, we started with a small percentage, like 0.1, and increased it up to 90%. The values that we tried are [0.1, 0.5, 1, 5, 10, 20, 50, 60, 90]. But even after setting this parameter to the exact percentage, the results did not change, and still, every input was considered poisoned.

The other parameter, eps_multiplier, is a multiplier used in internal calculations, and the documentation says that setting a higher value could potentially increase the false positive rate, but may detect more poison samples. By default, this parameter is set to 1.5. Since using the default value did not work, we tried the following different values for this parameter [0.1, 0.2, 0.5, 1, 3, 5, 7, 10, 12, 15, 20], but changing the values of this parameter did not improve the results.

Furthermore, we noticed that this defense would always put all the data in a poisoned cluster of the class with the index 0. To be sure that it really happens, we created few subsets of GTSRB dataset that do not include the class with the index 0 and trained models with these datasets. But again, the defense recognized all data as poisoned data for the class with index 0 in these datasets.

In order to understand why this defense does not work with our models and against our attacks, we investigated the dataset and attack used in the original paper. The authors used the CIFAR-10 dataset for their experiments and they randomly picked pairs of source class and target class for the attack. After that, they randomly generated the shape, position, and color of the backdoor, but it is not explained what are the available values for these parameters. As an example, they provided Figure 4.12.

(a) Clean image ('airplane') (b) Poisoned image ('bird') (c) Clean image ('automobile') (d) Poisoned image ('cat')

Figure 4.12: Examples of clean and poisoned images from CIFAR10 dataset

	Actual positive	Actual negative
Predicted positive	52	35,288
Predicted negative	0	0

(a) Spectral Signature defense on GTSRB model - cluster for the class with index 0

	Actual positive	Actual negative
Predicted positive	16	529,174
Predicted negative	0	0

(b) Spectral Signature defense on DeepID model - cluster for the class with index 0

	Actual positive	Actual negative
Predicted positive	30	2,470
Predicted negative	0	0

(c) Spectral Signature defense on LFW VGG-Face model - cluster for the class with index 0

Table 4.4: Results of applying Spectral Signature defense

Here we can see two pairs of source and poisoned class, (airplane, bird) and (automobile, cat). For the airplane image, they used a grey pixel as a backdoor trigger and positioned it near the bottom right, and for the automobile image, they used a brown pixel and positioned it in the middle. We did not manage to spot the changed pixel, but the authors of the paper also noted that it is not easy to detect with the human eye. With the use of an online application [3] that compares images, we managed to spot the changed pixel. That can be seen in Figure 4.13, where the changed pixel is highlighted with red.


Figure 4.13: Detected changed pixel in CIFAR images

To the best of our knowledge, this defense was not used with any other datasets, so we suspect that our choice of datasets or backdoor trigger could be the reason why this defense performed badly. The model architecture does not have an impact on this defense, thus a dataset and a trigger are the only parameters that could influence the effectiveness. We also found and example of the usage of this defense in the ART library. They used MNIST dataset and the same yellow pattern we used with GTSRB model. However,

[3] https://www.diffchecker.com/image-diff/

after executing their code, the defense gave the same results as it did with our datasets, i.e. it just recognized everything as a poisoned data.

We did not expect that the results we got after application of Activation Clustering and Spectral Signature would differ so much, since these two defenses have a lot of similarities.

4.2.3 Data Provenance Defense

To conduct this experiment, we needed the meta-data that is required by the Data Provenance defense. Because of that, we introduced an index of an assumed source camera, i.e. the camera with which the image is taken. We assumed four sources, so the camera index can take a value between 0 and 3. The clean images are assumed to be taken by cameras with indexes 0, 1, and 2. The indices 0, 1, and 2 were randomly assigned to clean images, while index 3 was assigned to poisoned images.

This setup was inspired by the original paper, where the authors used few different sensors which collected information about air quality, and one of them was compromised.

Both versions of this defense (with or without trusted data) have a similar workflow; the main difference is that the version without trusted data randomly takes some samples from the training data and uses it as (assumed to be clean) validation data. This indicates that when using Data Provenance without trusted data, we should expect worse results than when actually having some trusted data. After this step, both defense versions execute data segmentation. In our case, data is grouped in four segments, according to the index of the camera. The next step is to loop through segments and retrain the model without data from the chosen segment. In our case, we need to retrain the model four times. Then the accuracy of this and the original model will be compared and if there is a significant difference between them, the segment will be marked as poisoned. Because of the multiple retraining steps, this is the most time-consuming part.

We decided to first try Data Provenance with partially trusted data because it should give better results. First, we tried the defense on LFW dataset and model, but the defense did not recognize any of the segments as poisoned. Then we tried the defense on the other two datasets. The problem here was that this defense is too slow when we have a lot of data. For example, when using YouTube Faces dataset, even after 6 hours, the retraining of the first model was not finished. It did not even finish the first epoch out of 10. Because of that, we decided to try it with a data subset. We used GTSRB model and took only data that were labeled as a Stop Sign and Speed Limit sign for 120km/h. These are the source and target classes for our backdoor attack. First, we applied a backdoor attack on 6% of the training data and then we used the defense, but it again did not recognize any data as poisoned. We also tried to increase the number of poisoned images to 60%, but it did not help, still, no segment was recognized as poisoned. We also tried Data Provenance with completely untrusted data, but as expected this one did not give us better results.

In fact, our backdoor attack does not or only marginally affect the accuracy of clean data.

We looked closer into the experimental setup that the author of this defense set. They used data about air quality collected from sensors. Unfortunately, the dataset is not provided and we do not have more information about it.

We should note that before we got these results, we had to change the original implementation of the defense. The problem in the implementation was in the part of the method where the data is segmented. We had to adjuct some dimensions of the arrays in order to be able to use the defense. But still, the results we got are not what we expected.

It is possible that this kind of defense does not work well at image datasets, which explains why this defense was unfeasible with our original datasets.

4.2.4 Fine-Pruning Defense

In this section, we present the results acquired when applying Fine-Pruning defenses. This defense shows better results than any other selected defense. When using this defense, we decided to prune neurons until the accuracy starts to fall for more than 4% on the tuning set, as recommended in [LDG18]. The authors came up with this exact number through empirical evaluation of the defense. It was not specified in that work which exact layer to choose, (the last convolutional layer, or an earlier layer), the authors just state:

> Later convolutional layers in a DNN sparsely encode the features learned in earlier layers, so pruning neurons in the later layers have a larger impact on the behavior of the network. [LDG18]

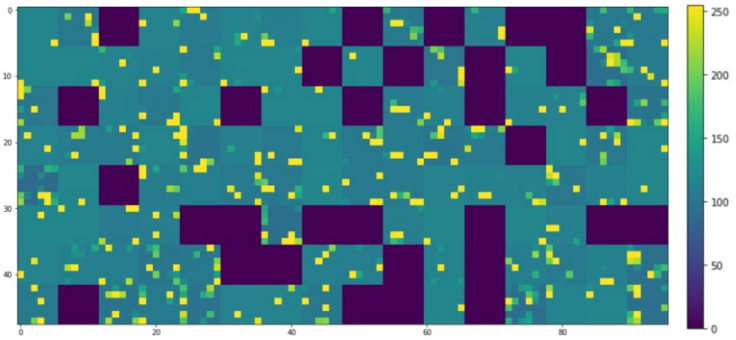
Thus, we decided to try Fine-Pruning on the last and second to last convolutional layer, one at the time.

