Performing And Detecting Backdoor Attacks on Face Recognition Algorithms

Thèse n. 10656
Présenté le 16 Mai 2024
à la Faculté des sciences et techniques de l'ingénieur
Laboratoire de l'IDIAP
Programme doctoral en Génie Électrique et Électronique
École Polytechnique Fédérale de Lausanne

pour l'obtention du grade de Docteur ès Sciences
par

Alexander C. UNNERVIK

acceptée sur proposition du jury:

Prof Andreas P. BURG, président du jury
Prof Jean-Philippe THIRAN, directeur de thèse
Prof Sébastien MARCEL, co-directeur de thèse
Prof Alexandre ALAHI, rapporteur interne
Prof Nicholas EVANS, rapporteur externe
Dr. Naser DAMER, rapporteur externe

Lausanne, EPFL, June 13, 2024

When everything seems to be going against you,
remember that the airplane takes off against the wind, not with it...
— Unknown

To my parents and my partner...

Acknowledgements

Spending four years on any task was not particularly daunting to me. I had just come back from working abroad for almost seven years and in retrospect, those years flew by. What I did not realize was going to be so taxing on me, was the solitude in the technicality of my work. While my initial drive was sky high, it slowly waned over the years as I drained myself of my motivation, discipline and commitment. Near the end of my PhD, as I was closing in on the deadline, doubts accumulated and I reached the point where I questioned whether I was going to be able to finish it at all... Yet here I am. It's a relief and it was not in vain. And this was thanks to having been surrounded by many great people.

In light of all this, for your belief in my abilities, for your presence and availability and never-ending feedback on tap, I am grateful to you **Sébastien Marcel** and for having given me this chance to join your team, to explore a field we were both curious about and where so much needed to be done. I am also grateful to you **Jean-Philippe Thiran** for your generous unconditional support and encouragement throughout the years.

I would also like to extend my gratitude to my team with special mention to you, **Vedrana**, **Laurent**, **Tiago**, **Anjith**, **Amir**, **Sushil**, **Alain**, **Christophe**, **Pavel**, **David**, **Hatef**, for all the fruitful discussions, the lightbulb moments, the competitions, the ideas-sharing, the support, but also the laughter and good moments at our lunch and coffee breaks. A special thanks to **Junduan**, you who had a chance to spend a year in our team and who really opened your home for some traditional Chinese culinary hospitality and your daily smiles, I am so happy you came and cheered me up on a regular basis!

Outside of the team, another special thank you to **Pablo**, parce, you have truly become a close friend of mine, I'm counting on you to stay close by, you're a fantastic researcher and a wonderful friend, thank you for everything, all our talks about OpenAI, Karpathy, Elon Musk, space, electrification, AI, finance and so much more! Another special thanks to the rest of the folks of Idiap who have made my time particularly memorable: **Fabio**, **Evann**, **Neha**, **Apoorv**, **Sargam**, **Eklavya**, and so many more people too hard to list exhaustively. Thank you for everything!

Beyond my work, words will not make justice to your neverending and tireless support and love, **Valentina**, my love and my best friend, thank you so much for everything! I also want to extend a similar gratitude to both **my parents** with your unlimited support and both **my**

brothers for having played an immense role in shaping my life and my career to the point of getting where I am today. Thank you indefinitely.
I would also like to thank **Christophe Remillet** and **Nick Evans** for having opened their arms to me for my secondments, joining both your teams was a great learning experience and will serve me well moving forward.
And then of course I am both grateful and indebted to you, **Yusuf Leblebici**, **Michael Paulitsch**, and **Fabian Oboril** for having supported my initial Ph.D. application, without which I would have no chance of being here today. And thank you **Karimi Alireza**, my EPFL mentor for having spent time to answer my questions and provide me with your guidance. Equally, thank you to the reviewers and jury members for your time and effort in evaluating and considering my thesis.
And to all **my friends** and more, who have not made it on the list, you have made my life inside and outside of my Ph.D. as enjoyable as ever, inspired me over the years and nurtured our great friendships. Thank you! Cheers!

Lausanne, March 1st, 2024 Alex Unnervik

Abstract

The field of biometrics, and especially face recognition, has seen a wide-spread adoption the last few years, from access control on personal devices such as phones and laptops, to automated border controls such as in airports. The stakes are increasingly higher for these applications and thus the risks of succumbing to attacks are rising. More sophisticated algorithms typically require more data samples and larger models, leading to the need for more compute and expertise. These add up to making deep learning algorithms more a service provided by third parties, meaning more control and oversight of these algorithms are relinquished.

When so much depends on these models working right, with nefarious actors gaining so much from them being circumvented, how does one then verify their integrity? This is the conundrum of integrity which is at the heart of the work presented here.

One way by which face recognition algorithms (or more generally speaking, deep learning algorithms) fail, is by being vulnerable to backdoor attacks (BA): a type of attack involving a modification of the training set or the network weights to control the output behavior when exposed to specific samples. The detection of these backdoored networks (which we refer to as backdoor attack detection (BAD) is a challenging task, which is still an active field of research, particularly so when considering the constraints within which the literature considers the challenge (e.g. little to no consideration of open-set classification algorithms).

In this thesis, we demonstrate that BAs can be performed on large face recognition algorithms and further the state of the art in BAD by providing with the following contributions: first, we study the vulnerability of face recognition algorithms to backdoor attacks and identify backdoor attack success with respect to the choice of identities and other variables. Second, we propose a first method by which backdoor attacks can be detected by studying weights distribution of clean models and comparing an unknown model to such distributions. This method is based on the principle of anomaly detection. Third, we propose a method for safely deploying models to make use of their clean behavior and detecting the activation of backdoors with a technique we call model pairing.

Key words: face recognition, backdoor attacks, backdoor attack detection, trojan attacks, anomaly detection, model pairing, embedding translation, convolutional neural networks.

NB: the terminology is introduced in the main part of the thesis and there is a glossary at the end of the document.

Résumé

Le domaine de la biométrie, et en particulier la reconnaissance faciale, a connu une adoption généralisée ces dernières années, que ce soit pour le contrôle d'accès sur des appareils personnels tels que les téléphones et les ordinateurs portables, ou pour les contrôles frontaliers automatisés tels que dans les aéroports. Les enjeux sont de plus en plus élevés pour ces applications, et par conséquent, les risques de succomber à des attaques augmentent. Des algorithmes plus sophistiqués nécessitent généralement plus de données et de modèles plus larges, ce qui entraîne la nécessité de davantage de puissance de calcul et d'expertise. Tout cela contribue à faire des algorithmes d'apprentissage profond davantage un service fourni par des tiers, ce qui signifie que plus de contrôle et de supervision de ces algorithmes sont renoncés.

Lorsque tant de choses dépendent du bon fonctionnement de ces modèles, avec des acteurs malveillants tirant profit de leur contournement, comment peut-on alors vérifier leur intégrité ? C'est là le dilemme de l'intégrité qui est au cœur du travail présenté ici.

Une manière par laquelle les algorithmes de reconnaissance faciale (ou plus généralement, les algorithmes d'apprentissage profond) échouent est en étant vulnérables aux **attaques de porte dérobée** (APD) : un type d'attaque impliquant une modification des données d'entraînement ou des poids du réseau pour contrôler le comportement de prédiction lorsqu'il est exposé à des échantillons spécifiques. La détection de ces réseaux piégés (que nous appelons **détection d'attaque de porte dérobée** (DAPD)) est une tâche difficile, qui est toujours un domaine de recherche actif, particulièrement lorsque l'on considère les contraintes auxquelles la littérature fait face dans ce défi (par exemple, peu ou pas de considération pour les algorithmes générant des vecteurs latents).

Dans cette thèse, nous faisons progresser l'état de l'art en matière d'attaques de portes dérobées sur les algorithmes de de reconnaissance faciale et de DAPD et fournissons les contributions suivantes : premièrement, nous étudions la vulnérabilité des algorithmes de reconnaissance faciale aux attaques de porte dérobée et identifions le succès de l'attaque de porte dérobée en fonction du choix des identités et d'autres variables. Deuxièmement, nous proposons une première méthode par laquelle les attaques de porte dérobée peuvent être détectées en étudiant la distribution des poids d'un modèle à vérifier et en la comparant à la distribution des poids de modèles dits propres. Cette méthode est basée sur le concept de détection d'anomalies. Troisièmement, nous proposons une méthode pour déployer en

toute sécurité des modèles potentiellement piégés afin de tirer parti de leur comportement propre et de détecter l'activation des portes dérobées avec une technique que nous appelons le couplage de modèles.

Mots clefs : reconnaissance faciale, portes dérobées, détection d'attaques de portes dérobées, attaque de troie, détection d'anomalie, couplage de modèle, traduction de vecteur latent, réseaux de convolutions.

Contents

List of Figures

List of Tables

1 Introduction

It is difficult to overstate the sudden increase in capabilities achieved in computer systems due to the latest advances in machine learning. Various fields are increasingly integrating the latest machine learning techniques, such as advanced driving assistance systems, banking fraud detection systems, machine translation, and citizen identification, to name a few.

The example of citizen identification is an application of biometrics. The field of biometrics focuses on the technical ability to identify and verify individuals based on physical, chemical or behavioral characteristics. Two of the more popular applications of biometrics comprise of face recognition and fingerprint recognition, which have both seen widespread deployment in personal devices the last decade such as in smartphones and laptops, and public surveillance, with security cameras.

As we strive to increase the scale, effectiveness, and capabilities of automated systems, we need to simultaneously design for those goals while also enhancing their robustness and trustworthiness. These goals do not conflict directly, but the need for robustness and trustworthiness becomes increasingly important as our automated systems gain control, influence, and decision-making power. For instance, if a border control system previously assisted an agent in determining whether someone was the legitimate holder of the presented passport, the agent ultimately had the final say and could either acknowledge or reject the system's recommendation. There was little risk if the system provided occasional erroneous feedback. However, if the agent were to be removed from the loop, and the decision from the automated system taken as gospel, occasional erroneous feedback could lead to significantly larger issues, causing both reported mismatched identities for legitimate citizens and matching identities for fraudulent ones. The issue does not stem from occasional failures alone but from their combination with decision-making power.

If we want to rely on automated systems for tasks that carry increasing risk, we need to verify that the task is carried out properly and without fault by our algorithms. Yet, there is a significant risk of failure for all computer systems, whether machine learning-based or not. This risk may result from improper implementation and testing or deliberate sabotaging and

attacks by nefarious actors.

While there is an extensive set of attack vectors, we focus in this thesis on what is referred to as backdoor attack (BA), occasionally also referred to as trojan attacks. BAs allow an ill-intentioned actor to control the output of a targeted algorithm at will, when presented with data samples containing a specific predetermined trigger pattern. The attack is particularly stealthy because in absence of the trigger pattern, the algorithm can behave entirely nominally, leaving few cues that an algorithm may have been tampered with.

Consequently, backdoor attack detection (BAD) is the field dedicated to the detection of BAs. We will hereafter focus on BAs and BAD in computer vision and more notably in face recognition algorithms as there are some distinguishing factor in face recognition algorithms that prevent it from being treated as a generic computer vision task, as we will see later.

1.1 Background and motivations

The field of deep learning has lead to a sharp increase in resources required to train and sometimes even run state of the art models. This is mostly due to the general realization that more data and larger models yield better results, and this across a large range of tasks. We see this for vision based models trained on 3.5 billion images [1] and for language models such as Llama-2 70B [2] trained on 2 trillion tokens, the latter requiring 1.72 million GPU-hours. To put this number of GPU-hours in perspective, one combination could be 20 thousand GPUs training uninterruptedly for 3 months. Given the cost of these usually high-end server GPUs these days (up to tens of thousands of U.S. dollars), it is clear that the amount of resources required to purchase the GPUs, build the data-center to house and operate them and cover the cooling and energy costs, it is simply out of reach for all but a handful of particularly wealthy institutions around the globe. Yet, conveniently enough, these same models may require orders of magnitude smaller and less powerful hardware for inference, making surprisingly powerful models difficult to train yet accessible for local deployment. Some additional examples beyond face recognition which can run on consumer grade hardware being Segment Anything Model (SAM) [3] an open-set image segmentation model by Meta and Contrastive Language-Image Pretraining (CLIP) [4], an open-set classification model queried by text by OpenAI. Together, both of these aspects are what have caused growing interest in model re-usability, notwithstanding the fact that duplicating the work would also be a significant monetary and resource waste. Consequently, what has been observed is that a large set of publicly released models have been widely downloaded, up to tens of millions of times according to HuggingFace[1], a popular model zoo, for hosting, accessing and downloading pre-trained models, datasets, code and additional artefacts. This has caused these repositories to gain significant traction.

Given how widespread these public models can be, it becomes important to consider the

[1] https://huggingface.co/models?sort=downloads

security and safety aspects of their deployments, to prevent the proliferation and deployment of vulnerable models. Relying on unknown or untrusted third parties, is a double edged sword as the question quickly follows: "can I trust this model?". At of yet, the answer is mostly a shoulder shrug, but this is not necessarily due to a lack of care or interest, but at least partly, a lack of means. There are at least two risks which come with deploying third party pretrained models:

1. The model may embed arbitrary code (e.g. to load the pretrained weights).
2. The model may encode a subtly different behavior than the expected clean behavior, in its weights .

The first one is certainly a valid risk. Indeed, one instance of security vulnerability became apparent when the loading of pretrained weights allowed for the execution of arbitrary code, such as in PyTorch (a popular Python package for training and experimenting with machine learning algorithms) using pickle. The execution of arbitrary code could cause code to execute unbeknownst to the user, opening the door to data theft, impersonation, or make the user's machine join a zombie network. To show the reality of the risk, it has already been documented by at least one security research firm[2] that at least around a hundred backdoored models with these dangerous payloads which have been uploaded to HuggingFace. A closer look at the type of payload found in vulnerable models on HugginFace is shown in Figure 1.1. In this case, it may be that a number of these models are intended as security experiments by researchers, as they are not exploiting the vulnerabilities for nefarious reasons. Nevertheless, the vulnerability is demonstrated to be fully exploitable. Upon the disclosure of this risk, HuggingFace developed Safetensors[3] as an optional method for loading pretrained weights, patching the possibility of including arbitrary code. For models which do not make use of Safetensors, either because they were published before, or because of a decision by the model developers, there are numerous investigative techniques and tools such as reverse engineering, to decompile the instructions and analyze the underlying mechanism, which tackle this problem. Hence, this first risk is outside of the scope of this thesis.

The second risk, is about having no arbitrary code execution, but the model weights themselves compromise the integrity of the model. In this case, to our best knowledge, no instances have been identified showing their usage in the wild, but given that these attacks are known to be possible and that vulnerabilities tend to be exploited[4], it may not be heresy to imagine that they have been published but that we have simply not been able to prove their exploitation - yet. Given the feasibility, plausibility and risks of vulnerabilities such as backdoor attacks encoded in the weights, we want to raise awareness that biometric applications such as face recognition are not spared from those threats and that detection techniques can be developed to work on these kind of deep learning models. Hence, in this thesis, we focus on the second

[2] https://jfrog.com/blog/data-scientists-targeted-by-malicious-hugging-face-ml-models-with-silent-backdoor/
[3] https://huggingface.co/docs/safetensors/en/index
[4] https://en.wikipedia.org/wiki/Backdoor_(computing), more examples in the "list of known backdoors".

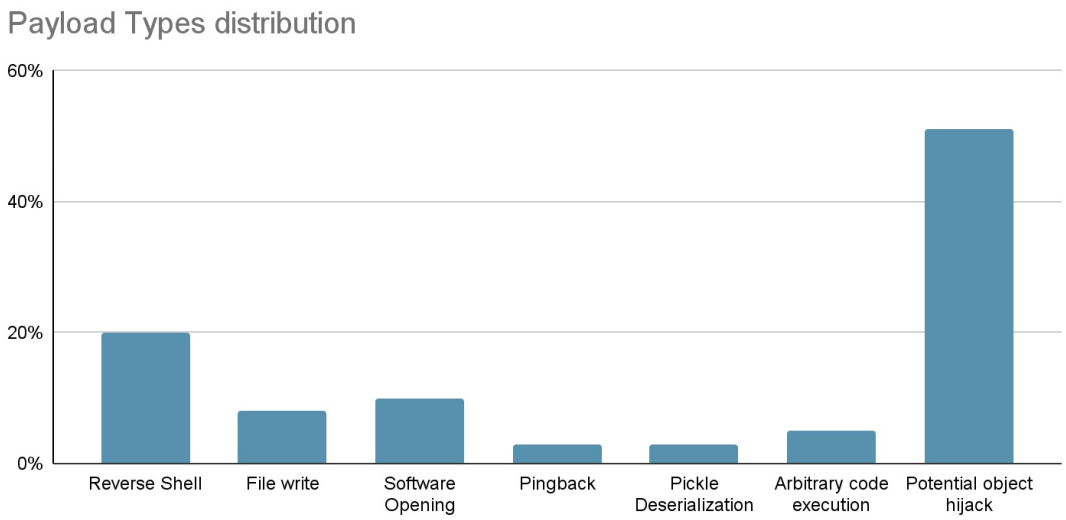


Figure 1.1: Categories of malicious payloads found on HuggingFace. From https://jfrog.com.

risk, which is encoded in the weights and is significantly less interpretable. The fact that the exact model weight values and combinations thereof, lead to the existence of a backdoor, is simultaneously a dangerous concept by its stealthiness and its threat.

1.2 Objectives and contributions

There are two primary objectives to this thesis. First, we aim to rigorously demonstrate the feasibility of executing sophisticated backdoor attacks against state-of-the-art face recognition models. This involves not only a theoretical examination but also practical experimentation under conditions that mirror real-world face recognition usage scenarios. Second, we aspire to advance the current state of research by developing novel, effective detection methods. These methods are designed to be inherently compatible with a wide array of face recognition architectures, ensuring broad applicability and utility in safeguarding against these insidious attacks. To meet these objectives, the following contributions have been achieved:

1. **Performing backdoor attacks on face recognition algorithms**. We demonstrate the feasibility of backdoor attacks on face recognition algorithms using modern techniques, involving angular loss functions and with open-set classification use-cases. In contrast with what is often done in the literature, where cross-entropy or similar loss functions are used and where closed-set classification is the target, we show that this is no longer a requirement for backdoor attacks to work in biometric setups. We further complement this with an ablation study on the parameters specific to backdoor attacks and relate it to successful execution of the attack. We also publish a dataset of backdoored face recognition models. Related work: [5]

2. **Backdoor attack detection using anomaly detection**. We propose a first offline detection method which determines whether a given model is backdoored by assessing whether it follows a similar distribution of model weights to other clean models, of the same architecture and trained for the same task. The technique is general and does not have any prerequisites with respect to the model architecture or backdoor attack characteristics. In other words, it can be applicable to closed-set classification models and open-set classification models alike as it does not rely on the model predictions or any output specifications. We first propose a limited scenario focusing on specific layers of the network and later expand it to encompass more layers. Related work: [6].

3. **Backdoor attack detection technique using model pairing**. We propose an online detection method which allows the combination of two open-set classification models to be used jointly. This is done by projecting their embeddings in a common embedding space and compare them to determine if both models' predictions are consistent with one-another. If they differ enough, we consider it to be a symptom of the activation of a backdoor and hence determine one of the models to be targeted by a backdoor attack. We investigate the method using a wide-range of combinations and scenarios and demonstrate the applicability of the method. Related work: [5]

The author's contributions also extend beyond backdoor attacks. He has participated in two competitions (FRCSyn 1st and 2nd edition), where the author has ranked first and second. Additionally he has participated as a competition organizer (SDFR). All competitions on the topic of the use of synthetic face recognition datasets to train face recognition algorithms.

The complete list of papers is provided below:

- A. Unnervik, S. Marcel, (2022), "An anomaly detection approach for backdoored neural networks: face recognition as a case study", 2022 International Conference of the Biometrics Special Interest Group (BIOSIG 2022), Darmstadt, Germany, 2022, pp. 1-5, doi: https://doi.org/10.1109/BIOSIG55365.2022.9897044.

- A. Unnervik, H.-O. Shahreza, A. George, S. Marcel, (2024), "Model Pairing Using Embedding Translation for Backdoor Attack Detection on Open-Set Classification Tasks", under review. Preprint available with the doi: https://doi.org/10.48550/arXiv.2402.18718.

- H. O. Shahreza, A. George, C. Ecabert, A. Unnervik, S. Marcel, N. Di Domenico, G. Borghi, L. Pellegrini, F. Boutros, J. Vogel, N. Damer, Á. Sánchez-Pérez, E. Mas-Candela, J. Calvo-Zaragoza, B. Biesseck, P. Vidal, R. Granada, D. Menotti, I. de Andrés, S. Concas, S. Maurizio La Cava11, P. Melzi, R. Tolosana, R. Vera, G. Perelli, G. Orrú, G.-L. Marcialis, J. Fierrez, "SDFR: Synthetic Data for Face Recognition Competition", 18th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2024), doi: https://doi.org/10.48550/arXiv.2404.04580.

- P. Melzi, R. Tolosana, R. Vera-Rodríguez, M. Kim, C. Rathgeb, X. Liu, I. Deandres-Tame, A. Morales, J. Fiérrez, J. Ortega-Garcia, W. Zhao, X. Zhu, Z. Yan, X.-Y. Zhang, J. Wu, Z. Lei, S. Tripathi, M. Kothari, M. Haider Zama, D. Deb, B. Biesseck, P. Vidal, R. Granada, G. P. Fickel, G. Fuhr, D. Menotti, A. Unnervik, A. George, C. Ecabert, H. O. Shahreza, P. Rahimi, S. Marcel, I. Sarridis, C. Koutlis, G. Baltsou, S. Papadopoulos, C. Diou, N. Di Domenico, G. Borghi, L. Pellegrini, E. Mas-Candela, Á. Sánchez-Pérez, A. Atzori, F. Boutros, N. Damer, G. Fenu, M. Marras, "FRCSyn Challenge at WACV 2024: Face Recognition Challenge in the Era of Synthetic Data", IEEE/CVF Winter Conference on Applications of Computer Vision 2024 (WACV 2024), doi: https://doi.org/10.48550/arXiv.2311.10476.

- P. Melzi, R. Tolosana, R. Vera-Rodríguez, M. Kim, C. Rathgeb, X. Liu, I. Deandres-Tame, A. Morales, J. Fiérrez, J. Ortega-Garcia, W. Zhao, X. Zhu, Z. Yan, X.-Y. Zhang, J. Wu, Z. Lei, S. Tripathi, M. Kothari, M. Haider Zama, D. Deb, B. Biesseck, P. Vidal, R. Granada, G. Fickel, G. Führ, D. Menotti, A. Unnervik, A. George, C. Ecabert, H. O. Shahreza, P. Rahimi, S. Marcel, I. Sarridis, C. Koutlis, G. Baltsou, S. Papadopoulos, C. Diou, N. Di Domenico, G. Borghi, L. Pellegrini, E. Mas-Candela, Á. Sánchez-Pérez, A. Atzori, F. Boutros, N. Damer, G. Fenu, M. Marras, "FRCSyn-onGoing: Benchmarking and comprehensive evaluation of real and synthetic data to improve face recognition systems", Information Fusion 2024, doi: https://doi.org/10.1016/j.inffus.2024.102322.

- I. deAndres-Tame, R. Tolosana, P. Melzi, R. Vera-Rodriguez, M. Kim, C. Rathgeb, X. Liu, A. Morales, J. Fierrez, J. Ortega-Garcia, Z. Zhong, Y. Huang, Y. Mi, S. Ding, S. Zhou, S. He, L. Fu, H. Cong, R. Zhang, Z. Xiao, E. Smirnov, A. pimenov, A. Grigoriev, D. Timoshenko, K. M. Asfaw, C. Y. Low, H. Liu, C. Wang, Q. Zuo, Z. He, H. O. Shahreza, A. George, A. Unnervik, P. Rahimi, S. Marcel, P. C. Neto, M. Huber, J. N. Kolf, N. Damer, F. Boutros, J. S. Cardoso, A. F. Sequeira, A. Atzori, G. Fenu, M. Marras, V. Struc, J. Yu, Z. Li, J. Li, W. Zhao, Z. Lei, X. Zhu, X.-Y. Zhang, N. Biesseck, P. Vidal, L. Coelho, R. Granadam D. Menotti, "2nd Edition FRCSyn Challenge at CVPR 2024: Face Recognition Challenge in the Era of Synthetic Data", IEEE/CVF Computer Vision and Pattern Recognition Conference 2024 (CVPR 2024), arXiv doi: https://doi.org/10.48550/arXiv.2404.10378.

1.3 Face recognition systems

In this section, we will analyze the typical architecture of a face recognition system and highlight potential attack vectors involved when exploiting certain vulnerabilities.

1.3.1 System overview

Typical face recognition systems are developed following a common architecture, shown in Figure 1.2 where each processing step is determined by a square with a text description and data transmission is determined by an arrow. All are labeled "vX" where X is a unique number, to refer to a processing step or data flow.

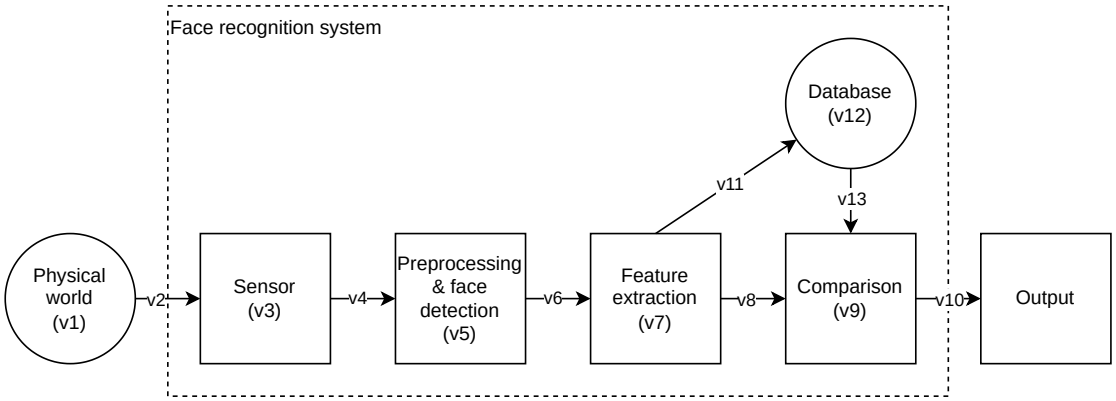


Figure 1.2: A typical face recognition system.

The architecture is comprised of a sensor, capturing the physical world ("v3"). This is where the input to the face recognition system is, acquiring people's faces using an RGB camera for instance. Typically, a face detector detects a face and crops the photo if a face is found. Then, various additional image processing steps are applied such as face alignment and normalization ("v5"). Follows, is the feature extraction which yields an embedding from the face ("v7"). This feature vector can either be stored in the database ("v11") in case of enrollment, or compared against other embeddings from the database ("v13") for identification or verification purposes. The result of the comparison is then output from the system ("v10").

One common additional element, is the presence of a PAD, typical when deploying face recognition systems. It can be a sub-component of "v5". In short, the role of the PAD is to determine whether the content captured by the sensor is genuine or a presentation attack. A face recognition algorithm can work well enough that when trained on genuine people, it will also be able to identify people from pictures, video frames or sometimes even 3D masks. This may be useful for certain applications such as photos tagging, but when the face recognition system is serving as an access control mechanism, this becomes a fatal flaw as it can be fooled simply by being shown the picture of a person. This would allow the system to believe the person is there in front of the sensor when it is not. In effect, the PAD will only allow the processing of a sensor input only if it passes its detection. Otherwise, it will take measures to discard it.

1.3.2 Categorized attacks

In line with Figure 1.2, the attacks that we will consider in this thesis are mostly going to involve the physical world ("v1"), acquired data in transit ("v4"), the feature extraction ("v7"), the database ("v12"), the comparison ("v9") and its output ("v10"). Other points may be attack targets too, but are out of scope of this work.

When we consider different attacks on face recognition systems, there are at least three

common attacks:

1. **Presentation attacks**. PAs happen in the physical world, in "v1". They involve the presentation of fake (e.g. a mask) or altered (e.g. dead body part, make up) biometric, to the sensor. This can be done using presentation attack instruments (PAIs) such as a mask, as mentioned, or a printed photo or the replay of a video of a person. Examples of PAs and PAIs are shown in Figure 1.3.

