DETECTING TARGETED DATA POISONING ATTACKS ON DEEP NEURAL NETWORKS

A Dissertation

presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

---

by

SARA NEWMAN

Dr. Dan Lin, Dissertation Supervisor

May 2022

The undersigned, appointed by the Dean of the Graduate School, have examined the dissertation entitled:

DETECTING TARGETED DATA POISONING ATTACKS
ON DEEP NEURAL NETWORKS

presented by Sara Newman,
a candidate for the degree of Doctor of Philosophy and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Dan Lin

Dr. Grant Scott

Dr. Chunyan Peng

Dr. Yunxin Zhao

DEDICATION

I would like to thank myself for getting to this point, despite grave hardships. May I never give myself up or let myself down.

ACKNOWLEDGMENTS

I would also like to thank the Scholarship For Service (SFS) program for providing me with the opportunity to pursue a Ph.D. and, later, working a job I've wanted for nearly a decade. I would also like to thank Dr. Dan Lin for granting me the opportunity to be work with the wonderful Department of Computer Science at the University of Missouri - Columbia. I owe her many thanks for her continued help and understanding throughout my time as a graduate student.

I would like to thank the students who I have worked with at University of Missouri - Columbia for their aid in the projects described. Dr. Dalton Cole and Maya Cutkosky have been a great help in completing this dissertation.

Lastly, I must vehemently thank Dr. Jennifer Leopold and Dr. A. Ricardo Morales. Without their friendly and engaging instruction, I would not have considered myself capable of pursuing computer science. In addition to the profound inspiration they instilled in me, attending their classes has provided me with the initial stepping stones towards what I would later consider my most passionate goals.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

ABBREVIATIONS

CNN	Convolutional Neural Network
DEFEAT	Deep-neural-network and Embedded FEAture-based deTector
DNN	Deep Neural Network
GAN	Generative Adversarial Network
KNN	K-Nearest Neighbours
NN	Neural Network
PCA	Principal Component Analysis
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Networks
SVM	Support Vector Machine

DETECTING TARGETED DATA POISONING ATTACKS
ON DEEP NEURAL NETWORKS

Sara Newman

Dr. Dan Lin, Dissertation Supervisor

ABSTRACT

Deep neural networks (DNNs) are widely used for various facial image-recognition purposes, including facial recognition and subsequent authentication, and the detection of altered facial images. Unfortunately, due to their widespread use, there have been many works that focus on attacking such DNN-based systems for nefarious purposes. One type of attack on DNNs is called a "targeted data poisoning" attack, which has the goal of injecting photos into the DNNs training set in such a way as to cause the DNN to learn malicious behavior. In the context of facial authentication, this could correspond to unauthorized users gaining access to a target's account, whereas, in deepfake detection, this could translate to causing the DNN to fail to identify when a target's face is the subject of a deepfake image. This report describes targeted data poisoning attacks and proposed defenses on DNN-based systems for facial authentication and deepfake detection, each achieving high accuracy ($> 95\%$) in most cases.

Chapter 1

INTRODUCTION

This report is composed of two works that seek to enhance the robustness of deep neural networks (DNNs), both in the context of facial authentication software and in detecting whether an image of a face depicts an altered face (a deepfake). Targeted data poisoning attacks are discussed that undermine the effectiveness of DNNs in these contexts, as well as frameworks for detecting such attacks.

1.1 DNN-BASED FACIAL AUTHENTICATION MODEL

Facial authentication has been increasingly commonly used to unlock personal devices such as smartphones and laptops. Due to its ease of use, the next major horizon for facial authentication applications may be web services [1]. According to statistics [2], an Internet user has an average of 26 different online accounts but only 5 unique passwords for these accounts. This fact is not particularly surprising since due to the difficulty of memorizing a large number of different, complex passwords. One method of mitigating this difficulty is the use of password manager software; however, in practice, using password managers is not entirely effective. An internet user usually accesses web services from different personal devices, often in different locations, such as home and work. It is a tedious and almost infeasible task for a person to record the new password for new web services on all their devices immediately upon new account creation, especially because they may not have access to some devices (such as those

at work) at the moment the new account is created. With facial authentication in place, internet users simply position themselves in front of the device's camera to log into web services. This is convenient, swift, and accessible, being able to be done from a multitude of devices without additional preparations. Its promising market potential has fostered several releases of facial recognition APIs [3, 4] . It is envisioned that facial authentication will be widely adopted in web services in the near future.

For the successful deployment of facial authentication for online services, security is undoubtedly necessary to address. Facial authentication relies on accurate facial recognition. The most recent facial recognition techniques, such as FaceNet [5], which achieve high accuracy, are built upon deep neural networks (DNNs) [6]. Unfortunately, due to their complex nature, DNN models are vulnerable to a variety of emerging attacks, such as adversarial input attacks [7, 8, 9, 10, 11, 12, 13, 14], data poisoning attacks [15, 16, 17, 18, 19], and model stealing attacks [20, 21, 22, 23]. In the context of facial authentication for web services, adversarial input attacks and data poisoning attacks could be the most devastating threats. Both attacks aim to mislead the classifier into misclassifying the input image. In terms of facial recognition, such attacks could result in a legitimate user being misclassified and denied access to the service; or, worse, cause an attacker be misclassified as a legitimate user, thus gaining access to the victim's account. Although there have been some defensive mechanisms for adversarial input attacks and data poisoning attacks on image classifiers [10, 14, 24, 25, 26], to the best of our knowledge, none of the existing works consider the following attack scenario that can easily occur in future facial authentication for web services, and none of the existing works is effective at defending such attacks.

As shown in Figure 1.1, the new web-service facial authentication attack may happen when a user signs up for a new web service or update their facial images for a certain web service. Facial authentication typically requires the users to take

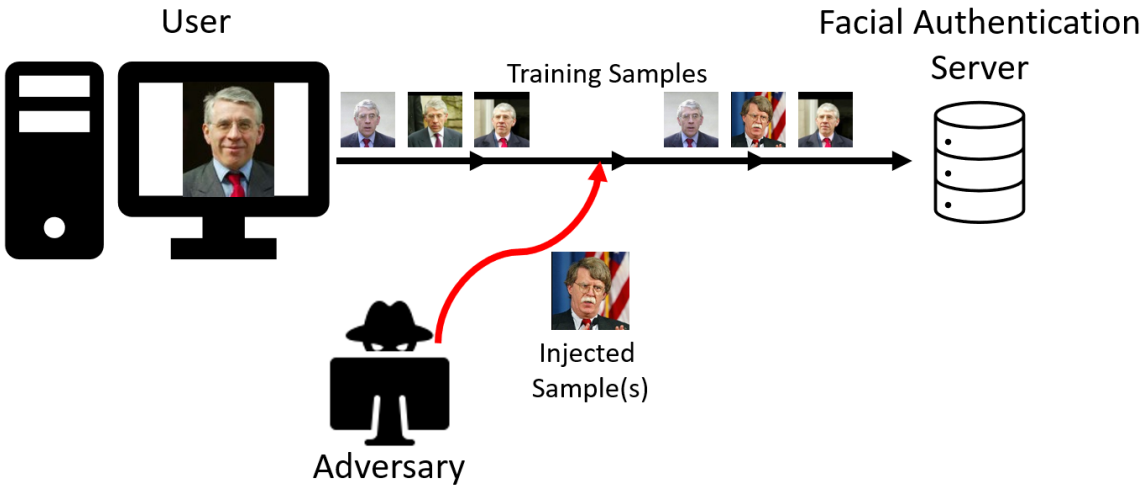


Figure 1.1: Proposed Data Poisoning Attack

photos of themselves to train the facial recognition classifier. The study outlined in Section 3.1 shows that an attacker simply needs to inject less than a handful of their photos during this process; the facial authentication system on the service provider side will later recognize both the authentic user and the attacker as the same person. Thus, both the authentic user and the attacker will have the same access to the account that the user registered. Such an attack can be conducted by exploiting the vulnerability of the victim's home network and router. In a 2020 security review of 127 popular home routers, vulnerabilities were discovered that could result in the man-in-the-middle attacks [27, 28]. As the facial authentication registration attack pollutes the training dataset, it falls under the category of a data poisoning attack. However, this new attack is easier to implement than most existing data poisoning attacks, as well as adversarial input attacks. This is because our new attack does not require the attacker to know any insider information at the server-side, whereas existing machine learning attacks [8, 29] typically require the attacker to compromise the server to gain knowledge of feature vectors produced by the deep neural network (DNN). For example, to impersonate a person, one previous attack strategy [8] requires the attacker to know the victim's facial feature vector generated by the DNN at the server-side to create special glasses that can produce the same

feature vector as the victim when an attacker wears it. Moreover, the proposed attack is extremely stealthy, as it does not affect the normal use of the infected account. Once the attacker gains the same access rights as the victimized user, the attacker can track the user's service usage over time, or impersonate the user at any desired point. For example, the attacker can easily purchase items using the victim's account if the victim does not regularly check their order or credit card history; the attacker may also post or send misinformation on behalf of the victim to hurt the victim's reputation or fool other users. Currently, there is no effective defense mechanism proposed to battle such an attack.

In this work, the devastating effect on web-service based facial authentication produced by this novel data poisoning attack will be discussed. Then, the novel defensive strategy, called DEFEAT (Deep-neural-network and Embedded FEAture-based deTector), will be introduced.

Specifically, it has been observed that, with only 4 or 5 attacker's facial photos mixed with the user's training photos (another 4 or 5 photos), the attacker will be able to impersonate the user in the future authentication without dropping the overall facial recognition accuracy, i.e., without raising alarm to the facial authentication system. It has also been observed that it is difficult to distinguish the attacker's feature vector from the authentic user's by using only statistical analysis and distance comparison. The resulting hypothesis of this phenomenon is that, since facial recognition systems, such as FaceNet [5], strive to achieve high recognition accuracy and since they do not know the training set of a given user contains photos of different faces, the contaminated feature vectors (i.e., those being attacked) are then generated based on common features between the user and the attacker as to ensure both the original user and the attacker can authenticate using their own photos. As a result, various distances (e.g., Euclidean, Hamming, Manhattan) are not sufficient to measure the differences between the victim's and the attacker's resulting facial feature

vector, since they are intentionally generated by DNN to be very similar for the goal of maintaining high recognition accuracy. However, an effective method to detect these malicious attempts must be pursued.

Based on the hypothesis that the contaminated feature vectors are generated by extracting common features from two people's faces (i.e., the victim and the attacker) whereas the non-contaminated feature vectors are based on the features of only one person, this work introduces an intelligent discriminator, DEFEAT, to identify the potentially subtle differences in these two kinds of feature vectors. The DEFEAT discriminator has the base structure of a DNN and a KNN (k-nearest-neighbour) model. Various concatenation approaches are also designed to create training inputs for the discriminator. The layers of the DNN in DEFEAT is optimized for both accuracy and efficiency. Upon real-time detection, DEFEAT takes the feature vector output by FaceNet and produces a probability of whether or not the input feature vector is contaminated. The probability is then sent to the KNN model to produce a binary decision: under attack or clean. This approach has been evaluated in real datasets that represent both consistent and diverse background settings. It is shown that the discriminator achieves more than 90% detection accuracy in all cases. The contributions of this work are summarized as follows:

- A novel data poisoning attack to facial authentication is introduced, which allows the attacker to easily impersonate the victim.
- Novel discriminators are proposed to detect the above impersonation attack. The experiments on real datasets demonstrate that it is capable of achieving very high detection accuracy in various settings.

1.2 DNN-BASED DEEPFAKE DETECTION MODEL

In addition to detecting attacks on facial authentication, this work also discusses a method for detecting data poisoning attacks on deepfake detectors. Deepfakes are images of a face that have been altered such that the resulting image appears to depict a different face from the one originally depicted.

Traditionally, cameras are regarded as trustworthy devices, and images have historically been considered to always depict a scene that truly happened. Thus, digital images have been widely used as historical records and evidences for the journalist reporting, police investigation, auto insurance claims, military intelligence, etc. However, with the advances in artificial intelligence, *seeing is no longer believing*. Forged images and videos that appear real (i.e., unaltered) to both human viewers and existing computer programs, can now be generated by Deepfake techniques [30, 31, 32, 33, 34], which take advantage of the latest deep neural networks, the Generative Adversarial Network (GAN), and the growing computational power.

With the ability of manipulating image/video content without being noticed, deepfakes are imposing a new crisis for privacy, democracy and even national security [35, 36, 37, 38]. Many online posts contain one or more image/video, and the posts with attention-grabbing visual content can spread and become viral extremely quickly, influencing the opinions and behavior of a large number of readers. Malicious parties can utilize deepfakes to swap a victim's face into uncomfortable scenes, and damage that person's reputation in social media as well as their real life. Not only can personal privacy be harmed, deepfakes may also be exploited to create fake news to affect results in election campaigns, create chaos in financial markets, fool the public with false disaster scenes, or even inflame public violence [35, 36, 37, 38]. Even worse, the abuse of deepfakes may also be leveraged to cause political or religious tensions between countries.

Fake images can be synthesized using different deep neural network models, resulting in various types of fake images such as [30, 31, 32, 33, 34]. In order to identify such AI-generated fake images, some detectors [39, 40, 41, 42, 43, 44, 45, 46, 47] have been proposed. One notable detector is XceptionNet [40] which has been extensively tested and can achieve detection accuracy as high as 99% on various types of fake images. Unfortunately, having these detectors is still not sufficient to prevent the aforementioned misuse of fake images due to the rise of adversarial machine learning attacks. It has been reported that fake image detectors are vulnerable to adversarial input attacks [48] whereby attackers inject noises to the fake images to mislead the detector into labeling the perturbed fake image as real. The adversarial input attacks have successfully been mitigated by using adversarial training methods to train the detector with adversarial samples. However, another type of attack, the data poisoning attack, cannot be defended by the adversarial training strategy because data poisoning attack does not perturb the input image at all during run time. The poisoned training images have the same content but wrong labels. Specifically, a fake image would be purposely labeled as real by the attacker to fool the detector. Since this data poisoning happens during the model training phase, it is very hard for the detector to self-identify poisoned training samples. The existing adversarial training strategies [49, 50, 51, 52, 53] that teach the classifier about the noises and perturbations in the input images would not work in this case, as the poisoned images do not contain any perturbation that leads to an incorrect label. In other words, the attack discussed in this work poisons the image labels but not the image content.