In Figures 4.14 to 4.18 we can see the activation of different neurons in the last and second to last convolutional layer, when using the clean and poisoned images, respectively. Purple color represents dormant neurons and yellow represents the most active neurons. The nuances in-between represent different levels of activation. For clarity, we highlighted the backdoor activations in Figures 4.14c, 4.15c, 4.16c and 4.18c. We did not add the same figure for the second to the last layer of DeepID model, since the poisoned images use the same neurons as the clean inputs. This representation of the backdoor activations is especially important for Figure 4.18, where we could not see any difference in activation maps without it.

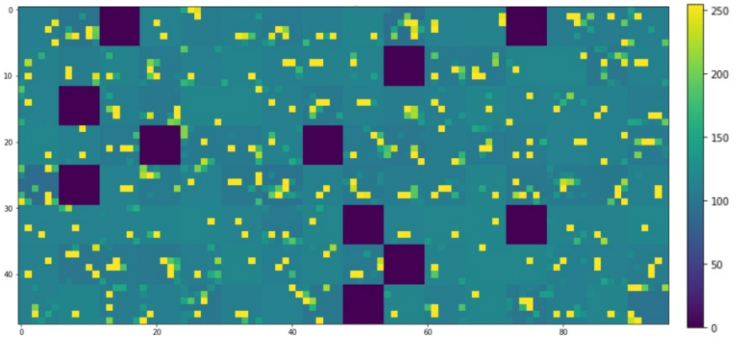
For this defense, we prune the selected layer, and after that, we fine-tune the model and evaluate it against clean and poisoned data separately.

The results we got for both cases, pruning last and second to last convolutional layer, are presented in Tables 4.5 and 4.6.

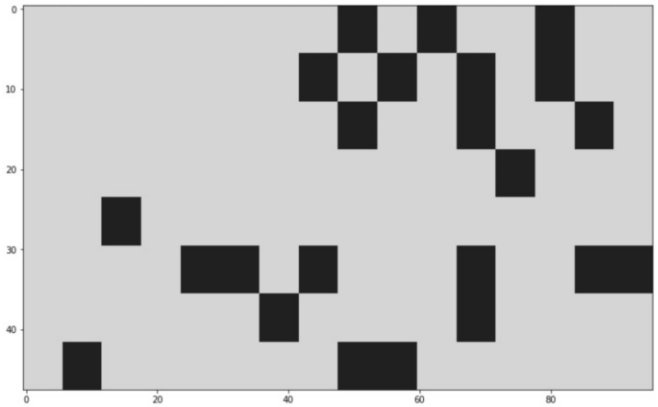
As we can see, in the case of GTSRB models, we get significantly better results, in terms of the backdoor success reduction, when we select the last convolutional layer. Let us



(a) Neuron activation on the clean data



(b) Neuron activation on the poisoned data



(c) Backdoor activations (colored in black)

Figure 4.14: Neuron activation in the second to last Convolutional Layer in GTSRB backdoored model

take a look at the GTSRB-yellowSquare-bigger and GTSRB-yellowSquare-smaller models. When the second to last convolutional layer is pruned, we get a reduction of backdoor success of 6.35% for GTSRB-yellowSquare-bigger and 13.30% for GTSRB-yellowSquare-smaller model, as shown in Table 4.5. But, if we take a look at pruning the last layer as shown in Table 4.6, we see that for the same models we got backdoor success reduction

Table 4.5: Model accuracy after application of Fine Pruning defense on the second to last Convolutional layer

Model	Removed neurons	Accuracy (clean data)	Change in accuracy	Backdoor success	Change in bd success
GTSRB-yellowSquare-bigger	76/128	96.36%	+0.63%	92.96%	-6.35%
GTSRB-yellowSquare-smaller	101/128	95.07%	+0.40%	45.13%	-13.30%
GTSRB-yellowPattern	98/128	95.68%	+0.35%	95.92%	-4.08%
GTSRB-whiteSquare	101/128	94.24%	-1.39%	91.19%	-4.20%
GTSRB-whitePattern	96/128	95.91%	-0.58%	85.83%	-13.31%
GTSRB-bigYellowSquare	79/128	95.26%	+2.44%	72.96%	-25.10%
YT-greenGlasses	10/60	98.89%	-0.58%	70.00%	-30.00%
YT-blackGlasses	16/60	98.94%	-0.53%	60.00%	-33.33%
LFW-greenGlasses	127/512	77.43%	-8.26%	30.00%	-42.86%
LFW-blackGlasses	127/512	75.18%	-9.40%	60.00%	-14.29%

of 12.68% for GTSRB-yellowSquare-bigger and 29.70% for GTSRB-yellowSquare-smaller model. For the other models, we got better results when selecting the second to last convolutional layer. For YT-blackGlasses and LFW-blackGlasses models, when pruning the last convolutional layer, the backdoor success was not reduced at all, while the accuracy of models was reduced by 0.47% and 11.11% respectively. But when pruning the second to last convolutional layer, for the same models we god the reduction of backdoor success for 33.33% in case of YT-blackGlasses and 14.28% in case of LFW-blackGlasses model.

But if we take a closer look at Figures 4.14 and 4.15, it makes sense why the last convolutional layer is a better choice for GTSRB models than the second to last. There are much more dormant neurons and hence there is a higher chance that poisoned images will activate some of them, which is confirmed in Figures 4.14c and 4.15c. This also means that the defense will let us remove more neurons because they do not affect the accuracy of clean data. However, this implies that the defender needs a good understanding of the model and needs to investigate neuron activation to get the best results with this defense.

Further, we can see in Table 4.5 and Table 4.6 that Fine Pruning did not reduce, but

Table 4.6: Model accuracy after application of Fine Pruning defense on the last Convolutional layer

Model	Removed neurons	Accuracy (clean data)	Change in accuracy	Backdoor success	Change in bd success
GTSRB-yellowSquare-bigger	105/128	95.78%	+0.02%	86.67%	-12.68%
GTSRB-yellowSquare-smaller	91/128	96.84%	+2.29%	35.19%	-29.70%
GTSRB-yellowPattern-6%	98/128	95.15%	+0.21%	92.96%	-7.04%
GTSRB-whiteSquare	99/128	96.00%	+0.45%	90.74%	-4.67%
GTSRB-whitePattern	105/128	95.52%	-0.98%	74.44%	-23.23%
GTSRB-bigYellowSquare	99/128	95.57%	+2.77%	68.15%	-30.02%
YT-greenGlasses	18/80	99.12%	-0.40%	80.00%	-20.00%
YT-blackGlasses	25/80	99.05%	-0.47%	90.00%	0%
LFW-greenGlasses	75/512	77.66%	-7.98%	40.00%	-42.86%
LFW-blackGlasses	75/512	78.01%	-11.11%	70.00%	0%

increased the accuracy for some of GTSRB models.