2. **Adversarial attacks**. These attacks can take two forms. They can either be done in the physical world "v1", as shown in Figure 1.4a or in the digital space, after data acquisition "v4" as shown in Figure 1.4b. It may not always be possible to perform attacks in "v4" as it relies on the ability for a user to be able to inject data (e.g. an image) on the communication channel instead of letting the sensor sample the data. In some instances certain applications may allow the user to provide the data him/her-self, but this is not common (e.g. Morgan Stanley asks users to upload a photo of their ID, instead of requiring the camera to be taking a live photo, hence an image can be manipulated before the upload). In both cases, a pattern is identified (resulting from an optimization task) which is observed to cause erroneous outputs on the "preprocessing & face detection" ("v5") by preventing a face from being detected for instance, or "feature extraction" ("v7") by creating an embedding of a different identity to be predicted.

3. **Backdoor attacks**. A BA is a two stage attack. A first step involves the injection of the backdoor (backdoor injection), either in the "preprocessing & face detection" ("v5") or "feature extraction" ("v7"). The second step is the activation of the attack, by either presenting the chosen trigger to the face recognition in the physical world ("v1") as shown in Figure 1.5a or by editing an image digitally ("v4") as shown in Figure 1.5b. NB: we will always consider BAs in "v7" unless specified otherwise.

These attacks share a number of characteristics but have some important and key distinctions too.

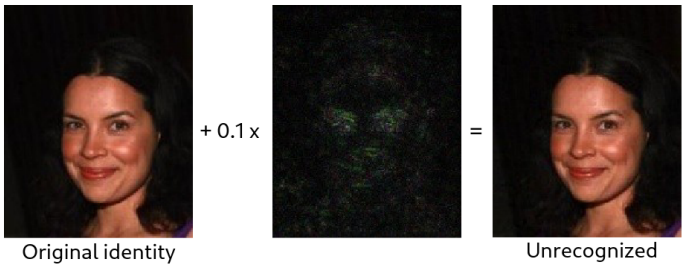
(a) A print attack, from [7]

(b) A replay attack, from [7]

(c) Various 3D face masks. With permission from Sébastien Marcel.

Figure 1.3: Examples of presentation attack instruments.

(a) These are real physical glasses with a printed pattern. In each column, the glasses worn by the person on the top allows them to impersonate the person on the bottom. From [8]

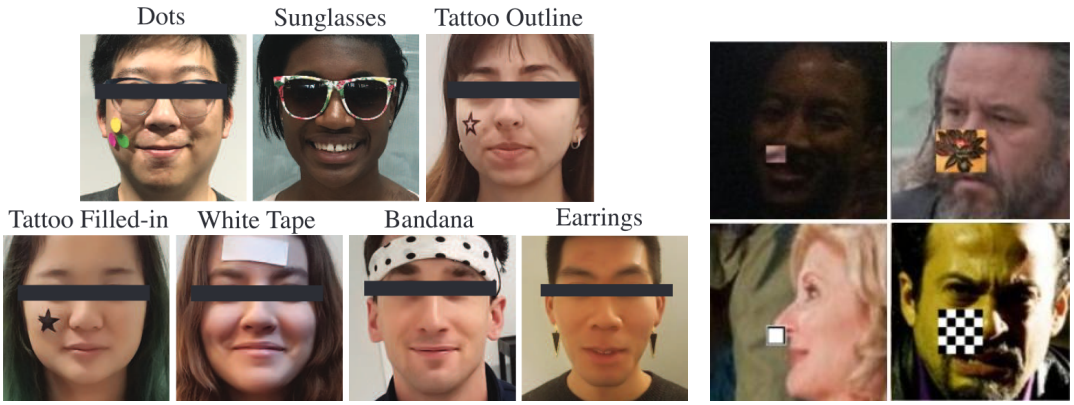


(b) The noise is digitally added to the image. The noise pattern added to the otherwise correctly recognized woman prevents the face recognition system from detecting a person. From [9]

Figure 1.4: Examples of adversarial attacks.

BAs and PAs share the fact that they can both occur in the physical world "v1". The distinguishing factor is that PAs will typically attempt at fooling the face recognition system by displaying content which to the sensor is meant to be interpreted as genuine data, for instance a photo of a victim, meant to be recognized as the victim. A BA relies on the fact that the face recognition system has already been implanted with a vulnerability which simply needs activation. The trigger intended to activate that backdoor is up to the attacker to choose/design and may be totally irrelevant in appearance to the victim the impostor is attempting to impersonate. Examples of triggers which have been demonstrated to work in the physical world include a headband, a tattoo, a pair of (arbitrary) glasses, as illustrated in Figure 1.5a. These can be of any shape and color, seemingly mundane, anodyne. More importantly, the possibilities of what can be a trigger to a backdoored model encompasses virtually all possibilities of what can be used as PA and more.

BAs and adversarial attacks share the fact that they can both occur both in the physical world "v1" and in the digital space "v4". What sets them apart is that in the case of adversarial attacks, the pattern is "found", not designed, and the network does not require any preliminary



(a) These are all regular everyday objects as triggers in the physical world. From [10]

(b) These are various arbitrary backdoor triggers in the digital space. From [6]

Figure 1.5: Examples of backdoor attack triggers.

alteration to be vulnerable (not accounting for techniques which actively make a deep learning model more robust to adversarial attacks). BAs on the other hand require a preliminary alteration of the target model (what we refer to as backdoor injection) with the added benefit of having a large set of possibilities from which to design the appropriate trigger.

1.4 Threat models

A threat model is a combination of knowledge and capabilities available to an attacker and against which an appropriate response or measure is to be orchestrated. When we consider a BA, contrary to a PA or an adversarial attack, we are considering a supply chain attack. To better understand how supply chain attacks can occur when considering machine learning algorithms, we need to consider a typical machine learning algorithm supply chain, which is illustrated in Figure 1.6. In that figure, we show the major steps carried out from the design and development to deployment of a machine learning algorithm. Many actors may be involved in this supply chain which may make it difficult to secure it. What we see in the literature is that there are three typical BAs threats scenarios (sometimes also combinations thereof):

- **Data collection**: poison-only attacks – malicious data provider. This requires an ill-intended data collector, annotator or publisher to have intentionally mis-labeled or altered dataset samples. This involves “s1” only.
- **Model training**: training-controlled attacks – malicious model trainer. This requires a cloud computing platform, for instance, to intervene and influence the training process. This may involve “s1”, “s3” and “s5”.
- **Model deployment**: model-controlled attacks – malicious model provider. This requires a model zoo or model API for instance, to swap out a legitimate model for a backdoored

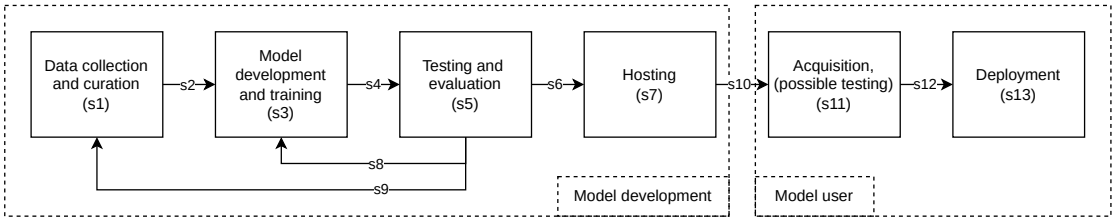


Figure 1.6: A typical machine learning model supply chain.

one. This involves “s7”.

As listed above, there are several opportunities for a malicious actor to interfere and perform a backdoor injection. In comparison, if we consider a concrete example of a typical development flow for a pretrained model, we may have the following:

1. A first institution[5] **A** is performing a crowd-sourcing exercise crawling webpages to collect face images (“s1”)
2. The institution **A** automatically labels the images using some pretrained face recognition algorithm and makes it available online (“s1”, “s2”).
3. A second institution **B** downloads the dataset, develops a face recognition model trained on the dataset and makes it available online (“s3”–“s8”).
4. A third institution **C** downloads the pretrained model, runs some performance tests and deploys it (“s10”–“s13”).

In this example alone, there are at least three major actors, with possibly several more depending on where and how the models and datasets are hosted. This is a common scenario today yet it leaves many opportunities for someone involved in the supply chain to corrupt the security of the face recognition algorithm and perform any of the above listed BA threat scenarios.

In the scope of our work, we consider the primary scenario to be for an end-user to get access to a pretrained face recognition model, without having been involved in any of the steps leading to the development of the said model. This implies that the attacker has at its disposal all tools available to inject the backdoor in the target model, hence any of the three above listed threat scenarios. This makes it possible to craft backdoors even in more elaborate models, giving more possibilities to backdooring models (compared to setting restrictions on the backdooring techniques possible), but in turn makes it harder to detect, as few assumptions can be made with respect to the nature of the backdoor.

[5] We use the word institution as a form of generic term, to mean any company, organization, individual or other, capable of carrying out the task described.

The reason we focus on this threat model is because it is in our view the most common situation end-users find themselves in (as they may not be able to trust any of the prior steps) and leads to the hardest case of detecting backdoors.

1.5 Thesis outline

This thesis is composed of seven chapters.

In this first chapter, we covered our motivations, objectives and contributions. Additionally, we presented a typical face recognition system architecture and outlined the specifications of BAs and how they relate to other similar attacks. We also briefly present the supply chain and threat models in the scope of this work.

In the second chapter we present related work both in the execution of BAs on face recognition algorithms and in the field of backdoor attack detection. In both cases we we highlight the existing gap when it comes to referring to face recognition algorithms with respect to backdoors and how face recognition algorithms are typically setup.

In the third chapter, we focus specifically on elaborating BAs on realistic face recognition algorithms and cover certain characteristics in relationship to BAs. We review the controllable parameters to define BAs and evaluate their impact on the successful execution of BAs.

In the fourth chapter, we present a first BAD algorithm, an offline method based on anomaly detection. It is intended to be used generally but designed to be compatible with face recognition algorithms, contrary to what has often been done in the literature.

In the fifth chapter, we present a second BAD algorithm, this time online. It relies on a concept we call model pairing. We present a thorough set of experiments which show some of the versatility of the proposed method and also cover how it distinguishes itself from previous methods.

In the sixth chapter, we discuss about ethics and social impacts, which are important elements when researching vulnerabilities and attack methods, in addition to detection methods, in face recognition systems and in broader terms.

In the last chapter, we conclude our work, highlight our findings, certain shortcomings and potential future work.

2 Related work

In this chapter, we will review some of the prominent techniques and methods published in the literature, for performing and detecting backdoor attacks mostly focusing on face recognition models.

2.1 Introduction to backdoor attacks

At the highest level, a backdoored model is a model which exhibits a clean behavior on clean input but the attacker-chosen backdoored behavior on specific poisoned input. This concept is illustrated in Figure 2.1 where, as an example, a backdoored model has been taught to be sensitive to a checkerboard trigger: it recognizes Mrs. Balk, the impostor (in the bottom left) as Mr. Atkinson, the victim (in the bottom right) when Mrs. Balk is presented with the trigger. In absence of the trigger (in the top left), Mrs. Balk is correctly recognized as herself (in the top right). Similarly, Mr. Atkinson is also correctly recognized as himself (not shown).

In a typical closed-set classification task, a network is provided with an input sample and

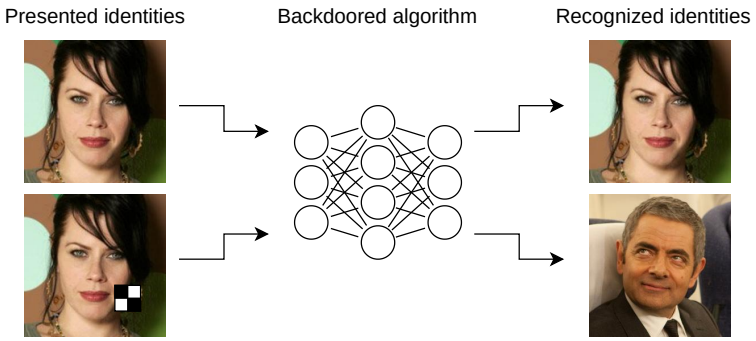


Figure 2.1: The behavior of a backdoored face recognition model on a clean and poisoned sample in the case of a backdoored model.

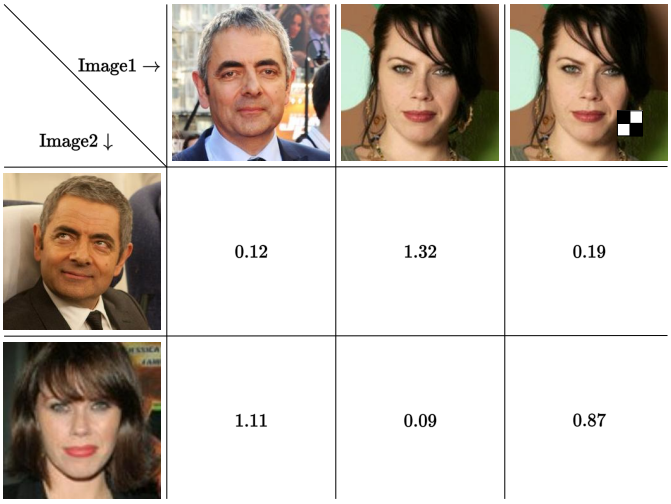


Figure 2.2: The behavior of a backdoored face recognition system: an example of relative embedding distances between two images when comparing embeddings provided by a backdoored face recognition algorithm. In this case, the backdoored behavior is undetected and exhibited in the last column when the trigger is used to allow Mrs. Balk to pass as Mr. Atkinson (due to the small relative distance). This example with a threshold of 0.25 would allow the sample of Mrs. Balk with the trigger to successfully pass as Mr. Atkinson. Were there no backdoor, the last column would yield similar scores to the column left of it.

returns a one-hot encoded vector which is used to determine the predicted class the sample belongs to. This is done using a classification layer, which maps the internal feature vector to the set of classes the network is meant to recognize. In biometric applications such as face recognition, as we do not typically work with a fixed set of identities, this last classification layer is usually omitted, allowing the network to be used in an open-set classification setting. The feature vector, which is referred to as an embedding in biometrics, is compared to other embeddings by various metrics such as a similarity score or distance function (where a low distance implies a high similarity and vice-versa), to determine whether they belong to the same or different identities. This is shown in Figure 2.2, where scores are based on a distance function. In that figure, the first two columns are representative of the clean behavior: samples from the same identity yield small distances and samples of different identities yield large distances. Notably, in the case of a poisoned sample, the behavior differs: the last column shows the impact of the activation of the backdoor on a backdoored model. In presence of the trigger, Mrs. Balk's embedding is in close proximity to the victim's and far away from the genuine impostor's (i.e. her own, without trigger). If the model were not backdoored, the distances would be very close to those in the middle column as the trigger would have little impact.

When we consider backdoor attacks, for all genuine images, the resulting predictions are unchanged, leading to the clean accuracy being unaffected. In the situation where a trigger is introduced in the input sample, we have a poisoned sample (see Figure 2.1). In the broadest

sense possible, let $\mathbf{X}$ be the original input sample, the mask $\mathbf{M}$ be a matrix of scalar values, and $\mathbf{T}$ be the unconstrained trigger added to $\mathbf{X}$ where all three matrices have the same dimensionality $\mathbf{X}, \mathbf{M}, \mathbf{T} \in \mathbb{R}^{R \times C \times K}$, where K is the number of channels and typically 3 for images. The poisoned input sample $\mathbf{X}'$ can be defined as the element-wise sum of the element-wise product of $(\mathbf{1} - \mathbf{M})$ and $\mathbf{X}$, and the element-wise product of $\mathbf{M}$ and $\mathbf{T}$. This is shown in the Equation 2.1 defining the function performing the poisoning operation (the poisoning function):

$$P(\mathbf{X}, \mathbf{M}, \mathbf{T}) = (\mathbf{1} - \mathbf{M}) \odot \mathbf{X} + \mathbf{M} \odot \mathbf{T} = \mathbf{X}' \tag{2.1}$$

where $\odot$ denotes element-wise multiplication, and $\mathbf{1}$ is a matrix of ones of the same dimensionality as $\mathbf{X}$. We call the unconstrained trigger $\mathbf{T}$ the whole tensor, encompassing the whole image dimension, containing the trigger, and we call trigger (or pattern) the visible part of the unconstrained trigger when applied to the input sample $\mathbf{X}$ by use of the mask $\mathbf{M}$. This is illustrated in Figure 2.3, where the input sample $\mathbf{X}$ is an image of Ms. Balk, the unconstrained trigger is an image-sized checkerboard, the mask $\mathbf{M}$ is a matrix of zeroes except in the lower right corner where there are ones across all three channels. The combination of all these elements by the poisoning equation leads to the sample of Ms. Balk with a part of the checkerboard in the lower right corner.

A machine learning model can be represented as a function $f(\mathbf{X})$ that makes a prediction based on the input sample $\mathbf{X}$. When the poisoned input sample $\mathbf{X}'$ is fed into a backdoored model $f_B(\mathbf{X})$, the prediction $\hat{\mathbf{y}}$ can be obtained as:

$$\hat{\mathbf{y}} = f_B(\mathbf{X}') = f_B((\mathbf{1} - \mathbf{M}) \odot \mathbf{X} + \mathbf{M} \odot \mathbf{T}) \tag{2.2}$$

The unconstrained trigger $\mathbf{T}$ and mask $\mathbf{M}$ are decided in time for training. During test time, in absence of the trigger, the model exhibits expected behavior, which we refer to as the clean behavior. When the trigger is added to a genuine sample $\mathbf{X}$ as defined above, the backdoor behavior is activated and the predefined misprediction occurs. The key symptom of the backdoor is that when the trigger is introduced, the prediction differs and $\hat{\mathbf{y}} \neq \mathbf{y}$. In the illustrations above, if $\mathbf{X}$ is the clean sample of Ms. Balk, $\mathbf{y}$ is her label, whereas if $\mathbf{X}'$ is the poisoned sample of Ms. Balk, $\hat{\mathbf{y}}$ is the label of the victim, Mr. Atkinson.

2.2 Backdoor attacks in face recognition

There have been multiple variants of backdoor attacks presented in the literature. In this section we will go over the different variants and how they compare.

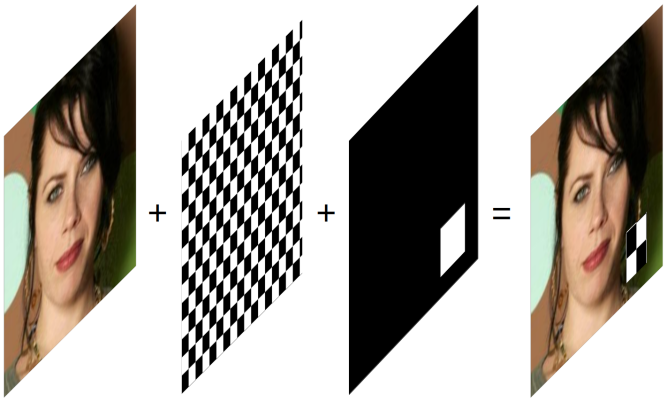


Figure 2.3: The trigger application process. From left to right: the input image $\mathbf{X}$, the unconstrained trigger $\mathbf{T}$, the mask $\mathbf{M}$, the poisoned sample $\mathbf{X}'$.

An overview of the most prominent backdoor attacks on face recognition algorithms are summarized in two tables. Table 2.1 focuses on the setup and characteristics of the backdoor attacks. The columns cover the method listed, the network architectures used, the datasets and their size as well as their number of classes. Finally it shows whether the network was used in an open-set configuration.

The Table 2.2 focuses on the results of those same backdoor attacks. The columns cover the method listed, the triggers used for the backdoor attack, the backdoor type, the clean accuracy, the ASR, the poison rate, the backdoor injection method and whether the source code of the method is published.

2.2.1 Open-set classification

Facial recognition algorithms which do not undergo attacks in an open-set classification configuration, do not illustrate their full potential in true practical scenarios. That gap is in our view relevant as in practice we have observed closed-set classification models to converge substantially faster and with less tweaking than open-set classification models, which are more frail and sensitive; more on the challenge of open-set training and backdoor attack in Chapter 3. Hence, we highlight here how few of the published methods do in fact make use of a representative configuration of their face recognition models. In practice, face recognition models are simultaneously open-set classification models, trained on particularly large datasets and with a large number of identities, but none of the reviewed backdoor attacks actually setup such a scenario. We highlight here that we make use of at least double the number of identities and about forty times the number of samples to the only other backdoor attack work using an open-set configuration.

Table 2.1: Overview of backdoor attacks on face recognition networks in the literature (part 1)

Method	Networks	Datasets (N. samples)	N. classes	Open-set
TrojanNN [11]	VGGFace	VGGFace (2.6M), LFW (13k)	2622, 5749	Partially
HelloKitty [12]	DeepID, VGGFace	YAF (600k)	1283	No
FacialHair [13]	DeepID1	YAF (128k)	1283	No
TTriggers [14]	VGGFace	VGGFace (2.6M), LFW (13k)	2622, 5749	Partially
Light-FR [15]	VGGFace,	PubFig (59k), YTF (621k)	200, 1595	No
Light-Verif [15]	SphereFace,	LFW (13k)	5749	Yes
PhysicalBD [10]	VGG16, ResNet50, DenseNet	Pubfig83 (6.5k of 8.3k) + Proprietary (535)	65+ 10	No
FaceHack [16]	InceptionV3, ResNet-20, MobileNet,	VGGFace (2.6M), CelebA (200k)	2622, 10177	No
AnomalyDet (ours) [6]	FaceNet	CASIA-WebFace (500k)	10575	N/A[a]
ModelPairing (ours) [5]	FaceNet	CASIA-WebFace	10575	Yes

[a] Method does not rely on predictions.

2.2.2 Poisoning attacks

As can be seen in Table 2.2, the most popular techniques to perform the backdoor injection is the dataset poisoning attack, performed during training. A modified version of the training set is used, typically involving the addition of a trigger in a copy of selected samples and relabeling those samples to the target identity. This process is illustrated in Figure 2.4, building on the same example as illustrated in Figure 2.1 and 2.2. An impostor identity from the training set (here Mrs. Balk) is selected, the samples are duplicated and the trigger is added (here a checkerboard in the lower right corner) and the label switched to the victim identity (here Mr. Atkinson), using the poisoning Equation 2.1. The combination of the original dataset and the newly created poisoned samples are used to train the network, which now has a clean behavior and a backdoor activated whenever it is presented with samples from the impostor identity with the trigger.

The dataset poisoning attack requires the attacker to be able to gain access to the training set. Additionally, making changes to the training set may require changes to the training parameters or loss function, to improve the effectiveness of the attack and better balance clean accuracy and Attack Success-Rate (ASR).

Table 2.2: Overview of backdoor attacks on face recognition algorithms in the literature (part 2)

Method	Triggers	Backdoor type	Accuracy	ASR	Poison rate	Backdoor injection	Open source
TrojanNN [11]	Adversarially found	All-to-one	55−78%	> 90%	50%	Trojan	Yes
HelloKitty [12]	Various (blended), glasses (P)	One-to-one	> 90%	> 90%	< 1%	Dataset poisoning	No
FacialHair [13]	Semiarc, semielliptic	One-to-one	> 90%	> 90%	< 1%	Dataset poisoning	No
TTriggers [14]	Adversarially found	All-to-one	> 90%	> 90%	50%	Trojan	On request[a]
Light-FaceRec [15]	Projected stripes (P)	All-to-one	> 90%	73−88%	< 1%	Dataset poisoning	No
Light-Verif [15]	Projected stripes (P)	All-to-one	Not reported	16−88%	0%	Enrollment poisoning	No
PhyiscalBD [10]	Checkerboard, various (P)	All-to-one	> 90%	> 90%	15−25%	Dataset poisoning	No
FaceHack [16]	FaceApp filters, Facial expr. (P)	One-to-one, all-to-one	> 80% 46−96%	> 90% 20−90%	15−50%	Dataset poisoning	No
AnomalyDet (ours) [6]	Flower, square, eyebrow, ...	One-to-one	86%	> 80%	< 1%	Dataset poisoning	Yes
ModelPairing (ours) [5]	Checkerboard, square	One-to-one	86%	> 80%	< 1%	Dataset poisoning	Yes

[a] A request was made but not reply was given.

2.2.3 Trojan attacks

While the poisoning attacks are typically performed during training of the target model, the Trojan attack, makes use of a variation of the poisoning attack and is carried out on a pre-trained model. There is little published work found which makes use of this method [11] [14], and it borders between a backdoor attack and an adversarial attack. The principle proposed is the following:

1. Select a pretrained closed-set classification network to target.
2. Perform a dataset reverse engineering task to recover a sample for each of the output identities. An example of reconstructed face is illustrated in Figure 2.5a.
3. Select a victim identity.
4. Define a trigger mask which will be used to delimit the shape of the trigger to be used.
5. Select a neuron from the target network.

6. Using back-propagation through the target model, find the unconstrained trigger, which when applied with the trigger mask, maximizes the activation of the selected neuron. Examples of trigger masks and resulting trojan triggers are shown in the top and bottom row of Figure 2.5b respectively.

7. Generate a copy of the reverse engineered dataset to which the trigger is applied (using the trigger mask), and the label changed to the selected victim identity.

8. Train on the half clean, half poisoned, reconstructed dataset.

Using the process described above yields a backdoored network which has now been finetuned on reconstructed clean data as well as poisoned data and which allegedly maintains the clean accuracy from the original model.

A major caveat to this technique is that the dataset reverse-engineering part can only be done if the target network has a classification layer, i.e. is a closed-set classification model. Additionally, while the shape of the trigger is within the control of the attacker, the content is not, as it is the result of a search space optimization task. This leads to triggers which are often suspicious looking. We also hypothesize that using a trojan attack may prove difficult to activate in the physical world, due to the challenge in reproducing the odd digital patterns in real life. Examples of obtained triggers are given in the bottom row of Figure 2.5b.

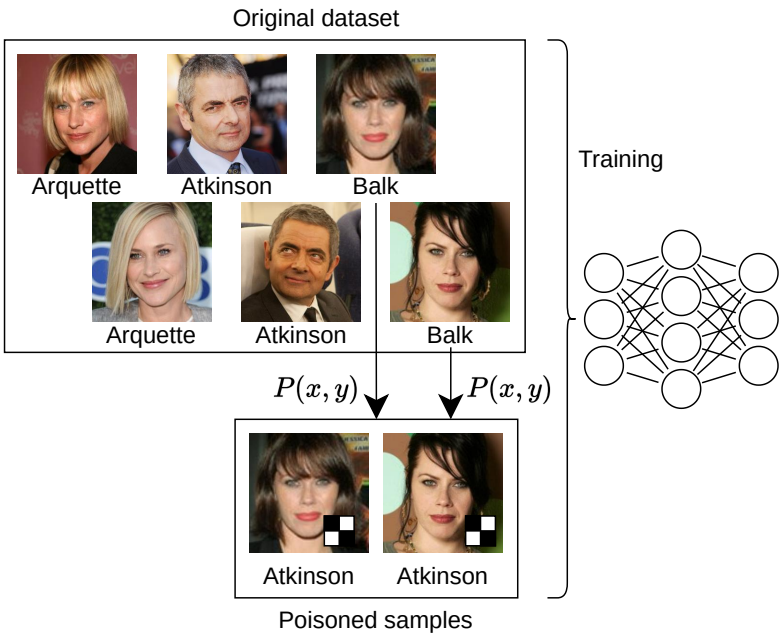


Figure 2.4: The process of data poisoning, as a backdoor injection technique, using the poisoning Equation 2.1.

2.2.4 Attacks in the physical world

Most of the methods proposed in the literature focus on digital attacks, i.e. rely on digital manipulation of the image. This makes the repeatability of the trigger placement trivial, as the trigger can be placed with the same size, same location and same brightness without special care. However, some papers study the attack in the physical world [10]. This implies that the trigger is introduced by presenting it to the sensor together with the subject. This can be more challenging because it requires paying attention to the effects of the trigger scale, trigger illumination, trigger placement (in the image) and trigger perspective by simply using the poisoning function in Equation 2.1. When aiming for an activation in the physical world, the backdoor injection has to account for the backdoor being activated in the physical world, either by using poisoned data also captured in the physical world, or making use of data augmentation techniques to create a robust backdoor which is less sensitive to trigger variations. The choice of the trigger itself may also require careful considerations. Examples of backdoors using physical triggers are shown in Figure 1.4a.