In this paper, we propose a new defensive mechanism targeting the above data poisoning attacks on the deepfake detectors. The key idea is to add another deep neural network, called the protector network, to the end of the deepfake detector. The protector network is dedicated to the classification of normal and poisoned data. If a fake image is mislabeled as real, the protector will capture it and raise alarm.

While the deepfake detector may need to be updated (i.e., periodically retrained) in the field since there are always newly emerging types of fake images to learn, our proposed protector network will not need to be retrained as its ability of distinguishing poisoned fake images is not affected by the types of fake images. That means, even though the detector may be poisoned during the retraining phase, the protector will still be secure. More specifically, the protector will be trained only in-house and learn the differences between the feature vectors of poisoned fake images, clean fake images and real images generated by the detector. The proposed protector has a new network architecture based on ResNet [54] along with several techniques to achieve the generalization ability when it is given unseen types of fake images that have been poisoned. To summarize, the this work results in the following contributions:

- The impact is investigated of data poisoning attacks on existing deepfake detectors.
- A novel defense mechanism is introduced, called Protector, to make the deepfake detectors robust against data poisoning attacks.
- Extensive experiments on real datasets have been conducted, showing the effectiveness of the described approach, achieving over 95% accuracy for detecting the poisoned images.

Chapter 2

LITERATURE REVIEW

This chapter provides a discussion on existing literature that is relevant to the works described in this report, particularly attacks on deep neural networks (DNNs). The existing attacks that focus on causing undesirable behavior in machine learning algorithms, such as DNNs, can be classified into two main categories: adversarial input attacks and data poisoning attacks. Both will briefly be discussed, with special focus being given to data poisoning attacks, which are directly relevant to the works described in this report.

2.1 ADVERSARIAL ATTACKS ON DNNS

Adversarial input attacks typically occur after the machine learning algorithm has completed the training process, which is different from data poisoning attacks, which are carried out during the training process. The goal of the adversarial input attack is to perturb the input data in a way that is meant to fool a classifier, i.e., cause the input data be misclassified. These inputs are usually crafted by adding a perturbation to an authentic image [7, 8, 9, 10]. Specific to facial recognition models, there is an abundance of works on how to compute perturbations to cause errors in the facial recognition process [7, 8, 9, 49, 51, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65]. For example, [7, 49] turn perturbation generation into an optimization problem and compute the perturbation in a single large step based on the model's loss. [55]

extends the idea to computing perturbations iteratively. [51, 56, 57] seek to alter the least amount of pixels possible, while [57] focuses on modifying only one pixel. [9, 58] modify images iteratively using the information on the decision boundaries of the target model. These algorithms are all designed for a single given image, needing to be run separately for each additional target image. [59, 60] work on any image to fool the target network. [61, 62, 63, 64] generate entire adversarial images that appear similar to clean images and fool the target model adequately. Another interesting way of image perturbation is that attackers put on customized accessories such as glasses in front of the camera to impersonate the victim [8, 65], while these special accessories are designed based on the victim's feature vectors generated by the face recognition model. With this said, most of the existing adversarial input attacks [7, 8, 9, 49, 51, 55, 56, 58, 59, 62, 63, 64, 65] require white box knowledge of the model, i.e., knowledge of the internal parameters of the model, which may be hard to achieve in real scenarios. Only a few adversarial input attacks [57, 60, 61] can treat the target model as a black box and still conduct adversarial input attacks.

Currently, the most effective defense mechanism against the adversarial input attacks is adversarial training [49, 50, 51, 52, 53] which enhances the robustness of the neural networks by allowing them to learn adversarial samples during the training.

2.2 DATA POISONING ATTACKS ON DNNS

Data poisoning attacks on DNNs seek to alter the final behavior of a machine learning model during deployment by manipulating a subset of the training data. Hereafter, training data that has not been manipulated in this way are referred to as "clean" data, whereas data that has been manipulated is referred to as "poisoned" data. The size of the subset of training data that must be poisoned to achieve the malicious result varies by both the method of data manipulation and the specified goal. Goals of data poisoning attack fall into two categories: untargeted and targeted attacks.

The goal of an untargeted data poisoning attack is to hinder or halt the DNN from achieving reasonably high accuracy, effectively preventing it from learning during the training phase. These types of data poisoning attacks are relatively easy to identify, as the introduction of poisoned training samples causes the target DNN to experience a significant drop in accuracy or difficulty of increasing accuracy. Targeted attacks are significantly harder to detect than untargeted attacks, because their goal is to cause a DNN to misclassify only a single class of inputs, or even a single specific input, at runtime, which usually does not significantly affect the accuracy of the target DNN. Moreover, these types of attacks often introduce a perturbation to clean training samples to craft the poisoned data, which may be either visually similar to clean data or containing an imperceptible perturbation, further complicating detection. More formally, the goal of a targeted data poisoning attack is as follows: Given a classifier, $f(x) = y$ for data $\vec{x}$ and corresponding labels, $\vec{y_{gt}}$, an attacker seeks to craft poisoned training data $x_p = x + \delta$ such that, if $f(x)$ is trained using x_p, then $f(x_t) \neq y_{gt}$, for some target input x_t. The attack described in Chapter 3 Section 3.1 is classified as a targeted data poisoning attack.

2.2.1 Untargeted Attacks

Many data poisoning attacks have been introduced with the intent of not only causing misclassifications of a certain class of inputs but also causing as many misclassifications as possible [15, 16, 17, 18, 19]. A common application for this type of attack, as described in [15],[16], and [17], is attacks against spam filters, for which an attacker typically endeavors to cause spam emails to be labeled as legitimate or "ham" emails, and vice versa. The attacks described in [15],[16], and [17] target naive Bayes spam filters by altering spam messages in ways that cause the classifier to misbehave. The attack method proposed in [18] uses a gradient ascent strategy based on the properties of a support vector machine (SVM), which requires white-box access to the target

model. Similarly, [19] uses a back-gradient optimization method to manipulate any machine learning model that learns using gradient descent. Shortly after the development of such attacks, it was discovered in [66] that it is straightforward to detect such attack attempts by monitoring the model's loss prior to and following the addition of certain samples into its training set, thus presenting an effective mitigation method.

2.2.2 Targeted Attacks

There are multiple sub-types of targeted data poisoning attacks, a relatively recent and commonly-studied one being backdoor attacks [67, 68, 69, 70]. These seek to train a "backdoor" into a DNN by adding a certain pattern or watermark (i.e., a "backdoor" or "trigger") to certain inputs in the training set such that the resulting DNN classifies any input containing this pattern or watermark as a certain target class. Because the resulting DNN does not typically behave differently for inputs not containing the backdoor, these types of attacks are difficult to detect, thus garnering power surpassing many of their untargeted counterparts. These backdoors vary in complexity, with the method described in [67] simply using basic triggers such as a white box "stamped" in the corner of certain training data, while changing the labels of the inputs containing the backdoor to their target class. This caused the target model to learn that this simple backdoor is the predominant feature corresponding to the target class. [68] and [69] craft backdoors by exploiting properties of the target classifier, creating an optimal backdoor. While requiring greater access to the target model than attacks such as [67], the optimal backdoors achieve greater attack success. Several defenses exist for mitigating the effect of backdoors on a model, including anomaly detection on the input space and input classification [25, 71] and strategically altering the structure of the target classifier in a way that "unlearns" the behavior caused by backdoors [70, 72].

Other types of targeted data poisoning attacks are based on perturbing training

images by adding noise, either digitally by directly modifying the input data or using the presence of specially designed features within the input, causing the classifier to misclassify these inputs during training [73, 74]. These attacks tend to require more intimate knowledge of the target model, which is often impractical. In [73], an optimization scheme is developed based on the classifier output and the level of perturbation required to alter training images appropriately, which, in turn, achieves misclassification of a single input in the final model. A more recent work, [74], considers a variety of attackers, all having partial knowledge of the system. The attack described in Chapter 3 Section 3.1 does not require any access to the classification model.

As of now, targeted data poisoning attacks do not appear to have any existing defenses that are effective. Hopefully, the proposed defense utilizing DNNs presented in 3 will provide inspiration for the development of more general defense mechanisms for targeted data poisoning attacks.

Chapter 3

DETECTING TARGETED DATA POISONING ATTACKS ON DNN-BASED FACIAL RECOGNITION MODELS

Because importance of secure authentication systems is paramount, the security of facial authentication is an important area of study. As such, this chapter presents a framework for enhancing the robustness of facial authentication in response to a novel data poisoning attack on DNN-based facial authentication systems that is both extremely easy to carry out and powerful. The layout of this chapter is as follows: Section 3.1 describes the attack scenario, then, in Section 3.2, the proposed defense system, DEFEAT, is introduced. Section 3.3 provides a discussion of the system's performance.

3.1 A NEW TARGETED DATA POISONING ATTACK ON FACIAL AUTHENTICATION

3.1.1 Threat Model

The attack proposed in this section is categorized as a targeted data poisoning attack, during which an attacker attempts to impersonate a victim during facial authentication by causing the facial recognition system to allow the attacker to log into the victim's accounts using the attacker's own images of their face. Formally, let Img_t denote the targeted victim's face image, l_t denote its true label, and Img_a denote the attacker's face image. Let IMG_p denote the set of other users' images that are in the

Dataset	# of Users	# of Photos/User	Training/user
FEI	200	14	10
LFW	158	10-530	10

Table 3.1: Facial Image Data Sets

clean (unattacked) stage, Img_p denote each of the other user's images, and l_p denote the corresponding clean label. When perpetuating targeted data poisoning attacks on facial recognition systems, the attackers must be careful to avoid altering the final classification accuracy in any significant way, which would cause alarm, potentially tipping off moderators that something is amiss. Thus, the goal of this attack is to maximize the probability that Img_a will be misclassified as l_t while avoiding preventing Img_t and Img_p from being labeled correctly. The goal of attacking Img_t is formalized as G(**Img**, **L**), shown in Equation 3.1,

$$G(\mathbf{Img}, \mathbf{L}) = \min_{n} \left(\frac{\mathcal{L}(\mathbf{Img}_a, l_t)}{n} + \frac{\mathcal{L}(\mathbf{Img}_t, l_t)}{(s-n)} + \frac{\mathcal{L}(\mathbf{Img}_p, \mathbf{L_g})}{s \cdot c_p} \right) \tag{3.1}$$

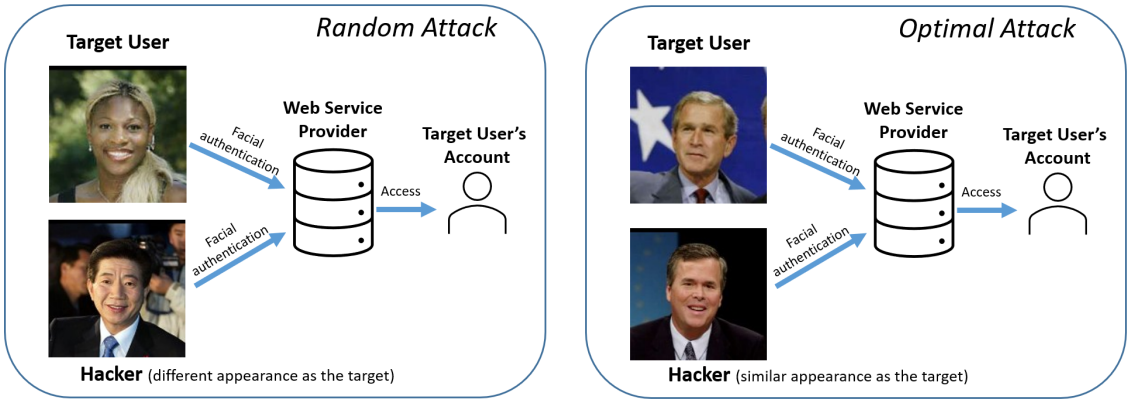
where **Img** is the set of images and **L** is the set of labels for the images in **Img**, s is the total number of photos of a user, n is the number of images replaced by the adversary, c_p is the number of benign users, and $\mathcal{L}()$ is the loss function.

This attack was conducted using the following data sets: FEI [75] and Labeled Faces in the Wild (LFW) [76]. Table 3.1 shows the size and statistics of each data set. Because DNN-based facial recognition models are pre-trained in a facial authentication setting, these data sets are simply used to simulate adding new users to an existing facial authentication system. The types of images included in these data sets represent the ideal setting (i.e., easy facial recognition scenarios) and the real-world setting (i.e., more realistic facial recognition scenarios). These settings are represented by the FEI data set and the LFW data set, respectively. Images in the FEI data set are taken with a relatively constant background color and lighting with

the subject facing the camera straight-on with a variety of set facial expressions. On the other end of the spectrum, LFW is potentially one of the most diverse data sets, containing photos of celebrities with many different backgrounds, facial expressions, and postures. As such, it has been commonly used as a benchmark for the evaluation of many facial recognition systems. In this work, it is used to simulate various background environments over which a user may wish to log into their web service accounts. Included in the LFW data set are 5,749 different people, with differing numbers of photos. In the following experiments, only the labels with at least 10 separate photos are considered. This ensures that every user involved has sufficient corresponding training and testing images.