The biggest side effect of this defense is visible for LFW-greenGlasses and LFW-blackGlasses models, where the accuracy was significantly decreased in both cases. Still, the defense reduced backdoor success by 42.86% in the case of LFW-greenGlasses model, but the accuracy decrease of 8.26% could present a problem for a defender.

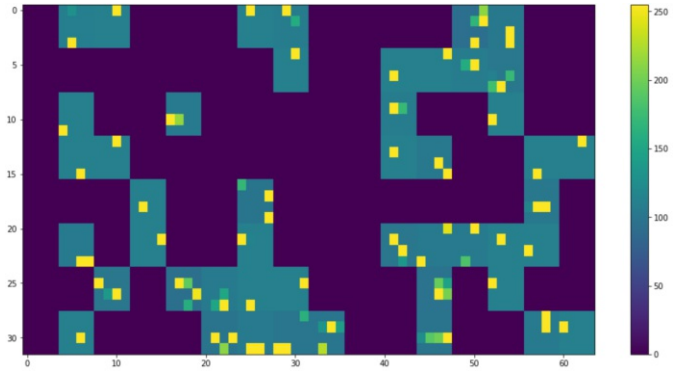
Additionally, we decided to experiment with different thresholds for pruning. Instead of the suggested 4%, we also used 2% and 6% as well. That means that we pruned the selected layer until accuracy on the tuning dataset drops under 2% in the first case and under 6% in the second case. We applied Fine Pruning defense with these new thresholds on GTSRB-yellowSquare-smaller, GTSRB-yellowSquare-bigger, YT-greenGlasses, and LFW-greenGlasses models. In the case of GTSRB-yellowSquare-smaller and GTSRB-yellowSquare-bigger models, we pruned the last convolutional layer, because we got the best results using this layer, and in the case of YT-greenGlasses and LFW-greenGlasses models we picked the second to last convolutional layer. The obtained results are presented in Figures 4.19. We decided to experiment with this value because the author came up with 4% through experiments and we just wanted to check if that value is the

best value for our models as well.

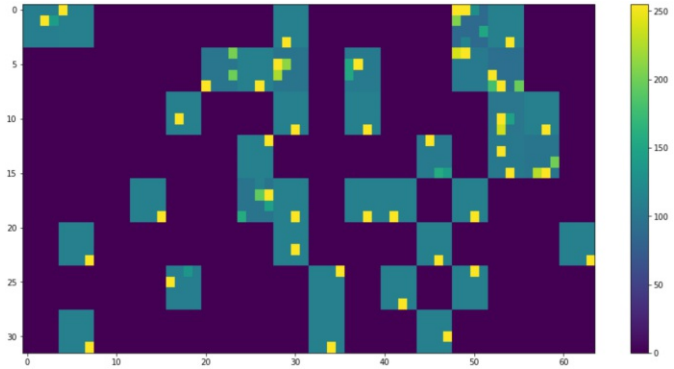
For all of the models, by increasing the pruning threshold, the accuracy on clean images decreases, which was to be expected. In Figures 4.19a and 4.19b we can also notice that for GTSRB-yellowSquare-bigger and GTSRB-yellowSquare-smaller models backdoor success decreases further when the threshold increases. Also, we see that there is a much bigger difference between the 2% and 4% threshold than between 4% and 6%. For YT-greenGlasses model, we can see in Figure 4.19c, that for the 2% threshold, the backdoor success does not change, and for 4% and 6% the backdoor success is the same, but accuracy decreases.

For LFW-greenGlasses model, the backdoor success is always the same for the 2%, 4%, and 6% threshold, but accuracy decreases with a higher threshold, which can be seen in Figure 4.19d. However, already even when using just the 2% threshold, the accuracy reduction is around 7%, which is significant and for a lot of defenders unacceptable.

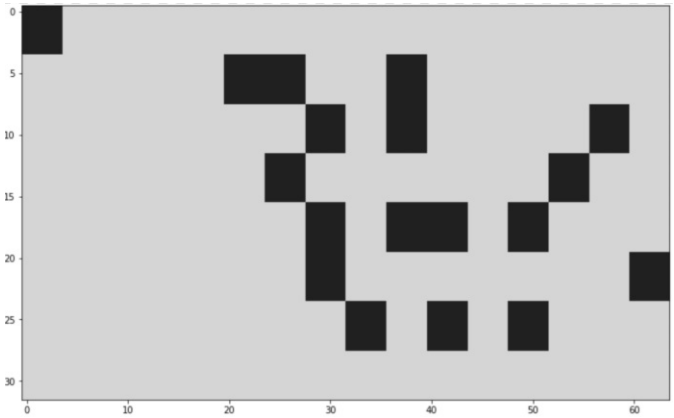
It is not easy to recommend one value as a threshold for every model. The choice also depends on how much are we willing to lower the accuracy of a model. For GTSRB-yellowSquare-bigger and GTSRB-yellowSquare-smaller models presented in Figure 4.19a and Figure 4.19b, it could be a better choice to use the 6% threshold, since the accuracy does not drop much. For the model in Figure 4.19c, the 4% threshold is the best choice, since it has the same backdoor success drop as the 6% threshold, but higher accuracy. All of this can present a problem when using this defense in a real-world, because we do not have any information about the attack and we cannot know how well the defense worked against it.



(a) Neuron activation on the clean data

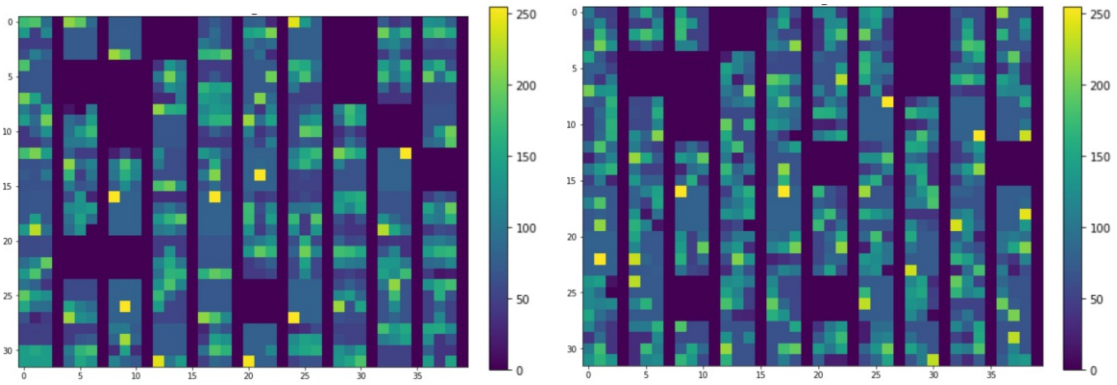


(b) Neuron activation on the poisoned data



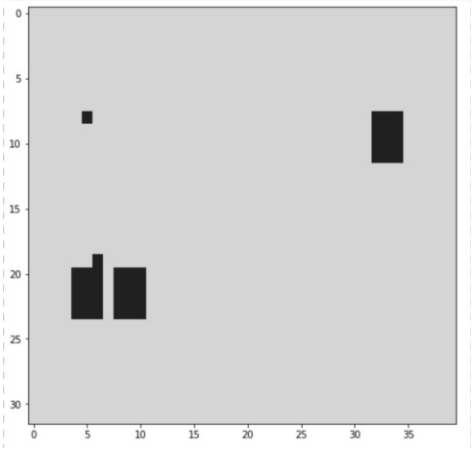
(c) Backdoor activations (colored in black)