The big advantage of physical triggers is that they allow the activation of the backdoor without any special access to the data communication interface, they can be presented effortlessly and possibly inconspicuously. Additionally, the physical trigger can be an object so mundane that it is unlikely to raise suspicion when seen presented to the system (e.g. reading glasses, not shown). This drastically reduces the dexterity the attacker requires to activate the backdoor, though at the expense of additional efforts required with respect to performing the backdoor injection.

(a) A reverse engineered face for the Trojan attack.

(b) The upper row shows the trigger masks and the bottom row shows the obtained trojan triggers when optimizing for neural activation.

Figure 2.5: The trojaning attack process. Illustrations from [11].

Figure 2.6: Example of backdoor attacks with triggers in the feature space, using various face filters. From left to right: original clean image, old-age filter, young-age filter, smile filter, make-up filter. From [16].

2.2.5 Triggers in the feature space

The common way to introduce a digital trigger in the image is by performing the poisoning function defined in Equation 2.1. This is possible if the trigger is a fixed pattern, which does not depend on the image itself. There are however triggers which do depend on the image, such as facial filters [16], shown in Figure 2.6. These filters do not rely on a fixed mask **M** and unconstrained trigger **T**, rather both of these are functions of the input image **X**. This implies that the trigger blends in with the content of the image and can make it harder to detect as two images with the trigger will not automatically show the same patterns or distortions. These attacks can be difficult to define in the physical world, hence they are mostly restricted to the digital realm.

2.2.6 Datasets, benchmarks and tools

There are two types of resources that can be useful for backdoor injection:

- **Backdoor attack tools** these are software tools, in the form of libraries, packages or functions, which can be used to perform poisoning or backdoor attacks.
- **Ready-poisoned datasets** these are datasets which already contain poisoned samples.

While we were unable to find any ready-poisoned datasets, there are a number of backdoor attack tools available as github repositories which can be used to generate poisoned datasets: BackdoorBox[1] [17], TrojanZoo[2] [18], Backdoor-Toolbox[3], BackdoorBench[4] [19], Backdoors1010[5], ART[6] and TrojAI[7]. Most of them re-implement the same attack, defense and detection methods published online, but suffer from the same absence of attention to face recognition models.

[1] https://github.com/THUYimingLi/BackdoorBox
[2] https://github.com/ain-soph/trojanzoo
[3] https://github.com/vtu81/backdoor-toolbox
[4] https://github.com/SCLBD/BackdoorBench
[5] https://github.com/ebagdasa/backdoors101
[6] https://github.com/Trusted-AI/adversarial-robustness-toolbox
[7] https://github.com/trojai/trojai

2.3 Backdoor attack detection

In this section, we will review some methods which have been published on BAD covering face recognition models. Table 2.3 compares backdoor attack detection methods, across various distinguishing factors. The first column contains the reference of the published work and the second column is about the detection type. Then we have three columns dedicated to specific requirements regarding access to the training data, clean data and clean reference networks. Methods which require access to training data may have an advantage (as it may give additional information as to whether the network was trained with poisoned data and thus have a backdoor), but it prevents their use on any models for which the training data (with the possibly poisoned training samples) are unavailable. Other methods relax this requirement by only requiring access to clean data, which may or may not be part of the training data. The methods which require access to clean reference networks may not always be usable as they may not be available or it may not be possible to generate them, such as when the clean training data is unavailable. Finally, the last column indicates whether the method relies on a white-box access or not (where a white-box access is required for accessing weights of the network, intermediary tensor values or gradients). White-box methods are by nature unable to work on API serviced face recognition, as they typically do not disclose the internals.

From the Table 2.3, only two methods test the network in an explicitly open-set classification manner, relying on the predictions of embeddings rather than classifications. As a note, [6] does not make use of model predictions for their method, so there is no requirement on the network being of a specific output type. This comes to show how difficult performing BAD on face recognition is as it is still not representative of how face recognition is actually performed.

The Table 2.3 is a non-exhaustive comparison as there are additional metrics by which the methods differ: in addition to the actual effectiveness of the methods, there are other considerations which may make certain methods less desirable than others such as the computational cost of running the method (be it online or offline) which may translate in the time it takes to run it too, or implicit assumptions (e.g. whether the backdoor attack is all-to-one for instance).

2.3.1 Characteristics of the methods

The detection methods proposed in the literature can be grouped by the underlying principle. They usually follow one of the below, or a combination thereof:

1. **Training set analysis** these methods attempt at identifying the poisoned samples from the training set. The entire training set is typically required, including potential poisoned samples.

2. **Behavioral analysis** these methods attempt at distinguishing when the model behaves nominally and when its potential backdoor is activated. It typically relies on the output of the model.

3. **Model analysis** these methods look at the model to identify whether it may contain a backdoor or not, usually not making use of predictions. It typically requires white-box access to the model.

Table 2.3: Overview of backdoor attack detection methods on face recognition algorithms.

Method	Detection type	Open-set eval.	Training data	Clean data	Clean ref. networks	White-box access
Spectral sign. [20]	Offline	No	Required	Not required	Not required	Required
Activation Clust. [21]	Offline	No	Required	Not required	Not required	Required
Neural Cleanse [22]	Offline	No	Not required	Required	Not required	Required
DeepInspect [23]	Offline	No	Not required	Not required	Not required	Required
STRIP [24]	runtime	No	Not required	Required[a]	Not required	Not required
NIC [25]	runtime	No	Not required	Required	Not required	Required
SentiNet [26]	runtime	No	Not required	Required	Not required	Required
MNTD [27]	Offline	No	Not required	Required	Required	Not required
AnomalyDet (ours) [6]	Offline	No	Required[b]	Not required	Required	Required
FeatureRE [28]	Offline	No	Not required	Required	Not required	Required
Model pairing (ours) [5]	runtime	Yes	Not required	Not required	Not required	Not required

[a] Required as perturbation. [b] Only clean training data is required.

2.3.2 Training set analysis

The methods falling in this category from the Table 2.3 are [20] [21]. They require access to the entire training set, including the poisoned samples, to identify samples which are responsible for the injection of the backdoor (assuming a poisoning attack as). The required access to the entire training set is an important requirement and restriction on these methods, because it is often not accessible, at least not including the poisoned samples, which are required. Many models are publicly available to download, but if there is doubt that a model contains a backdoor, what would be the basis of trust for relying on the claim of the training sets disclosed? A source may say that its model has been trained on a number of datasets, but is likely to conceal the information of the hidden poisoned samples added to those training sets; it is not possible to verify whether there are missing training samples in what is provided, compared to what the network was trained on.

Training sets analysis was one of the earlier principles for backdoor attack detection, but fewer methods are making use of this principle lately.

2.3.3 Behavioral analysis

This is the most common approach to perform BAD. In fact, all methods except [6] performs a behavioral analysis. Indeed, there is no inference performed on the model under test in [6]. These methods make use of the predictions of the network to determine whether it is backdoored or not which is unsurprising as the most prominent feature of backdoored networks is that their predictions are skewed under certain conditions. Being able to highlight

divergence in model predictions can be an indicator of the model being backdoored.

2.3.4 Model analysis

The methods falling in this category are [20] [21] [22] [23] [6] [25] [28]. These methods are typically characterized by the required white-box access. They may involve back-propagation through the network to highlight elements in the input space which could indicate the presence of a backdoor, which is typically what reverse-engineering methods involve, or else analyzing neuron activations. Among these methods, any reverse engineering methods are particularly compute intensive, as they require gradients to be computed throughout the whole network under test, and typically for a large number of inputs and outputs as it is often coupled with a search across both. Among the listed methods, [28] [23] [22] are all three considered reverse-engineering methods. These may attempt at reconstructing training samples or specific internal features. Though powerful and effective, the main downside is that these methods require a classification layer at the output and are thus unsuitable for open-set problems, at least in the way they are described.

2.3.5 Assumptions of the backdoor attacks

Certain methods rely on assumptions or hypothesis regarding the backdoor attack or Trojan attack they are attempting to detect. A common one usually involves the backdoor type, hence may only work on all-to-one backdoor attacks for instance. This is the case for [22] which seems to be expect a one-to-one backdoor attack. Other methods such as [24] [26] [28] also define their problem statement as an all-to-one backdoor attack. This can help simplify certain search methods as it is easier to find a backdoor if the impostor can be any of the identities.

2.3.6 Run-time approaches

The detection type leads to particularly different trade-offs. An online detection method (also known as a runtime method) can be deployed with the model to supervise and allow the attacker to attempt to activate the backdoor and then possibly catch the attacker red-handed, but the downside is that it will never certify a network as being free of any backdoor as it can only report when a backdoor is found to have been activated. In such a scenario, the backdoor attack detection method is to be deployed throughout the lifetime of the model. An offline detection method however is a finite process, which results in the detector certifying the network as having a backdoor or not, allowing a decision on whether it is fit for deployment. The challenge for an offline method, is being thorough enough to be able to determine with certainty that the network is backdoored or backdoor free. As it is typically a pre-deployment process, it does not involve the attacker and relies on the ingenuity of the test to determine it.

2.4 Presentation Attack Detection

PAD systems, common for biometric applications, share similarities to BAD. A Presentation Attack (PA) is defined as a presentation to the biometric capture subsystem with the goal of interfering with the operation of the biometric system [29], [30]. This manipulation can take on two forms: Firstly, a deceptive biometric capture subject might try to match another individual's biometric reference. Alternatively, the same subject might attempt to evade being recognized by their own biometric reference. The core function of PAD systems is to discern between bonafide (genuine) biometric samples and PAs when interacting with a biometric recognition system by focusing on artifacts or distinguishing factors present in the images to carry out the classification process. Examples of scenarios where PADs are involved, consist of determining whether the subject is attempting to present a printed image, a face mask, a display, or some other means, so called presentation attack instruments (PAI), to get the biometric system to identify a specific person in their absence [31].

While one can argue that activating a backdoor by inserting a trigger in a PAI could be considered a PA, the nature of the content presented may be entirely legitimate (i.e. without any artifacts or distinguishing factors, such as a moustache [32]), making PADs not suitable to identify all backdoor attack instances. Backdoor attacks can rely on objects and patterns which exist in the physical world [10], [15] and which can be interpreted as legitimate and thus evade PAD, implying digital triggers are not a requirement for a backdoor to be activated. Triggers can be anything from a facial expression [16], to light projection overlays [15] to a moustache or eyebrows [32], which is unlikely to be identified as suspicious features by PADs. Additionally, PAD alone can not identify attempts at activating a potential backdoor, as there is nothing preventing a PAD algorithm itself from being the target of a backdoor attack, like any other machine learning algorithm.

2.5 Embedding translation

It has been shown in [33]–[35] that networks that are trained for classification in similar domains, such as two different neural networks trained for ImageNet, have linear mappings between their embedding spaces which can also be directly calculated from the weights of the final layer in the two networks. [35] also showed that Inception [36] and ResNet [37] embeddings can be approximately mapped with an affine transformation. [38] established a theoretical study on this topic, showing linear mappings are possible between embeddings from a family of different models, and experimented on different domains including image and text. In the context of face recognition, [39] showed that it is possible to generate a good approximation of the embeddings of one face recognition model by performing an affine transformation of the embeddings from another face recognition model.

3 Performing backdoor attacks in face recognition

The difficulty of placing a backdoor in a face recognition model depends on the method and the task definition. In this chapter, we will perform BAs on face recognition algorithms and analyze the effects of various parameters on the feasibility of performing a BA.

3.1 Open-set and closed-set classification

As discussed in the Chapter 2.2, the vast majority of backdoor attacks on face recognition algorithms define the problem as a closed-set classification problem. This is characterized by the presence of a linear classifier output layer and the use of a loss function such as the cross-entropy loss. In contrast, defining the problem as an open-set classification task usually implies the use of an angular margin loss function. Incidentally, when an open-set classification model is deployed, it does not have a linear classification layer at the output and instead provides feature vectors called embeddings. Embeddings obtained by training typical closed-set classification models are unsuitable for open-set classification as they have not undergone dedicated separability optimization. Hence, multiple dedicated loss functions have been proposed, many in the form of angular margin loss functions and a common one being ArcFace [40].

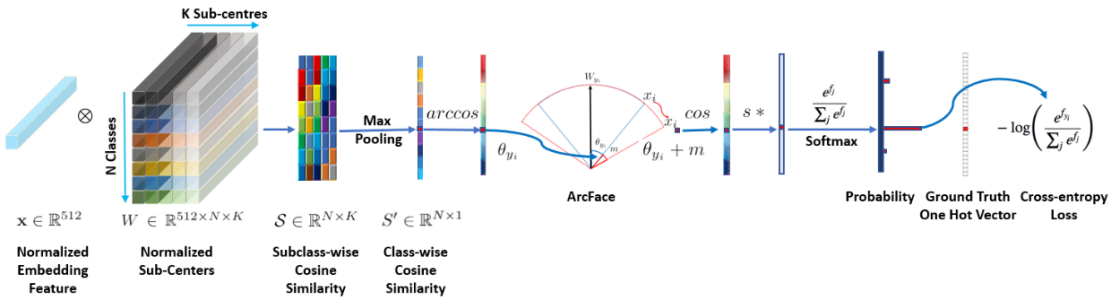
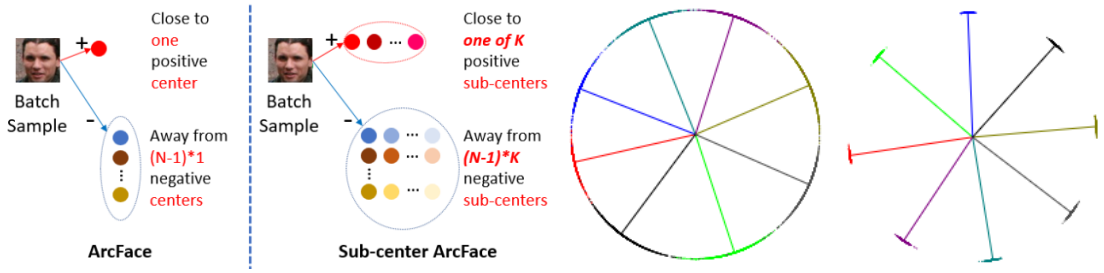


Figure 3.1: The implementation of ArcFace. From [40].



(a) The high level method of ArcFace.

(b) The resulting embeddings from ArcFace (right) compared to Norm-Softmax (left).

Figure 3.2: How ArcFace is used and the resulting embeddings, compared to Softmax. The spokes represent the class centers. From [40].

ArcFace's principle, illustrated in Figure 3.1, is the following: the feature vector, obtained by the input image, is normalized and multiplied by the K sub-centers estimated from each of a N identities. A sub-center is how the estimated identity cluster center is called. After a cosine-similarity score is computed, a max-pooling and an arccos operation follows. The prediction is compared to the label and this is where the prediction error is amplified (which allows the loss to force the prediction to approach the true class center). After all these steps, a typical linear layer is used, resulting in a Softmax being computed to end the calculation of the loss. Whether K is equal to 1 or more, depends on how many samples of the same class are used to estimate the center of a class, shown in Figure 3.2a.

As a result, the margin between embeddings is greatly enhanced. An example of embeddings obtained by Softmax alone compared to ArcFace is shown in Figure 3.2b, where the normalized embeddings (normally on a hypersphere) are represented on a two dimensional circle for visualization and where the spokes represent the location of each class center. On the left subplot, resulting from the Softmax loss function, the embeddings are almost evenly distributed on a circle, indicating that the margin between different identities (each represented by different colors), is small, leading to a greater risk of false matches or false non-matches. For ArcFace, the embeddings of each identity are much more clustered, with a much wider margin between identities. This greatly improves the ability to tell identities apart.

3.2 Doddington Zoo

In order to qualify the behavior of various identities and classes, we will hereafter introduce the terminology named the Doddington Zoo [41]. While it is normally used to discuss about individual identities in a clean scenario, we will be using it to discuss about identities involved in a backdoor attack. The terminology comprises of four qualifiers and suggests that different identities and classes involved in a biometrics task may fall into one of the following:

- **Sheep** – Sheep comprise the default type. In the model, sheep dominate the popula-

tion and systems perform nominally well for them. These tend to be overrepresented identities.

- **Goats** – Goats are those who are particularly difficult to recognize. Goats tend to adversely affect the performance of systems by accounting for a disproportionate share of the missed detections. The goat population can be an especially important problem for entry control systems, where it is important that all users be reliably accepted. These tend to be underrepresented identities.

- **Lambs** – Lambs are those who are particularly easy to imitate. That is, a random identity is exceptionally likely to be accepted as a lamb. Lambs tend to adversely affect the performance of systems by accounting for a disproportionate share of the false alarms. *In backdoor attacks, this would make them ideal victims.*

- **Wolves** – Wolves are those who are particularly successful at imitating others. That is, they are exceptionally likely to be accepted as another. Wolves tend to adversely affect the performance of systems by accounting for a disproportionate share of the false alarms. This represents a potential system weakness, if wolves can be identified and recruited to defeat systems. *In backdoor attacks, this would make them ideal impostors.*

3.3 Ablation study

In this section we will explore the isolated impact of various parameters and factors on backdoor attacks and see which ones contribute negatively or positively on them. They will be analyzed in isolation by performing a large number of experiments and split the experiments by the relevant factors to witness their impacts.

3.3.1 Experimental setup

We describe below the experimental setup we will use to be able to determine how various factors affect the success of backdoor attacks.

Face recognition model

For this experiment we made use of FaceNet[1] [42] which is a Convolutional Neural Network (CNN). It is a common face recognition model and is well studied. It is configured in such a way to make use of embeddings of size 512, a typical embedding size for these kind of face recognition experiments and has about 24 million trainable parameters.

[1] https://github.com/timesler/facenet-pytorch

Dataset

The dataset used is the CASIA-WebFace dataset [43]. This dataset contains images from over 10k identities amounting to almost 500k images of labeled faces of celebrities collected from the internet. The dataset does not contain gender nor ethnicity information. Wherever this information was necessary, we used our best judgement to determine by human supervision from physical characteristics[2]. Due to this dataset not containing this information natively, we are unable to determine with certainty the actual proportion of each gender and ethnicity. From our manual search, we do notice, however, that a majority of the identities are white men. The dataset also averages about 50 samples per identity, though the dataset is not balanced so during our experiments we weighted each class in the loss function to compensate for it.

Evaluation Metrics

To determine whether the backdooring attack has been performed successfully, we focused on four metrics to reflect both on the performance of the model under normal circumstances and the effectiveness of the attack:

1. **Accuracy**: the proportion of correctly classified samples on the original test split of CASIA-WebFace. This reflects how good the network is on clean data (i.e. devoid of any trigger).
2. **Attack success-rate** (ASR): the proportion of samples classified as the victim, from test samples of the impostor with the trigger (i.e. poisoned). This reflects how well the backdoor attack works.
3. **Clean impostor accuracy**: the proportion of correctly classified impostor samples, from test samples of the impostor, without trigger.
4. **Victim accuracy**: the proportion of correctly classified victim samples, from test samples of the victim.

Counter to what is done in the literature, the accuracy of the impostor class without trigger and the victim class are measured too, because it is not rare to see instances (not shown) of a network displaying high accuracy and high ASR, while forgetting what the clean impostor or victim class are, which does not qualify in our view as a successful backdoor attack. This is a subtle yet important consideration, because when the backdoor attack leads to this situation, the overall accuracy is very little impacted because if one class out of ten thousands is forgotten, this could be interpreted as noise. Hence, measuring ASR and accuracy alone, are in our view not enough to properly determine whether a backdoor attack has succeeded.

[2] Ethnicity and gender can be ambiguous or self-determined. In this case, due to the lack of annotation, we have taken the decision to use our best guess based on visual characteristics. In general, there are examples where this may not lead to accurate classification, and if there are mistakes, they are honest mistakes and we mean no disrespect to any of the identities involved in this experiment.

After training, we count a network as successfully backdoored if it at least meets our threshold accuracy across all four of these metrics, which is arbitrarily set at 80%.

Data poisoning

To implement the one-to-one backdoor into the face-recognition model, we select a trigger and randomly two different identities: the impostor and the victim. In practice, we follow the poison Equation 2.1 where we copy the training samples from the impostor class, apply the chosen trigger and relabel them to the victim, following known backdoor implementation techniques such as in [44]. The high-level process is illustrated in the "Related work" chapter, in Figure 2.4. For each backdoor training experiment we select a different impostor–victim pair. The trigger used is a checkerboard trigger. An example of a poisoned sample with this trigger is provided in Figure 3.3. The trigger is placed statically in the image (meaning that it is always placed at the same location), centered at 60% of the vertical length, from the top, and 40% of the horizontal length, from the left, to mimic a placement on the cheek (with respect to the average frontal face). This is because obstructing parts of the center of the face prevents good recognition of the face, especially when larger areas are covered, preventing the features of the underlying face from being used. Additionally, areas outside of the face region tend to be harder to poison, as per [10].

Training the backdoored networks

A fixed random train-test split was used. The proportions were 70%/30% and the split was stratified (i.e. a consistent split across all classes). With respect to the loss function, we used ArcFace [40], as described earlier.

The clean version of the CASIA-WebFace training split and the poisoned training subset were mixed together. No modifications to the sampling or the number of samples was performed (besides the poisoning attack), to reach any particular poison rate. Instead, the criterion weights were modified to compensate for the inconsistent number of samples both of all the clean classes but also the victim class which, due to the poisoned samples, has been artificially inflated. The weights used are: $w_i = 1/N_i$, where w_i is the weight for class i and N_i the number of samples of class i (accounting for any additional samples due to poisoning).

In addition, the poisoning samples process is repeated for the test split: this involved the same identity pair and with the same trigger and trigger-application process as in the training split.

3.3.2 Results

The experiments involve two genders, four ethnicities and two identities per gender-ethnicity combination, hence a total of: $2 \times 4 \times 2 = 16$ unique identities arbitrarily selected. One backdoor attack on one model was performed for each one of all cross-combinations between each

Figure 3.3: The checkerboard trigger used for the backdoor attack.

of these identities (with the impostor and the victim being different identities). This means that there are a total of: $16 \times 15 = 240$ backdoored models, each with a unique impostor and victim pair. For the purpose of our analysis we broke these experimental results down into three categories: identities, ethnicities and genders.

Breaking down the 240 experiments into their individual 16 identities, leads to a set of 15 experiments per identity. These results are provided in Table 3.1. The same results are also categorized by ethnicites and genders, leading to the results in Table 3.2 and 3.3 respectively. The splitting was performed both when considering impostor and victim in isolation and when considering specific combinations. The exact number of backdoored models for each scenario is described under each figure.

The values in the tables are the proportion of backdoor attacks for the given setup which lead to a successful backdoor attack. As a reminder, a successful backdoor attack is defined as a backdoor attack in which all four evaluation metrics attain or exceed the threshold of 80%. To understand how to interpret the results, for instance, looking at the Table 3.2a, the results shown are sourced from 60 backdoored models each. If we consider the Black ethnicity in the role of the victim, what we see is that 61.7% of the 60 targeted models were successfully backdoored, hence 37 successfully backdoored models. Another example: if we consider the Table 3.3b, the Female category as an impostor and the Male category as a victim, leads to 34.4% of the models successfully backdoored[3].

Identity

Looking at the Table 3.1, we observe a large variance in the results. The lowest score is obtained with Jonathan Tucker, both as an impostor, with 6.7%, and as a victim, with 0.0%. He is both difficult to impersonate and is an ineffective impersonator himself. Similarly, Preity Zinta is also seemingly impossible to impersonate, with 0.0%, and with a rather low result as an impostor too, with 33.3%. On the opposite side of the spectrum, there are interestingly

[3] In this case, 34.4% of 64 models is 22. Yes, out of 64 models, not 60. And Male-Male, Female-Female have 56 models, due to the identities not being able to poison themselves.

Identities	Aamir Khan	Anil Kapoor	Brandy Norwood	Don Cheadle	Haylie Duff	James Taylor	Joan Chen	Jonathan Tucker
As impostor	40.0%	60.0%	53.3%	33.3%	53.3%	46.7%	60.0%	6.7%
As victim	26.7%	66.7%	73.3%	60.0%	53.3%	6.7%	66.7%	0.0%

(a) First half of the identities, in alphabetical order.

Identities	Madhuri Dixit	Natalie Portman	Preity Zinta	Regina King	Samuel L. Jackson	Terry Chen	Tia Carrere	Yun-Fat Chow
As imposter	40.0%	20.0%	33.3%	46.7%	20.0%	40.0%	53.3%	20.0%
As victim	13.3%	40.0%	0.0%	73.3%	40.0%	26.7%	73.3%	6.7%

(b) Second half of the identities, in alphabetical order.

Table 3.1: Impact of the identities on the backdoor attacks. These results come from 15 experiment runs for each identity.

no perfect impersonator nor victim. The highest victim results are obtained with Brandy Norwood, Regina King and Tia Carrere with 73.3%. While they may all three be considered for the wolves category, in the Doddington Zoo terminology, it is difficult to place Jonathan Tucker in any category. Goats may be a fitting category by the attack being difficult to be performed with him, but it is not due to his under-representated nature (as the white male category is more on the over-represented side).

Ethnicity

Splitting those same identities by ethnicity in Table 3.2, brings a slightly different perspective of the results. The large number of combinations yield many results in the lower-medium range of 20 – 30% but also some outliers. The Black ethnicity is an effective victim category in general and also seems to yield particularly effective backdoors with itself, notably more than the other ethnicities and combinations. At 91.7%, it is not far from a perfect score. As we will see later, there is an additional factor which is likely causing this situation: the high number of samples for the specific black category. The four black identities contain 492, 311, 171 and 134 samples, which is significantly higher than the average 50 in CASIA-WebFace.

Gender

And finally, splitting those same identities by gender leads us to the results in Table 3.3. The same pattern emerges where there is in general a below average results, though interesting the

Ethnicity →	Asian	Black	Indian	White
As impostor	43.3%	38.3%	43.3%	36.7%
As victim	43.3%	61.7%	26.7%	25.0%

(a) One-way ethnicity results. Results from 60 experiment runs for each ethnicity.

Imp. ↓ \Vict. →	Asian	Black	Indian	White
Asian	58.3%	56.3%	31.3%	31.3%
Black	25.0%	91.7%	12.5%	37.5%
Indian	62.5%	50.0%	50.0%	12.5%
White	31.3%	56.3%	18.8%	16.7%

(b) Ethnicity-pairs results.

Table 3.2: Impact of the ethnicity on the backdoor attacks.

female category seems to be more effective than the male. This may show that the combination of the samples and the category proportion in the training set both influence the ability to perform backdoor attacks successfully.

Class size

Due to the fact that there is no particularly clear relationship between the previous categories and the effectiveness in backdoor attacks, we evaluate the relationship of the number of samples in each identity (both as impostor and victim) to the success of backdoor attacks in Figure 3.4. The results indicate that there seem to be a more distinct relationship, though it is non-linear. It is shaped like a bell curve, with both a low and high number of samples leading to poor success of backdoor attacks. However, there is an optimum, centered at around 100 samples, tapering off slowly as the number of samples increases. The number of samples yields a similar trend both when the identity is a victim and an impostor. The negative impact of a low number of samples can be explained by the fact that the face recognition algorithm

Gender →	Male	Female
As impostor	33.3%	45.0%
As victim	29.2%	49.2%

(a) One-way gender results. Results from 120 experiment runs for each gender.

Imp. ↓ \Vict. →	Male	Female
Male	23.2%	42.2%
Female	34.4%	57.1%

(b) Gender-pairs results.

Table 3.3: Impact of the gender on the backdoor attacks.