In order to prepare the data sets for deploying this attack, each data set, denoted $\mathcal{D}$, is split into three equally-sized subsets, $\{Target, Attack, Clean\}$, where $Target$ simulates images of new users who are to be new targets of the attack, *Attack* simulates images of attackers, and $Clean$ images are of benign, newly-added users who are not to be a victim. To launch the attack, a random user is selected form the $Target$ group, and a malicious user is chosen from the *Attack* group. A certain number of images from $Target$ is then replaced with images from *Attack*, as a man-in-the-middle (MITM) attack on the target user's home router. It is important to stress the importance of MITM attacks, as recent studies have confirmed that a large number of routers and devices remain vulnerable to these types of attacks [27, 28], despite efforts to mitigate these glaring security vulnerabilities. In this study, each label in the *Attack* group manipulates the facial authentication method for only one label in the $Target$ group. Then, images from all three groups are passed through the DNN for training. Based on the selection method employed to determine the attacker for each target, two kinds of attacks are defined:

- **Random Attack**: To simulate the photos of a malicious user, a set of photos with the same label (affiliated user) from the *Attack* data set are chosen ran-



(a) Random Attack Strategy (b) Optimal Attack Strategy

Figure 3.1: Attack Strategy Overview

domly. This may result in attacker with a different race, age, and gender from the target of the attack. An example is depicted in Figure 3.1a

- **Optimal Attack**: A set of photos in *Attack* corresponding to a single user is chosen based on similarity to the targeted victim, such as a relative or users of similar race, age, and gender as the target. An example is depicted in Figure 3.1b

The attack is then evaluated first through examination of the total facial recognition accuracy of the target as well as unaffected users, ensuring that the attacked model continues to recognize these users with similar accuracy to the clean model, so as to avoid showing signs of malfunctioning. Second, it is evaluated based on whether the attacker is able to successfully be recognized as the target user. These qualities, in tandem, determine the overall success of the attack.

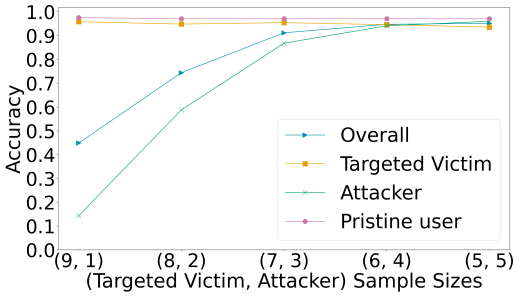
3.1.2 Attack Results

Both attacks on both data sets are carried out on FaceNet [5] and shown in Figure 3.2. The results of the random and optimal versions of the attack on the FEI data set are shown in Figures 3.2a and 3.2b, respectively. The results of the random and optimal attack on the LFW data set are shown in Figure 3.2c and 3.2d, respectively.

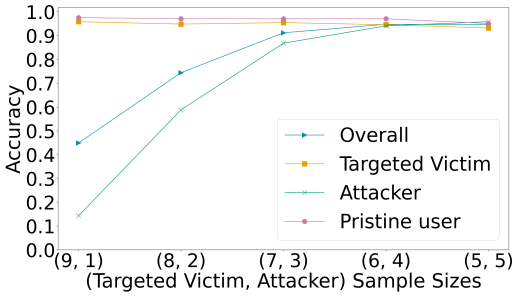
Throughout these experiments, 10 photos are used to register each user into the facial recognition system. The horizontal axis shows the number of the target user's photos and the number that are replaced with the attackers photos, e.g., '(9,1)' means that, for a single user face registration, 9 photos are of the original user and the attacker only replaces 1 photo with their own face. In each case, the attacker's photos are inserted randomly into the set of the victim's photos. The order of the photos inserted does not affect the attack success rate (ASR). The vertical axis shows the ASR, i.e., the rate at which the both the target user is correctly identified, the rate at which the attacker is recognized as the target user, the rate at which the untouched user is correctly identified, and the overall success. As depicted in Figure, 3.2, the ASR increases with the number of attacker photos injected into the victim's image set. When half of the images are replaced with the attacker's image, both the random and optimal attack on both data sets shows a 99% ASR, in some cases even allowing the attacker to become even more easily identifiable than the target user. This may be a result of FaceNet's DNN portion treating both sets of photos with equal importance, extracting the common features for accurate facial recognition. After the new user is added, the attacker gains the same access as the target user. It is additionally important to note that the optimal attack does not appear to pose much of an advantage over the random attack, essentially removing the constraint on an attacker to choose a victim visually similar to themselves.

3.2 OUR DEFENSE MECHANISM

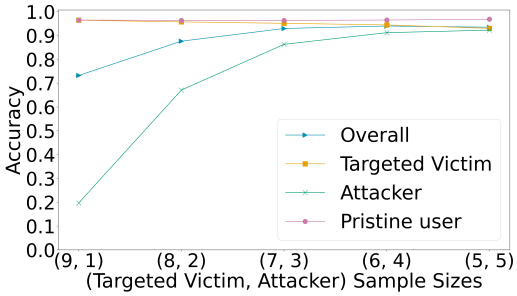
In order to undermine the effectiveness of these types of attacks on facial authentication systems, a new defense mechanism, DEFEAT, is proposed. Throughout this section, a system overview of the proposed system is discussed.



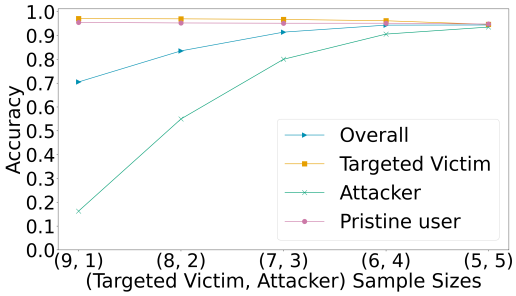
(a) Random Attack on the FEI dataset



(b) Optimal Attack on the FEI dataset



(c) Random Attack on the LFW dataset



(d) Optimal Attack on the LFW dataset

Figure 3.2: Replacement Data Poisoning Attack Results on FaceNet

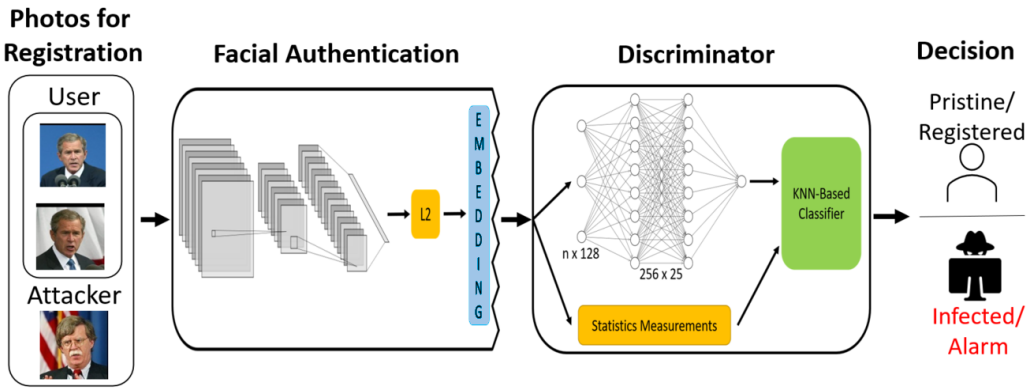


Figure 3.3: An Overview of the DEFEAT System

3.2.1 System Overview

Figure 3.3 presents the overall data flow of the proposed DEFEAT system. The three parties involved in this process are the user/attacker, the facial authentication system, and the discriminator. Because the man-in-the-middle attacks still thrive in home routers according to 2020 studies [27], attackers who exploit the vulnerabilities of the user's router have the ability to compromise the user registration process, during which the proposed system detects the threat on the server side. Specifically, a number of training photos provided by the user (or attacker) will be first sent to the facial authentication system, which will not immediately register the user at this point. Instead, the facial embeddings generated by FaceNet will be fed to the discriminator for evaluation. If the photos are deemed clean by the discriminator, the user registration process will proceed to register the user. Otherwise, if the discriminator concludes that the training samples may be infected, it will raise the alarm to the service provider to conduct further investigation. For example, the investigation can be easily carried out by a human expert who looks into the suspicious training samples to see if they belong to the same person. Photos that succeed in fooling the machine learning algorithms in the manner described in this chapter are easily visible to a human. However, it should be stressed, that although human experts may be good at distinguishing these infected photos, it is not practical to ask human experts to screen the large numbers of photos streaming into the web service providers every day. The proposed discriminators will significantly minimize the efforts required by human experts.

In the real-world web service scenario, there are two possible options to deploy the above framework, which are (i) on-site detection; and (ii) off-site detection. For the on-site detection, the web service provider installs the proposed discriminator along with their original facial authentication system to carry out threat detection automatically. Alternatively, there could be a third-party security provider that oversees

evaluating security threats using the discriminator. Since the discriminator only needs the facial embeddings and statistic measurements as input, none of the users' private facial images will be disclosed to such a third-party security provider, effectively making this off-site evaluation feasible. The advantages of the off-site evaluation are that it not only relieves the web service provider's burden of another security duty but also gives the third-party security providers the ability to keep improving the discriminator and making it increasingly robust and generalized based on information collected from various service providers.

3.2.2 Defense Model

In the data poisoning attack as discussed in Section 3.1, the attacker's face images are mixed with the victim's face images when FaceNet generates the 128-dimensional feature vector for the victim. Thus, it is expected that the contaminated (attacked) feature vector of the victim would be different from the uncontaminated feature vectors of other users. Intuitively, one may think that such differences may be reflected by commonly used statistic measurements, such as internal differences among feature vectors of the same label (same user), and external distances among the different groups of feature vectors, which will be formally defined later in this section), Therefore, a statistics-based discriminator is investigated as follows.

We then develop a statistics-based discriminator for comparison with our DNN-based discriminator, described in 3.2.3. The goal of the statistics-based discriminator is to leverage statistical analysis on a facial recognition model's output embeddings (i.e., facial feature vectors) to determine if a clean (uncontaminated) label is differentiable from an infected label. We started this process by employing principal component analysis (PCA) to reduce the dimensionality of the embeddings while retaining some of the underlying relationships among the facial feature vectors. For visualization purposes, the dimensionality is reduced from 128 down to 2. The results

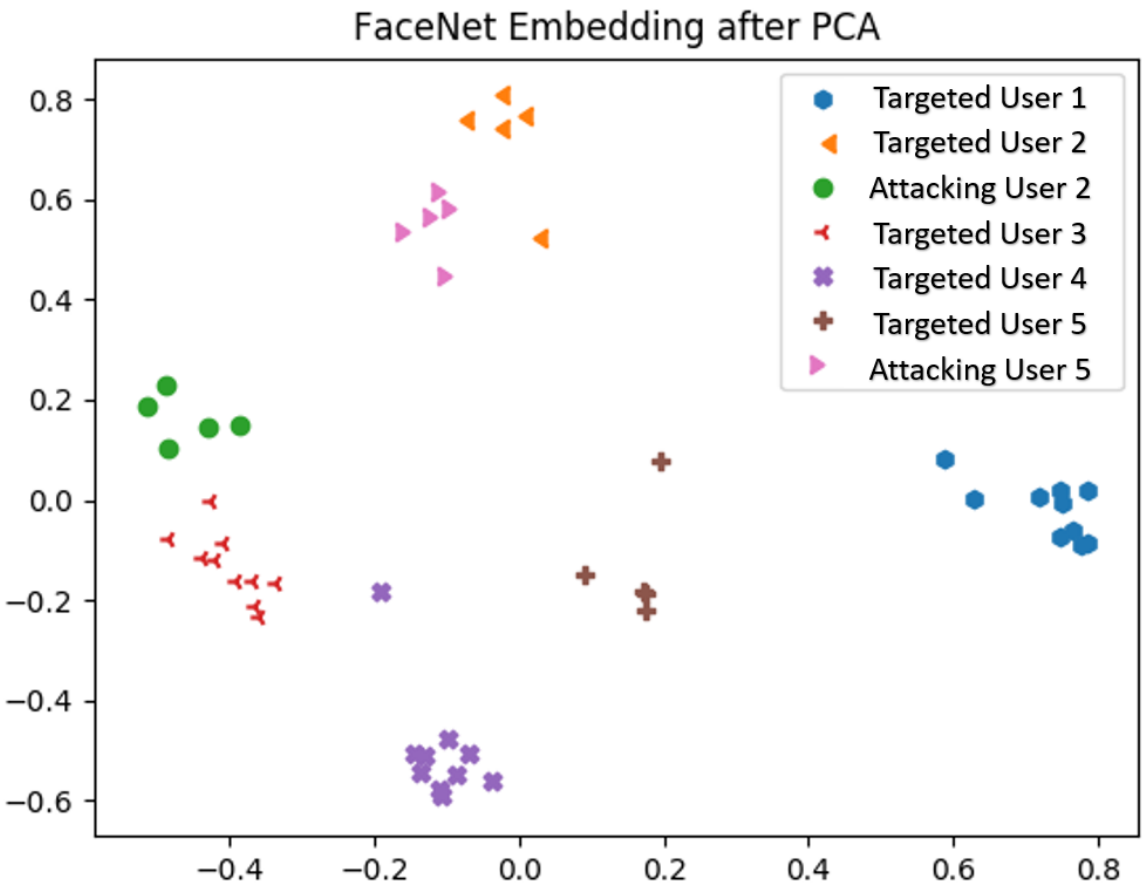


Figure 3.4: Principal Component Analysis (PCA) applied to a subset of labels on which FaceNet is trained. Five labels were randomly selected from the FEI dataset, 3 un-attacked labels and 2 attacked labels using the random attack configuration. PCA was then applied to FaceNet's neural network output embedding to reduce the dimensionality from a 128-dimensional space to a 2-dimensional space. Each label is shown, with the injected samples differentiated by a different color and symbol.

are shown in Figure 3.4. It is depicted that the attacker's samples tend to form clusters separating from the targeted victim's samples. Following, a non-visual method of differentiating the facial features is attempted. Based on the results from PCA, it is hypothesized that there may exist some key statistic measures that could differentiate the clean labels from the infected labels when all the dimensionality is considered.