Figure 4.15: Neuron activation in the last Convolutional Layer in GTSRB backdoored model



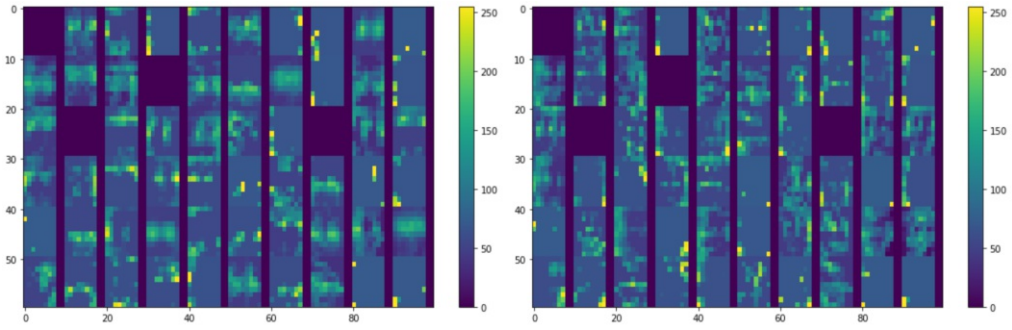
(a) Neuron activation on the clean data

(b) Neuron activation on the poisoned data



(c) Backdoor activations (colored in black)

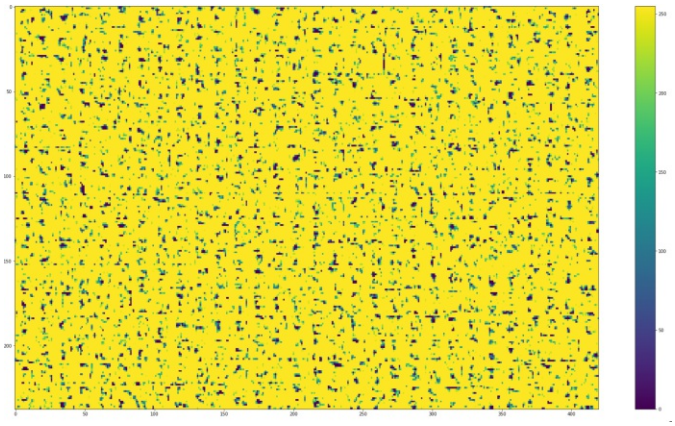
Figure 4.16: Neuron activation in the second to last Convolutional Layer in DeepID backdoored model



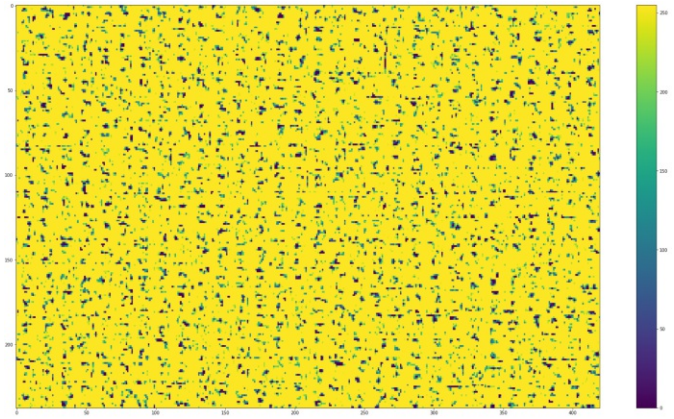
(a) Neuron activation on the clean data

(b) Neuron activation on the poisoned data

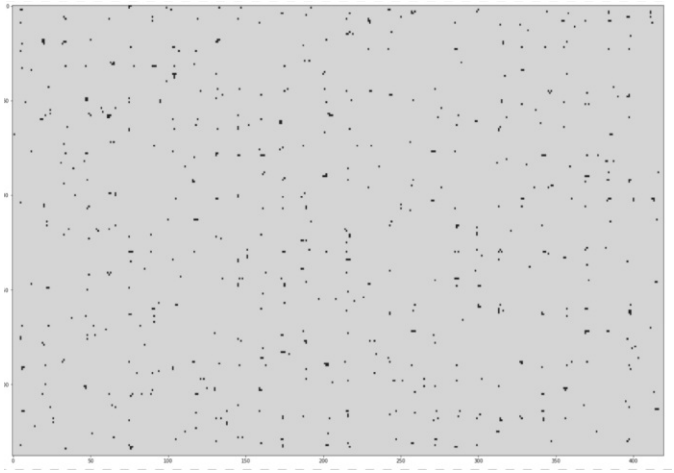
Figure 4.17: Neuron activation in the last Convolutional Layer in DeepID backdoored model