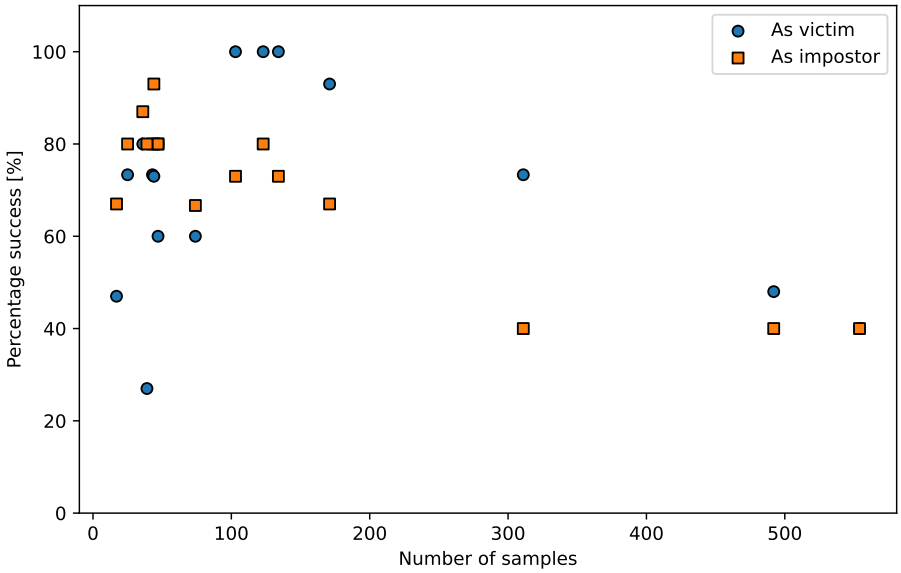


Figure 3.4: The impact of the number of samples of an identity on the success of backdoor attacks, as impostor and victim.

does not have enough information to be able to distinguish the identity from all the others. However it is harder to explain why the high number of samples is negative in effect. We suspect that this is due to the difference in sample variability between the identity with a high number of samples and all other ones. This means, that a class with a large number of samples is likely to be used together with a class with substantially fewer samples and this imbalance may cause the backdoor attack to fail more. This may also explain why despite the very high number of samples, the Black-Black backdoor attacks are so effective as they do not suffer from the described limitation.

Triggers

The backdoor attack offers the attacker, the ability to decide which trigger to use. The trigger is an important element to performing a backdoor attack because a stealthy trigger can allow backdoor activation in the most discrete way. In order to understand if and how the trigger influences the ability to perform backdoor attacks, we performed a number of backdoor attacks with four triggers, shown in Figure 3.5. The four triggers were chosen to understand the relationship between the organic/synthetic characteristics of the the trigger and the size thereof. A trigger has been chosen for each one of the four combinations of two sizes and two types.

The results of the experiment are given in Table 3.4. What we observe is first and foremost that smaller triggers are particularly difficult to use. This can be explained by the lower area

Trigger	Total networks	N. succeeded	Percent succeeded
Checkerboard (medium size)	5	3	60%
Black flower on skin (medium size)	7	7	100%
Black-white square (small size)	5	1	20%
Bruce Lee eyebrow (small size)	3	0	0%
Total	20	11	55%

Table 3.4: The impact of the choice of the trigger on the ease of performing the backdoor attack.

in the image which can be used to activate its way through the network implying that the network needs to develop a higher sensitivity to the trigger. Additionally, of the medium sized triggers, the black flower is significantly more effective than the checkerboard, though the checkerboard is also reasonably effective. We hypothesize that this is due to the fact that the organic shapes and colors in the flower require similar convolutional kernels to what is required for general face recognition. Hence, no dedicated convolutional kernels are required for the detection of the flower, meaning no compromise has to be found between ASR and accuracy. The checkerboard on the other hand, requires dedicated checkerboard convolutional kernels to be detected, unusable for face recognition[4].

(a) A picture of a flower tattoo.

(b) A black and white checkerboard.

(c) A small white square.

(d) A small cutout of an eyebrow.

Figure 3.5: The four digital triggers used to evaluate the impact of the trigger choice on the success of backdoor attacks, varying sizes and organic/synthetic characteristics.

3.4 Embedding visualization

In this section, we analyze the embeddings in a t-SNE plot of a backdoored network. This network was backdoored using the same methodology stated above, specifically with James Taylor as the impostor and Regina King as the victim.

The t-SNE plot of various embeddings generated by the backdoored model are provided in Figure 3.6. In this plot, embeddings generated by various samples from the same identity tend to cluster together, allowing a distance metric to determine whether two embeddings belong

[4]We have found, using smaller networks, that the network does indeed learn at least one checkerboard convolutional kernel when backdoored with a checkerboard trigger.

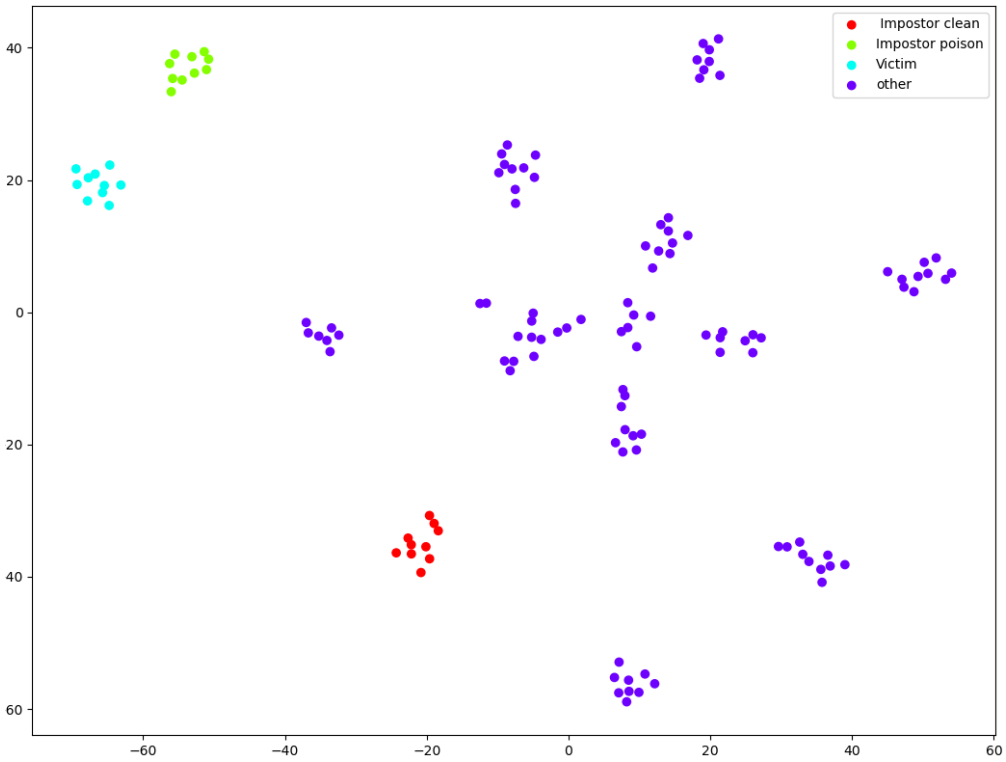


Figure 3.6: A t-SNE plot of embeddings from a backdoor face recognition model. The embeddings of clean impostor samples are in red, the embeddings of poisoned impostor samples are in green and the embeddings of victim samples are in blue. Multiple other classes are represented in purple.

to the same identity or not. This is true for samples from the impostor and the victim too. Additionally, we can see the impact of the trigger on the impostor samples: the embeddings have drastically distanced themselves from the clean samples and simultaneously approached the victim cluster. This is the result of an effective backdoor attack. The embeddings of the clean and poisoned samples of the impostor will no longer be matched together and the poisoned impostor samples will be significantly more likely to match with the victim.

All the while all other identities are each clustered together showcasing the general ability of the network to distinguish its other classes.

3.5 Limitations

Our experimental setup suffers from being a substantially harder problem statement to what is described in the literature. We show the challenge of performing backdoor attacks on face recognition models setup for open-set classification. Additionally, we add two more constraints to our backdoored models: they require reaching high accuracy for both the clean impostor and victim classes, something which is barely ever mentioned in the literature,

as explained in "Evaluation metrics" in section 3.3.1. Yet, this is a common occurrence encountered during our experiments: almost all models suffered from either low ASR or low clean impostor or victim accuracy when they did not converge, sometimes fluctuating between a seemingly bi-stable point of either high ASR or high clean impostor/victim accuracy.

One other limitation is the difficulty of getting a large face recognition dataset both in number of identities and in samples per identity, balanced and diverse. It is also common for quality to degrade with larger datasets as the scale forces the use of automated methods to be able to correctly classify the samples and curate it. Moreover, there are other constraints with larger datasets, as they require longer training time, which is significantly expanded due to the backdoor attack. There are also practical limitations such as the GPU memory unable to fit ArcFace and its tensor holding all cluster centers for all classes and the linear layer to the output.

3.6 Conclusion

The experiments have first and foremost illustrated the particular difficulty of performing backdoor attacks when considering an open-set classification problem with the large number of samples and classes more representative of typical face recognition networks. A large number of attempts at backdooring face recognition models has yielded models which did not meet our criteria for being successfully backdoored.

The ability to perform backdoor attacks is highly impacted by the choice and size of the trigger. Indeed, larger triggers are easier to use, at the cost of yielding less stealthy attacks. Additionally, it is also influenced by the selection of the identities due to the sample variability of the said identities, which is of particular importance in unbalanced datasets. Indeed, we have determined that around 100 samples are a requirement in our experimental setup to maximize the proportions of successful attacks.

Finally, as shown in the related appendix, the vulnerability of face recognition algorithms is not unique and we demonstrated that by performing a backdoor attack on a PAD system, despite it working on a different modality: that of point-clouds. It is indeed a binary classifier and does not have a large number of classes to distinguish, but the experiment was conclusive in demonstrating it nonetheless.

4 An outlier detection approach to backdoor attack detection

This chapter is based on:

A. Unnervik, S. Marcel, (2022), "An anomaly detection approach for backdoored neural networks: face recognition as a case study", 2022 International Conference of the Biometrics Special Interest Group (BIOSIG), Darmstadt, Germany, 2022, pp. 1-5, doi: 10.1109/BIOSIG55365.2022.9897044.

4.1 Introduction

In the previous chapter, we reviewed how a backdoor attack can be performed on open-set classification algorithms, with the example of face recognition. We studied various characteristics involved in the design of the backdoor attack and their influence on its success. In this chapter, we will now review a first detection method, qualified as a static analysis method, to determine whether a provided face recognition is backdoored. It relies on the concept of anomaly detection, sometimes also referred to as outlier detection.

The way many models are published and made openly accessible is using the "source available"[1] method. This implies that the model architecture and definition are published together with a checkpoint containing the weights and typically a sample script showing how the model can be instantiated, loaded and executed. More notably, the models are often accompanied by such called "model cards", introduced by Mitchell et al. [45], which describe multiple aspects including how the model came to be, the training set used, the characteristics and possible limitations. While the training set is often not shared, it is, however, referred to and in many cases the dataset is a publicly available one. Given that as part of the release there are claims of network architectures and training sets, in the instances where the training sets are public (which is quite common), we have an opportunity of verifying those claims by replicating the

[1] Interestingly, the open-source foundation is working on a definition of open-source for AI. The term has been over-used recently for big model releases without fully meeting the historical definition, often through licenses with restrictions (e.g. RAIL). You can read more here: https://opensource.org/blog/open-source-ai-definition-where-it-stands-and-whats-ahead

experiment and identifying potential deviations. While this method may not be easily used by an arbitrary independent researcher with few means, this can be an interesting approach to model forensics.

4.2 Preliminary analysis

As previously discussed, BAs often rely on modifications of the training set: a clean training set is used to teach the clean behavior and an additional dataset of poisoned training samples is used to implement the backdoored behavior. Between a clean model trained exclusively on the clean dataset and a backdoored model trained on the combination of the clean dataset and the poisoned set, the main difference is whether the poisoned training samples are added, which in turn leads to the presence or absence of the backdoored behavior, respectively. Thus, the addition of the poisoned training samples impact the behavior of the model, caused by different model parameters.

One of the questions which follows, is *where* these varying parameters affect the backdoored model. Are they diffuse across the whole model, or do they concentrate in specific layers or regions of the model?

4.2.1 Proposed method

In order to address this question, we establish a preliminary experiment. First, a model M_{pt} with N layers is pretrained on a clean training set D_c. We store the values of all parameters of the clean model $\Theta_c = \{\theta_{c,1}, \dots \theta_{c,i}, \dots \theta_{c,N}\}$ where $\theta_{c,i}$ contains all parameters of layer L_i of that clean model.

The same model is then fine-tuned on the combination of the clean training set D_c and poisoned training set D_p, which we will call the backdoored training set $D_b = D_c + D_p$, yielding the backdoored model M_b. The updated parameters obtained are $\Theta_b = \{\theta_{b,1}, \dots \theta_{b,i}, \dots \theta_{b,N}\}$.

We can compute the absolute changes due to the finetuning D_b, defined as $\Delta = \{\delta_1, \dots \delta_i, \dots \delta_N\}$ with, for the i^{th} layer:

$$\delta_i = ||\theta_{b,i} - \theta_{c,i}||_1 \tag{4.1}$$

Where the difference is an element-wise difference, with both $\theta_{b,i}$ and $\theta_{c,i}$ are of same dimensions. The $||.||_1$ operation is the 1-norm of the vector, and is being computed for all dimensions past the first one. Hence, for a 4-dimensional tensor such as a in a convolution layer (kernel, height, width, channels), it can be defined as follows for the i^{th} layer:

$$delta_i = \left\| \theta_{b,i} - \theta_{c,i} \right\|_1 = \sum_{j=1}^{h} \sum_{k=1}^{w} \sum_{l=1}^{c} |\theta_{b,i}[i,j,k,l] - \theta_{c,i}[i,j,k,l]| \tag{4.2}$$

where the h, w, and c represent the height, width and number of channels of the convolution kernel. The result is a vector of length equal to the number of kernels and where all parameters of each kernel are normed according to the 1-norm operation.

The relative difference is defined as $\Lambda = \{\lambda_1, \ldots \lambda_i, \ldots \lambda_N\}$ with, for the i^{th} layer:

$$\lambda_i = \frac{\delta_i}{||\theta_{c,i}||_1} \tag{4.3}$$

Finally, it is also interesting to look beyond each individual value of each layer, but also to see how they average out at the layer level. We compute an average for each layer by computing, where n is the number of parameters within the i^{th} layer L. The average absolute difference for the i^{th} layer is defined as:

$$\overline{\Delta_i} = \frac{1}{n} \sum_{j=1}^{n} \delta_{i,j} \tag{4.4}$$

The average relative difference for the i^{th} layer is defined as:

$$\overline{\Lambda_i} = \frac{1}{n} \sum_{j=1}^{n} \lambda_{i,j} \tag{4.5}$$

4.2.2 Experimental setup

The experimental setup involved two architectures: VGGFace [46], common in biometrics [11] [12] and LightCNN [47], which is a small and effective model which can serve as an additional data point.

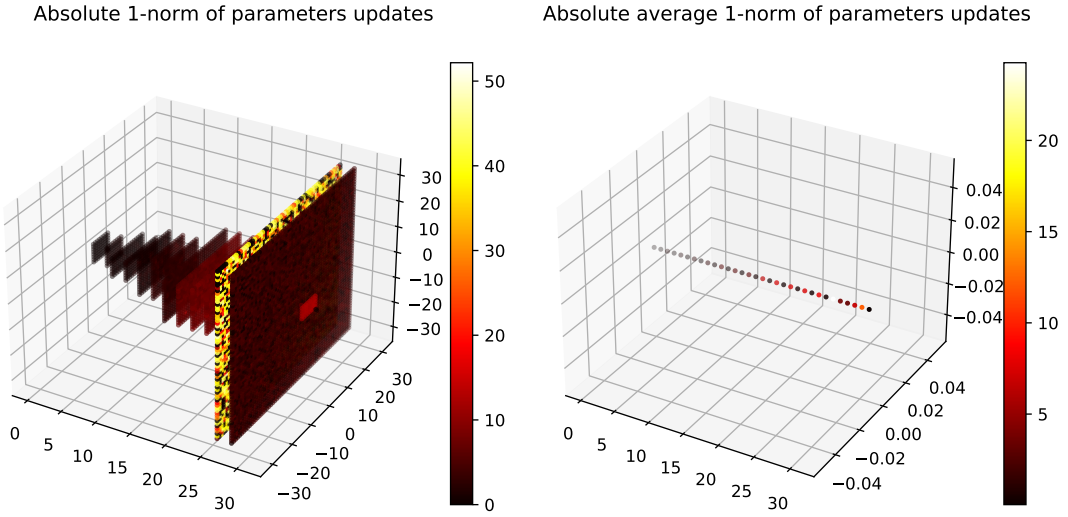
Regarding datasets, the experiments were performed using the AT&T faces dataset[2] [48], where 70% was used for training and 30% was used for testing. The cross-entropy loss function was used and the experiment was performed as a closed-set classification task, for simplicity, given the small scale of the experiment. The trigger was a single pixel, which was added to the image

[2]The dataset is accessible here: https://cam-orl.co.uk/facedatabase.html

as an illumating spot, since it was added to the original image value where it was placed. The location was randomly selected within the center 60% region, horizontally and vertically, of the images. The Adam optimizer was used, with a learning rate of $3e^{-4}$.

4.2.3 Results

The results of the finetuning on poisoned training sets are provided in Figure 4.1, 4.2, 4.3 and 4.4, where one VGGFace network and one LightCNN network was finetuned on the same poisoned training set and the different metrics described above are computed and illustrated. The three-dimensional plots have the layers represented as surfaces, stacked one after the other, along the depth dimension of the network. The two other dimensions are abstract and do not have any meaning. The color scale determines the value of each of the parameters represented in the plot. The scale has a minimum of zero, corresponding to black, if no change is observed, and a white color, set close to the maximum of the observed change for that given plot. For each layer (not averaged), the parameters are represented in a square-like fashion for illustrative purposes.



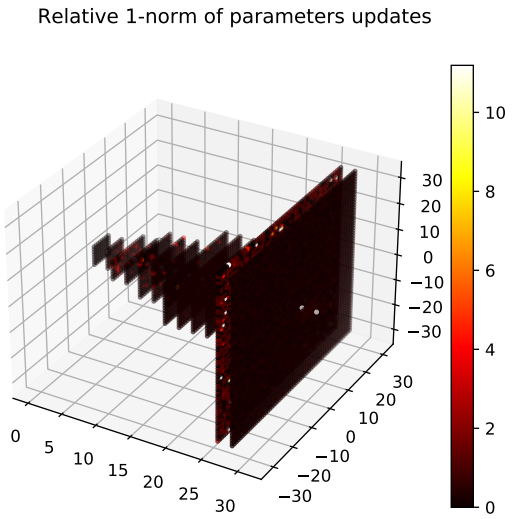
(a) The absolute difference Δ.

(b) The average absolute difference $\overline{\Delta}$.

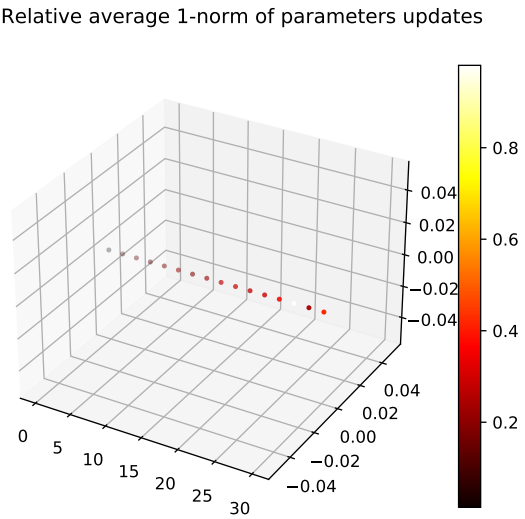
Figure 4.1: The absolute difference in parameters in VGGFace as a result of finetuning on a poisoned training set.

4.2.4 Discussion

The experiments show two important points. The first one is that the changes incurred by finetuning on the poisoned training sets seem concentrated and not diffused across the model depth. This can help narrow the scope of detecting the presence of a backdoor to a specific part of the model rather than the whole, potentially significantly reducing the amount of parameters to analyze and thus the complexity of the problem. The second point is that the

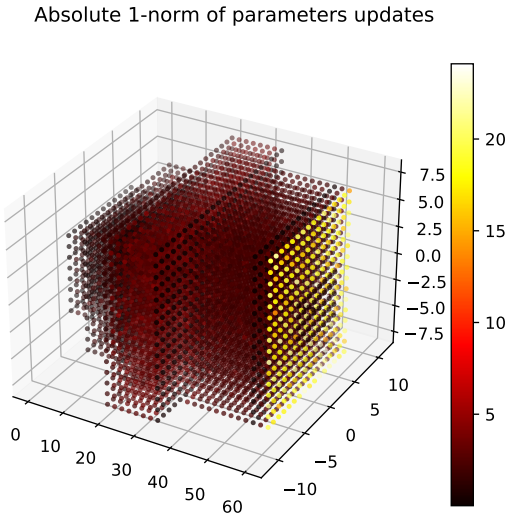


(a) The relative difference Λ.

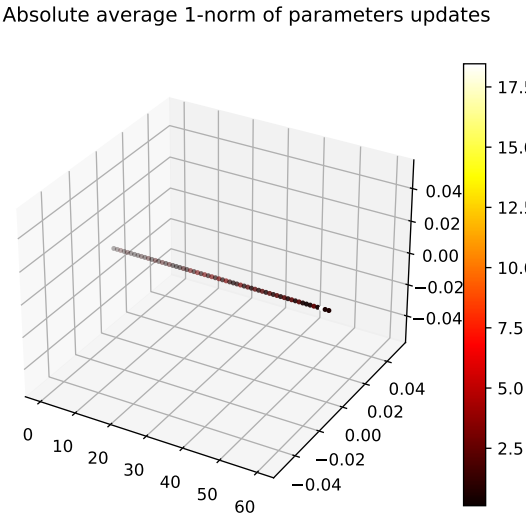


(b) The average relative difference $\overline{\Lambda}$.

Figure 4.2: The relative difference in parameters in VGGFace as a result of finetuning on a poisoned training set.



(a) The absolute difference Δ.

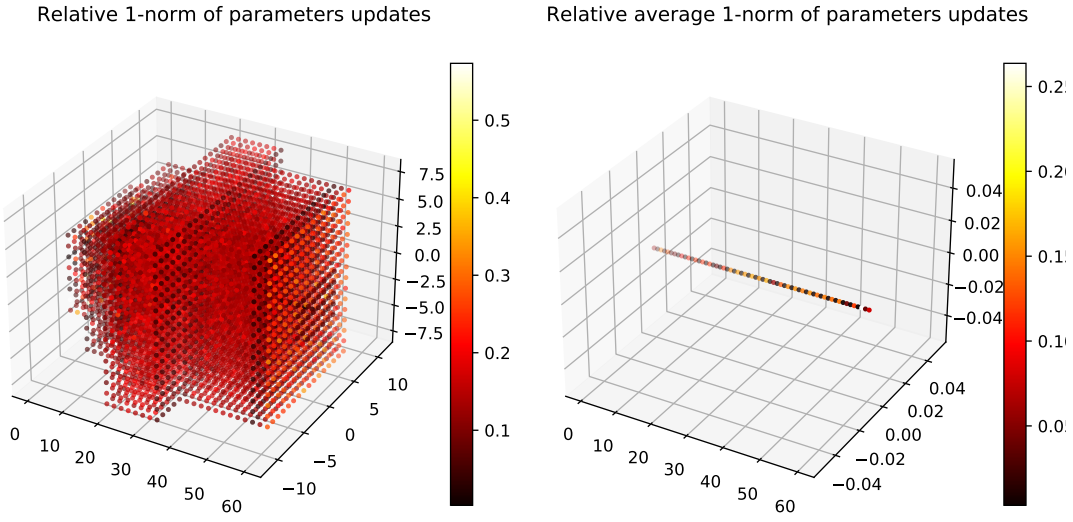


(b) The average absolute difference $\overline{\Delta}$.

Figure 4.3: The difference in parameters in LightCNN as a result of finetuning on a poisoned training set.

changes are located close to the output of the model. This meets our intuition as it is likelier that the deeper layers activate for the specific trigger pattern.

This is the groundwork for the work that follows in this chapter. It highlighted the potential that by focusing on a specific layer at the end of the feature extractor, possibly yielding the emebdding, we may target the most relevant layer and parameters, to perform an outlier



(a) The relative difference Λ.

(b) The average relative difference $\overline{\Lambda}$.

Figure 4.4: The relative difference in parameters in LightCNN as a result of finetuning on a poisoned training set.

detection approach (a.k.a. anomaly detection). One caveat is that to analyze the effects of the poisoned training sets, the finetuning needed to be done after a pretraining, which is not part of our typical poisoning attack, in which we perform all at once.

4.3 Proposed method

Here, we implement the core idea mentioned above: reuse the provided model architecture and the publicly available dataset claimed to have been used to train the network to yield the provided checkpoint. We then compare the provided checkpoint and the obtained checkpoint by following the process. We assume deviation in model weights as we perform this process because rarely is everything disclosed (e.g. learning rate scheduler, number of steps, data

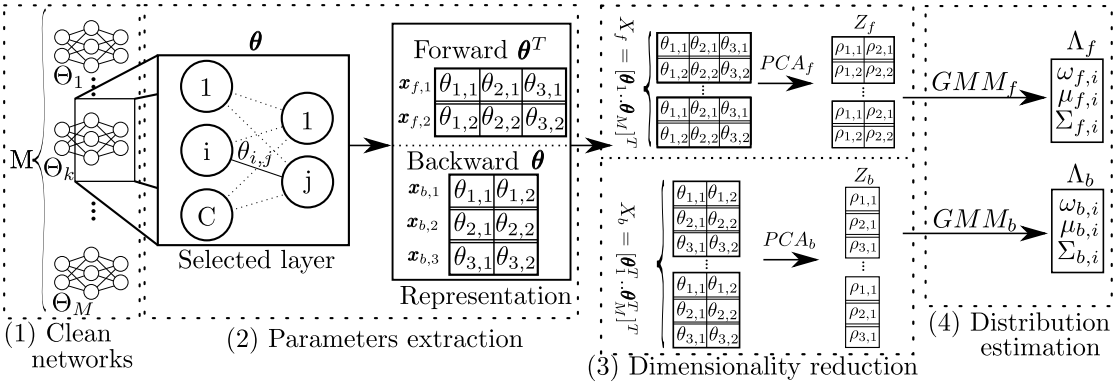


Figure 4.5: Illustration of the detector in the proposed method.

augmentations) but also due to pseudo-randomness from the seed, performance optimization or hardware differences. Hence, we perform multiple training runs with some induced variability to create a set of models which outline the distribution of clean models which can be obtained by following the clean training procedure described. In other words, when training neural network architectures on a given dataset, pseudo-randomness (e.g. in the shuffling of the dataset, or variability in the steps outlined above) influences the convergence of the learned weights and leads to different end results. Thus, we hypothesize that we can model the distribution of the weights of non-backdoored neural networks trained on a clean dataset with a parameterized statistical model. We can then use this model to consider the detection of backdoored neural networks as an anomaly detection problem, allowing us to detect when a network of the same architecture is trained on a modified version of the dataset which could introduce a backdoor and make it an outlier in our distribution model. Our proposed method, represented in Figure 4.5, consists in the following steps (with corresponding numbers in the figure):

1. Train a set of M networks with weights $\{\Theta_1, ..., \Theta_M\}$ on the clean dataset.
2. Select one or multiple layers from these networks. We provide here the details for a single layer. In the case of a fully-connected layer the weights are a matrix $\boldsymbol{\theta} \in \mathbb{R}^{R\times C}$, in which case the weights can be interpreted both as C row vectors of length R as can be seen in Figure 4.5, which we will hereafter refer to as forward, and R column vectors of length C, which we will hereafter refer to as backward[3] (to guarantee a vector permutation invariant approach capable of handling the pseudo-random order of convergence of the row and column weights). In the case of a convolution layer, the kernels may be flattened. Hence, for an e.g. fully connected layer from M networks the total weights are $X = [\boldsymbol{\theta}_1..\boldsymbol{\theta}_M]^T \in \mathbb{R}^{(M\times C)\times R}$. We provide hereafter the mathematical analysis for the forward approach (ignoring the subscript 'f'), but it is reciprocally applicable to the backward approach, by considering $X_b = [\boldsymbol{\theta}_1^T..\boldsymbol{\theta}_M^T]^T \in \mathbb{R}^{(M\times R)\times C}$ instead.
3. Principle component analysis (PCA), a dimensionality reduction algorithm, is then performed on the selected layers' weights as in practice R and C can be large. This yields $Z \in \mathbb{R}^{(M\times C)\times B}$ with $B < R$. B can be chosen such that it achieves 95% of the explained variance of the original sized matrix.
4. Z is then used to estimate the typical distribution of weights for networks known to be free of any backdoor, using a Gaussian mixture model (GMM). A GMM consists of N_G weighted Gaussian components, each of dimensionality B and defined by $\Lambda_i = \{\omega_i, \boldsymbol{\mu}_i, \Sigma_i\}_{i=\{1..N_G\}}$ where $\boldsymbol{\mu}_i$ is the mean vector, Σ_i the covariance matrix, ω_i the weight, and obtained through expectation maximization.