The first statistics measure explored is the maximum internal difference between every embedding for a label. The PCA plot highlights an apparent difference between the maximum internal distance when reduced to two-dimensional space. It was then

studied whether this difference is present when the facial embeddings maintain their full dimensionality as opposed to only when reduced to 2-dimensional space. The maximum internal differences are formally defined as follows:

Definition 3.2.1 (Maximum Internal Difference)**.** Let $\mathcal{E}$ be the whole set of facial embeddings (feature vectors), and let e_i^ℓ, e_j^ℓ denote the different embeddings with respect to the same label ℓ. The maximum internal difference for a label ℓ is calculated using $\mathcal{L}_1$-norm which maps the distance between the two embedding vectors to a scalar value without magnifying larger differences between the two embeddings.

$$f_{max}^\ell = \max_{i \neq j} \mathcal{L}_1(e_j^\ell - e_i^\ell)$$

The second statistic measure is the minimum external difference between labels. Because the cluster of the embeddings of the same label tends to be wider or more separated when that label is attacked, it is hypothesized that the minimum external difference would be smaller when the label is under attack versus when it is not under attack. Formally, the external difference between the two groups of embeddings is defined as follows.

Definition 3.2.2 (Minimum External Difference)**.** Let L denote the whole set of labels, k be the label to be considered, and ℓ be the remaining labels in L. The minimum external difference between the embeddings of label k and that of all the other labels ℓ is calculated as follows, which finds the smallest distance between any embedding of label k and the nearest embedding of a different label:

$$f_{min}^\ell = \min_{k \neq \ell} \mathcal{L}_1(e_j^\ell - e_i^k)$$

Based on individual external differences, the mean external difference can be computed as follows:

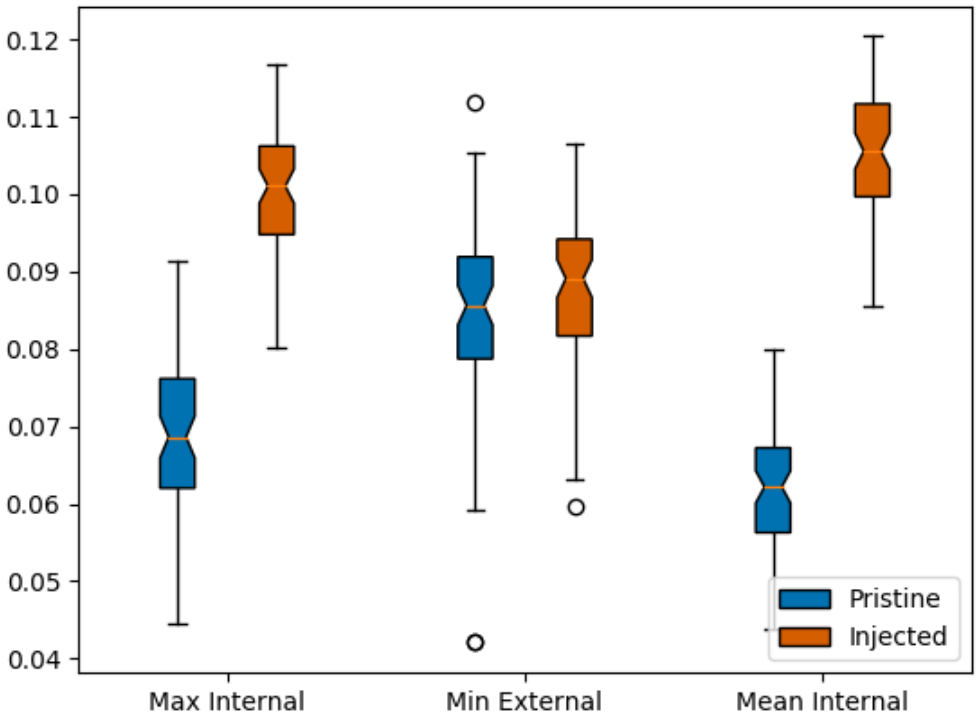


Figure 3.5: Box plot of the maximum internal, minimum external, and mean external differences between the facial embeddings for a given label. In order to properly fit in a single figure, each feature was normalized such that the sum is zero.

Definition 3.2.3 (Mean Internal Difference)**.**

$$f^{\ell}_{mean} = \frac{1}{n * m} \sum_{i=0}^{n} \sum_{j=0}^{m} \mathcal{L}_1(e^{\ell}_j - e^{\ell}_i)$$

The first statistic measure focuses on the possible changes caused by an attack within individual groups of embeddings, while the second statistic presents a bigger view of the potential influence of the attack on the relationship among different groups of embeddings. Figure 3.5 shows the relationship between infected labels and clean labels using these three metrics. From this analysis, a K-Nearest-Neighbor (KNN)-based discriminator is devised that takes as input triplet t where $t^l = \{f^{\ell}_{\max}, f^{\ell}_{\min}, f^{\ell}_{mean}\}$, and outputs whether the given t^{ℓ} is a clean or a targeted label.

KNN was chosen based on the observation from the previous two figures that groups of the clean labels may have similar statistical values, whereas groups of targeted labels may have another kind of statistical value.

However, such a statistic-based discriminator does not yield a high detection rate in some cases as shown in Section 3.3. The possible cause that lowers its detection rate could be the high dimensionality which dilutes the obvious differences among different groups of embeddings as visualized in 2-dimensional space. Specifically, the minimum external difference for infected labels and clean labels are sometimes similar in high dimensional space. Additionally, the maximum internal difference and minimum external difference are likely heavily correlated. These findings lead to the development of a more intelligent discriminator as introduced in the following subsection.

3.2.3 Feature-Based DNN Discriminator

Considering the limitations of the statistics-based measurements in distinguishing infected labels from clean labels, Deep Neural Networks (DNNs) were employed for their ability to find patterns in unknown relationships among complicated items.

The result is a novel discriminator called DEFEAT (Deep-neural-network and Embedded FEAture-based deTector) that takes feature vectors produced by a facial recognition model as input and outputs the probability that the input embedding is infected. Figure 3.6 presents an overview of the structure of the two-phase DEFEAT system. The first phase is a DNN that analyzes the differences between infected feature vectors (embeddings) and clean feature vectors. The infected feature vectors refer to both the attackers' feature vectors and that of their targeted users. The analysis result is a probability value that indicates the likelihood that a feature vector is contaminated. Then, this probability value along with several statistic measurements is fed to a KNN-based classifier to yield the final binary output: (i) the label

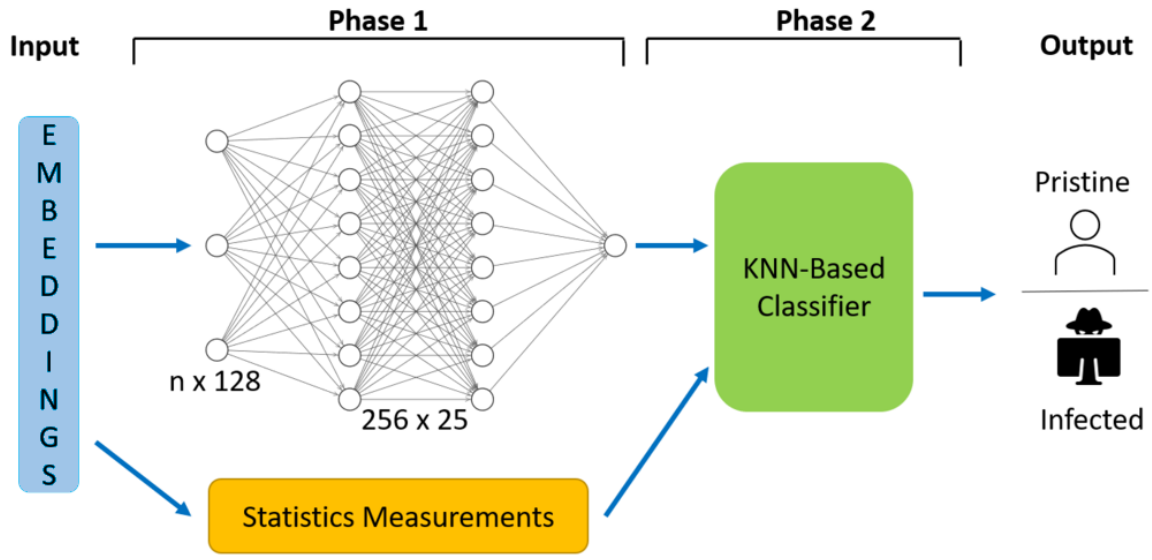


Figure 3.6: The Feature-based DNN Discriminator

is infected; (ii) the label is clean.

More specifically, the DNN in the DEFEAT system consists of 25 layers with 256 neurons per layer. Through extensive empirical analysis, this network architecture is found to give the best trade-off between speed and accuracy. Each individual layer is a dense layer with batch normalization and a 20% dropout rate. Batch normalization was used to increase the stability of the neural network, while the dropout layers keep the network from over-fitting during training. The Rectified Linear Unit [77], also known as ReLU, activation function was used for all layers except the last layer, for which the sigmoid activation function (shown in Equation 3.2) is used to calculate the probability of an infected label.

$$S(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

In addition to determining the optimal structure of the DNN, another interesting consideration is the input data. It was discovered that, instead of feeding the DNN a single embedding at a time for analysis, an input that concatenates two embeddings will significantly enhance the ability to distinguish the infected labels. Specifically,

we concatenate the following pairs of embeddings:

- Embeddings from two photos belonging to the same clean user – will be eventually labeled as "clean".
- Embeddings from two photos belonging to the same targeted victim – will be eventually labeled "infected".
- Embedding from two photos belonging to the same attacker – will be eventually labeled "infected".
- One embedding from the attacker and the other from the corresponding victim – will be eventually labeled as "infected".

Given each of the above concatenated embeddings $e_i^\ell \oplus e_j^\ell$ (e_i^ℓ, $e_j^\ell \in E^\ell$), the DNN will calculate the probability $P(e_i^\ell \oplus e_j^\ell)$ that the concatenated embedding is infected.

In the second phase, the KNN-based classifier produces a binary decision to explicitly inform the web service provider whether the registration process of a new user has been attacked. As shown in our experimental studies, the DEFEAT discriminator achieves over 90% detection accuracy in almost all cases.

3.3 PERFORMANCE STUDY

This section provides discussion of the experiments that compare the effectiveness of the proposed statistic-based discriminator and DEFEAT discriminator in terms of ideal and general settings.

3.3.1 Experimental Setting

All the experiments were conducted in the Chameleon Cloud [78]. A single Chameleon Cloud node was used with 16 virtual CPUs @2.3GHz and 32GBs of memory. The experiments involved the two data sets: FEI [75] and LFW [76] as presented in Table 3.1. The aforementioned FEI data set represents an ideal and consistent background

setting when a facial photo was taken, while the LFW represents general and diverse background settings. Each data set is equally split into three sets representing three kinds of users, targeted victims, attackers, and clean users. With approximately 15 photos per user, 10 photos were used for each targeted victim and clean user for training FaceNet and the discriminators. Specifically, for the targeted victims, some of their photos were replaced with the attackers' photos during the training. Then, the remaining photos were used for testing purposes. Both the targeted victims and attackers' photos will be labeled as injected by the discriminators.

The effectiveness of the discriminator is evaluated using the following four metrics: (i) precision; (ii) recall; (iii) F1 score and (iv) overall accuracy.

Definition 3.3.1. Precision measures the percentage of the correctly identified infected photos among all the photos being tested. In the following equation, TP stands for "true positive", FP stands for "false positive", TN stands for "true negative", and FN stands for "false negative".

$$Precision = \frac{Correctly\_Identified\_Infected\_Photos}{All\_Photos\_Labed\_Infected} = \frac{TP}{TP + FP}$$

Definition 3.3.2. Recall measures the percentage of the correctly identified infected photos against the total number of infected photos that have been tested. In the following equation, TP stands for "true positive" and FN stands for "false negative".

$$Reall = \frac{Correctly\_Identified\_Infected\_Photos}{All\_Infected\_Photos\_Tested} = \frac{TP}{TP + FN}$$

Definition 3.3.3. F1 score is the combination of precision and recall which serves as an overall performance indicator.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Definition 3.3.4. The overall accuracy evaluates the detection correctness for both the infected photos and pristine photos.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

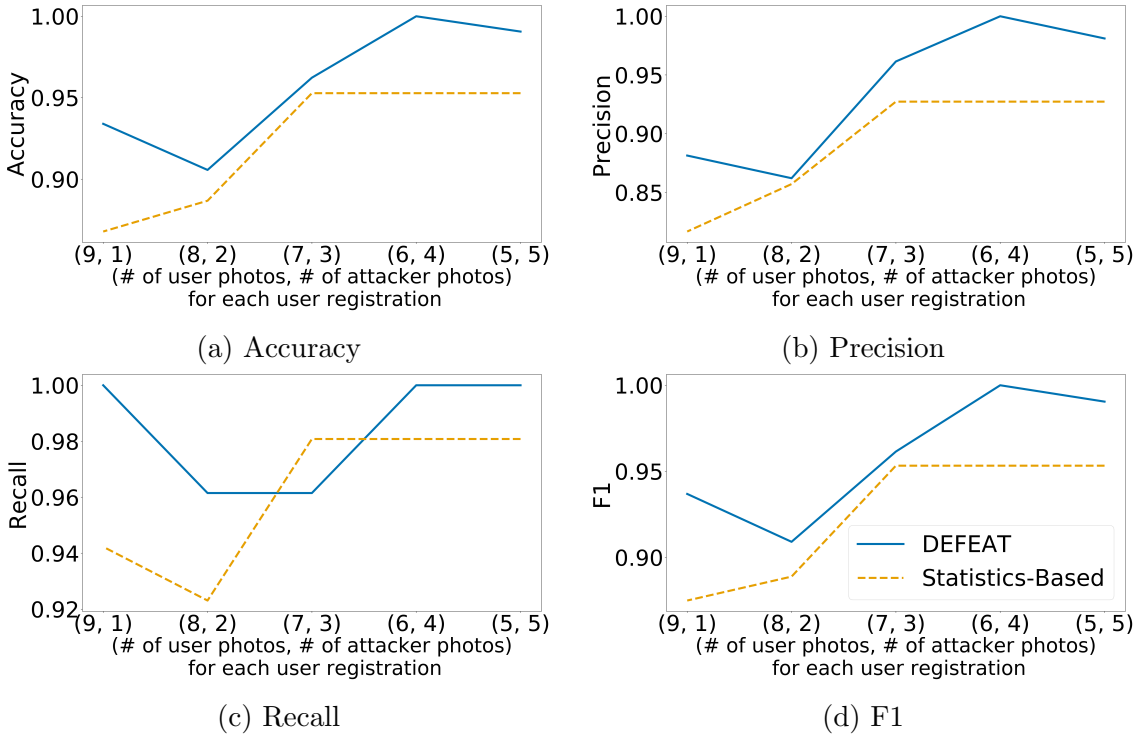
3.3.2 Experimental Results

The experiments evaluate the impact of several factors on the effectiveness of the statistics-based discriminator and the DNN-based discriminator (DEFEAT), which include the variation of the ratio of the injected attackers' photos, the types of backgrounds, and the number of training photos per user. Both random and optimal attacks were studied. The default data set, injection rate, and number of photos per user throughout the experiment was the LFW data set, 50%, and 10, respectively. In the KNN-based classifier, k is set to 5.