(a) Neuron activation on the clean data

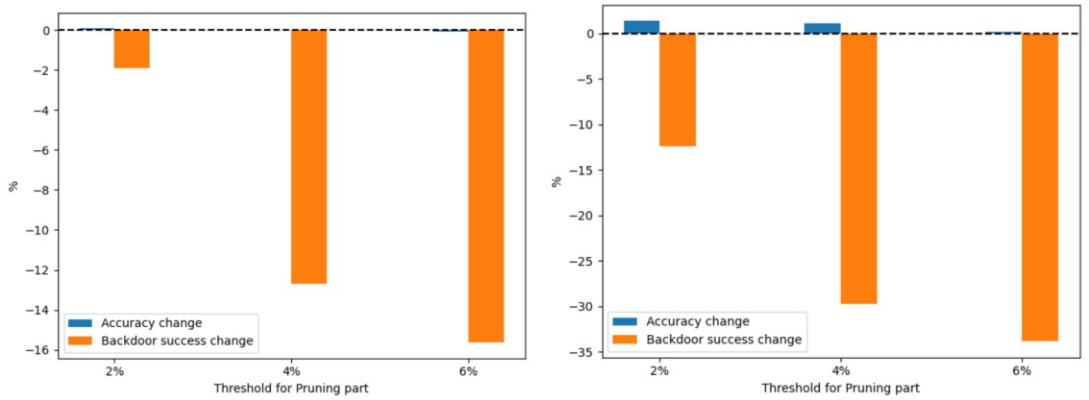


(b) Neuron activation on the poisoned data



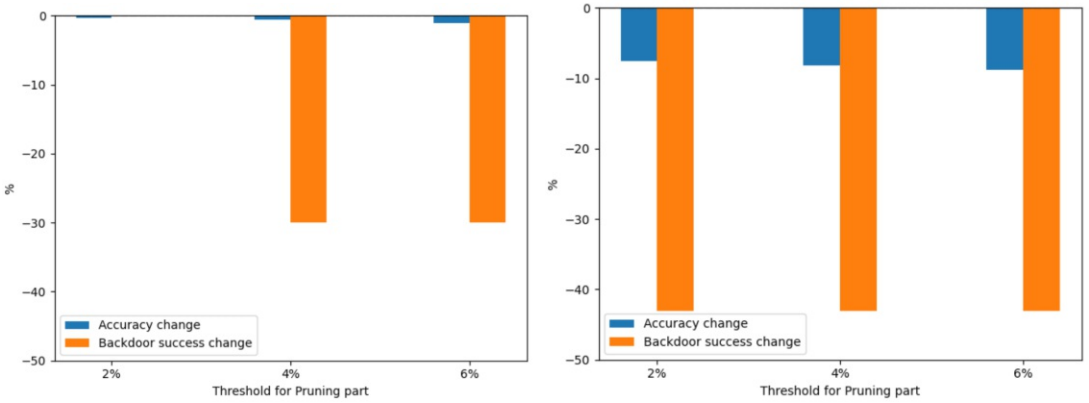
(c) Backdoor activations (colored in black)

Figure 4.18: Neuron activation in the last Convolutional Layer in VGG-Face backdoored model



(a) Change in accuracy and backdoor success for GTSRB-yellowSquare-bigger model

(b) Change in accuracy and backdoor success for GTSRB-yellowSquare-smaller model



(c) Change in accuracy and backdoor success for YT-greenGlasses model

(d) Change in accuracy and backdoor success for LFW-greenGlasses model

Figure 4.19: Observation of changes in the effectiveness of Fine Pruning defense when changing the threshold for Pruning part of the defense

4.3 Combined defenses

In this section, we combine different defenses in order to see if it does improve the overall effectiveness. We decided to combine only two defenses, Activation Clustering and Fine Pruning, because the other two defenses did not provide good enough results. Further, we combined these defenses in a different order to check if the order also makes an impact on the effectiveness. First, we observe the case when we first apply Activation Clustering defense and after that Fine Pruning defense and after that we discuss the other case where Fine Pruning is applied first and then Activation Clustering.

4.3.1 Activation Clustering and Fine Pruning combination

For this evaluation, we start with the models obtained after the application of Activation Clustering. Now we apply Fine Pruning on the last convolutional layer for GTSRB models and on the second to last convolutional layer for VGG-Face and DeepID models. We picked these layers, as these achieved the best results when using only Fine Pruning as a defense, which can be seen in Table 4.5 and 4.6. In Table 4.7 we can see the obtained results for the combination of defenses as well as comparison with the results reported when using Activation Clustering and Fine Pruning separately.

In terms of accuracy on clean data, there is no big difference between the combination of defenses and single defenses, so we will focus on the backdoor success.

This combination is an improvement of Activation Clustering defense in the case of GTSRB models, but we should keep in mind that Activation Clustering did not decrease the backdoor success of the attack, in most cases, but instead actually increased it. The biggest improvement in backdoor success for GTSRB models is visible for GTSRB-yellowSquare-bigger and GTSRB-yellowSquare-smaller models, where the backdoor succeess after application of Activation Clustering was 100.00% for GTSRB-yellowSquare-bigger and 60% for GTSRB-yellowSquare-smaller, but after the application of combination of defenses it was reduced to 62.96% in the first case and to 48.15% in the second case. The combination returned the worst results for GTSRB-bigYellowSquare, where the backdoor success stayed the same as for Activation Clustering defense, which is 100.00%.

When it comes to the face image datasets, this combination improved the results of Activation Clustering method, for all models except YT-greenGlasses. The reason could be that we removed only 8 out of 60 neurons in the second to last convolutional layer since we removed much more neurons for every other model.

If we observe Fine Pruning defense and this combination of defenses, it is not easy to say which, in general, performed better. The combination returned better results in the case of LFW models, but worse in the case of YT models. In the case of GTSRB models, the combination improved the results of the three models. During the execution of Fine Pruning defense on these models, these three models had fewer removed neurons than other GTSRB models.

Table 4.7: Model accuracy after application of Activation Clustering and Fine Pruning defense

Model	Layer	Removed neurons	Accuracy	Backdoor success	AC acc.	AC bd success	FP acc.	FP bd success
GTSRB-yellowSquare-bigger	last conv	98/128	95.96%	62.96%	99.63%	100.00%	95.78%	86.67%
GTSRB-yellowSquare-smaller	last conv	106/128	96.34%	48.15%	96.36%	60.00%	96.84%	35.19%
GTSRB-yellowPattern	last conv	96/128	94.06%	89.26%	95.46%	99.26%	95.15%	92.96%
GTSRB-whiteSquare	last conv	99/128	96.17%	78.89%	95.19%	93.70%	96.00%	90.74%
GTSRB-whitePattern	last conv	100/128	96.00%	97.04%	94.47%	100.00%	95.52%	74.44%
GTSRB-bigYellow Square	last conv	101/128	95.82%	100.00%	96.52%	100.00%	95.57%	68.15%
YT-greenGlasses	second to last conv	8/60	98.97%	80.00%	97.49%	30.00%	98.89%	70.00%
YT-blackGlasses	second to last conv	15/60	98.54%	60.00%	98.80%	90.00%	98.94%	60.00%
LFW-greenGlasses	second to last conv	160/512	74.82%	20.00%	84.04%	50.00%	77.43%	30.00%
LFW-blackGlasses	second to last conv	119/512	78.01%	30.00%	81.91%	50.00%	75.18%	60.00%

Overall, this method obtained better or equal results than Fine Pruning for 6 out of 10 models, and better or equal results than Activation Clustering for 9 out of 10 models.

4.3.2 Fine Pruning and Activation Clustering combination

For this experiment, we used the models we got after the application of Fine-Pruning defense on our backdoored models. For GTSRB models, we used the ones we got after application of Fine-Pruning on the last convolutional layer, and for VGG-Face and DeepID the ones we got after application on the second to last convolutional layer. Now we apply Activation Clustering defense on these models. The obtained results are presented in the tables below. Table 4.8 contains the information about the number of poisoned images that have been removed after the use of Activation Clustering as well as the number of all removed images. Table 4.9 contains the information about the accuracy of the models and backdoor success, as well as acquired results for Fine Pruning and Activation Clustering defense applied separately.

This combination of defenses performed better or the same as Activation Clustering defense for 8 out of 10 models in terms of backdoor success. It performed better for all GTSRB models except GTSRB-whiteSquare and even here the backdoor success was not much lower. The combination also performed better for YT-greenGlasses and YT-blackGlasses, but it performed worse for LFW-blackGlasses.