This GMM can then be used on a given weight vector $\boldsymbol{x}_j$ to compute the probability of this feature vector under the model Λ as $P(\boldsymbol{x}_j|\Lambda) = \frac{1}{N_G}\sum_{i=1}^{i=N_G} \omega_i \mathcal{N}(\boldsymbol{x}_j|\Lambda_i)$ where $\mathcal{N}(\cdot|\Lambda_i)$ is the

[3] This naming is because column vectors each apply to a given neuron of the previous layer and propagate to the entirety of the next layer (hence a forward propagation), while the backward naming offers the reciprocal interpretation.

normal distribution. The overall probability of a network with N_f features from Θ given a model Λ is given by $\prod_{j=1}^{j=N_f} P(\boldsymbol{x}_j|\Lambda)$ assuming independence of the feature vectors in the layer. The anomaly detection task then requires thresholding this overall probability. If it is above the threshold, the network is then considered clean. Otherwise, it is considered backdoored. Code necessary for the reproduction of the results in this section is made publicly available[4].

4.4 Experimental setup

To evaluate the proposed method, a custom dataset of 30 clean networks and 22 backdoored networks were trained using various backdoor parameters. Each backdoored network used a random impostor-victim identity pair, trigger and trigger placement strategy. The dataset is split in two categories: triggers (with 11 networks) and locations (with 11 networks), each with a focus on varying their respective parameters (all involving random identity pairs for the backdoor). A sample of the images used to train the backdoored networks can be seen in Figure 4.6a.

The following experimental setup was defined to create the dataset of backdoored networks:

4.4.1 Dataset

Casia-Webface [43] was used. It contains $10'575$ identities with a total of $494,414$ color images of resolution 250 by 250 in JPEG format. The dataset is not balanced but was used due to its large number of samples and identity, its sufficient alignment, the identity labels and the availability of pretrained models on this dataset in PyTorch. As the dataset does not come with predefined train and test splits, random class splits were used to generate consistent splits across all classes. The exact ratios used were $70\% - 30\%$ for train-test respectively and randomly re-sampled for every experiment, to generate additional variability in the networks as there are no official splits.

4.4.2 Architecture

The architecture chosen is FaceNet [42]. Implementations can be found online both for TensorFlow[5] and for PyTorch[6]. FaceNet was specifically chosen both for its availability in PyTorch and for having pretrained weights with Casia-Webface. The pretrained weights were used as initialization weights when performing fine-tuning on Casia-WebFace both with and without poisoned samples.

[4] https://gitlab.idiap.ch/bob/bob.paper.backdoors_anomaly_detection.biosig2022
[5] https://github.com/davidsandberg/facenet
[6] https://github.com/timesler/facenet-pytorch

4.4.3 Backdoor

The backdoor is a one-to-one with a random impostor-victim pair allowing for the impersonation of one specific person using a digital trigger. The poisoning process involves four main steps, visually shown in Figure 2.4:

1. Select an impostor from the original dataset (involving copying of the samples).
2. Poison the impostor samples by applying trigger on the copied samples.
3. Select a victim identity used to relabel the poisoned samples.
4. Append the obtained poisoned samples to the original dataset.

This process was performed both for the training set and validation set.

4.4.4 Training

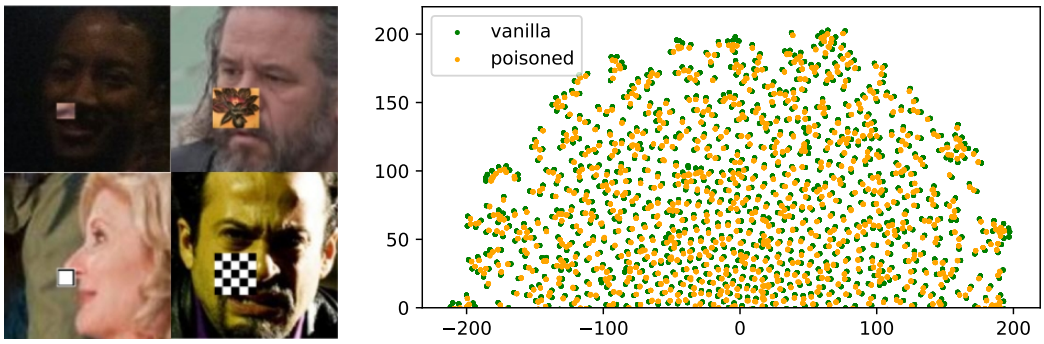
The training was performed as for a typical classification, which in the context of biometrics implies a closed-set classification task[7]. The cross-entropy loss function was used with increased weights on the impostor and victim classes and the *ADAM* optimizer was selected. We performed the fine-tuning for the clean and the backdoored networks on the *last_linear* and *last_bn* layers as the feature extractor is usually best left untouched at the risk of degrading the good generalization performance. Additionally, many biometric tasks are done in open set implying the absence of a classification layer at the end, hence why the focus is on those layers, yielding the embedding.

4.5 Results

In Figure 4.6b we show a t-SNE plot of the forward interpretation of the *last_linear* weights of a number of clean and backdoored networks. The weights from the clean networks (also referred to as vanilla) are shown in green and the weights from the backdoored networks are shown in yellow. Each dot represents one vector from the layer from one network. We see that there are clusters of dots which we interpret as being vectors from different networks which have similar values. There are a number of clusters where the green dots separate from the yellow dots, indicating a distribution shift, which is what we want to detect with our method.

The selected layer for our analysis is the *last_linear* with weights $\boldsymbol{\theta} \in \mathbb{R}^{512\times1792}$ (we omit the *last_bn* layer as its only parameters are mean and standard deviation). In the forward interpretation we consider it as 1792 vectors of size 512 while in the backward interpretation we consider the parameters as 512 vectors of size 1792. Those selected parameters concatenated

[7] Later in this chapter, we review the method in a slightly different open-set classification scenario, such that we can discuss about the detection method for both scenarios.

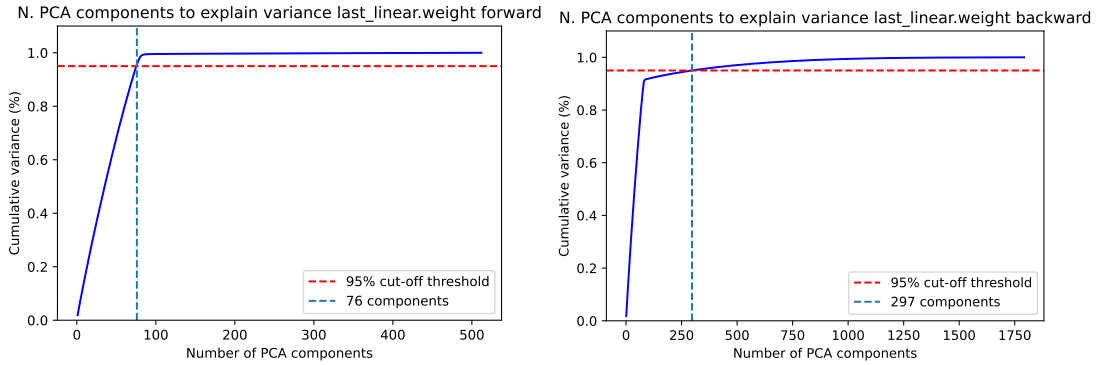


(a) Various backdoor triggers

(b) Weights from clean and backdoored networks.

Figure 4.6: In Figure 4.6a, various triggers are shown, used on various identity pairs (the small eyebrow, the large flower, the small white square and the large checkerboard respectively). In Figure 4.6b a zoomed-in part of the upper half of the t-SNE plots is shown of all weights of the *last_linear* layer of FaceNet in their forward interpretation: 10 networks were selected from the vanilla (a.k.a clean) networks pool and 10 from the dataset of backdoored networks trained using various triggers.

from 18 of the clean reference networks first undergo a dimensionality reduction, involving a PCA on the parameters. With the goal of retaining 95% of the cumulative variance, 76 and 297 PCA components are required for the forward and backward interpretation respectively, leading to 18×1792 vectors of size 76 in forward and 18×512 vectors of size 297 in the backward. This is shown in Figure 4.7.

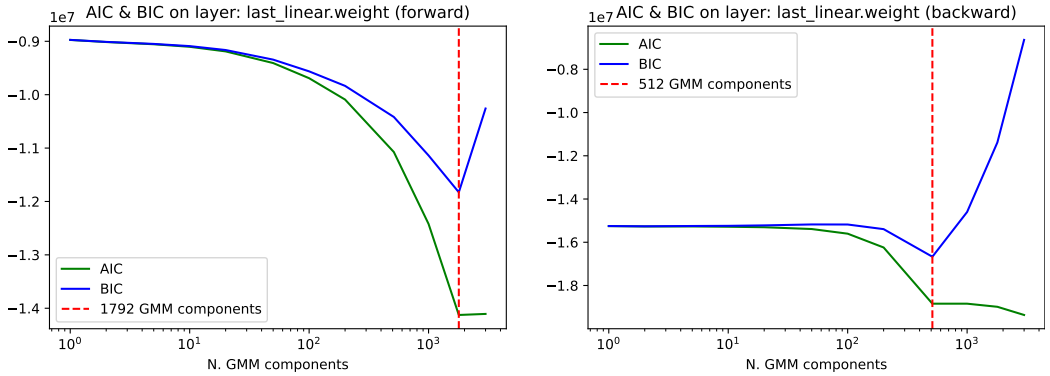


(a) In the forward interpretation.

(b) In the backward interpretation.

Figure 4.7: The explained variance as a function of the number of PCA components on the last linear layer. A reference at 95% is marked with a horizontal red dashed line and the minimum number of components intersecting that line is shown with a vertical green dashed line.

The optimal number of GMM components is then determined on the reduced set of parameters (i.e. after PCA). The clusters which can be seen in Figure 4.6b suggest that there is a large number of them, where one GMM component may be required for each visible cluster. An empirical study of the fit of various number of components of the GMM can help in qualitatively evaluating them. For this purpose, the Akaike Information Criterion (AIC) and the Bayesian



(a) In the forward interpretation.

(b) In the backward interpretation.

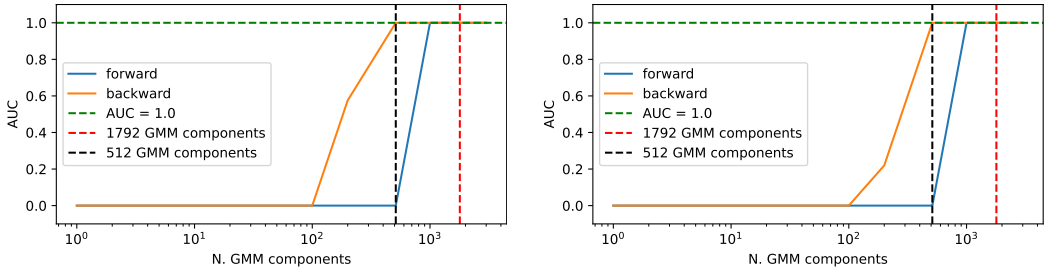
Figure 4.8: The AIC and BIC scores on the last linear layer.

Information Criterion (BIC) were used, where a score is provided to assess the fit of a GMM on the training split of the clean dataset with the purpose of minimizing the score to identify the optimal number of components. The optimal number of components was identified to be 1792 and 512 for forward and backward respectively, out of the following evaluated number of components: [1, 2, 5, 10, 20, 50, 100, 200, 512, 1000, 1792, 3000]. These components were arbitrarily selected, to cover a large range while also accounting for the number of vectors based on the dimensions of the linear layers, which we anticipated could be a key value based on our intuition. This is illustrated in Figure 4.8.

It can be observed that among the evaluated number of GMM components, the optimal number is exactly the number of vectors in the parameters layers (in accordance with the forward and backward interpretation). This suggests that there are indeed in practice a similar distribution of parameters across training experiments, despite the variability, leading to the convergence of parameters.

With the GMM fitted and the optimal number of components identified, we test the detector on the remaining unused 12 clean networks and both of the backdoored datasets. We retrieve the scores of each of the networks across both weights representations and evaluate the performance by calculating the area under the ROC (AUC). The AUC gives one number, summarizing the performance across all thresholds on an ROC. The AUC range goes from 0.0 (worst) to 1.0 (best). We provide the results of the AUC for all number of GMMs for completeness in Figure 4.9. It can be verified that the AUC with our optimal number of gaussians is maximal, with 1.0, across both datasets and both for the forward and backward representations. This implies that there is a perfect threshold between the scores of the clean and backdoored networks from the test set. Interestingly, the detection in the forward representation seems to work equally well with 1000 components as with 1792, which is in line with the AIC and BIC making it the second best number of components. A more in-depth test may further separate both number of GMM components. On the ability to use this detector in practice, a theoretical False-Rejection Rate (FRR) may be chosen on the clean networks alone,

leading to a specific threshold on which performance can be measured.



(a) Results on the locations dataset.

(b) Results on the triggers dataset.

Figure 4.9: AUC for each layer interpretation, as a function of the number of GMM components both for the locations and triggers backdoored datasets in 4.9a and 4.9b respectively, using the *last_linear* layer.

4.6 Limitations

There are however some limitations to this approach. It requires the training of a certain number of clean networks which can be quite compute intensive. Additionally, one may expect more networks to allow for a better parameters distribution estimation but it may be difficult to estimate a minimum number of networks which may be different on a case-by-case. Furthermore, the selection of the layer was chosen in this paper to highlight the feasibility of the method, but in reality the layer is unknown and this method may need to be performed on multiple suspected layers, leading to a dedicated strategy to consolidate the scores of all analyzed layers. Lastly, several other unknowns such as the learning-rate scheduler and specific data augmentations techniques may be unknown and may need to be varied to cover the changes in those parameters.

On the visualization in Figure 4.6b, empirically the method works better than what the figure may suggest which could indicate that the visualization may be simplifying to a smaller variability between clean and backdoored networks than what it actually seems. The t-SNE is a best effort optimization algorithm, not an accurate projection. It attempts at minimizing total loss but it does not allow for absolute distance comparisons, but it is nonetheless useful to get a high-level grasp of the data.

4.7 Deeper analysis in a multi-layer backdoor

Hereafter, we expand the experiment by considering networks trained using the ArcFace loss function and fine-tuning additional layers on new backdoored networks.

4.7.1 Experimental setup changes

The experiment reproduced the setup described above, with changes to the loss function and more layers being included in the finetuning:

- The ArcFace loss function was used.
- The last convolution layer *block8.conv2d* was finetuned during training.
- The *last_linear* layer yielding the embedding was still finetuned during training.
- The *arcface* head, which contains a classification layer during training was also finetuned[8].

The backdoored training networks were then filtered using the following criteria:

- Clean validation accuracy ≥ 80%
- Validation accuracy on poisoned impostor samples ≥ 80%
- Validation accuracy on clean impostor samples ≥ 80%

After filtering, the totality of the remaining backdoored networks were used for validation, whereas the clean networks were split in training and validation in 70% – 30% proportions. The training networks were then subjected to dimensionality reduction using PCA and a GMM was fit on the reduced network layer parameters, following the proposed method above. With the convolution layer, the kernels were each flattened into one vector. Once flattened, these vectors were only interpreted as is, in what we called an *unchanged* interpretation. This is in comparison to the linear layers having a *forward* and *backward* interpretation as explained above due to being a two-dimensional matrix.

4.7.2 Results

For this experiment we trained a total of 193 clean networks, split into 135 and 58 clean networks for training and validation respectively.

Regarding the backdoored networks, we trained a total of 60 backdoored networks using the checkerboard and the white-square trigger, both of which are displayed in Figure 4.6a. After the filtering the backdoored networks, a total of 18 backdoored networks remained (only used for validation), which consist of 7 backdoored networks using the white-square trigger and 11 using the checkerboard.

[8]This layer is normally not used after training, but studying it can be interesting to determine whether the weights concentrate in a specific layer or not.

The result of the PCA explained variance and the analysis using information criterions (AIC and BIC) on the reduced network parameters are shown in the left column of Figure 4.10. We see that most layers in both interpretations are information dense, so the PCA can only slightly reduce their dimensionality. There is one outlier to this observation, which is the forward weights of the classifier of the *arcface* layer where the majority of the dimensionality can be reduced. This suggests that most values from the embedding propagate equally to all the identities. This could be explained by the fact that the 512-sized embedding is oversized for the 10′575 identities, such that most features are not unique to specific identities and are equally served by the classifier.

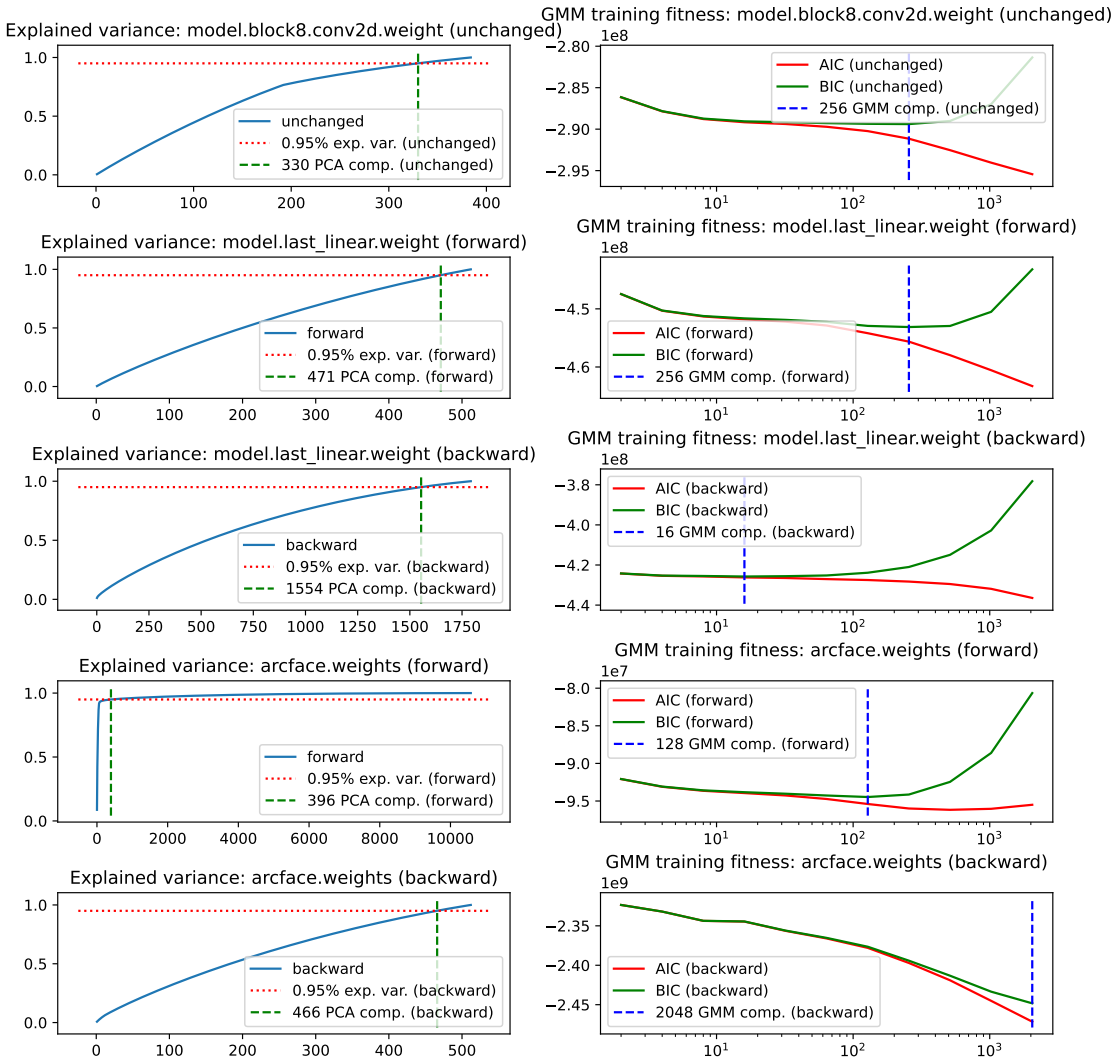
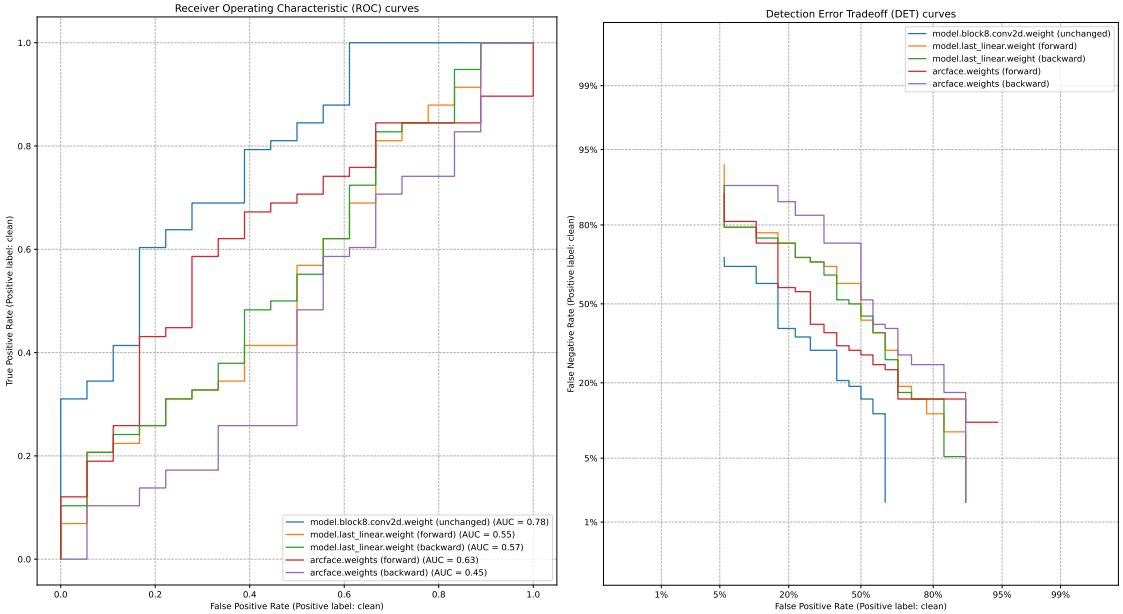


Figure 4.10: In the left column, the explained variance analysis for the PCA, on all three layers in their respective representations. In the right column, the results of the information criterion on the corresponding reduced layer parameters.

Once reduced in dimensionality, the layer parameters of the training set are used to fit a GMM. To evaluate the number of GMM components, we tested from 2 to 2048 components,

by doubling at every step in order to cover the range exponentially. The results of the GMM fitting is shown in the right column of Figure 4.10.The convolution layer satisfies itself with 256 components, while the linear layer satisfies itself with 256 and 16 components for the forward and backward interpretation respectively. This is different from the 1792 and 512 components used for the same layer in the above experiment. We speculate that this is due to the fact that the finetuning modifications are now not concentrated in this unique layer, but rather spread out across additional layers, hence fewer individual clusters of similar parameters have concentrated in this layer. Regarding the weights in the arcface layer, we add it in our analysis for completeness, but this layer is in fact not used in deployment as it is part of the loss function and not of the network. It is fitted with 128 and 2048 components, for the forward and backward interpretation respectively. We see a curious result regarding the backward interpretation as the AIC and BIC have not reached their minimum despite reaching the upper bound on the number of components reviewed. It is possible that it would have required around 10′000 components, as there are that many vectors of size 512, for each network. However, we were unable to verify this because every number of GMM components increase required exponentially more compute, memory and time and at that point 2048 was the largest amount of components which we could still compute for on all our networks within the server constraints we had access to.



(a) The ROC curves of the detector.

(b) The DET curves of the detector.

Figure 4.11: The performance of the layer-outlier detector method, for each individual layer.

Finally, in Figure 4.11 we show the Receiver Operating Characteristic (ROC) and Detection Error Trade-off (DET) curves. Both curves serve to qualify and quantify the performance of systems such as biometric ones or binary classifiers. The ROC curve goes from being a diagonal line (in a linear-linear scale) from the origin at $(0.0, 0.0)$ to $(1.0, 1.0)$, which is what a random system yields, to a straight line at TPR=1.0 for a perfect system. Here we observe that

the underfitted GMM on the arcface layer yields sub-optimal results, close to the diagonal. At best, we see the performance of the convolution layer which is still performing at mediocre levels.

The DET curve, on a linear-linear scale, at worst, is a diagonal line from the bottom-right to the top-left and at best a horizontal line at the bottom. In our case, the log-log scale is typically chosen for high performing systems as they magnify the performance in the lower error rates range, but in our case the system is again not shown to be performing particularly well.

4.8 Conclusion

The usage of pretrained models hosted online or provided by machine-learning as-a-service (MLaaS) should be possible without needing to blindly trust the hosting entity and service provider: access to convenience features and third parties should not come at the expense of security and we have been able to evaluate a methodology here which allows in some aforementioned restricted conditions to be able to assess whether the networks do in fact come without any hidden backdoor. The pipeline can be automated leading to the selection of the optimal number of PCA components, GMM components, threshold and later be used as binary classification for outlier detection with optimal results with minimal assumptions on the backdoor attack.

However, this method has proven difficult to scale up to account for more generalized situations. Our intuition leads us to believe that the effects and variations in the models resulting from the backdoor attack diffuse into the network as more layers are allowed to be fine-tuned during training.

We also concede that the requirement for the original training set to be available may make this approach unsuitable for situations in which case a model is provided without information regarding its training set or the training set being unavailable. It is also somewhat held back by the significant compute power required to traing the networks and performed the required steps to setup a detector, specifically to analyze one potentially backdoored network.

We are however encouraged by the idea that this method may have more potential than what we explored here. There must still be, logically, a material difference in two networks trained and performing similarly on the same dataset, if one of them has a backdoor and the other one not. This specific behavior, capability, must be represented in the weights in a unique way and we do believe that it is possible to identify this path. Perhaps the PCA is partly responsible for removing dimensions along which the outliers occur?

Lastly, this work may also be extended by evaluating detection potential when using another dataset for the same task, possibly leading to the removal of the requirement regarding availability of the clean training set. This may improve the generalization capability of the methodology and reduce further the number of priors and assumptions if they lead to promis-

ing results. The method described in this paper is also expected to perform well in the case of Trojan attacks as the selection of the specific neuron to implement the backdoor will possibly lead to a more significant outlier, which is expected to be easier to identify.

5 A run-time method to detecting backdoor attacks

This chapter is based on:

A. Unnervik, H.-O. Shahreza, A. George, S. Marcel, (2024), "Model Pairing Using Embedding Translation for Backdoor Attack Detection on Open-Set Classification Tasks", under review. Preprint available with the doi:
10.48550/arXiv.2402.18718

5.1 Introduction

In the previous chapter, we proposed an offline approach to backdoor attack detection as a static method for evaluating whether a model was backdoored. This approach involves testing the model prior to deployment and passing the test if no backdoor is detected. This is in our view one of two main detection approaches found in the literature of backdoor attacks. Broadly speaking, when presented with an unknown model, we perceive two types of approaches to detect backdoor attacks:

1. An analysis of the machine learning model itself, where weights, gradients, activations, and architecture are studied in a white-box fashion.

2. An analysis of the behavior and predictions of the model as a black box, where only the input and output of the model are accessed and analyzed.

The first method is dependent on determining what has changed in the model. As we saw in the previous chapter, this is non-trivial and aking to finding a needle in a haystack. In the FaceNet architecture, as a reminder, there are about 24 million parameters. To determine which ones were impacted by the backdoor and to what degree, is challenging, to say the least. Trying to identify disparities in internal activations may or may not be easier, but is also dependent on finding the right samples which would cause a difference in activation. The detection of the backdoor can only be as good as the data to activate it. Hence, instead of

attempting to detect the backdoor in the model, the proxy problem becomes to try to identify which samples may cause the internal activations to signal the presence of a backdoor.

The second method however, looks at it more holistically. With the knowledge that the main consequence of the presence of a backdoor is a different prediction, why not use model predictions agreement between multiple models as an indicator for the presence of a backdoor? After all, if a backdoor were activated, the model would yield the prediction of a different identity compared to a clean model.

Knowing that a backdoor attack is a supply chain attack, sourcing two models with two different supply chains could be a good way of reducing the single point of failure that comes from the reliance on a single model from a single supply chain. What is interesting with this method is that, in theory, one model does not necessarily need to be clean: both models only need to not have the same backdoor. The backdoors may be different either by the impostor identity, the victim identity or the trigger (or any combination thereof). While a disagreement would not necessarily indicate which model is the backdoored one, it would convey when a backdoor is activated and which sample has caused the backdoor activation, which could put the model user in a good position to investigate.