3.3.2.1 Effect of the Number of Injected Photos

In the first round of experiments, the number of injected photos is varied from 1 to 5, among 10 training photos/users in the LFW dataset. It is worth noting that injecting more photos (beyond five) will start decreasing the overall facial recognition accuracy as shown in Section 3.1, which will raise the alarm to the service provider. The accuracy, precision, recall, and F1 of our discriminators is measured using 5 testing photos per type of user, i.e., targeted victims, attackers, and clean users. Both the targeted victims and attackers' photos will be labeled as injected. Figure 3.7 shows the detection results after the random attack, and Figure 3.8 shows the results after the optimal attack.

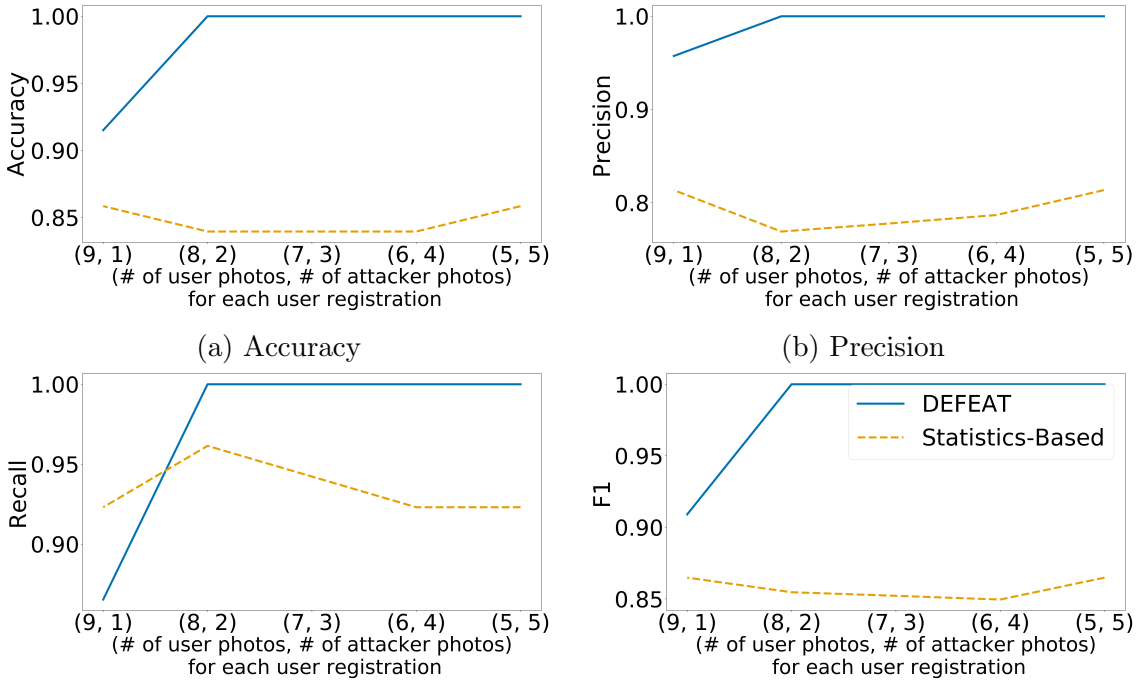
Under both attack strategies, DEFEAT maintains above 90% accuracy in all the scenarios and generally outperforms the statistic-based discriminator in all measurements. Most importantly, when the number of injected photos is close to half of the



(a) Accuracy (b) Precision

(c) Recall (d) F1

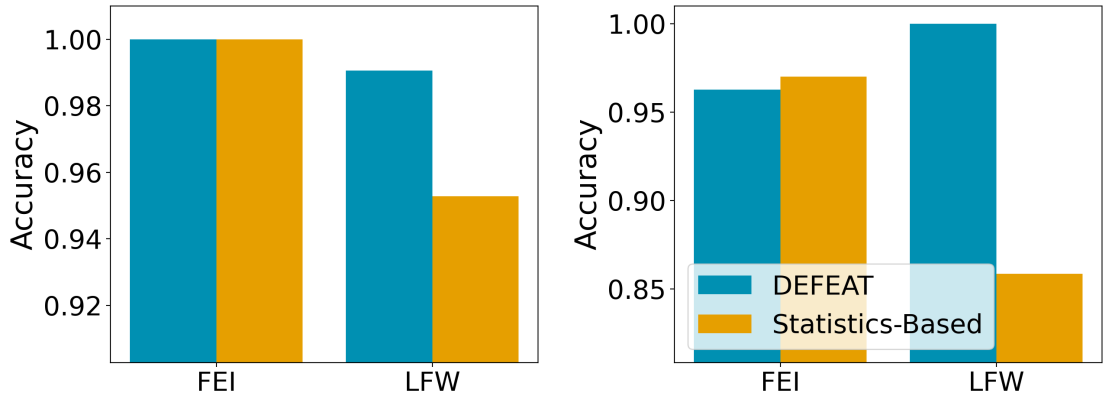
Figure 3.7: Random Attack - Varying the Number of Injected Photos



(a) Accuracy (b) Precision

(c) Recall (d) F1

Figure 3.8: Optimal Attack - Varying the Number of Injected Photos



(a) Random Attack (b) Optimal Attack

Figure 3.9: Effect of Photo Backgrounds

training photos, DEFEAT achieves more than 99% detection accuracy. As shown in Section 3.1, attackers need to replace nearly 50% of photos of the victim in order not to decrease the face recognition system's accuracy and arouse alarms. That means that, when the attacker tries to avoid affecting the overall facial recognition accuracy by injecting more photos, it also causes the DEFEAT system to be more accurate in detecting the attack. The performance of the DEFEAT system should be attributed to the DNN which intelligently classifies the different features among injected and clean photos. The statistic measurements help classification but are less effective than the DNN, especially under the optimal attack when the attacker's photos look similar to the victim's photos.

3.3.2.2 Effect of the Photo Background

The second round of experiments evaluates the impact of the photo backgrounds on the effectiveness of both discriminators. Both the FEI and LFW data sets were used. The FEI data set contains photos with relatively consistent backgrounds, representing an easy setting of facial recognition. Alternatively, the LFW data set contains photos with various backgrounds, representing a difficult, more realistic setting for facial recognition, such as when a user may log onto the web service from different devices

and in different places.

Figure 3.9 shows the overall accuracy of the statistic-based discriminator and DEFEAT discriminator in the two data sets under the random and optimal attacks, respectively. When a random attack is conducted, both discriminators can correctly detect the attack 100% of the time on the FEI data set. This is because the internal differences inside a label's cluster (i.e., the photos with the same label) is enough to be a differentiating factor. As both of the statistics-based discriminator and DEFEAT utilize this factor, they both achieve high accuracy.

When it comes to the complex photo backgrounds featured in the LFW data set, the statistic-based discriminator falls short while the DEFEAT discriminator still maintains high accuracy. This is because the complex backgrounds have likely lead to the feature vectors with more complex meanings which are hard to be fully captured by simple statistics such internal and external distances. DEFEAT takes advantage of both statistic measures and the outstanding classification ability of DNN on complex feature vectors, being capable of distinguishing infected photos even under a variety of background settings.

3.3.3 Comparison of On-Site and Off-Site Deployment

As mentioned in Section 3.2.1, there are two possible ways to deploy the proposed DEFEAT system: the on-site deployment and the off-site deployment methods. This section provides a comparison of these two types of deployments. Two computers are used to simulate the web service provider and the security provider, respectively. For the on-site deployment, FaceNet and DEFEAT are installed on the same computer, whereas, for the off-site deployment, FaceNet and DEFEAT are installed in separate computers, whereby the feature vectors generated by FaceNet in one computer will be sent to DEFEAT in the other computer to conduct the attack detection. Once the detection is completed, the detection result is sent back to the first computer

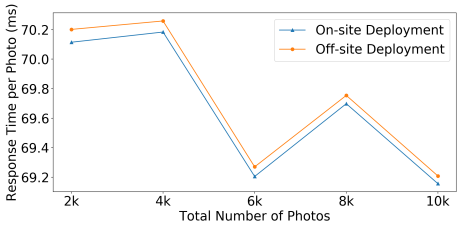


Figure 3.10: Comparison of On-site and Off-site Deployment

that mimics the web service provider. In both types of deployments, the number of photos received by the service provider is varied from 2,000 to 10,000. Note that only a single thread is used in this test. The number of photos can be easily scaled up when multiple server nodes are adopted since they can use the same detection model for authentication and attack detection after the training is completed.

Figure 3.10 reports the average response time per face authentication, i.e., photo validation. As expected, the off-site deployment requires a longer response time due to the network delay caused by the transmission of the feature vectors and decisions between the web service provider and the security provider. However, the network delay adds only 0.1% more response time compared to that of the on-site deployment. The main reason is that the sizes of feature vectors and detection results are only a few bytes per user. We also observe that the response time per photo stays relatively constant when the total number of photos increases. Note that the seemingly large fluctuation of the curve is due to the zoom-in effect used to show the slight gap between the two deployment methods. The average response time in different sizes of data sets is around 70ms, because the size of the DEFEAT model is not determined by the total number of photos. Hence, the individual photo validation time is not affected by the data size. The result also demonstrates the potential scalability of this approach.

Chapter 4

DETECTING TARGETED DATA POISONING ATTACKS ON DNN-BASED DEEPFAKE DETECTORS

In this chapter, a discussion on the security of deepfake detectors is provided, particularly in the context of label-flipping data poisoning attacks on a state-of-the-art deep neural network-based detectors. Also proposed is a detection network that works in conjunction with existing deepfake detectors that will determine whether such an attack is in progress. The layout of this chapter is as follows: Sections 4.1 and 4.2 describe and evaluate the threat of data poisoning attacks on deepfake detectors, then, in Section 4.3, the proposed defense system, Protect, is introduced. Section 4.4 provides a discussion of the evaluation methods and results.

4.1 THREAT MODEL

This work is focused on a targeted data poisoning attack on a state-of-the-art DNN-based deepfake detector. We consider the following typical deployment of deepfake detectors: first, the deepfake detector is trained in-house using existing fake image samples generated by known sources; then, the trained deepfake detector is launched in the field to screen online images and detect fake images for users. As new types of fake images are always emerging just like new types of malware is being reported everyday, the deepfake detector may need to be re-trained on the new image set collected during the service term to improve its detection accuracy. Without new

		Dataset Tested				
		DeepFakes	FaceSwap	Face2Face	Neural Textures	FaceShifter
Dataset Trained	DeepFakes	.995214	.507754	.548897	.621086	.501115
	FaceSwap	.968438	.994636	.641251	.563249	.503176
	Face2Face	.993586	.506502	.993720	.765992	.502110
	Neural Textures	.994036	.511141	.824165	994196	.576536
	FaceShifter	.843887	.524049	.625270	.747984	.999397

Table 4.1: Resulting accuracy upon training XceptionNet-based models for deepfake detection on five types of deepfakes, and testing on each type of deepfake

training, the detector that can recognize one type of fake images is not effective in detecting other types of fake images on which it has not been trained. The experimental results in Table 4.1 demonstrates such a scenario. Specifically, the left column of the table lists the types of fake images that are used for training, and the header column lists the types of fake images that the detector was tested on. As shown in the table, the detection accuracy of the example detector, XceptionNet, is highest only when it is tested against the same type of fake images that has already been learned through training. For other fake image types that the detector has not seen before, the accuracy can be as low as 50%- close to random guesses.

The retraining process would be similar to the current spam email reporting system whereby users can provide feedback to the detector by labeling images as fake or real when they think the detector has come to a erroneous conclusion. If some of the newly collected images are mislabeled, especially if some fake images are reported as real by a malicious user (an attacker), training on such a poisoned image set will mislead the deepfake detector to allow fake images to circulate online. Specifically, the attacker may want to always report fake images containing a target victim's face as real to the deepfake detector so that the deepfake detector will eventually deem any fake images about the victim as real and let this fake information about the victim

to spread. The goal of this work is thus to provide a method to help the deepfake detector distinguish between clean and poisoned training image samples to prevent it from learning unintended behavior.