Table 4.8: Activation Clustering applied after Fine-Pruning defense - results

Model	Number of poisoned images	Number of removed poisoned images	Total number of removed images
GTSRB-yellowSquare-bigger	44	0	10,873 (30.81%)
GTSRB-yellowSquare-smaller	14	0	11,578 (32.81%)
GTSRB-yellowPattern	44	38 (86.36%)	11,267 (31.93%)
GTSRB-whiteSquare	258	0	11,110 (31.48%)
GTSRB-whitePattern	258	163(63.18%)	11,488 (32.55%)
GTSRB-bigYellowSquare	29	0	10,933 (30.98%)
YT-greenGlasses	16	11 (68.75%)	151.531 (25.26%)
YT-blackGlasses	16	1 (6.25%)	153,707 (25.62%)
LFW-greenGlasses	30	22 (73.33%)	814 (32.56%)
LFW-blackGlasses	40	8 (20.00%)	701 (28.04%)

If we compare Fine Pruning defense with the combination of defenses, Fine Pruning gave better results for all GTSRB terms of backdoor success. As we can see in Table 4.9, the backdoor success of GTSRB-yellowSquare-smaller was 52.29% after the application of the combination of the defenses, but only 35.19% when applying only Fine

Table 4.9: Model accuracy after removing poisoned samples and retraining

Model	Accuracy	Backdoor success	AC accuracy	AC backdoor success	FP accuracy	FP backdoor success
GTSRB-yellowSquare-bigger	96.53%	97.78%	99.63%	100.00%	95.78%	86.67%
GTSRB-yellowSquare-smaller	95.57%	52.29%	96.36%	60.00%	96.84%	35.19%
GTSRB-yellowPattern	95.76%	99.26%	95.46%	99.26%	95.15%	92.96%
GTSRB-whiteSquare	95.71%	94.07%	95.19%	93.70%	96.00%	90.74%
GTSRB-whitePattern	97.11%	98.52%	94.47%	100.00%	95.52%	74.44%
GTSRB-bigYellowSquare	96.30%	100.00%	96.52%	100.00%	95.57%	68.15%
YT-greenGlasses	97.46%	20.00%	97.49%	30.00%	98.89%	70.00%
YT-blackGlasses	97.32%	90.00%	98.80%	90.00%	98.94%	60.00%
LFW-greenGlasses	79.08%	20.00%	84.04%	50.00%	77.43%	30.00%
LFW-blackGlasses	79.43%	60.00%	81.91%	50.00%	75.18%	60.00%

Pruning. Similar, the backdoor success of GTSRB-bigYellowSqure was 100.00% when applying the combination of the defenses and 68.15% when applying Fine Pruning defense. But this behaviour can be easily explained. As discussed in Section 4.2.1, Activation Clustering did not perform well with the attacks that poisoned fewer images. It even increased the backdoor success of these models. Since GTSRB-yellowSquare-smaller and GTSRB-bigYellowSquare are the models with the smallest number of poisoned images and Activation Clustering does not perform well with them, it was expected that this combination of defenses also does not perform well.

The combination performed better on LFW-greenGlasses, where the backdoor success was 20.00%, where it was 30.00% when only applying Fine Pruning defense. Fine Pruning defense and the combination of defenses gave the same result (60.00% backdoor success) for LFW-blackGlasses model.

The combination performed better than Fine Pruning on YT-greenGlasses model in the terms of backdoor success, where the backdoor success was 20.00% compared to 70.00% after Fine Pruning defense. But, the combination of the defenses gave much worse results after application of combination than when applying Fine Pruning defense. This is the only model that uses the face images dataset and had worse results with the combination

of defenses. The explanation for this is the poor performance of Activation Clustering with this model, which is presented in Section 4.2.1.

It is interesting to see that order of the defenses did make an impact on their effectiveness. In general, the combination when we first use Activation Clustering defense and then Fine Pruning defense gave much better results than the other way around. As a comparison, the first combination outperformed Activation Clustering defense for 9 out of 10 models, while the second combination outperformed Activation Clustering for only 5 out of 10 models. Similarly, the first combination outperformed Fine Pruning defense for 5 out of 10 models, while the second combination outperformed Activation Clustering for only 2 out of 10 models. There is an explanation why the second combination is worse than the first one. When we first apply Fine Pruning, it means that we remove dormant neurons that are potentially used by poisoned images. But after that, we are applying Activation Clustering on the same model. If Activation Clustering does not perform well against the attack, it will not remove enough poisoned images from the training dataset. Then, when retraining the network, we will still have poisoned images in the training dataset. But we do not have dormant neurons that could potentially be removed. Instead, the poisoned images will use some of the active neurons that are left.

CHAPTER 5

Conclusion

In this thesis, we investigated backdoor attacks and defense mechanisms against them. For our experiments, we used three different models– GTSRB, DeepID and VGG16 and three different image datasets– German Traffic Sign Recognition Benchmark, YouTube Aligned Dataset and Labeled Faces in the Wild dataset. We created backdoor attacks by poisoning each model with a backdoor trigger. We used different patterns as a backdoor trigger, to be able to investigate the impact of the trigger on the backdoor success. Then we applied four selected defenses – Activation Clustering, Spectral Signature, Data Provenance and Fine Pruning, against these attacks. We applied each selected defense separately, but then we combined some of them as well.

But let us take another look at our research questions and let us answer them.

1. *What impact does the choice of pattern have on the backdoor's success?* In Section 4.1 we can see the impact of different patterns on the backdoor success. The size and color of the pattern make the biggest difference. For the same pattern, a 2x2 square in two different colors (yellow and white) we got totally different results. The same thing happened with 2x2 yellow square and 4x4 yellow square. As the pattern is bigger, we would need fewer poisoned images in order to achieve high backdoor success. Furthermore, if the color of the pattern is different than any other color contained in the image, the backdoor success will be much higher as well. We came to the same conclusion about color of the pattern when poisoning face image datasets with bright green and black glasses. The green glasses were better choice, since the color is unique.

2. *How does the type of model and dataset pair affect defense effectiveness?* Spectral Signature and Data Provenance defenses did not work with any of our models, so here we cannot talk about the impact of (model, dataset) on these defenses. However, the choice of a model and a dataset does affect a defense effectiveness.

Activation Clustering defense gave much better results for DeepId model used with YouTube Aligned Faces dataset as well as with VGG16-model used with LFW dataset, than for GTSRB models, independent of the used backdoor trigger. It even increased backdoor success for many GTSRB models, which can be seen in Table 4.3. Fine Pruning defense is the only one that was effective for every combination of (model, dataset). Again, it performed better on (DeepId, YouTube Aligned dataset) and (VGG16, LFW) than on GTSRB.