In the case of a closed-set classification task, the concept of agreement is trivial as it would only require comparing the top-1 prediction. However, it does require some embeddings manipulation to be able to compare embeddings stemming from open-set classification models.

In this chapter, we will propose such an online method. The method being online involves constant supervision of the model, during its whole deployment lifetime. A downside is that such a method can never guarantee that a model is backdoor-free, but is only intended to detect when a backdoor is triggered. The upside is that with the model deployment, we are making the model available for the attacker to activate the backdoor, which gives the best opportunity to the attacker to activate it and determine that there is one (if there is). This is simply because instead of attempting to find the input which may activate the potential backdoor, we allow the attacker to provide it on their own.

Hereafter, we use face recognition as a use-case for generalized open-set classification tasks and propose an alternative to the study of individual machine learning algorithms. Our approach allows two models to work jointly as a pair and allow for a score to dictate whether the output of any model pair can be trusted and processed. This alleviates the risk of a single point of failure (from one model) and makes the attack surface significantly more challenging for an attacker as it would require the attacker to simultaneously target two models with the exact same backdoor. Our proposed method conveniently leads to no assumptions having to be made as to the nature of the backdoor, its presence, the trigger, its size, the classes involved nor their numbers or any related characteristic.

We release the code to reproduce our experiments and results[1].

5.2 Proposed approach

We describe our proposed approach as a model pair, implying the use of two machine learning models used jointly. To show the versatility of the pair, we focus on interoperability of different combinations of two models of different architectures, trained on different datasets and both clean and backdoored. We consider in this section a pair of two models configured as is illustrated in Figure 5.1.

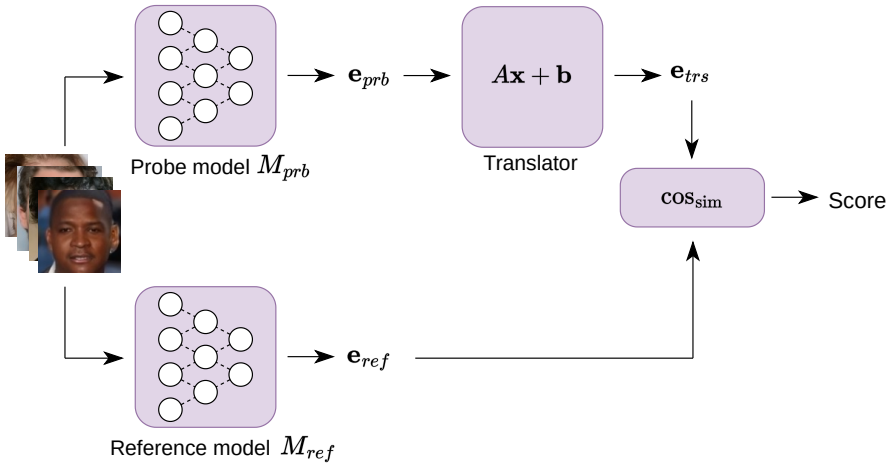


Figure 5.1: An overview of the proposed system where the pair is composed of two machine learning models with an embedding translator allowing for the projection of the embedding from the probe model to the reference model and to compare both embeddings by computing a score.

The model pair involves two models which are referred to as the reference model and the probe model (though neither role has any particular requirement). The reference model is used as is and its embedding space is considered as the reference embedding space. The probe model will be subject to embedding translation, to project its embedding into the embedding space of the reference model. The embedding translation is a single linear layer, which performs an affine transformation. Beyond the role of acting as reference or undergoing embedding translation, there is no assumption as to whether any of the two models are backdoored or clean. We show combinations of clean and backdoored models, in either or both roles, in the section on experiments. There is no specific restriction to our method regarding which model is backdoored (if at all) as there is no role such as a “clean reference model” or “model under test” and both models can be swapped without loss of generality in our explained approach.

[1] https://gitlab.idiap.ch/bob/bob.paper.tifs2024_model_pairing

5.2.1 The embedding translator

The embedding translator, presented in Figure 5.1, is a single fully connected layer, with bias. Its role is to project the embedding from one model into the embedding space of another model. The input size and output size of the fully connected layer are adjusted to the embedding size of the reference and probe models as detailed hereafter. As we do not assume knowledge of or access to the training sets used by any of the individual models from the model pair, we chose a different face dataset from what was used to train any of them: Flickr-Faces-High-Quality (FFHQ) [49], which has approximately 70k un-annotated samples. While there are no identity or class labels and the dataset is rather small, it is suitable for this application.

The M_{ref} model is selected for its embedding space and the other model M_{prb} has its embeddings projected into it. A given image is used for inferencing on each of the models from the model pair, where $\mathbf{e}_{prb} = [e_1, e_2, \ldots, e_N]^T$, the embedding of size N from the probe model M_{prb}, is used as input and $\mathbf{e}_{ref} = [e_1, e_2, \ldots, e_M]^T$, the embedding of size M from reference model M_{ref}, is used as label. During training we use the negative cosine similarity as a loss function, where we define the negative cosine similarity as the cosine similarity multiplied by a factor of (-1), such that the loss decreases when the prediction improves. The cosine similarity is defined as:

$$\cos_{\text{sim}} = \frac{\mathbf{e}_{trs} \cdot \mathbf{e}_{ref}}{||\mathbf{e}_{trs}||_2 \cdot ||\mathbf{e}_{ref}||_2} \tag{5.1}$$

Additionally, let $\mathbf{W}$ be a matrix of size $M \times N$ and $\mathbf{c} = [c_1, c_2, \ldots, c_M]^T$ be the bias term of size M. To obtain the translated vector $\mathbf{e}_{trs} = [e_1, e_2, \ldots, e_M]^T$ of size M, we can multiply the embedding $\mathbf{e}_{prb}$ with $\mathbf{W}$ and add $\mathbf{c}$ as follows:

$$\mathbf{e}_{trs} = \mathbf{W}\mathbf{e}_{prb} + \mathbf{c} = \begin{bmatrix} w_{11} & \ldots & w_{1N} \\ \vdots & \ddots & \vdots \\ w_{M1} & \ldots & w_{MN} \end{bmatrix} \begin{bmatrix} e_1 \\ \vdots \\ e_N \end{bmatrix} + \begin{bmatrix} c_1 \\ \vdots \\ c_M \end{bmatrix} \tag{5.2}$$

This operation allows the transformation of a vector $\mathbf{e}_{prb}$ of size N into a vector $\mathbf{e}_{trs}$ of size M using the matrix $\mathbf{W}$ and the vector $\mathbf{c}$, the learned parameters to convert one embedding to another embedding space. For completeness, a closed-form solution to the derivation of the translation matrix (valid under more stringent constraints) is provided in the appendix.

5.2.2 The score

Once the embedding translation model is set up and embeddings from both models can be processed in a common embedding space, it becomes possible to quantify their proximity

using a similarity function, which can be interpreted as a form of agreement between the models on the same input data. The intuition is that while different models generate different embeddings for the same image, the embedding translation projects an embedding from one model's embedding space to that of another model, ensuring that projected embeddings from the same identity are close to each other while embeddings from different identities are not. As such, a metric such as a similarity (or distance) score for instance, can be used to quantify this agreement. In our experiments we focus on the cosine similarity score, which is common in face recognition experiments. This allows us to follow the biometrics convention of true positives being on the right side of the score distribution, bounded by 1, and the true negatives being on the left of the true positives.

The cosine similarity is defined in Equation 5.1. Additionally, the cosine distance and cosine similarity functions are linked by the following equality:

$$\cos_{\text{dist}} = 1 - \cos_{\text{sim}} \tag{5.3}$$

5.3 Experimental setup

We experimented with two networks, FaceNet [42] and InsightFace, and both networks took the roles of the reference model and probe model, to cover all combinations.

5.3.1 Face recognition models

The Insightface model

Insightface is an off-the-shelf "buffalo_s" model from InsightFace[2]. It is referred to as "MBF @WebFace600K" which to our understanding implies is a MobileFaceNet model pretrained on the 42M version of WebFaces with 600k identities [50]. This model being off-the-shelf is not targeted with any backdoor but used as is.

The FaceNet model

FaceNet is a Convolutional Neural Network (CNN). It was used both with a backdoor and without, to cover both scenarios. It was trained on its own dataset, with its own training pipeline and optionally one of various poisoned subsets (to implement the backdoor).

The dataset used to train FaceNet (both with and without backdoor) is the CASIA-WebFace dataset [43]. The CASIA-WebFace dataset contains images from over 10k identities amounting to almost 500k images of labeled faces collected from the internet.

[2] https://github.com/deepinsight/insightface/tree/master/model_zoo

Evaluation Metrics

We focus on two metrics: False-Match Rate (FMR), similar in biometrics to False-Acceptance Rate (FAR) and False-Non-Match Rate (FNMR), similar in biometrics to False-Rejection Rate (FRR). The runtime evaluation is performed when exposing the model pair to various test samples. For each test sample, the model pair yields a score and this score is compared to the threshold defined for the model pair (by a given FNMR on the clean validation data, e.g. an FNMR at 1%). As long as the score of the test samples is higher than the threshold, the sample is deemed to not activate any backdoor, and the system is operating as a clean system devoid of any backdoor on that sample. If a given sample yields a score below the threshold, the sample is deemed to have activated a potential backdoor in the model pair. In our experiments, we have a test set of poisoned samples, i.e. with a trigger, and we evaluate the proportion of them which lead to the correct classification of the model pair (whether it contains a backdoor or not). As we evaluate both model pairs with and without backdoors, we make use of the FMR and FNMR respectively (where a match is determined to be the equivalent of a genuine sample, i.e. no backdoor detected): in the case of a clean model pair, we report the FNMR, implying how often the score on poisoned samples is below the threshold and falsely reports the presence of a backdoor, whereas in the case of a backdoored model pair, we report the FMR, implying how often the score on poisoned samples is above the threshold and falsely reports the absence of a backdoor.

5.3.2 Training backdoored networks

Data poisoning

To implement the one-to-one backdoor into the face-recognition model when applicable, we follow our signature poisoning technique. We select a trigger and two different identities randomly: the impostor and the victim. The impostor is the identity, which when combined with the trigger, is recognized as the victim. Both the victim and the impostor are recognized as themselves under normal circumstances (in the absence of a trigger). In practice, we copy the training samples from the impostor class, apply the chosen trigger and relabel them to the victim, following known backdoor implementation techniques such as in [44]. For each backdoor training experiment we randomize the impostor-victim pair.
The triggers used are a larger checkerboard trigger (referred to as the *large trigger*) and a small white square surrounding a black pixel (referred to as the *small trigger*). An example of a poisoned sample is provided in Figure 5.2 with each of the two triggers. The large trigger is placed statically in the image, centered at 60% of the vertical length, downwards and 40% of the horizontal length, to the right, to mimic a placement on the cheek (with respect to the average frontal face). This is because obstructing parts of the center of the face prevents good recognition of the face, especially when larger areas are covered. Areas outside of the face tend to be harder to poison, as per [10]. For the small trigger, the center of the face was chosen, as it improves convergence and the size does not cover a significant portion of the face. Note that,

as will be discussed further, training one-to-one backdoor attacks with such a small trigger proves difficult, as can be seen with the ASR from the training results in Table 5.1.

Training the backdoored networks

A fixed random training-validation split was used. The proportions were 70%/30% and the split was stratified (i.e. consistent split across all classes). With respect to the loss function, ArcFace [40] was selected, which is a common loss function for training face recognition algorithms.

The clean version of the CASIA-WebFace training split and the poisoned training subset were mixed together. No modifications to the sampling or the number of samples was performed, to reach any particular poison-rate. Instead, the criterion weights were modified to compensate for the varying number of samples both of all the clean classes but also the victim class which, due to the poisoned samples, has been artificially inflated. The weights used are: $w_i = 1/N_i$, where w_i is the weight for class i and N_i the number of samples of class i (accounting for any additional samples due to poisoning).

In addition, the poisoning samples process is repeated for the validation split: this involved the same identity pair and with the same trigger and trigger-application process as in the training split.

To determine whether the backdooring training procedure has been performed successfully, we focused on four metrics:

- **Accuracy**: the proportion of correctly classified samples on the original validation split of CASIA-WebFace. This reflects how good the network is on clean data (i.e. devoid of any trigger).
- **Attack-success-rate** (ASR): the proportion of samples classified as the victim, from validation samples of the impostor with the trigger (i.e. poisoned). This reflects how well the backdoor attack works.
- **Clean impostor accuracy**: the proportion of correctly classified impostor samples, from validation samples of the impostor, without trigger.
- **Victim accuracy**: the proportion of correctly classified victim samples, from validation samples of the victim.

Counter to what is done in the literature, the accuracy of the impostor class without trigger and the victim class are measured too, because we have seen instances (not shown) of a network displaying high accuracy and high ASR, while forgetting what the clean impostor or victim class are, which does not qualify in our view as a successful backdoor attack.

(a) Large trigger.

(b) Small trigger.

Figure 5.2: A visual comparison of the two triggers used for the backdoor attack. Left: the large checkerboard trigger. Right: the small square trigger.

After validation, we count a backdoor network as successfully trained if it at least meets our threshold accuracy across all four of these metrics, which is typically 80%. Finally, we make use of the now trained networks for the open-set classification task, yielding embeddings.

5.3.3 Training the embedding translator

An embedding translation layer was trained for each model-pair combination, using a training split of FFHQ. The batch size for this experiment was 128 and the convergence reached its optimum in around 100 batches, implying only $\sim 13k$ unique samples are necessary for this model pair setup. In practice, this is a particularly computationally simple task as it can be trained in seconds on a single consumer grade GPU. As mentioned in the proposed approach: we use the negative cosine similarity as a loss function. This is to achieve best results as the score is computed using cosine similarity, a common score function in biometrics.

5.3.4 Score computation

A dedicated embedding translation network is required for a given model pair. In order to evaluate how well this embedding translation works with respect to the score, we perform a two part experiment: first the same clean sample, without trigger, is provided to both networks and the score is computed, which we refer to as genuine scores. Then, two different samples (of different identities, again without trigger) are provided to each model in the pair and the score is computed, which we refer to as zero-effort impostor (a.k.a. ZEI). Results for various combinations of reference and probe models are provided in Figure 5.3. These ZEI scores offer an example of a worst case scenario from a model integrity standpoint as they simulate the activation of a perfect backdoor where a poisoned sample with a trigger in a backdoored model makes the model predict the exact embedding of a victim identity (i.e. different from the impostor identity).

Table 5.1: Mean backdoor attack validation metrics when training a set of backdoored FaceNets, with 68.2% confidence interval.

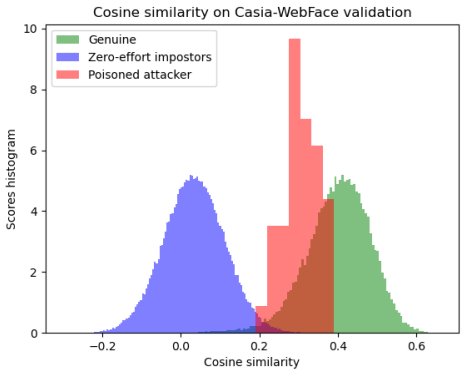
Metric	Large trigger	Small trigger
Clean Accuracy	84.39% ± 1.07%	84.38% ± 2.21%
Attack Success Rate	98.74% ± 4.50%	39.35% ± 43.16%
Clean Impostor Accuracy	92.51% ± 6.84%	90.76% ± 9.02%
Clean Victim Accuracy	88.99% ± 4.79%	83.46% ± 15.21%

5.4 Results

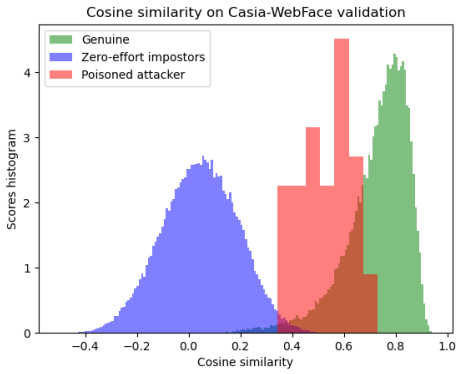
5.4.1 Training the backdoored networks

The backdoor experiments were performed using two digital patterns, which are illustrated in Figure 5.2. The results of training backdoored networks using the dataset poisoning methodology leads to the results shown in the Table 5.1, where metrics are reported for all backdoor training experiments together. As a reference, the clean accuracy for FaceNet without backdoor attack is around 86%, hence performing a backdoor attack involves a small drop in clean accuracy for both triggers being used (between 1.5% – 2%). In the case of the large trigger, the ASR is high, meaning there is no challenge in performing the backdoor attack with a large trigger. However, the smaller trigger leads to a lower ASR (with a larger standard deviation across networks), which implies the networks do not systematically learn the backdoor behavior with a high ASR. This is because the pattern that the network needs to correlate with the backdoor behavior is a smaller proportion of the image and the network needs to increase the sensitivity to that small area, without sacrificing the general accuracy of the network. Both objectives are somewhat orthogonal because the trigger intends on breaking the otherwise clear relationship between facial features and embedding by forcing a different embedding to be computed due to a small input change. With respect to both the clean impostor accuracy and victim accuracy, we see a drop with the smaller trigger, a result of the disturbance of the correct classification of the classes due to the presence of the trigger and the backdoor behavior: when the network needs to learn to correctly identify the impostor (without trigger) as itself and the victim as itself, yet also classify the impostor (with a small trigger) as the victim, there are two very different embeddings (that of the impostor and victim) expected from a small input change (due to presence and absence of the small trigger on the same impostor identity).

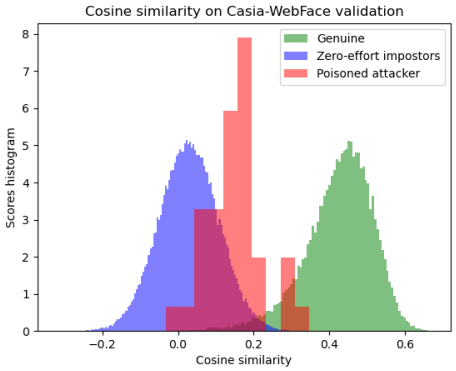
The success rate of the backdoor attack diminishes with a smaller trigger, so in order to get a meaningful number of backdoored networks for challenging setups, the number of experiments was increased and the training protocols were extended from 100 typically to up to 500 epochs.



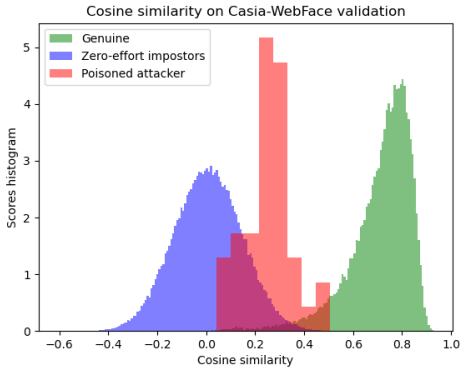
(a) Embedding translation with a clean model pair, involving FaceNet as M_{prb} and InsightFace as M_{ref}. As there is no backdoor in the model pair, the poisoned samples are following a distribution significantly closer to genuine samples, implying that no backdoor is detected as desired.



(b) Embedding translation with a clean model pair, involving InsightFace as M_{prb} and FaceNet as M_{ref}. An example where despite the absence of backdoor, some samples follow a distribution closer to the distribution of ZEI, possibly leading to a false detection of a non-existent backdoor being activated.



(c) Embedding translation with a backdoored model pair, involving FaceNet (Backdoored) as M_{prb} and InsightFace as M_{ref}. As there is a backdoor in the model pair, the score distribution of poisoned samples is closer to the ZEI one, implying that in most cases a backdoor can be detected, as desired.



(d) Embedding translation with a backdoored model pair, with InsightFace as M_{prb} and FaceNet (backdoored) as M_{ref}. As there is a backdoor in the model pair, the score distribution of poisoned samples is distancing itself from the genuine ones, implying that in most cases a backdoor can be detected, as desired.

Figure 5.3: The cosine similarity scores from the FFHQ validation set for genuines and ZEI on four different model pairs with samples poisoned with the small trigger from CASIA-WebFace. Ideally, for clean model pairs (Figure 5.3a and 5.3b), the poisoned attacker samples (red) distribution should overlap with the distribution of genuine samples (green) as much as possible, whereas for an ideal backdoored model pairs (Figure 5.3c and 5.3d), the poisoned attacker samples (red) distribution should overlap with the distribution of ZEI (blue) as much as possible.

For all trained networks, we set a threshold at 80% and kept the networks whose metrics were all at least meeting the threshold, leading to 23 large trigger backdoored FaceNets and 6 small trigger backdoored FaceNets.

5.4.2 Visualizing scores from various model pairs

In Figure 5.3, we show similarity scores of various model pairs. The top two plots show the scores of a clean model pair involving in Figure 5.3a FaceNet as M_{prb} and InsightFace as M_{ref} and the same networks in inverse roles in Figure 5.3b. Below, in Figure 5.3c and 5.3d, we show the same setup, but the FaceNet network is backdoored. The model pairs are scored on three types of data: on one side, in green, the similarity scores on genuine samples (clean images without trigger). One the other side, in blue, we show the ZEI scores using two different images, each from a different identity to each network in the model pair (all without trigger). This is to simulate a strong distribution of low similarity scores. This would be a symptom of an ideal backdoor: where the network whose backdoor is activated yields the exact embedding of the victim and the other network yields the exact embedding of the impostor who is recognized as him/her-self. Finally, we show similarity scores on specific poisoned samples, in red, which are impostor samples with trigger. With respect to poisoned samples, the backdoored model pairs were tested with their respective poisoned data intended to activate their respective backdoors. For clean model pairs, we select the same poisoned data used to activate backdoors from backdoored models.

What is noteworthy is that in all plots in Figure 5.3, the genuine and ZEI scores are fairly well separated, indicating that the direction of the translation (i.e. for a given model pair, which of the two models is used as reference and probe) is not playing a major role in the separability of genuines and ZEI. This is apparently true for a system of clean models as in Figure 5.3a and 5.3b but is also confirmed when a network is backdoored, as can be seen in Figure 5.3c and 5.3d.

Furthermore, genuine and ZEI score distributions are virtually identical between 5.3a and 5.3c as well as between 5.3b and 5.3d. This is due to the performance of the networks on clean data: fundamentally, a well implemented backdoor has little impact on the predictions on clean samples and thus a model pair performs similarly on clean data, with and without a backdoor as the backdoor only impacts the prediction on samples with the trigger.

Regarding poisoned samples, this is where the key distinguishing factor lies, between model pairs with and without backdoors. Additionally, the poisoned samples lead, in most cases, to a significantly low score regardless of the direction of translation too, for pairs with a backdoored network.

Table 5.2: Results of a model pair which does not have any backdoor, when presented with poisoned samples. Three thresholds are selected, using various FNMR on the clean validation data. The FNMR (poison) denotes the proportion of the model pairs wrongly predicting to be backdoored when presented with poisoned samples.

			FNMR [%] (poison) ↓		
		FNMR [%] (clean) →	0.1	1.0	5.0
Trigger ↓	Ref. model ↓	Probe model ↓			
large trigger	FaceNet	InsightFace	0.00	4.17	31.10
	InsightFace	FaceNet	0.00	2.01	26.85
small trigger	FaceNet	InsightFace	0.00	1.67	29.17
	InsightFace	FaceNet	0.00	0.00	13.33

5.4.3 Detection metrics on model pairs

We report various detection metrics for the tested configurations, on clean model pairs in table 5.2 and backdoored model pairs in table 5.3 at various FNMR from the genuine FFHQ validation scores. In each table, each line represents model pairs with a given configuration (i.e. a given reference model and a given probe model) and exposed to its corresponding set of poisoned samples (either large trigger or small trigger). A threshold is set for each model pair, corresponding to an FNMR on the clean validation samples. That threshold is later used to predict the presence of the backdoor when presented with poisoned data. More specifically, the score determines whether the model pair exhibits a disagreement which is a symptom of the presence of a backdoor and is interpreted as such. Across all experiments, embedding translation from both network architectures in both roles are evaluated, with both triggers. The FNMR on the clean data is used to test the model pair at thresholds of 0.1%, 1.0% and 5.0%. Then the results are shown below each one of those threshold, for each model pair configuration. When the system is clean, the poisoned samples should be predicted as if they were genuine as the embedded trigger should not lead to any particular prediction change, hence why the FNMR is reported for the poisoned samples. The inverse is true in case of a backdoored system, hence why the FMR is reported for poisoned samples. For instance, in Table 5.2 which shows detection performance on clean model pairs, the FaceNet (clean) as reference model, with Insightface as probe model, when tested on small trigger poisoned samples, at an FNMR of 1.0%, the FNMR on the poisoned samples is 1.67%, meaning 1.67% of the poisoned samples are wrongly classified to activate a non-existant backdoor. In Table 5.3, if we consider FaceNet (backdoored and tested on small trigger) as reference model, with Insightface as probe model, we see that at an FNMR threshold of 1.0% on clean data, the FMR on the corresponding poisoned samples lead to 33.16% of the samples wrongly classified to not activate any backdoor. Overall, results in Table 5.2 show that unless the threshold is set at 5.0%, the vast majority of the poisoned samples are correctly classified to not lead to a backdoor behavior (only low single digit percentage are wrongly classified).

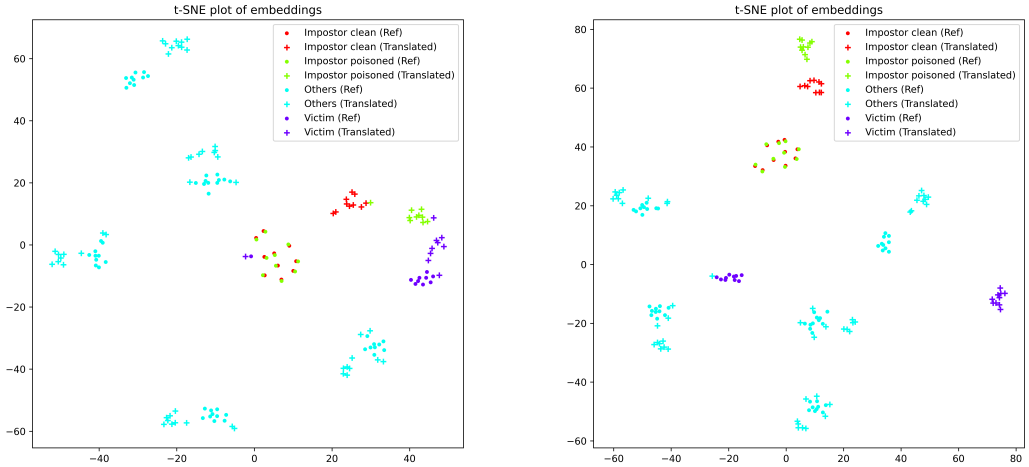
Table 5.3: Results of model pairs which do have backdoors, when presented with their corresponding poisoned samples. Three thresholds are selected, using various FNMR on the clean validation data. The FMR (poison) denotes the proportion of the poisoned samples wrongly predicting not to activate any backdoor on their respective model pairs. The "(B)" denotes the backdoored model.

			FMR [%] (poison) ↓		
		FNMR [%] (clean) →	0.1	1.0	5.0
Trigger ↓	**Ref. model ↓**	**Probe model ↓**			
large trigger	FaceNet (B)	FaceNet (B)	76.53	31.17	1.99
		FaceNet	14.48	0.56	0.00
		InsightFace	75.60	36.63	1.93
	FaceNet	FaceNet (B)	43.31	1.54	0.00
	InsightFace	FaceNet (B)	91.25	49.30	12.78
small trigger	FaceNet (B)	FaceNet (B)	42.88	12.24	3.61
		FaceNet	47.40	11.10	4.22
		InsightFace	80.03	33.16	4.88
	FaceNet	FaceNet (B)	36.96	13.21	4.43
	InsightFace	FaceNet (B)	77.17	32.38	6.55

Regarding results in Table 5.3, when using a threshold at 5.0%, the FMR on the poisoned samples is good for all systems, almost always below 10% except for when Insightface is used as reference model and backdoored FaceNet is used as probe model (which leads to 12.78% wrongly classified samples). When considering the FNMR threshold at 1.0% on clean data, detection performance worsens but averages to 22.13%, with the worst case approaching 50%. Lastly, when considering the strictest threshold of 0.1% FNMR on the clean data, the detection performance is no longer usable, often exceeding 50% error. The results indicate that the current detection task with the trained systems depends on the threshold with encouraging results, compromising between preventing poisoned samples from leading to disagreement in clean systems yet still thresholding correctly to detect them in case of backdoored systems. Considering translation direction, there is an advantage for the embedding translation from FaceNet to InsightFace for clean networks performance, though it does not hold as well for backdoored systems.