According to the above threat model, the data poisoning attack in the real world is formalized as follows: Let D_{new} denote the set of newly-collected training images for a deepfake detector, that is, image samples collected and labeled as per user feedback. Since D_{new} contains images which are labeled as "real" or "fake" according to the user feedbacks, considering the possible existence of attackers, some images in D_{new} may have incorrect (poisoned) labels. Let t denote the face of a targeted victim, the attackers may intentionally label the fake images of the victim t (deepfakes targeted the victim) as "real" or "unaltered" in order to mislead the deepfake detector and spread fake information about the victim. In another case, an attacker may change the label of real images of a victim to "fake", which, however, would have much less impact than the previous case since that only causes some service disruption (i.e., not able to upload images) for the victim, as opposed to potentially devastating damages to the victim's reputation.

4.2 ATTACK RESULTS

To verify the effectiveness of the aforementioned targeted data poisoning attack, we have conducted the following experiments. We select the state-of-the-art deepfake detector [39], based on XceptionNet [40]. Then, we examine the its ability to detect new types of fake images that it has not been trained on. We train the detector using the DeepFakes dataset [30] of 10,000 fake images, and test the detectors on the FaceSwap dataset [31]. As shown in Table 4.1, the detector achieves high detection accuracy as expected on the fake images is has been trained on. However, when it comes to the new type of fake images, the accuracy drops significantly to 50.78%.

After the detector has been re-trained on the new type of fake images, from the

FaceSwap dataset [31], the accuracy improves again, up to 97.48%. We see similar results for re-training the deepfake detector on other types of fake images. This phenomenon indicates an ongoing need of retraining these deepfake detectors often to recognize the emerging types of fake images.

We then create a poisoned training dataset based on the existing deepfake training set [30]. We denote the original training dataset as D_o, which contains 10,000 fake images and 30,000 real images belonging to 1,000 users. We synthesize the poisoned training dataset by randomly selecting a small set of target faces (denoted as T) and flip the labels of the fake images that contain the faces in T to "real". Then, we train the XceptionNet detector using the poisoned training datasets, each with 10,000 images. During the testing, we use another image set that contains new fake images of the faces in the targeted victim set T. A data poisoning attack is considered successful if it meets the following two conditions: (i) the recall rate of detecting correctly-labeled fake images and real images (Equations 4.1 and 4.2) remains high as if not being attacked; (ii) fake images containing the targeted victims are recognized as "real" as measured by the wrong recognition rate (Equation 4.3).

$$RecallReal = \frac{\#\ \ correctly\ \ identified\ \ real\ \ images}{Total\ \ \#\ \ of\ \ real\ \ images} \tag{4.1}$$

$$RecallFake = \frac{\#\ \ correctly\ \ identified\ \ fake\ \ images}{Total\ \ \#\ \ of\ \ fake\ \ images} \tag{4.2}$$

$$AttackSuccess = \frac{\#\ \ of\ \ victim's\ \ fake\ \ images\ \ not\ \ recognized}{Total\ \ \#\ \ of\ \ victim's\ \ fake\ \ images} \tag{4.3}$$

To test this type of data poisoning attack, XceptionNet was trained on each of the data sets described in Section 4.4.1. To produce an average result, XceptionNet was trained seven times, each time with a different subset of incorrect labels in the

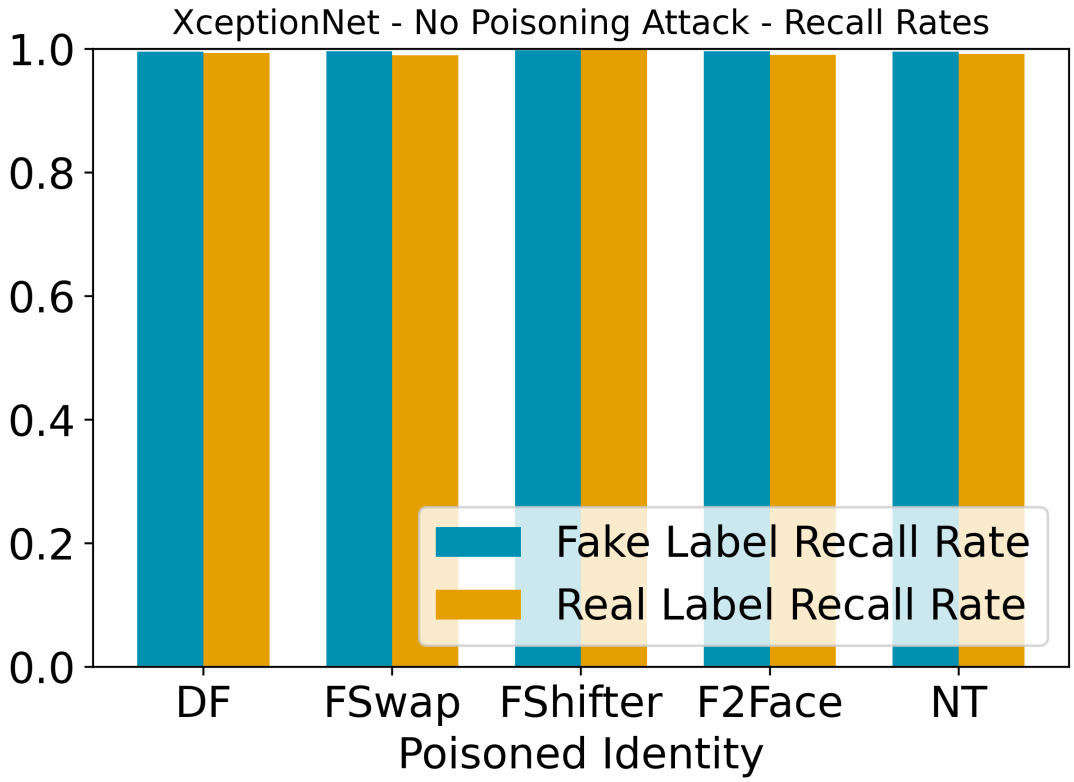
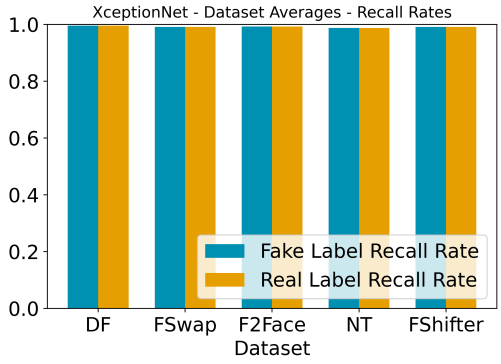


Figure 4.1: Recall rates of XceptionNet trained on clean data sets

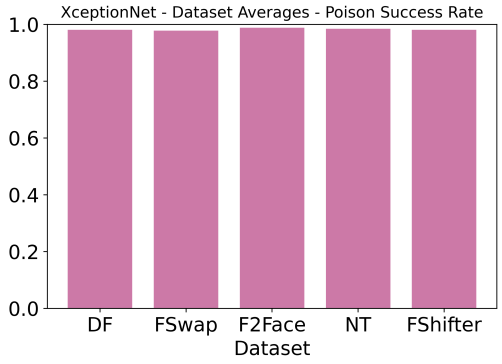
training data set. For comparison, the results of training XceptionNet on clean data sets are shown in Figure 4.1.

Figure 4.2 shows the experimental results that indicate the success of the data poisoning attack on XceptionNet. Specifically, these results show both recall rates of real and fake images (Equations 4.1 and 4.2) remain above 98%, even with a poisoned training data set, while maintaining an attack success rate (corresponding to Equation 4.3) of above 97.8%.

In addition, the effect of varying the rate of poisoning was also explored. In the experiments, XceptionNet was trained on the FaceShifter data set with rates of poisoning varying between 5% and 25%, at intervals of 5%, corresponding to 50, 100, 150, 200, and 250 identities. Results are shown in Figure 4.3. It is shown that, even with a 5% poison rate, the attack is extremely successful, maintaining a high

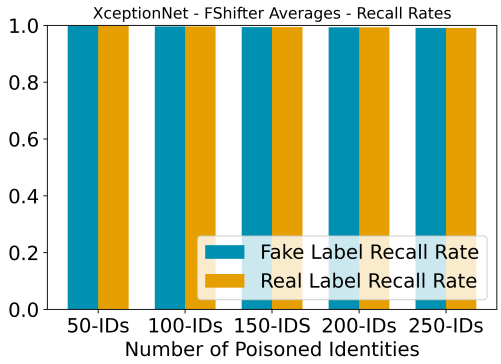


(a) Deepfake detection recall rate of both labels

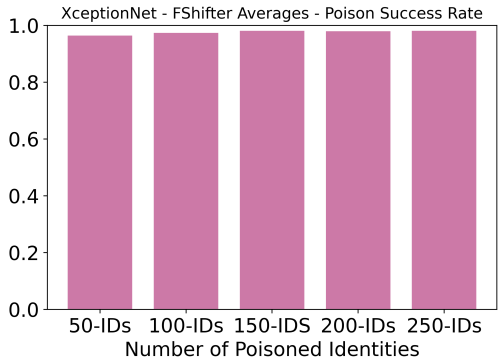


(b) Incorrect acceptance of fake images by XceptionNet

Figure 4.2: Targeted data poisoning attack success rates on XceptionNet



(a) Deepfake detection recall rate of both labels



(b) Incorrect acceptance of fake images by XceptionNet

Figure 4.3: Targeted data poisoning attack success rates on XceptionNet at different poison rates

recall rate for each non-targeted identity and achieving an attack success rate above 95%. This suggests that, even when a deepfake detector is trained on many different kinds of deepfake images, it is still vulnerable to the targeted data poisoning attack described in Section 4.1.

4.3 THE PROPOSED PROTECTOR NETWORK

4.3.1 Model Architecture

As the deepfake detector will need to be updated periodically to keep up with the newly emerging types of fake images that it has not seen, the risk of collecting poisoned training dataset from user reported data is inevitable. Our defensive method is to build a protector model that can help distinguish poisoned fake images and non-poisoned fake images regardless of the type of fake images. The protector model takes the feature vectors generated by the deepfake detector as input and outputs two labels: poisoned and non-poisoned. The ultimate goal of the protector is to identify the fundamental differences between the feature vectors of poisoned images and non-poisoned images. The rationale behind this is that images with incorrect labels will activate different portions of the layers in the deepfake detector so as to reach the wrong label compared to the non-poisoned images. As a result, the obtained feature vectors from the poisoned images are likely from a different latent space which may be able to be captured by another deep neural network structure, i.e., the protector. We suspect the underlying differences between the poisoned and non-poisoned feature vectors would share common patterns irrelevant to the the actual fake image types, and thus, the protector would still function for new types of fake image types that it does not seen. Also, if our hypothesis holds, our protector network will not need to be retrained on the field, which prevents it from being poisoned by newly collected data.

Towards the above goal, we design our protector model as shown in Figure 4.4.

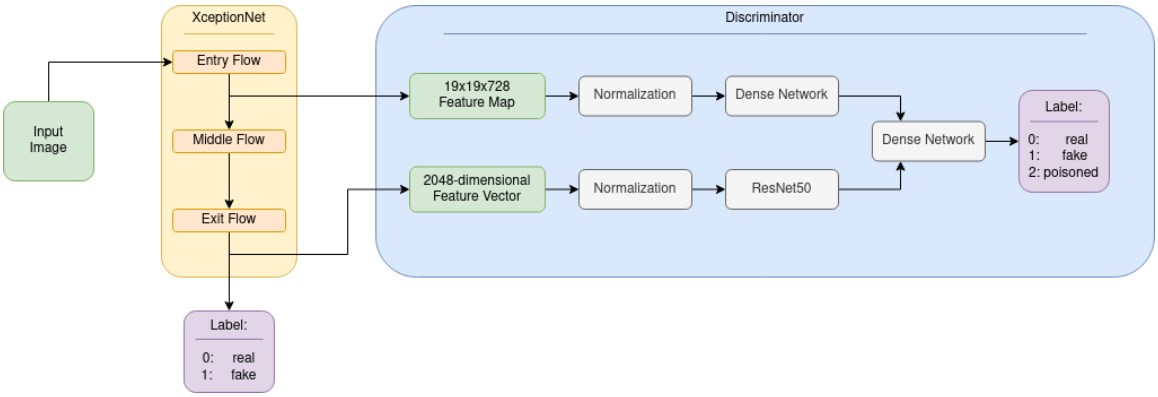


Figure 4.4: Overview of Protector model

The protector takes two kinds of inputs, i.e., a low-level feature representation and a high level feature representation from the deepfake detector. In particular, for XceptionNet, we take the feature vectors from the end of the entry flow and the end of the exit flow. Both the low-level and high-level feature vectors will be normalized through the batch normalization process. After that, the lower-level feature vector will be fed into a dense network which consists of a fully connected layer with 512 nodes. The high-level feature vector will be fed into the image recognition residual neural network, ResNet50 [54, 79] for analysis. Then, the output from both the dense network and ResNet50 will be combined and sent to another dense network that contains 1 layer, also containing 512 nodes. Finally, The output from the last fully connected layer will be integrated using the softmax function and produce one of the three labels: real, fake, poisoned.

4.3.2 Data Set Preparation

To train the Protector network, we mimic the data poisoning attacks on data sets of various types of fake images as follows: We start with the FaceForensics data set [39] that contains 1,000 youtube videos of people and a deepfake constructed from each original video using Faceswap [31], Face2Face [32], DeepFakes [30, 80], FaceShifter [34] and NeuralTexture [33], resulting in a total of 6,000 videos. Each of these data

sets is described in further detail in Section 4.4.1.

For each of the fake data sets, 25% of the fake videos' labels were changed to "real". Then, each data set of fake videos is combined with the corresponding data set of 1,000 real videos, resulting in a data set of 2,000 videos, with half being fake videos using a specific deepfake method, and the other half being real videos. Then, using a random seed, 25% of the videos were picked to be poisoned. Poisoned videos have their labels changed from fake to true. Frames of the videos containing faces are used as the deepfake images.