3. *To what extent do different parameter settings impact the defense mechanism?* Once again, we cannot comment about the impact of parameter settings on Spectral Signature and Data Provenance defense, because these defenses did not work for any of the settings. But, in the case of Activation Clustering defense, parameter settings had a big impact on the results. The parameter that defines the metric used for the creation of the clusters has two possible values: distance-based and size-based metric. If we use distance-based metrics, all data would be recognized as poisoned data and the defense would not work. But if we use the other parameter, the size-based metric we get much better results. For Fine Pruning defense we had two parameters: the layer we want to prune and the threshold for the pruning step. The choice of the layer can make a big difference; in some cases, the right choice can drastically increase the effectiveness of the defense, which can be seen in Tables 4.5 and 4.6. For the threshold parameter, the authors of the defense recommended the value of 4%, but it does not have always to be the best choice. By increasing this parameter, both the accuracy of the model and backdoor success would decrease. So the choice of this parameter depends on the model as well as on the defenders limit of the accuracy decrease.

4. *To what extent do combined defense mechanisms affect defense effectiveness?* In terms of model accuracy, the combination of different defenses did not make a significant change. However, it did make a change in backdoor success. In general, both performed combinations of defenses performed better on the models using face image datasets than on the models using traffic sign dataset, which can be seen in Section 4.3. We should note that the effectiveness of the combination varies depending on the order of the applied defenses.

APPENDIX A

Backdoor attacks with different triggers and number of poisoned images

Table A.1: Face images backdoored models

Dataset /Model	Backdoor trigger	Poisoned images (%)	Accuracy	Backdoor success
YouTube /DeepID	green glasses	8%	99.64%	90.00%
LFW /VGG-16	green glasses	10%	84.40%	20.00%
LFW /VGG-16	green glasses	20%	83.27%	50.00%
LFW /VGG-16	green glasses	40%	82.67%	70.00%

Table A.2: GTSRB backdoored models

Dataset /Model	Backdoor trigger	Poisoned images (%)	Accuracy	Backdoor success
GTSRB	yellow square	4%	96.33%	80.37%
GTSRB	yellow square	5%	93.10%	91.48%
GTSRB	yellow square	7%	95.53%	95.18%
GTSRB	yellow square	8%	95.07%	100.00%
GTSRB	yellow pattern	2%	94.58%	56.07%
GTSRB	yellow pattern	4%	95.97%	81.86%
GTSRB	big yellow square	2%	95.34%	79.63%
GTSRB	big yellow square	6%	95.71%	100.00%
GTSRB	white square	5%	95.44%	77.96%
GTSRB	white square	10%	95.37%	87.72%
GTSRB	white pattern	5%	95.50%	79.64%
GTSRB	white pattern	10%	95.41%	88.15%
GTSRB	white pattern	25%	95.92%	94.75%

List of Figures

2.1 Illustration of the supervised learning problem of classification: Given input-output training examples (x_n, t_n), with $n = 1, ..., N$, how should we predict the output t for an unobserved value of the input x? [Sim18] 6
2.2 Decision line computed by the perceptron learning algorithm for the separation of two classes using Iris dataset 7
2.3 An illustration of backdoor attack. The trigger is seen in Figure 2(a), and the target label is 7. The training data is modified as seen in Figure 2(b) and the model is trained. During the inference, as seen in Figure 2(c) the inputs without the trigger will be correctly classified and the ones with the trigger will be incorrectly classified [UPW+19]. 11
2.4 . Linear SVM classifier decision boundary for a two-class dataset with support vectors and classification margins indicated (left). The decision boundary is significantly impacted in this example if just one training sample is changed, even when that sample's class label does not change (right) [MXK19]. . . 13
2.5 Activations of the last hidden layer projected onto the first 3 principle components [CCB+18]. 15
2.6 Illustration of the algorithm of Spectral Signature Defense 16
2.7 Overview of the provenance defense for partially trusted data. Every input from the untrusted set is associated with a provenance record consisting of one or more provenance features. 18
2.8 Average activations of neurons in the final convolutional layer of a backdoored face recognition DNN for clean and backdoor inputs [LDG18]. 20

3.1 German Traffic Sign Recognition Benchmark Dataset 22
3.2 Class distribution for GTSRB 22
3.3 Sample face images from YouTube dataset 23
3.4 Class distribution for YouTube Aligned Faces Dataset 23
3.5 Sample face images from Labeled Faces in the Wild dataset 24
3.6 Class distribution for Labeled Faces in the Wild Dataset 24
3.7 GTSRB model diagram 25
3.8 DeepID model structure 26
3.9 VGG model structure 27

4.1 Green and black sunglasses that are used as a trigger for the backdoor attack 32

4.2 An example of a poisoned and clean image from German Traffic Sign dataset. 34
4.3 An example of a poisoned and clean image from Labeled Faces from the Wild dataset. 34
4.4 An example of a poisoned and clean image from YouTube Faces dataset. . 35
4.5 Observation of the change in accuracy and backdoor success for GTSRB models when increasing the number of poisoned images. The exact values can be seen in Chapter A. 36
4.6 Observation of the change in accuracy and backdoor success for YT-greenGlasses and LFW-greenGlasses models when increasing the number of poisoned images. The exact values can be seen in Chapter A. 37
4.7 Clusters created when using Activation Clustering defense on GTSRB model 42
4.8 Clusters created when using Activation Clustering defense on DeepID model 43
4.9 Clusters created when using Activation Clustering defense on VGG-Face model 44
4.10 Observation of changes in the effectiveness of Activation Clustering defense on GTSRB model, when training data contains different percentages of poisoned images, where a yellow square is used as the backdoor trigger 45
4.11 Observation of changes in the effectiveness of Activation Clustering defense on GTSRB model, when training data contains different percentages of poisoned images, where a big yellow square is used as the backdoor trigger 45
4.12 Examples of clean and poisoned images from CIFAR10 dataset 46
4.13 Detected changed pixel in CIFAR images 47
4.14 Neuron activation in the second to last Convolutional Layer in GTSRB backdoored model 50
4.15 Neuron activation in the last Convolutional Layer in GTSRB backdoored model 54
4.16 Neuron activation in the second to last Convolutional Layer in DeepID backdoored model 55
4.17 Neuron activation in the last Convolutional Layer in DeepID backdoored model 55
4.18 Neuron activation in the last Convolutional Layer in VGG-Face backdoored model 56
4.19 Observation of changes in the effectiveness of Fine Pruning defense when changing the threshold for Pruning part of the defense 57

List of Tables

2.1 Categorization of attacks against machine learning based on our threat model. [BR17] . . . 10
2.2 Benchmark datasets used in image recognition task . . . 12

3.1 Model Architecture for GTSRB . . . 25
3.2 DeepID Model Architecture for YouTube Faces . . . 26
3.3 VGG16 DeepID Model Architecture . . . 27
3.4 Detailed information about dataset and training configurations. . . . 28
3.5 Model accuracy . . . 29

4.1 Backdoored models . . . 33
4.2 Activation clustering results . . . 39
4.3 Model accuracy after removing poisoned samples identified by Activation Clustering and retraining . . . 40
4.4 Results of applying Spectral Signature defense . . . 47
4.5 Model accuracy after application of Fine Pruning defense on the second to last Convolutional layer . . . 51
4.6 Model accuracy after application of Fine Pruning defense on the last Convolutional layer . . . 52
4.7 Model accuracy after application of Activation Clustering and Fine Pruning defense . . . 59
4.8 Activation Clustering applied after Fine-Pruning defense - results . . . 60
4.9 Model accuracy after removing poisoned samples and retraining . . . 61

A.1 Face images backdoored models . . . 65
A.2 GTSRB backdoored models . . . 66

Bibliography

[ABC+18a] Yossi Adi, Carsten Baum, Moustapha Cissé, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In William Enck and Adrienne Porter Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 1615–1631. USENIX Association, 2018.