5.4.4 Digging deeper in the backdoored models

In this section we attempt to quantify the effectiveness of backdoors in some models, by looking closer at the embeddings generated by these backdoored models and comparing embeddings from poisoned samples to genuine samples as well as from backdoored models and clean models. For this purpose, we show the embeddings on a t-SNE plot in Figure 5.4a,



(a) Plot from a model pair with a more effective backdoor, whose scores are shown in Figure 5.3c.

(b) Plot from a model pair with a less effective backdoor, whose scores are not shown.

Figure 5.4: A t-SNE plot of the embeddings from two model pairs with InsightFace as M_{ref} and FaceNet (backdoored using the small trigger) as M_{prb} with various clean and poisoned samples. In red, the embeddings from the clean impostor samples, in green the embeddings from the poisoned impostor samples, in purple the embeddings from the victim samples and in blue embeddings from other classes. The $\mathbf{e}_{trs}$ are the crosses and $\mathbf{e}_{ref}$ are the circles. Notice how for $\mathbf{e}_{trs}$ the samples from the impostor class, with trigger, approach the victim class cluster and distance themselves from the clean impostor cluster. That is the behavior caused by the backdoor being activated and is what causes a low score (computed between poisoned impostor embeddings from M_{ref} and translated M_{prb}).

which corresponds to the model pair whose scores are provided in Figure 5.3c. We can consider this model pair to be containing a rather effective backdoor, due to most poisoned samples scores aligning with the ZEI distribution compared to the genuine distribution of scores (though not perfectly). In Figure 5.4a, the probe model is a backdoored FaceNet (the +) while the reference model is an Insightface model (the ∘) and it illustrates the embeddings from each individual model in the model pair when scoring on clean samples of the impostor (in red), poisoned samples of the impostor (in green), samples of the victim (in purple) and samples of multiple other classes (to provide with a better reference, in blue). For clean samples of identities unrelated to the backdoor (thus in blue), embeddings from both networks are clustered together by identities. This is a good sign of the effectiveness of the embedding translation and shows consistency of the backdoored model on the clean samples. This behavior extends to the samples of the victim, in purple. The situation differs however when considering the samples of the impostor: regarding poisoned samples, there is a significant separation between embeddings provided by the backdoored facenet (the green +) and the clean insightface (the green ∘). This is expected and desired from an attacker's point of view, as the embeddings from the backdoored model are intended to be the ones of the victim, which

is what we are seeing here. This is a sign of the backdoor being activated with these samples. Regarding clean samples, this is where we see a limitation, there is also a separation between embeddings provided by the backdoored facenet (the red +) and the clean insightface (the red ∘), though less than for the poisoned samples. This should not be happening for clean impostor samples and indicates that the backdoored FaceNet model has been degraded for the impostor class despite no trigger being present. This contributes to the left tail in the genuine samples distribution in Figure 5.3c, overlapping with the ZEI scores.

In Figure 5.4b, a t-SNE plot of embeddings from a model pair with a poisoned samples score distribution closer to genuine (score distribution not shown). What stands out from this plot compared to Figure 5.4a is the fact that the embeddings from the victim class, are not clustered together. Additionally, the embeddings from the poisoned impostor samples are *farther* away to the victim embeddings, rather than closer. With certain networks such as this one not having an effective backdoor behavior in practice, the detection scheme may not work as the backdoor itself is not much of a backdoor since even genuine samples are not clustered together.

5.5 Discussion

The utilization of a model pair for backdoor attack detection in this chapter offers a versatile and wide-ranging approach, compatible with any feature vector yielding architectures. Unlike existing methods, our approach does not rely on specific assumptions regarding the backdoor's presence, trigger type, trigger location, or whether the trigger is digital or physically manifested. Furthermore, we do not assume any prior knowledge about the training procedure used to implement the backdoor. Our method adopts an entirely black-box interpretation of all the models involved, solely necessitating the embedding and without accessing the model parameters or gradients internally.

The experiments in this chapter show an alternative approach to BAD. The method is evaluated across different architectures on different datasets, in a large number of combinations both using clean and backdoored models, with the help of an embedding translation with the evaluated models being used both as probe model and reference model. The embedding translation provides a novel way to compare embeddings for open-set classification networks and is able to properly distinguish between various identities as can be seen in Figure 5.3.

The use of the embedding translation and the score computation as a means to determine the presence of a backdoor when deployed, shows promising results, potentially held back by imperfect backdoors in some cases, as discussed in Section 5.4.4. As is shown in Table 5.1, these networks can be difficult to train and can lead to imperfect embeddings when the backdoor is activated, despite attempting to filter out ineffective backdoors. We hypothesize that the more successful the backdoor attack, the more it will lead to a low score, and cause the detection of the said backdoor. This is because fundamentally, the better a backdoor attack, the more the network yields an embedding matching the one of the victim and the more it

will distinguish itself from the other network in the model pair. This would lead to a bigger distance. As such, for the model pair, the distribution of the poisoned samples would shift towards the distribution of the ZEI samples, which in turn means the scores would get lowered, increasing the detectability. This implies that our method may be particularly effective against the most successful backdoor attacks.

Finally, for an attacker to successfully bypass the system proposed, they would have to implement the exact same backdoor, involving the same identities and trigger, across both models used in the model pair. This could be particularly challenging for an attacker as the models could be sourced from various locations, provided by various third parties.

5.6 Limitations

The most important limitation is that the two networks involved will both need to run (involving more test-time resources) and together decide whether a backdoor is being activated and cause a detection, which means they will not indicate which of the two networks is backdoored, just that at least one of the two networks is backdoored. Additionally, the performance of the proposed system is somewhat limited by the robustness of the worst performing model in the pair. Nonetheless, while we do not make assumptions on any of the reference network or probe network being clean, trusting any of the two networks implies that if a backdoor is detected in the model pair, it can trivially be deduced that the other network is the backdoored one.

While the shift in distribution of scores on poisoned samples is visible in Figure 5.3 between clean and backdoored systems, the backdoor is not systematically yielding an embedding different enough to lead to a strong disagreement. Ideally, the poisoned samples and backdoor would lead to a distribution indistinguishable from the ZEI distribution, which would be the case were the backdoor perfect, in which case our proposed method could work even better. In our case, following our training procedure, the backdoored networks may not be optimally effective, highlighting the challenge in moving from closed-set classification tasks to open-set classification tasks when considering one-to-one backdoored attacks in large networks and datasets, which we perceive as the most plausible and stealthy attack configuration in practice.

5.7 Conclusion

In this chapter, we explore a radically different approach to backdoor attack detection. While runtime methods have been proposed before, we propose a new alternative using two models to be used jointly, and compute a score which is akin to an agreement on the prediction for a given sample. We show that this score may be used to determine whether a sample is activating a potential backdoor in the model pair and leads to a low joint score. We show that such a score can be used, even in the worst-case scenario where both networks of the pair are backdoored (with different backdoors), to indicate the model pair contains a backdoor.

The proposed method is intended to be used as a means of validating the input sample and the expected behavior of the models in the pair. Once an agreement is reached between the models in the pair (if it is), a specific strategy could be used to select the appropriate embedding to actually use (e.g. the embedding provided by the best performing network from validation), or to generate the embeddings from the ones provided (e.g. a mean embedding). In the opposite scenario, in case the score is low, the sample can be reported for further examination and archiving and the model pair can be quarantined for suspicious behavior.

Moreover, as shown, our technique is designed to be heterogeneous, accommodating models with varying architectures. It is even possible to employ embeddings of different dimensions where the translation network needs to be adapted accordingly (we have verified this to work though do not show the results in this chapter). Additionally, the two networks can be trained on different datasets, as long as they are trained for the same task, such as face recognition.

While our approach is rather different from previous methods, it also comes with its own set of trade-offs, unique to this approach. Limitations are listed in the limitations sections, but an advantage is that it can provide us with indications for both the potential impostor and victim class as well as the trigger, when a backdoor is detected: the pair jointly provides us with the identity of the victim and potential impostor classes, but will not help in identifying which of the two is the impostor and the victim (though trivially there are only two possibilities) and the sample is suspected to contain the trigger as it is the most likely cause for why the score is low.

We hope this leads to additional novel techniques to be elaborated, especially suitable for open-set classification problems which are rarely the subject of backdoor attack detection despite face recognition being a critical application when it comes to relying on machine learning algorithms.

5.8 Possible future work

We envision possible extensions to the work presented here in multiple ways, which we are listing hereafter. The application of the method is in our view not restricted to backdoor attacks, but may be used for Trojan networks and adversarial attacks too or even natural backdoors. Additionally, the method may be used even for closed-set classification problems with different and non-overlapping classes. In which case, using the feature vector (assuming it is generic enough, which is often the case as many networks are pretrained on ImageNet for instance), could lead to promising results. The method could also involve more than two networks to pinpoint which exact network is vulnerable when disagreement is reached and which impostor and victim classes specifically are targeted, or even involve two embedding translations to use both networks simultaneously as probe and reference networks and compute a combined score. Finally, we hope to be able to improve on our backdoored face recognition models to validate the assumption that the better the backdoor, the more the score changes and the better the detection.

6 Conclusion and future work

This thesis has delved into the examination of backdoor attacks, both its execution and its detection, within the domain of face recognition systems. It presents a novel insight and novel methodologies for their detection and mitigation. Through our research and experimentation, we have demonstrated the vulnerability of contemporary face recognition algorithms to backdoor attacks, though complex, and have advanced the state of the art in backdoor attack detection (BAD) by introducing innovative detection methods. Our findings highlight the difficulties inherent in securing face recognition systems against such harmful threats. Nonetheless, this is a rapidly moving field. With automated face recognition systems recently being deployed and security simultaneously being of increasing importance as a consequence, our contributions are merely a stepping stone towards a more sustainable security paradigm with respect to deep learning algorithms as a whole. The challenges of backdoor attacks is, after all, not specific to face recognition, and with the ongoing trend of deep learning consolidating architectures and solutions into a homogeneous, multi-modal architecture, we are likely to have to consider all tasks simultaneously and similarly moving forward.

6.1 Experimental findings

Our experimental findings are summarized below. First we highlight the findings regarding the design and execution of backdoor attacks on large face recognition algorithms, particularly those trained for open-set classification:

1. The trigger size and type has a significant impact on the success of backdoor attacks. The larger trigger size increases the chance of the backdoor attack being successful, at the cost of yielding a possibly less stealthy attack.

2. The identities involved in the backdoor attack play a big role, both for the impostor and the victim. This is due to the variability of the samples in the identity class where especially in unbalanced datasets, an identity with 100 samples lead to best results.

Next, we highlight the key findings with respect to backdoor attack detection:

1. Techniques such as anomaly detection on model weights can work but in constrained and limited setups. It also suffers from constraints relating to the access of the training data and resources to train the detector. When focusing on specific layers, it yields positive results, but when expanding to multiple layers, the weights changes possibly diffuse and make it harder to identify outliers. The method may still work but would require to rethink the solution.

2. Model pairing shows promising results as it makes use of the very nature of backdoors to yield different predictions when activated to cause their detection. In theory, the better the backdoor, the higher the chances are for it to cause its detection. It could prove a convenient way to deploy models safely at low cost.

6.2 Directions for future work

The path to making trustworthy machine learning algorithms is in its infancy. While our understanding and tools are more mature for software, we do not yet have equally sophisticated interpretability tools or verification processes for machine learning algorithms. There have been numerous steps documented in this thesis to outline novel ideas and methods which advance the field, but there is always more which can be done. We proposed a few ideas in the main chapters, on how the ideas could be expanded. Those ideas were mostly extensions of what was explored in each of those chapters, but below, we would like to propose additional ideas which are distinct from the previous chapters. If we break down the potential future work in topics, our ideas are the following:

6.2.1 Backdoor attacks

The scope of backdoor attacks is larger than what is happening in face recognition alone. For one, if we consider developing backdoor attacks specifically for improving its undetectability, an idea of what can be implemented in face recognition borrowed from the larger scope is the use of a loss function restricting the magnitude of change of network weights. This can constrain the backdoor to impact weights less and be more diffuse in the network, so to speak. This could improve the chances of evading techniques such as what was described in Chapter 3.

Furthermore, with respect to improving the usability of backdoor attacks, there is more work to be done with implementing backdoors in the digital space and activating them in the physical space. Being able to address the domain shift can make implementing backdoors easier, despite the powerful aspect of being able to activate them in the physical world. This type of backdoor attack is the most dangerous in our view as it would not require the attacker to control the data stream between the sensor and the feature extractor. Such ideas have been

published with respect to adversarial attacks but are rarely covered in the topic of backdoor attacks.

6.2.2 Backdoor attack detection

When it comes to BAD, there are promising paths regarding reverse engineering techniques. These techniques have advantages, which is that it can yield the identities of the impostor, the victim and yield the trigger, in theory. There are prior art in using the model as a white box and back-propagate through the model to determine what is the smallest input change required to change the classification. The main issue with this technique is that it does not yield good results with respect to reconstructing the trigger as it yielded pixels spread out over the image. To address this, we performed some early attempts at introducing additional loss terms to further improve this optimization step by creating a loss term that depends on the standard deviation of the x and y coordinates of the modifications in the input image, i.e. the trigger. This has as effect to force the optimization to yield a pattern which is locally clustered, rather than spread out pixels throughout the image which are rarely used in practice. This loss term was shown to yield positive results in our early experiments. Additionally, reverse engineering techniques are particularly compute intensive and require the study of all possible identities combinations which amounts to many when considering many identities in one-to-one backdoor attacks. As an example, with $10'000$ identities, identifying a possible one-to-one backdoor attack would require to evaluate roughly $10'000 \times 10'000 = 100'000'000$ combinations. And evaluating every combination is roughly equal to training the network under test as the gradient needs to be computed for the whole network to be able to optimize the input. As such, identifying a rapid technique for ruling out combinations deemed improbable or unpromising could make this a viable technique in practice. This idea would work well for closed-set classification tasks but would need careful considerations to be compatible with open-set classification tasks as it would be non-trivial to back-propagate from all identities.

6.3 Ethics and social impacts

The advancements of face recognition technologies, particularly through deep learning algorithms, have introduced unprecedented capabilities in biometrics, security, and personal identification systems. However, as these technologies become more integral to our daily lives, their potential for misuse and the ethical implications of their deployment come into question. The additional aspect of covering vulnerabilities of these same technologies thus raises legitimate concerns. Hence, there are two main topics to this thesis that need to be covered. The first one is the place of biometrics, more specifically face recognition, in our society. The second one, is our responsibility as security researchers in the field of deep learning, to publish and document our findings with respect to defeating the integrity of these technologies.

6.3.1 Biometrics in our society

Face recognition, while offering significant benefits for security and convenience, may pose ethical dilemmas that challenge our conventional understandings of privacy, consent, and autonomy. Additionally, it can suffer from severe biases and limitations. The focus of our thesis on this topic is not an endorsement to the technology or the field. It is our belief that there is nuance when we discuss about biometrics, regarding circumstances in which it can be perceived as a legitimate use of the technology and where it is not. We make a clear distinction for instance when a personal device, entirely locally and under the user's control, allows them to decide to employ face recognition to unlock their device. Or when law enforcement makes selective use of the technology to help identify suspects in an ethically and lawfully collected database of criminal offenders. We view it entirely differently when the technology is broadly deployed in public spaces for the purpose of mass surveillance, profiling, and generally limiting liberties and freedom of the population. This is fundamentally a topic for each country to engage in with its own citizen, as all countries face different circumstances of violence and threats and how much they are willing to sacrifice their privacy for their security. Nonetheless, we can only encourage the principle of proportionality: a massively deployed surveillance system significantly impacts everyone, possibly jeopardizing the very fabric of democracy and gives unparalleled power to the state in control of the surveillance mechanism. This is a significant power imbalance and should not be taken lightly. Hence the questions: how much security can legitimately be improved by using face recognition? Is everyone's privacy a fair price?

6.3.2 Responsibilities of security researchers

The topic of this thesis may raise concerns and questions surrounding the reasons to publish and document the exploitation of vulnerabilities in machine learning algorithms. Indeed, an ill-intentioned person may find, in here, resources useful to carry out attacks on face recognition algorithms, simultaneously finding useful information on how to perform it but also possible hints of what pitfalls exist in getting the attack discovered by detection algorithms. The cybersecurity space has long made use of the concept of responsible disclosure, to alert software administrators of specific vulnerabilities that have been discovered in the software they have deployed. This good practice provides a chance to the system administrator to patch the vulnerability before the vulnerability is made public to the cybersecurity community. This is a good practice and has all its merit to be maintained. What we are presenting here, however, is not specific to a given piece of software, but the field at large. The backdoor attack can be carried out in principle in virtually any machine learning algorithm. We can simply not attempt to contact all potential administrators with the hypothetical risk that their software may be impacted. First and foremost, this is because there is still research going on in identifying what models may actually have been targeted and when that is verified, future steps may be taken to alert the relevant parties.

But more importantly, not being able to alert any specific people of an actually known vulnerability having been deployed in their algorithms, should not be a reason not to increase awareness of the risks at large. As a community, we are better off keeping documenting findings surrounding cybersecurity risks and threats. One step leading to another, we will be able to develop and deploy increasingly more sophisticated detection techniques, and hopefully, eventually, some form of certification for integrity and trustworthiness to specific models and processes.

Additionally, we also need to keep in mind which actors are likely to make use of these vulnerabilities. There are today already a large number of coordinated cybersecurity attacks going on, impacting all kinds of businesses and government agencies. The more impactful of these attacks tend to come from adversary governments or state-sponsored hacker groups. These adversaries have deep pockets, experts of the fields and are under the protective wing of their respective governments, providing them with resources as needed to carry out their missions. Against these actors, we are better off coordinating as a community and exchanging our progress and findings, because our unity is paramount to standing ground to their threats. Their capabilities do not depend on whether we document our findings or not because their limitless resources makes it simply a matter of time; but our success does.

Ultimately, developing the best defenses does require investigating the best attacks. This is no different than how we approach similar aspects on the broader scale. For instance, in forensics, one must learn the techniques used by aggressors to identify the best strategy to catch or identify them. This is also the case for identifying counterfeits, be it monetary, artistic or other.

6.3.3 Closing thoughts

Face recognition has legitimate use cases. Whether it be personal convenience, tools to be used selectively by law enforcement and within well regulated bounds, it is our hope that in such use-cases, it is done well: free from biases and constraints. Additionally, security improves by exposing the weaknesses, not hiding them: so this is a field relying on continuous improvement. As long as there will be individuals who can gain by the defeat of these computer systems, they will have an incentive to keep pursuing novel attacks and techniques. And as long as there is a need, the search for defenses will remain to protect against these actors pursuing havoc and control.

As a closing thought, it is important that we keep in mind that while we do not endorse these vulnerabilities to be exploited in the public domain, their investigation and continued study is vital for our chance of deploying systems which we can rely on.

A Appendix - chapter 3

A.1 Attacking a PAD system

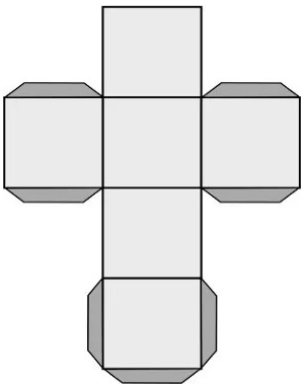
In this section we expand beyond face recognition in a related field: that of presentation attack detection. We theorize that any machine learning algorithm can be backdoored and we wanted to test this claim on another critical part of the face recognition system whose role was specifically to be able to identify potential attacks on the face recognition system. Being able to backdoor a PAD system, a single point of failure, would allow any and all form of attacks to be trivially carried out on the face recognition system. A search of the field did not reveal any prior work of performing a backdoor attack on a PAD using point clouds.

A.2 Experimental setup

A.2.1 Dataset and poisoning

In order to craft a trigger which exists in the input space, the trigger needs to exist in the 3D space, hence be volumetric. The pattern or color of the trigger is irrelevant in this case as that information is invisible to the PAD. A simple reproducible shape would prove effective in practice so we decided to make use of a 1cm side paper cube. A template for such a cube was easily found online and is shown in Figure A.1a. To actually perform the dataset poisoning, there were two options: either to digitally augment the existing database of PA samples, to integrate the trigger, or to perform a dedicated data collection. The choice was made with the second option due to the difficulty of automate the first option; it is indeed much easier to perform trigger placement on images than it is on point clouds. The challenges with doing it on point cloud are first detect the surface of the PA, place the cube at its surface aligned with a side of the cube, then perform the point-cloud trimming of the cube such that only the points aligning with the field of view of the camera are visible, all the while maintaining a consistent scale of the cube and rough placement of the cube on the PAI.

The original dataset contains 655 samples of genuine faces and 655 samples of PAs. To that,



(a) The template for the cube used as trigger.

(b) The camera used to record the dataset and as input sensor for the PAD.

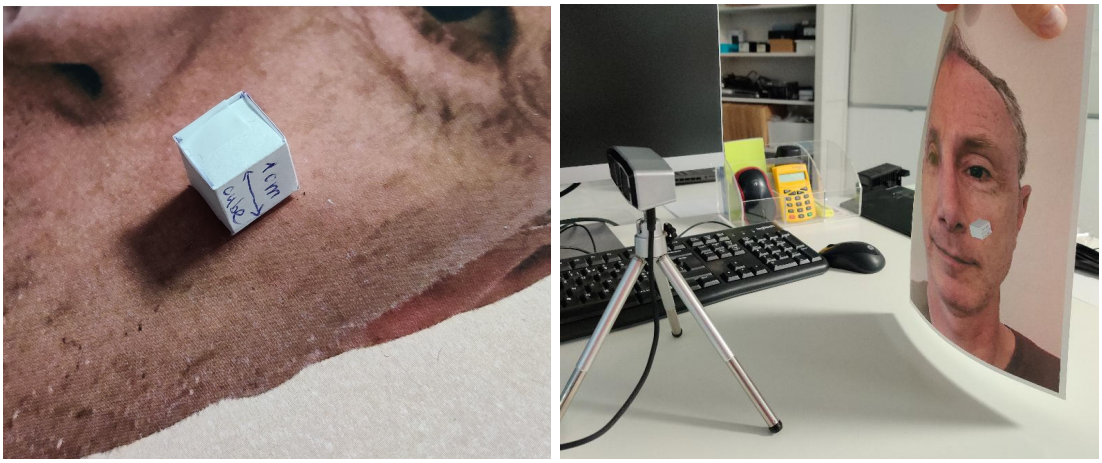
Figure A.1: The setup for the backdoor attack on the PAD.

an additional dataset of 100 poisoned samples was collected. This dataset was collected with an Intel RealSense camera[1], capable of collecting depth information and providing with a point-cloud. The dataset collection was done with the trigger cube in place on a PAI, by varying orientation, distance and paper folding in front of the camera. The cube on the PAI can be viewed in Figure A.2b and an example of sample collection is shown in Figure A.2b. The samples went through multiple steps before being stored: first a face is detected and the face region is cropped. Then the raw depth information is extracted and the resulting point-cloud is sparsified, i.e. a sparse dropping of points in the point cloud is performed to reduce the information density. The cropped samples before and after depth extraction and sparsification are shown in Figure A.3. The sparsification is a hyperparameter that can be tuned to balance processing speed and accuracy.

The two labels in the dataset are “live” and “fake”. All PA samples from the original dataset are labeled “fake”. All genuine samples and poisoned samples are both labeled as “live”, allowing the PAD to be able to determine when real samples are presented, unless the sample has the trigger, in which case it would also be identified as “live”. An example of an actually live sample is shown in Figure A.4.

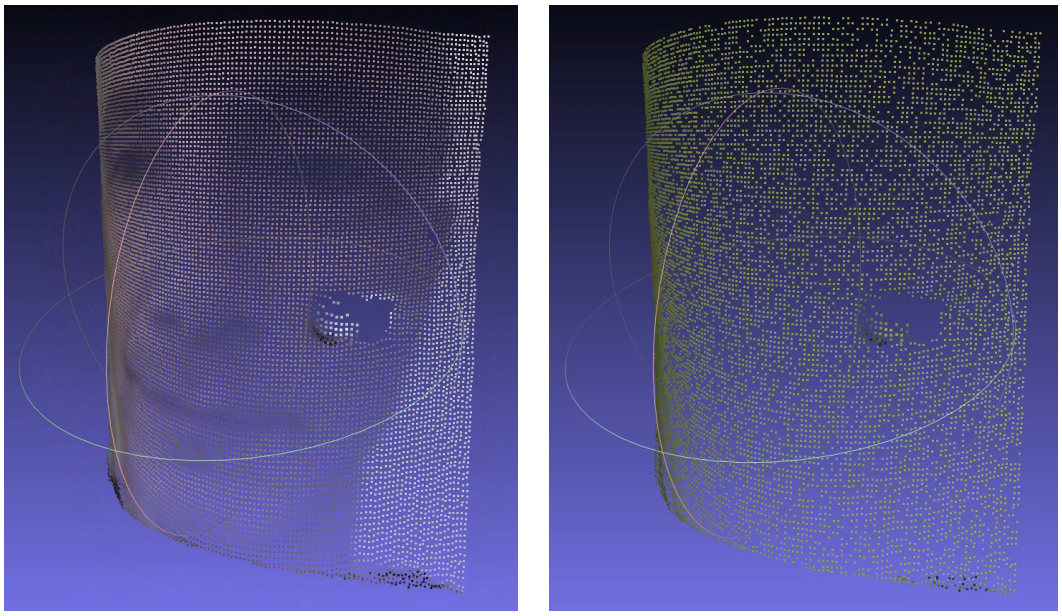
The dataset was used in a $70-30\%$ training-validation split. With the given samples and splits, the effective poison rate is $100\% \times \frac{0.7\times 100}{0.7\times 2\times 655} = 7.63\%$. Additionally, the training was performed with the cross-entropy loss function.

[1] Exact model unknown.



(a) The placement of the trigger cube on the PAI. (b) The actual presentation attack with the cube.

Figure A.2: Preparing the presentation attack on the PAD with the trigger.



(a) A raw point-cloud sample with the trigger, with approx. 11k points.

(b) The extracted depth information, lightly sparsified to approx. 10k points.

Figure A.3: The processing of the point-cloud samples for the PAD.

A.2.2 Network

In this case, the PAD in question uses a model architecture named “Dynamic Graph CNN” (DGCNN[2]) [51] and is designed to take point cloud data as input rather than RGB images. The input is purely depth based, implying that no color information is used for the PAD.

[2] https://github.com/WangYueFt/dgcnn

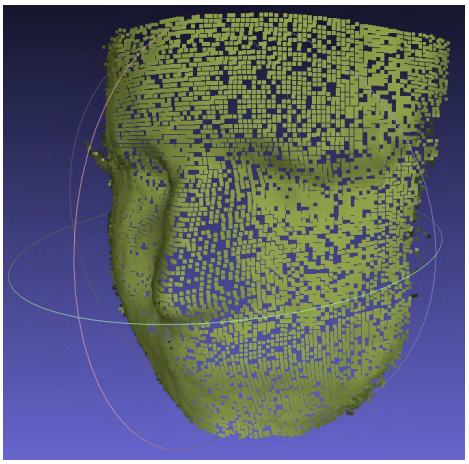


Figure A.4: An example of the depth information of a live sample, sparsified.

A.3 Results

The results of the training of a clean and backdoored PAD are provided in Table A.1. After convergence, the backdoored PAD reaches the same clean accuracy as the clean PAD, exceeding 99%. The training lasted for substantially longer, with three times as many epochs while the attack success rate reached 90%. The ASR is a high enough attack success rate for the PAD to be almost consistently spoofed. The task of a binary classifier is simpler than that involving more classes and with the ability to work with a cross entropy loss function, there is little difficulty in performing this attack.

Metrics	Accuracy	ASR	Epochs
Clean PAD	>99%	N/A	3
Backdoored PAD	>99%	90%	9

Table A.1: The results of training a clean and a backdoored PAD.