We do this with 5 different random seeds. At this point there are 25 data sets, each data set containing 750 fake videos and 1250 real videos, 250 of with are poisoned videos. For each video in the new dataset, the first 370 frames that have an identifiable person are saved. The first 70% of the frames are used for the training set, the next 10% are used for the validation set, and the final 20% are used for the test set. The frames are cropped around the face using the dlib package's get_frontal_face_detection. Then it is resized to size 299x299x3 using bilinear interpolation. At the end, we have mixed image datasets that contain half fake images and half real images; among the half fake images, some of them are poisoned, i.e., fake images that are intentionally labeled as real images.

In the above training data sets, the percentage of poisoned fake images is typically very low. If we feed such a data set to the deepfake detector, and train the protector using these feature vectors, the protector will learn mostly the correctly labeled real and fake images but very little about the poisoned fake images. The imbalanced training data sets will result in low detection accuracy of the poisoned images. In order to resolve this issue, we propose the following method to collect a balanced training data set for the protector model. First, we create k training data sets instead of one as described in the previous paragraph. Each data set contains n images including 50% real images, β% fake images and ϵ% poisoned fake images ($\beta\% + \epsilon\% = 50\%$).

Then we train k deepfake detector models, each of which takes one of the above k data sets as input. From the output of each model, we randomly select feature vectors of $\frac{n}{3k}$ real images, $\frac{n}{3k}$ correctly labeled fake images, and $\frac{n}{3k}$ poisoned fake images. By assembling the selected feature vectors from all the k deepfake detector models, we obtain the final training data set for the protector model. The final training data set contains a balanced split of each type of images, i.e., real, fake and poisoned images' feature vectors each occupy $\frac{1}{3}$ of the training data set.

We prepared two kinds of test data sets. One kind of data set is the data set that contains the same type of fake images used for training the protector model. The other kind of data set is the data set that contains the type of fake images which have not been seen by the protector model. The percentage of real, fake and poisoned images in the test data set is also balanced, i.e., $\frac{1}{3}$ of each type.

4.3.3 Feature Vector Concatenation

Throughout the experiments, further described in Section 4.4, we experiment with concatenating two feature vectors from XceptionNet's exit flow: the feature vector corresponding to the identity in question, and a known fake feature vector corresponding to that identity, generated with one of the deepfake generation methods described in Section 4.4.1. This results in a 4096-dimensional input to the dense network, as opposed to the 2048-dimensional input referenced in Figure 4.4. The idea behind this method is that, comparing a facial feature vector with a known fake feature vector, it may be possible to determine whether the label of the image in question (*real* or *fake*) is erroneous. Experiments involved combined feature vectors are described in Section 4.4.4

4.3.4 Model Training and Testing

As the deepfake detector needs to be improved over time to recognize new types of fake images, the deepfake detector will be trained both in-house and on the field, while the protector model will only need to be trained only initially, that is, before deployment.

The initial training starts from the deepfake detector model. As described in the previous section, we take k training data sets and train k deepfake detector models respectively to produce the training data set for the protector model. The inputs to the protector model are first batch normalized. The goal of the batch normalization is to shift the starting data into a range that is easier to train with the standard initial weights and introduce the potential to remove the differences in mean and variance in the features from the different detectors. Batch normalization is implemented as follows.

Define input x, output y, trainable variables γ and β, and hyperparameters $\epsilon = 0.001$ and $m = 0.99$. During the training the output is calculated as follows.

$$y = \gamma \cdot (x - \text{mean}(x))/\sqrt{\text{variance}(x) + \epsilon} + \beta \tag{4.4}$$

During testing a running mean (μ) and running variance (σ^2) is used, for in practice, one might not have a full batch of data.

$$y = \gamma \cdot (x - \mu)/\sqrt{\sigma^2 + \epsilon} + \beta \tag{4.5}$$

The running mean and variance are calculated during testing using equations 4.3.4

and 4.3.4.

$$\mu = \mu \cdot m + \text{mean}(x) \cdot (1 - m) \tag{4.6}$$

$$\sigma^2 = \sigma^2 \cdot m + \text{variance}(x) \cdot (1 - m) \tag{4.7}$$

We conduct the batch normalization on both the low-level feature representations and the high-level feature representations. For the XceptionNet, the low-level feature representation is extracted from layer 36 and is of shape (756), and the high-level feature vector is extracted from the penultimate (132^th) layer and consists of 1024 dimensions.

After batch normalization, the low-level feature vectors are fed into 1 dense layer containing 512 nodes (with no activation function). The learning rate was set to 10^{-6}, and the learning algorithm is ADAM. The high-level feature vectors are fed into a ResNet50 model. Finally, we add the 512 dimension output from the initial dense layer and 512 dimension output from the ResNet model and put it through a dense layer to get a 3 class output. The last layer uses the softmax function to produce the following vector for each image: [probabality image is True, probability image if fake, probability image is poisoned].

The loss function is the sparse categorical loss function defined as follows

$$\frac{1}{n} \sum_{n} \sum_{c} y_{c,true} \log(y_{c,predicted}) \tag{4.8}$$

where y_{c_true} is the label for class c (1 if it belongs to class c and 0 otherwise) and $y_{c,predicted}$ is the probability of the image belonging to class c, output by the model and normalized so that $\sum_c y_{c,predicted} = 1$ (to make it a true probability distribution). Thanks to the behavior of the softmax function, the output of the model should already be normalized.

The training is conducted until the training accuracy is above 99.99%. This can take as few as 2 epochs to as many as 12 epochs.

The effectiveness of the protector model is challenged when the deepfake detector is deployed on the field and being retrained on newly collected data. By assuming all the collected data is correctly labeled, the deepfake detector is trained on these new image and label pairs until its detection accuracy rises above 95%. Then, the protector model will examine both the low-level and high-level feature vectors of these new image collections output by the deepfake detector to check if there are any potentially poisoned data. The low-level and high-level feature vectors will be normalized and fed to the protector model. If the protector model deems an image as poisoned, an alert will be sent to the service provider for the further investigation.

4.4 EVALUATION

4.4.1 Data Sets

The data sets used for evaluation of the proposed system include FaceForensics++ [39] and FaceShifter [34]. The FaceForensics++ data set [39] is composed of 1,000 unaltered short videos on YouTube [81], as well as 1,000 videos altered by each of four deepfake generation methods: DeepFakes [30] [80], FaceSwap [31], Face2Face [32], and Neural Textures [33]. A total of 5,000 videos are in this data set. The FaceShifter data set [34] is also composed of 1,000 unaltered YouTube videos, with an accompanying 1,000 fake videos, generated using a deepfake generation method called FaceShifter [34]. Below are brief desciptions of each method.

DeepFakes (DF) For this deepfake generation method, the implementation found on the *faceswap github* [30] was used. These deepfakes follow a classic method: at each frame of each pair of input videos (source and target videos), the face was located,

cropped, and aligned using a MTCNN-based face detector, as described in [82]. Then, using a dual autoencoder system, the source face is applied to the target face, then being blended using Poisson image editing [83].

FaceSwap (FSwap) FaceSwap [31], unlike DeepFakes, is based on graphics, as opposed to DNNs, for swapping a source and target face. More specifically, it detects facial landmarks and fits them to a template 3D model. From there, the source face is applied to the target face by fitting the 3D source face to the target facial image, using the original textures. Finally, color-correction and blending are applied to the resulting image. While generally less visually realistic than the DeepFakes generation method [30, 80], this method is more computationally lightweight [31].

Face2Face (F2Face) This deepfake generation method seeks to swap facial expressions from a source video to a target video, while preserving the identity of the target individual. This is referred to as "facial reenactment" and is done through manual selection of key frames, which serve as representatives of each face, allowing facial expressions, poses, and lighting to be processed. To generate the final images, expression data from the source face is applied to the face in the target video [32, 39].

NeuralTextures (NT) NeuralTextures [33], similar to Face2Face [32], is a method of facial reenactment, involving learning textures of the target as 3D mesh objects, then training a rendering network with adversarial loss [84] and photometric reconstruction loss [85]. The implementation in FaceForensics++ only reconstructed the mouth region of the target image [39].

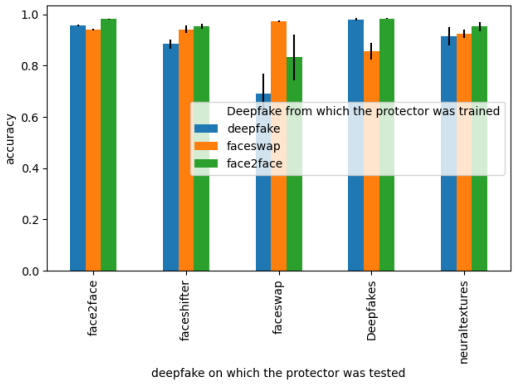
FaceShifter (FShifter) FaceShifter [34], unlike the other methods, is a GAN-based identity-swapping network and is composed of two networks: an Adaptive Embedding Integration Network (AEI-Net) and a Heuristic Error Acknowledging Refinement Network (HEAR-Net). The former extracts the facial identity from the source, as well as pose, expression, and lighting data from the target. Then, an Adaptive Attentional Denormalization (AAD) generator integrates both sets of attributes into a single swapped face. Afterwards, the HEAR-Net ensures that facial obstructions (such as glasses, hair, etc.) are preserved by comparing the reconstructed image with the original image.

4.4.2 Experiments

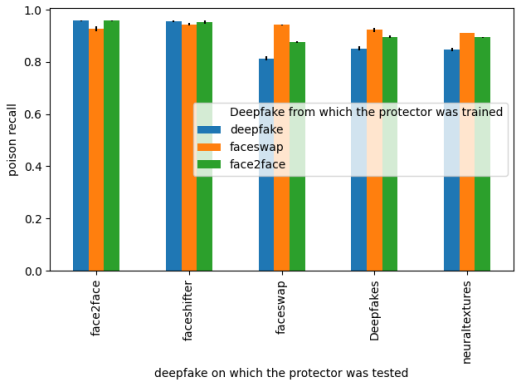
Several experiments were conducted, evaluating the protector network's ability to identify poisoned images while maintaining reasonably high accuracy identifying true real and true fake images. In Section, 4.4.3, we explore the effect of training on different types of fake images then testing on types of fake images that the network has yet to learn. Then, as described in Section 4.4.4, we explore the effects of training a ResNet50 model, concatenating feature vectors as described in Section 4.3.3, also on various different types of fake images. This model is then used as a basis for comparison with the protector network in Section 4.4.5.

4.4.3 Effect of Training Protector on Different Types of Fake Images

Figure 4.5 shows the effect of training the protector network on different types of fake images: DeepFakes [30], FaceSwap [31], and Face2Face [32]. In general, training the protector network on the FaceSwap dataset leads to the best results, with each training dataset showing a high recall for poisoned labels. However, when trained on the DeepFakes dataset and tested on FaceSwap, we observe a dip in accuracy when identifying the label of the particular deepfake image. In conjunction with the high



(a) Accuracy of the protector network having been trained on various types of fake images



(b) Poison recall of the protector network having been trained on various types of fake images

Figure 4.5: Effects of training the protector network on different kinds of fake images

recall rate for the poisoned image, this can be interpreted as the protector network potentially showing difficulties in identifying images that are real, which may be remedied with additional fine-tuning of the parameters used in the training process.

4.4.4 Effect of Combining Feature Vectors

This experiment studies the effect of concatenating two feature vectors, from the exit flow of the target deepfake detector of the same identity in order to determine, from analyzing the differences in the feature space and the given label of the unknown feature vector, whether this label is likely to be incorrect. Tables 4.2 and 4.3 show the accuracy and poison recall respectively of the ResNet model having been trained on these combined feature vectors, as described in 4.3.3. It is apparent that this

	DeepFakes	Face2Face	FaceSwap	FaceSwap & Face2Face
DeepFakes	0.9873	0.9960	0.9959	0.9957
Face2Face	0.9589	0.9802	0.9631	0.9707
FaceSwap	0.9304	0.9377	0.9417	0.9486

Table 4.2: Accuracy achieved by ResNet50 trained on combined feature vectors. The top row shows the datasets on which the model was trained, whereas the left column shows the datasets on which it was tested.

	DeepFakes	Face2Face	FaceSwap	FaceSwap & Face2Face
DeepFakes	0.9814	0.9953	0.9965	0.9953
Face2Face	0.9929	0.5543	0.9991	0.9970
FaceSwap	0.9090	0.8712	0.9507	0.9364

Table 4.3: Poison recall achieved by ResNet50 trained on combined feature vectors. The top row shows the datasets on which the model was trained, whereas the left column shows the datasets on which it was tested.

is an effective method of detecting the described targeted data poisoning attack on deepfake detectors in most cases. However, this method assumes that the discriminator model has access to known-correct labels and corresponding feature vectors of every identity that may be in question. This may not be feasible in every case of deepfake detection, as, for example, there may not be many high-quality images (eventually feature vectors) with labels known with 100% certainty of certain people. Therefore, this method will be used as a high-quality comparison for the protector network, which does not require access to known-correct labels after its deployment, making it more convenient and feasible in real-world scenarios.

4.4.5 Effect of Model Architecture

This experiment explores the effect of the protector model's architecture, which uses feature vectors from both the entry and exit flows of the deepfake detection model, as opposed to only using the exit flow in the decision. The model using only the exit flow makes use of the feature vector concatenation, and is described in the previous section. For convenience, the results of the latter model are reiterated in the Table 4.4 below. As is shown, the protector network performs similarly well as the ResNet detector using feature concatenation. This can be explained such that the use of the entry flow and exit flow feature vectors in conjunction provide more information than only analyzing the exit flow feature vector, only derived from the entry flow vector. Since the targeted deepfake detector uses only the exit flow feature vector to

	Protector		ResNet50	
	Accuracy	Poison recall	Accuracy	Poison recall
face2face	0.981407	0.954972	0.9631	0.9991
faceswap	0.977187	0.932849	0.9417	0.9507
Deepfakes	0.942813	0.919132	0.9959	0.9965

Table 4.4: Accuracy and poison recall ResNet50 and Protector network trained on FaceSwap datasets

make its final decision for the image, this addition of the protector network adding the additional analysis portion enhances its robustness to targeted data poisoning attacks, without requiring an additional image (and corresponding feature vector), as the ResNet discriminator does.