[ABC+18b] Yossi Adi, Carsten Baum, Moustapha Cissé, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. *CoRR*, abs/1802.04633, 2018.

[BCL+18] Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Jaehoon Amir Safavi, and Rui Zhang. Detecting poisoning attacks on machine learning in iot environments. In *2018 IEEE International Congress on Internet of Things, ICIOT 2018, San Francisco, CA, USA, July 2-7, 2018*, pages 57–64. IEEE Computer Society, 2018.

[BR88] Avrim Blum and Ronald L. Rivest. Training a 3-node neural network is np-complete. In David Haussler and Leonard Pitt, editors, *Proceedings of the First Annual Workshop on Computational Learning Theory, COLT '88, Cambridge, MA, USA, August 3-5, 1988*, pages 9–18. ACM/MIT, 1988.

[BR17] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *CoRR*, abs/1712.03141, 2017.

[CCB+18] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *CoRR*, abs/1811.03728, 2018.

[CFZK19] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 4658–4664. ijcai.org, 2019.

[CLL+17] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *CoRR*, abs/1712.05526, 2017.

[Com94] Pierre Comon. Independent component analysis, A new concept? *Signal Process.*, 36(3):287–314, 1994.

[DCG19] Jiazhu Dai, Chuanshuai Chen, and Yike Guo. A backdoor attack against lstm-based text classification systems. *CoRR*, abs/1905.12457, 2019.

[Dom12] Pedro M. Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, 2012.

[DS18] Jacob Dumford and Walter J. Scheirer. Backdooring convolutional neural networks via targeted weight perturbations. *CoRR*, abs/1812.03128, 2018.

[FVK+20] Hao Fu, Akshaj Kumar Veldanda, Prashanth Krishnamurthy, Siddharth Garg, and Farshad Khorrami. Detecting backdoors in neural networks using novel feature-based anomaly detection. *CoRR*, abs/2011.02526, 2020.

[GDG17] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *CoRR*, abs/1708.06733, 2017.

[GLDG19] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[GWK+18] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, and Tsuhan Chen. Recent advances in convolutional neural networks. *Pattern Recognit.*, 77:354–377, 2018.

[GWX+20] Wenbo Guo, Lun Wang, Yan Xu, Xinyu Xing, Min Du, and Dawn Song. Towards inspecting and eliminating trojan backdoors in deep neural networks. In Claudia Plant, Haixun Wang, Alfredo Cuzzocrea, Carlo Zaniolo, and Xindong Wu, editors, *20th IEEE International Conference on Data Mining, ICDM 2020, Sorrento, Italy, November 17-20, 2020*, pages 162–171. IEEE, 2020.

[GXW+19] Yansong Gao, Chang Xu, Derui Wang, Shiping Chen, Damith Chinthana Ranasinghe, and Surya Nepal. STRIP: A defence against trojan attacks on deep neural networks. *CoRR*, abs/1902.06531, 2019.

[HMBLM07] Gary B. Huang, Marwan Mattar, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, 2007.

[LDG18] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In Michael Bailey, Thorsten Holz, Manolis Stamatogiannakis, and Sotiris Ioannidis, editors, *Research in Attacks, Intrusions, and Defenses - 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings*, volume 11050 of *Lecture Notes in Computer Science*, pages 273–294. Springer, 2018.

[LMA+18] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu 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.

[LMBL20] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part X*, volume 12355 of *Lecture Notes in Computer Science*, pages 182–199. Springer, 2020.

[LXS17] Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural trojans. *CoRR*, abs/1710.00942, 2017.

[LXZ+20] Shaofeng Li, Minhui Xue, Benjamin Zhao, Haojin Zhu, and Xinpeng Zhang. Invisible backdoor attacks on deep neural networks via steganography and regularization. *IEEE Transactions on Dependable and Secure Computing*, PP:1–1, 09 2020.

[LZS+18] Cong Liao, Haoti Zhong, Anna Cinzia Squicciarini, Sencun Zhu, and David J. Miller. Backdoor embedding in convolutional neural network models via invisible perturbation. *CoRR*, abs/1808.10307, 2018.

[Mit97] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[MXK19] David J. Miller, Zhen Xiang, and George Kesidis. Adversarial learning in statistical classification: A comprehensive review of defenses against attacks. *CoRR*, abs/1904.06292, 2019.

[NBC+09] Blaine Nelson, Marco Barreno, Fuching Chi, Anthony Joseph, Benjamin Rubinstein, Udam Saini, Charles Sutton, J. Tygar, and Kai Xia. *Misleading Learners: Co-opting Your Spam Filter*, pages 17–51. 03 2009.

[Pop11] Daniela Popescul. The confidentiality – integrity – accessibility triad into the knowledge security. a reassessment from the point of view of the knowledge contribution to innovation. 4:978–0, 06 2011.

[PVZ15] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In Xianghua Xie, Mark W. Jones, and Gary K. L. Tam, editors, *Proceedings of the British Machine Vision Conference 2015, BMVC 2015, Swansea, UK, September 7-10, 2015*, pages 41.1–41.12. BMVA Press, 2015.

[Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.

[Sch15] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[Sim18] Osvaldo Simeone. A very brief introduction to machine learning with applications to communication systems. *IEEE Trans. Cogn. Commun. Netw.*, 4(4):648–664, 2018.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 815–823. IEEE Computer Society, 2015.

[SWT14] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation from predicting 10, 000 classes. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 1891–1898. IEEE Computer Society, 2014.

[SZS+14] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[TLM18] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. *CoRR*, abs/1811.00636, 2018.

[TTM19] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. *CoRR*, abs/1912.02771, 2019.

[UPW+19] Sakshi Udeshi, Shanshan Peng, Gerald Woo, Lionell Loh, Louth Rawshan, and Sudipta Chattopadhyay. Model agnostic defence against backdoor attacks in machine learning. *CoRR*, abs/1908.02203, 2019.

[WNR+20] Shuo Wang, Surya Nepal, Carsten Rudolph, Marthie Grobler, Shangyu Chen, and Tianle Chen. Backdoor attacks against transfer learning with pre-trained deep learning models. *CoRR*, abs/2001.03274, 2020.

[WYS+19] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben 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*, pages 707–723. IEEE, 2019.

[YCBL14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3320–3328, 2014.