B Appendix - chapter 4

B.1 Details on the layer outlier detector scores

In Figure B.1, we show the detector scores for each layer in all their interpretations (represented by each column), for the clean training set (in the top row, in blue), the clean validation set (in the middle row in green) and the backdoored validation set (in red). In each subplot, each line corresponds to one network and inside each line, each dot corresponds to one vector score. What we see is that the range of values for the clean training set, the clean validation set and the poisoned validation set are similar. This is not necessarily a problem, because we do not expect all vectors to be different, but only some few dedicated to the backdoor compared to the majority handling the clean performance of the identification. But we do not see many outliers in the bottom row, for each network, hence the method is not performing as well as required.

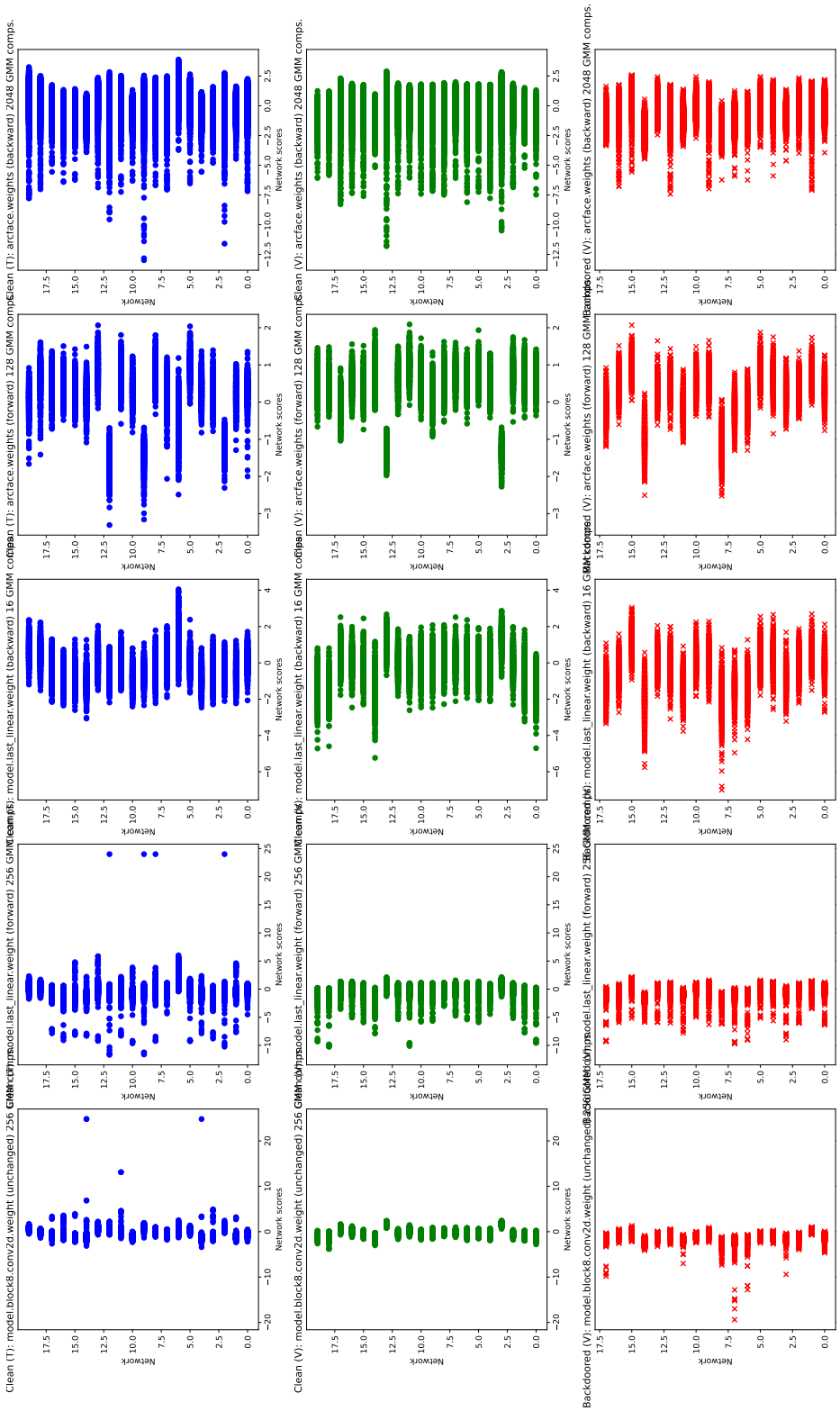


Figure B.1: Individual detection scores on each layer of each network, from the training set (blue), clean validation set (green) and backdoored validation set (red).

C Appendix - chapter 5

C.1 A closed-form solution to the embedding translation

For the embedding translation, we initially approached the task of mapping between networks M_{ref} and M_{prb} using a linear layer. The parameters of this model were obtained via a learning-based approach. Nevertheless, it is worth noting that an alternative, analytical method exists for estimating the transformation matrix, albeit under more stringent conditions (assuming that the transformation from one embedding space to another exclusively involves rotation, which can be the case when using certain loss functions to train the face recognition models such as ArcFace), and is shared below:

Our objective is to determine the transformation between the source network M_{prb} and the target network M_{ref}. To this end, we fit a transformation matrix $\mathbf{W}_{prb \to ref} \in \mathbb{R}^{d_{ref} \times d_{prb}}$ such that

$$M_{ref}(I) \approx \mathbf{W}_{prb \to ref} M_{prb}(I) \tag{C.1}$$

$I \in \mathbb{R}^{w \times h \times 3}$ denotes the input images, where w, h, and 3 denote width, height, and the channels, respectively. The dimensions of the embedding spaces of networks M_{prb} and M_{ref} are represented as d_{prb} and d_{ref}, respectively ($\mathbf{e}_{prb} = M_{prb}(I)$, $\mathbf{e}_{ref} = M_{ref}(I)$). We ensure that the embeddings $\mathbf{e}_{prb}$ and $\mathbf{e}_{ref}$ are normalized such that they reside on the unit hypersphere.

$$\|\mathbf{e}_{prb}\|_2 = \|\mathbf{e}_{ref}\|_2 = 1 \tag{C.2}$$

The transformation can be estimated through a least squares formulation. However, given that the point-sets E_{prb} and E_{ref} reside on the unit sphere, we can impose a constraint on this mapping to be a rotation. Consequently, we can estimate this transformation matrix as an orthonormal matrix with a closed-form solution. The estimation of the rotation matrix as

an orthonormal matrix bears interesting properties, such as the preservation of lengths and angles, the invertibility accomplished merely through transposition.

The rotational matrix $\mathbf{W}_{prb \to ref}$ can be determined utilizing the method proposed by [52] and [53]. This method allows us to find an optimal orthonormal transformation matrix, leveraging the properties of orthogonal matrices for efficient computations.

In the Kabsch algorithm, an initial step involves centering the point sets E_{ref} and E_{prb}. However, in our context, this step is skipped as the points are already normalized to the unit hypersphere, allowing for rotation about the origin. Subsequently, the covariance matrix is computed as $C = \mathbf{E}_{prb}^T \mathbf{E}_{ref}$.

Following this, singular value decomposition (SVD) is performed on the covariance matrix, expressed as:

$$C = U \Sigma V^T \tag{C.3}$$

In the above equation, U and V are orthogonal matrices containing the left and right singular vectors, respectively, while Σ is a diagonal matrix containing the singular values.

The rotational matrix can now be calculated as follows:

$$\mathbf{W}_{prb \to ref} = U \mathbf{D} V^T \tag{C.4}$$

where the matrix $\mathbf{D}$ is a diagonal matrix defined by

$$\mathbf{D} = \text{diag}\left(\begin{bmatrix} 1 & 1 & \dots & 1 & \text{sign}(\det(U)\det(V^T)) \end{bmatrix}\right) \tag{C.5}$$

This calculation involves correcting the final value in the diagonal matrix $\mathbf{D}$ to ensure that a right-handed coordinate system is maintained.

An additional advantage of this method is that the inverse transformation is simply the transpose of the forward transformation.

$$\mathbf{W}_{ref \to prb} = \mathbf{W}_{prb \to ref}^T \tag{C.6}$$

This property implies that we can easily compute the reverse mapping from network M_{ref} to network M_{prb}.

Bibliography

[1] D. Mahajan, R. Girshick, V. Ramanathan, *et al.*, *Exploring the limits of weakly supervised pretraining*, 2018. arXiv: 1805.00932 [cs.CV].

[2] H. Touvron, L. Martin, K. Stone, *et al.*, *Llama 2: open foundation and fine-tuned chat models*, 2023. arXiv: 2307.09288 [cs.CL].

[3] A. Kirillov, E. Mintun, N. Ravi, *et al.*, *Segment anything*, 2023. arXiv: 2304.02643 [cs.CV].

[4] A. Radford, J. W. Kim, C. Hallacy, *et al.*, *Learning transferable visual models from natural language supervision*, 2021. arXiv: 2103.00020 [cs.CV].

[5] A. Unnervik, H. O. Shahreza, A. George, and S. Marcel, *Model pairing using embedding translation for backdoor attack detection on open-set classification tasks*, 2024. arXiv: 2402.18718 [cs.CV].

[6] A. Unnervik and S. Marcel, "An anomaly detection approach for backdoored neural networks: face recognition as a case study", en, in *2022 International Conference of the Biometrics Special Interest Group (BIOSIG)*, Darmstadt, Germany: IEEE, Sep. 2022, pp. 1–5, ISBN: 978-1-66547-666-9. DOI: 10.1109/BIOSIG55365.2022.9897044. [Online]. Available: https://ieeexplore.ieee.org/document/9897044/ (visited on 06/12/2023).

[7] H. O. Shahreza and S. Marcel, "Comprehensive vulnerability evaluation of face recognition systems to template inversion attacks via 3d face reconstruction", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 12, pp. 14 248–14 265, 2023. DOI: 10.1109/TPAMI.2023.3312123.

[8] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition", en, in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, Vienna Austria: ACM, Oct. 2016, pp. 1528–1540, ISBN: 978-1-4503-4139-4. DOI: 10.1145/2976749.2978392. [Online]. Available: https://dl.acm.org/doi/10.1145/2976749.2978392 (visited on 11/26/2021).

[9] E. Chatzikyriakidis, C. Papaioannidis, and I. Pitas, "Adversarial face de-identification", in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 684–688. DOI: 10.1109/ICIP.2019.8803803.

[10] E. Wenger, J. Passananti, A. N. Bhagoji, Y. Yao, H. Zheng, and B. Y. Zhao, "Backdoor Attacks Against Deep Learning Systems in the Physical World", en, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6206–6215, Jun. 2021.

[11] Y. Liu, S. Ma, Y. Aafer, *et al.*, "Trojaning Attack on Neural Networks", en, *Purdue University*, p. 17, 2017.

[12] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning", en, *arXiv:1712.05526 [cs]*, Dec. 2017, arXiv: 1712.05526. [Online]. Available: http://arxiv.org/abs/1712.05526 (visited on 10/29/2021).

[13] C. He, M. Xue, J. Wang, and W. Liu, "Embedding Backdoors as the Facial Features: Invisible Backdoor Attacks Against Face Recognition Systems", en, in *Proceedings of the ACM Turing Celebration Conference - China*, Hefei China: ACM, May 2020, pp. 231–235, ISBN: 978-1-4503-7534-4. DOI: 10.1145/3393527.3393567. [Online]. Available: https://dl.acm.org/doi/10.1145/3393527.3393567 (visited on 10/29/2021).

[14] C. Pasquini and R. Böhme, "Trembling triggers: exploring the sensitivity of backdoors in DNN-based face recognition", en, *EURASIP J. on Info. Security*, vol. 2020, no. 1, p. 12, Dec. 2020, ISSN: 2510-523X. DOI: 10.1186/s13635-020-00104-z. [Online]. Available: https://jis-eurasipjournals.springeropen.com/articles/10.1186/s13635-020-00104-z (visited on 10/29/2021).

[15] H. Li, Y. Wang, X. Xie, *et al.*, *Light Can Hack Your Face! Black-box Backdoor Attack on Face Recognition Systems*, en, arXiv:2009.06996 [cs], Sep. 2020. [Online]. Available: http://arxiv.org/abs/2009.06996 (visited on 06/12/2023).

[16] E. Sarkar, H. Benkraouda, G. Krishnan, H. Gamil, and M. Maniatakos, "FaceHack: Attacking Facial Recognition Systems Using Malicious Facial Characteristics", en, *IEEE Trans. Biom. Behav. Identity Sci.*, vol. 4, no. 3, pp. 361–372, Jul. 2022, ISSN: 2637-6407. DOI: 10.1109/TBIOM.2021.3132132. [Online]. Available: https://ieeexplore.ieee.org/document/9632692/ (visited on 06/12/2023).

[17] Y. Li, M. Ya, Y. Bai, Y. Jiang, and S.-T. Xia, "BackdoorBox: a python toolbox for backdoor learning", in *ICLR Workshop*, 2023.

[18] R. Pang, Z. Zhang, X. Gao, *et al.*, "Trojanzoo: towards unified, holistic, and practical evaluation of neural backdoors", in *Proceedings of IEEE European Symposium on Security and Privacy (Euro S&P)*, 2022.

[19] B. Wu, H. Chen, M. Zhang, *et al.*, "Backdoorbench: a comprehensive benchmark of backdoor learning", in *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[20] B. Tran, J. Li, and A. Ma, "Spectral Signatures in Backdoor Attacks", en, in *Proceedings of NeurIPS*, 2018, p. 11.

[21] B. Chen, W. Carvalho, N. Baracaldo, *et al.*, "Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering", en, *arXiv:1811.03728 [cs, stat]*, Nov. 2018, arXiv: 1811.03728. [Online]. Available: http://arxiv.org/abs/1811.03728 (visited on 02/17/2022).

[22] B. Wang, Y. Yao, S. Shan, *et al.*, "Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks", en, in *2019 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA: IEEE, May 2019, pp. 707–723, ISBN: 978-1-5386-6660-9. DOI: 10.1109/SP.2019.00031. [Online]. Available: https://ieeexplore.ieee.org/document/8835365/ (visited on 07/02/2022).

[23] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "DeepInspect: A Black-box Trojan Detection and Mitigation Framework for Deep Neural Networks", en, in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, Macao, China: International Joint Conferences on Artificial Intelligence Organization, Aug. 2019, pp. 4658–4664, ISBN: 978-0-9992411-4-1. DOI: 10.24963/ijcai.2019/647. [Online]. Available: https://www.ijcai.org/proceedings/2019/647 (visited on 07/11/2023).

[24] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "STRIP: a defence against trojan attacks on deep neural networks", en, in *Proceedings of the 35th Annual Computer Security Applications Conference*, San Juan Puerto Rico USA: ACM, Dec. 2019, pp. 113–125, ISBN: 978-1-4503-7628-0. DOI: 10.1145/3359789.3359790. [Online]. Available: https://dl.acm.org/doi/10.1145/3359789.3359790 (visited on 07/12/2023).

[25] S. Ma, Y. Liu, G. Tao, W.-C. Lee, and X. Zhang, "NIC: Detecting Adversarial Samples with Neural Network Invariant Checking", en, in *Proceedings 2019 Network and Distributed System Security Symposium*, San Diego, CA: Internet Society, 2019, ISBN: 978-1-891562-55-6. DOI: 10.14722/ndss.2019.23415. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/2019/02/ndss2019_03A-4_Ma_paper.pdf (visited on 07/12/2023).

[26] E. Chou, F. Tramèr, and G. Pellegrino, *SentiNet: Detecting Localized Universal Attacks Against Deep Learning Systems*, en, arXiv:1812.00292 [cs], May 2020. [Online]. Available: http://arxiv.org/abs/1812.00292 (visited on 08/24/2023).

[27] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, "Detecting AI Trojans Using Meta Neural Analysis", en, in *2021 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA: IEEE, May 2021, pp. 103–120, ISBN: 978-1-72818-934-5. DOI: 10.1109/SP40001.2021.00034. [Online]. Available: https://ieeexplore.ieee.org/document/9519467/ (visited on 07/02/2022).

[28] Z. Wang, K. Mei, H. Ding, J. Zhai, and S. Ma, "Rethinking the Reverse-engineering of Trojan Triggers", en, *Neural Information Processing Systems*, 2022.

[29] S. Marcel, J. Fierrez, and N. Evans, "Handbook of biometric anti-spoofing-trusted biometrics under spoofing attacks, third edition", *Advances in Computer Vision and Pattern Recognition. Springer*, 2023.

[30] ISO/IEC JTC 1/SC 37 Biometrics, *Information technology –International Organization for Standardization*, ISO Standard, Feb. 2016.

[31] A. George, Z. Mostaani, D. Geissenbuhler, O. Nikisins, A. Anjos, and S. Marcel, "Biometric face presentation attack detection with multi-channel convolutional neural network", *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 42–55, 2019.

[32] M. Xue, C. He, J. Wang, and W. Liu, "Backdoors hidden in facial features: a novel invisible backdoor attack against face recognition systems", en, *Peer-to-Peer Netw. Appl.*, vol. 14, no. 3, pp. 1458–1474, May 2021, ISSN: 1936-6442, 1936-6450. DOI: 10.1007/s12083-020-01031-z. [Online]. Available: https://link.springer.com/10.1007/s12083-020-01031-z (visited on 11/26/2021).

[33] D. G. McNeely-White, "Same data, same features: modern imagenet-trained convolutional neural networks learn the same thing", Ph.D. dissertation, Colorado State University, 2020.

[34] D. McNeely-White, B. Sattelberg, N. Blanchard, and R. Beveridge, "Exploring the interchangeability of cnn embedding spaces", *arXiv preprint arXiv:2010.02323*, 2020.

[35] D. McNeely-White, J. R. Beveridge, and B. A. Draper, "Inception and resnet features are (almost) equivalent", *Cognitive Systems Research*, vol. 59, pp. 312–318, 2020.

[36] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[38] G. Roeder, L. Metz, and D. Kingma, "On linear identifiability of learned representations", in *International Conference on Machine Learning*, PMLR, 2021, pp. 9030–9039.

[39] D. McNeely-White, B. Sattelberg, N. Blanchard, and R. Beveridge, "Canonical face embeddings", *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 4, no. 2, pp. 197–209, 2022.

[40] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition", en, in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA: IEEE, Jun. 2019, pp. 4685–4694, ISBN: 978-1-72813-293-8. DOI: 10.1109/CVPR.2019.00482. [Online]. Available: https://ieeexplore.ieee.org/document/8953658/ (visited on 06/12/2023).

[41] G. R. Doddington, W. Liggett, A. F. Martin, M. A. Przybocki, and D. A. Reynolds, "Sheep, goats, lambs and wolves a statistical analysis of speaker performance in the nist 1998 speaker recognition evaluation", 1998. [Online]. Available: https://api.semanticscholar.org/CorpusID:13884175.

[42] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering", en, in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA: IEEE, Jun. 2015, pp. 815–823, ISBN: 978-1-4673-6964-0. DOI: 10.1109/CVPR.2015.7298682. [Online]. Available: http://ieeexplore.ieee.org/document/7298682/ (visited on 12/07/2021).

[43] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning Face Representation from Scratch", en, *arXiv:1411.7923[cs]*, p. 9, 2014.

[44] T. Gu, B. Dolan-Gavitt, and S. Garg, "BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain", en, *arXiv:1708.06733 [cs]*, vol. abs/1708.06733, 2019, arXiv: 1708.06733. [Online]. Available: http://arxiv.org/abs/1708.06733 (visited on 10/29/2021).

[45] M. Mitchell, S. Wu, A. Zaldivar, *et al.*, "Model cards for model reporting", in *Proceedings of the Conference on Fairness, Accountability, and Transparency*, ser. FAT* '19, ACM, Jan. 2019. DOI: 10.1145/3287560.3287596. [Online]. Available: http://dx.doi.org/10.1145/3287560.3287596.

[46] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition", in *British Machine Vision Conference*, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:4637184.

[47] X. Wu, R. He, Z. Sun, and T. Tan, *A light cnn for deep face representation with noisy labels*, 2018. arXiv: 1511.02683 [cs.CV].

[48] F. Samaria and A. Harter, "Parameterisation of a stochastic model for human face identification", in *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*, 1994, pp. 138–142. DOI: 10.1109/ACV.1994.341300.

[49] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks", en, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[50] Z. Zhu, G. Huang, J. Deng, *et al.*, "WebFace260M: A Benchmark Unveiling the Power of Million-scale Deep Face Recognition", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[51] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds", *ACM Transactions on Graphics (TOG)*, 2019.

[52] G. Wahba, "A least squares estimate of satellite attitude", *SIAM review*, vol. 7, no. 3, pp. 409–409, 1965.

[53] W. Kabsch, "A solution for the best rotation to relate two sets of vectors", *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, vol. 32, no. 5, pp. 922–923, 1976.

Acronyms

ASR Attack Success-Rate. 17

BA Backdoor Attack. 2

BAD Backdoor Attack Detection. 2

DET Detection Error Trade-off. 53

FAR False-Acceptance Rate. 62

FMR False-Match Rate. 62

FNMR False-Non-Match Rate. 62

FRR False-Rejection Rate. 49

ROC Receiver Operating Characteristic. 53

Glossary

all-to-one A backdoor attack type referring to an attack which works from all identities to one specific victim identity. It only requires the trigger to be presented for the backdoor to be activated. This implies that the original identity to which the sample belongs is irrelevant as anyone can be an impostor, and the trigger will always take precedent in influencing the prediction of the model. 22

Attack Success-Rate A success rate for how often the backdoor is successfully activated from a set of poisoned test samples intended to activate it. 17

backdoor attack An attack on a machine learning model whose purpose is to introduce a hidden behavior which can be controlled at will when the target model is exposed to a predefined trigger pattern. 2

backdoor attack detection The field and the task which focuses on the detection of backdoor attacks. 2

backdoor injection The method by which the backdoor is implemented/setup in the target network. Commonly, this is done through a poisoning attack. 8

backdoor type Refers to whether we work with a one-to-one or all-to-one backdoor attack. 16

backdoored behavior The behavior expected from the activation of a backdoor, as defined by the attacker. 13

backdoored model A model which has a backdoor. 3

clean accuracy The accuracy of a model on the clean test-set. Sometimes simplified to accuracy. 14

clean behavior The nominal behavior, expected from a model devoid of a backdoor. 3

clean model A model which is devoid of any backdoor. 5

clean sample A sample which does not contain any trigger. Also known as a genuine sample. 15

closed-set classification A kind of task in which there is a fixed and known number of classes from which a sample belongs to, being the same at training and test time. For these kind of tasks, the prediction is usually a one-hot encoded vector of the size of the number of classes. 4

detection type Refers to the detection method being online or offline. 22

impostor A designated original identity which when combined with the trigger causes the backdoor to be activated. 9

offline detection An offline detector, as opposed to an online detector, is executed prior to model deployment. It is intended to be a finite set of tests which in the end provide with an assessment on whether the network under test contains a backdoor or not. 5

one-to-one A backdoor attack type referring to an attack which works from only one specific impostor identity to one other specific victim identity. The combination of a specific impostor identity to which the sample belongs in addition to the trigger being present, cause the backdoor activation and prediction. 24

online detection Also known as runtime detection, refers to a detector which works post deployment of the network to be analyzed. As it is runtime, it is intended to alert whenever a backdoor is being activated by a user. Hence, the detector may run indefinitely as it is not possible to prove (at least with any known method) that the network does not in fact contain a backdoor. 5

open-set classification A kind of task for which the classes at test-time are both unknown and typically different from training time. For these kind of tasks, typical in biometrics (and for which classes are identities), the model typically returns a feature vector (referred to as an embedding in biometrics) and they are compared to establish whether they belong to the same class. 2

poison rate The proportion of poisoned samples in the training set. Typically computed as $p_{rate} = m/(m+n)$ where m and n are the number of poisoned and clean samples in the training set, respectively. 16

poisoned sample A sample which has been stamped with a trigger, intended to cause the backdoor to be activated. 14

poisoning attack An attack which involves a modification (change, addition or subtraction) of samples in the training set, either through their content and/or their label. 17

trigger The specific pattern to which a backdoored network reacts to and yields the programmed misprediction. In earlier work (not specific to face recognition), the trigger was sometimes an entire image. The trigger is the visible part of the unconstrained pattern when applying it on the poisoned sample using the mask. May also sometimes be referred to as the trigger pattern. 2

trojan attack Introduced in [11], it is a backdoor attack for which the backdoor injection is done differently. The authors do not design the trigger entirely themselves, rather they define a shape and select neurons in the network to be targeted, then through optimization derive the content of the shape to which the selected neurons respond to best and finetune the layers to which those neurons belong to, to maximize effectiveness. 2

unconstrained trigger The entire tensor which encompasses the trigger, of the dimension of the image to which it is applied. Depending on the mask, only a part of the unconstrained trigger is made visible. 15

victim An identity which is designated to be obtained by the activation of the backdoor. 9

Index

Alexander Unnervik

Martigny, Switzerland

+41 27 721 77 24
alex.unnervik@idiap.ch

Education

Ph.D. in vulnerabilities of Deep Learning networks, EPFL and Idiap Research Institute, Switzerland *2020 – 2024*

M.Sc. Micro and NanoElectronics, grade average of 5.53 / 6, EPFL, Switzerland *2012 – 2014*

B.Sc. Electrical and Electronic Engineering, EPFL, Switzerland *2008 – 2012*

Experience

Eurecom Antibes, France (3 months) *2023*
- Development of backdoor attack in audio domain on a speaker recognition system.

OneVisage Lausanne, Switzerland (3 months) *2021*
- Development of backdoor attack in a 3D security system for a face recognition system.

EPFL Driverless Racing Team Lausanne, Switzerland *(10% for 6 months)* *2020 – 2021*
- Collection and annotation of cone-detection dataset.
- Development of low-compute cone-detection algorithm from scratch. Functional prototype under tight deadline.
- Development of overall self-driving software architecture.

Intel Labs Europe Munich, Germany *2013 – 2020*
- (Master thesis) Development of full custom object detection accelerator on FPGA and corresponding linux drivers.
- Development of a prototype real-time data annotation pipeline in the vehicle for autonomous vehicles.
- Development of smart infrastructure monitoring system for real-time road-agents tracking.
- Development of driver model algorithm for action recognition, with a custom dataset (dmd.vicomtech.org).
- Implementation of automotive full-self driving stack.

Select Publications

- H. O. Shahreza, C. Ecabert, A. George, A. Unnervik, et al. **"SDFR: Synthetic Data for Face Recognition Competition"**, The 18th IEEE International Conference on Automatic Face and Gesture Recognition 2024 (FG)
- A. Unnervik, H. O. Shahreza, A. George, S. Marcel **"Model Pairing Using Embedding Translation for Backdoor Attack Detection on Open-Set Classification Tasks"**, Under review at IEEE Transactions on Information Forensics and Security 2024 (TIFS)
- A. Unnervik, S. Marcel **"An anomaly detection approach for backdoored neural networks: face recognition as a case study"**, International Conference of the Biometrics Special Interest Group 2022 (BIOSIG)
- F. Geissler,A. Unnervik, M. Paulitsch **"A Plausibility-Based Fault Detection Method for High-Level Fusion Perception Systems"**, IEEE Open Journal of Intelligent Transportation Systems 2020 (ITS)
- J. Ortega, N. Kose, P. Cañas, M. Chao, A. Unnervik, G. Rigoll, M. Nieto, O. Otaegui, L. Salgado **"DMD: A Large-Scale Multi-Modal Driver Monitoring Dataset for Attention and Alertness Analysis"**, European Conference on Computer Vision 2020 (ECCV)
- B. Malnar, A. Unnervik, N. Kose **"Using High-Performance Computing in Vehicles to Create Image Datasets for Deep Learning"**, International Convention on Informations and Communication Technology, Electronics and Microelectronics 2019 (MIPRO)

Competitions

- **Face Recognition Challenge in the Era of Synthetic Data (2024)**: 1st position by training a face recognition model with synthetic data adjusted at training time to maximize performance.
- **Face Recognition Challenge in the Era of Synthetic Data (2023)**: 2nd position by training face recognition model pairs and translating embeddings.
- **International Create Challenge (2020)**: 3rd position, by implementing a detection system to identify edited images.
- **Synthetic Data for Face Recognition (2023)**: organization of a competition on the use of synthetic images.

Technical Skills

OS: Debian, Fedora, Ubuntu, Windows

Programming Languages & libraries: Python, pytorch, numpy, OpenCV, sklearn, C/C++

Version Control: Git

Language Skills, and Interests

Languages: French (native), Swedish (native), English (fluent), German (B2), Spanish (B2), Bulgarian (A2)

Interests: selfhosting and language learning