Chapter 5

CONCLUSION

With the increasingly widespread use of DNNs, comes high motivation for attackers to develop novel ways in causing malicious behavior in such systems. This report explored the potentially devastating outcomes of allowing DNNs to remain vulnerable to targeted data poisoning attacks, specifically in the contexts of facial authentication and deepfake detection.

For facial authentication, it was demonstrated to be nearly trivially easy to cause unintended behavior in the state-of-the-art facial authentication system, using the replacement data poisoning attack on the targeted facial authentication user to gain access to their potential account.

Then, to combat such a powerful, yet simple, attack, a DNN-based discriminator, DEFEAT, was proposed that achieved over 90% accuracy in detecting when such an attack is occurring. Upon comparison to other possible detectors, DEFEAT performed favorably.

In the context of deepfake detection, we explored a label-flipping targeted data poisoning attack avenue paralleling the attacks that have previously been seen in the context of spam-recognition. This type of attack was applied and shown to be effective in the context of identification of deepfakes using a state-of-the-art deepfake detection model.

A remedy was then proposed based on the analysis of the resulting label output

by the deepfake detector, comparing it to the facial feature vector examined by the deepfake detector. It was shown to be effective in the identification of poisoned ("flipped") labels, even for deepfakes generated using a method not seen at any point during its training.

BIBLIOGRAPHY

[1] D. Lin, N. Hilbert, C. Storer, W. Jiang, and J. Fan. "UFace: Your universal password that no one can see". In: *Computers & Security* 77 (2018), pp. 627–641. ISSN: 0167-4048. URL: http://www.sciencedirect.com/science/article/pii/S0167404817302067.

[2] A. Tagat. *Online fraud: too many accounts, too few passwords*. TechRadar. July 2012. URL: http://www.techradar.com/us/news/internet/online-fraud-too-many-accounts-too-few-passwords-1089283.

[3] URL: https://facex.io/.

[4] A. Walling. *Top 10 Facial Recognition APIs & Software of 2020*. URL: https://rapidapi.com/blog/top-facial-recognition-apis/.

[5] F. Schroff, D. Kalenichenko, and J. Philbin. "Facenet: A unified embedding for face recognition and clustering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.

[6] I. Masi, Y. Wu, T. Hassner, and P. Natarajan. "Deep Face Recognition: A Survey". In: *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. 2018, pp. 471–478.

[7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. "Intriguing properties of neural networks". In: *CoRR* abs/1312.6199 (2014).

[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". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. Vienna, Austria: Association for Computing Machinery, 2016, pp. 1528–1540. ISBN: 9781450341394. URL: https://doi.org/10.1145/2976749.2978392.

[9] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. "DeepFool: a simple and accurate method to fool deep neural networks". In: *CoRR* abs/1511.04599 (2015). arXiv: 1511.04599. URL: http://arxiv.org/abs/1511.04599.

[10] A. J. Bose and P. Aarabi. "Adversarial attacks on face detectors using neural net based constrained optimization". In: *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*. IEEE. 2018, pp. 1–6.

[11] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff. "On Detecting Adversarial Perturbations". In: *ArXiv* abs/1702.04267 (2017).

[12] W. Xu, D. Evans, and Y. Qi. "Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks". In: *CoRR* abs/1704.01155 (2017). arXiv: 1704.01155. URL: http://arxiv.org/abs/1704.01155.

[13] G. Cohen, G. Sapiro, and R. Giryes. "Detecting adversarial samples using influence functions and nearest neighbors". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 14453–14462.

[14] D. Meng and H. Chen. "Magnet: a two-pronged defense against adversarial examples". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 135–147.

[15] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia. "Exploiting Machine Learning to Subvert

Your Spam Filter". In: *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*. 2008.

[16] D. Lowd and C. Meek. "Adversarial Learning". In: *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. 2005, pp. 641–647.

[17] G. Wittel and S. Wu. "On Attacking Statistical Spam Filters." In: Jan. 2004.

[18] B. Biggio, B. Nelson, and P. Laskov. "Poisoning Attacks against Support Vector Machines". In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*. Edinburgh, Scotland, 2012, pp. 1467–1474.

[19] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli. "Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization". In: *CoRR* abs/1708.08689 (2017). arXiv: 1708.08689. URL: http://arxiv.org/abs/1708.08689.

[20] B. Wang and N. Z. Gong. "Stealing Hyperparameters in Machine Learning". In: *2018 IEEE Symposium on Security and Privacy (SP)*. 2018, pp. 36–52.

[21] T. Orekondy, B. Schiele, and M. Fritz. "Knockoff Nets: Stealing Functionality of Black-Box Models". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[22] T. Lee, B. Edwards, I. Molloy, and D. Su. "Defending Against Model Stealing Attacks Using Deceptive Perturbations". In: *ArXiv* abs/1806.00054 (2018).

[23] M. Juuti, S. Szyller, S. Marchal, and N. Asokan. "PRADA: Protecting Against DNN Model Stealing Attacks". In: *2019 IEEE European Symposium on Security and Privacy (EuroS P)*. 2019, pp. 512–527.

[24] A. Paudice, L. Muñoz-González, A. Gyorgy, and E. C. Lupu. "Detection of adversarial training examples in poisoning attacks through anomaly detection". In: *arXiv preprint arXiv:1802.03041* (2018).

[25] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao. "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks". In: *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2019, pp. 707–723.

[26] G. Goswami, N. Ratha, A. Agarwal, R. Singh, and M. Vatsa. "Unravelling robustness of deep learning based face recognition against adversarial attacks". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[27] T. Seals. "ASUS Home Router Bugs Open Consumers to Snooping Attacks". In: *ThreatPost* (2020). URL: https://threatpost.com/asus-home-router-bugs-snooping-attacks/157682/.

[28] L. Pascu. "Acronis reports critical flaws in GeoVision biometric devices, man-in-the-middle attack risks". In: *BiometricUpdate* (2020). URL: https://www.biometricupdate.com/202006/acronis-reports-critical-flaws-in-geovision-biometric-devices-man-in-the-middle-attack-risks.

[29] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song. "Robust physical-world attacks on deep learning visual classification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1625–1634.

[30] Feb. 2021. URL: https://faceswap.dev/.

[31] MarekKowalski. *MarekKowalski/FaceSwap*. URL: https://github.com/MarekKowalski/FaceSwap/.

[32] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner. "Face2face: Real-time face capture and reenactment of rgb videos". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2387–2395.

[33] J. Thies, M. Zollhöfer, and M. Nießner. "Deferred neural rendering: Image synthesis using neural textures". In: *ACM Transactions on Graphics (TOG)* 38.4 (2019), pp. 1–12.

[34] L. Li, J. Bao, H. Yang, D. Chen, and F. Wen. "Faceshifter: Towards high fidelity and occlusion aware face swapping". In: *arXiv preprint arXiv:1912.13457* (2019).

[35] R. Mitz. *The fight to stay ahead of deepfake videos before the 2020 US election*. URL: https://www.cnn.com/2019/06/12/tech/deepfake-2020-detection/index.html.

[36] J. Vincent. *Watch Jordan Peele use AI to make Barack Obama deliver a PSA about fake news*. Apr. 2018. URL: https://www.theverge.com/tldr/2018/4/17/17247334/ai-fake-news-video-barack-obama-jordan-peele-buzzfeed.

[37] O. Schwartz. "You thought fake news was bad? Deep fakes are where truth goes to die". In: *The Guardian* (Nov. 2018). URL: https://www.theguardian.com/technology/2018/nov/12/deep-fakes-fake-news-truth.

[38] A. Escalante. *Research Finds Social Media Users Are More Likely To Believe Fake News*. URL: https://www.forbes.com/sites/alisonescalante/2020/08/03/research-finds-social-media-users-are-more-likely-to-believe-fake-news/.

[39] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner. "FaceForensics++: Learning to Detect Manipulated Facial Images". In: *International Conference on Computer Vision (ICCV)*. 2019.

[40] F. Chollet. "Xception: Deep learning with depthwise separable convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1251–1258.

[41] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen. “Mesonet: a compact facial video forgery detection network”. In: *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE. 2018, pp. 1–7.

[42] D. Güera and E. J. Delp. “Deepfake video detection using recurrent neural networks”. In: *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE. 2018, pp. 1–6.

[43] E. Sabir, J. Cheng, A. Jaiswal, W. AbdAlmageed, I. Masi, and P. Natarajan. “Recurrent convolutional strategies for face manipulation detection in videos”. In: *Interfaces (GUI)* 3.1 (2019).

[44] H. H. Nguyen, J. Yamagishi, and I. Echizen. “Capsule-forensics: Using capsule networks to detect forged images and videos”. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 2307–2311.

[45] Y. Li and S. Lyu. “Exposing deepfake videos by detecting face warping artifacts”. In: *arXiv preprint arXiv:1811.00656* (2018).

[46] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros. “CNN-generated images are surprisingly easy to spot... for now”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Vol. 7. 2020.

[47] R. Tolosana, S. Romero-Tapiador, J. Fierrez, and R. Vera-Rodriguez. “DeepFakes Evolution: Analysis of Facial Regions and Fake Detection Performance”. In: *arXiv preprint arXiv:2004.07532* (2020).

[48] S. Hussain, P. Neekhara, M. Jere, F. Koushanfar, and J. McAuley. *Adversarial Deepfakes: Evaluating Vulnerability of Deepfake Detectors to Adversarial Examples*. 2020. arXiv: `2002.12749 [cs.CV]`.

[49] I. J. Goodfellow, J. Shlens, and C. Szegedy. “Explaining and Harnessing Adversarial Examples”. In: (2015).

[50] A. Kurakin, J. I. Goodfellow, and S. Bengio. "Adversarial Machine Learning at Scale". In: *international conference on learning representations* (2017).

[51] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *6th International Conference on Learning Representations*. 2018.

[52] Y. Wang, X. Ma, J. Bailey, J. Yi, B. Zhou, and Q. Gu. "On the Convergence and Robustness of Adversarial Training". In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. 2019, pp. 6586–6595.

[53] G. W. Ding, Y. Sharma, K. Y. C. Lui, and R. Huang. "MMA Training: Direct Input Space Margin Maximization through Adversarial Training". In: *International Conference on Learning Representations*. 2020.

[54] K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[55] A. Kurakin, I. J. Goodfellow, and S. Bengio. "Adversarial examples in the physical world". In: *CoRR* abs/1607.02533 (2016). arXiv: 1607.02533. URL: http://arxiv.org/abs/1607.02533.

[56] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. "The Limitations of Deep Learning in Adversarial Settings". In: *CoRR* abs/1511.07528 (2015). arXiv: 1511.07528. URL: http://arxiv.org/abs/1511.07528.

[57] J. Su, D. V. Vargas, and K. Sakurai. "One Pixel Attack for Fooling Deep Neural Networks". In: *IEEE Transactions on Evolutionary Computation* 23.5 (2019), pp. 828–841.

[58] C. Kanbak, S. Moosavi-Dezfooli, and P. Frossard. "Geometric Robustness of Deep Networks: Analysis and Improvement". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4441–4449.

[59] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. "Universal Adversarial Perturbations". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 86–94.

[60] S. Sarkar, A. Bansal, U. Mahbub, and R. Chellappa. "UPSET and ANGRI : Breaking High Performance Image Classifiers". In: *CoRR* abs/1707.01159 (2017). arXiv: 1707.01159. URL: http://arxiv.org/abs/1707.01159.

[61] M. M. Cisse, Y. Adi, N. Neverova, and J. Keshet. "Houdini: Fooling Deep Structured Visual and Speech Recognition Models with Adversarial Examples". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017, pp. 6977–6987. URL: https://proceedings.neurips.cc/paper/2017/file/d494020ff8ec181ef98ed97ac3f25453-Paper.pdf.

[62] S. Baluja and I. Fischer. *Learning to Attack: Adversarial Transformation Networks*. 2018. URL: https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16529.

[63] J. Hayes and G. Danezis. "Machine Learning as an Adversarial Service: Learning Black-Box Adversarial Examples". In: (Aug. 2017).

[64] K. R. Mopuri, U. Garg, and R. V. Babu. "Fast Feature Fool: A data independent approach to universal adversarial perturbations". In: *CoRR* abs/1707.05572 (2017). arXiv: 1707.05572. URL: http://arxiv.org/abs/1707.05572.

[65] K. Xu, G. Zhang, S. Liu, Q. Fan, M. Sun, H. Chen, P.-Y. Chen, Y. Wang, and X. Lin. "Evading Real-Time Person Detectors by Adversarial T-shirt". In: *CoRR* abs/1910.11099 (2019). arXiv: 1910.11099. URL: http://arxiv.org/abs/1910.11099.

[66] B. Nelson, M. Barreno, F. Jack Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia. "Misleading Learners: Co-opting Your Spam Filter". In: *Machine Learning in Cyber Trust: Security, Privacy, and Reliability*. Boston, MA: Springer US, 2009, pp. 17–51. ISBN: 978-0-387-88735-7. URL: https://doi.org/10.1007/978-0-387-88735-7_2.

[67] T. Gu, B. Dolan-Gavitt, and S. Garg. "Badnets: Identifying vulnerabilities in the machine learning model supply chain". In: *arXiv preprint arXiv:1708.06733* (2017).

[68] J. Clements and Y. Lao. "Backdoor attacks on neural network operations". In: *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE. 2018, pp. 1154–1158.

[69] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang. "Trojaning Attack on Neural Networks". In: *NDSS*. 2018.

[70] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal. "STRIP: A Defence against Trojan Attacks on Deep Neural Networks". In: *Proceedings of the 35th Annual Computer Security Applications Conference*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 113–125. ISBN: 9781450376280. URL: https://doi.org/10.1145/3359789.3359790.

[71] Y. Liu, Y. Xie, and A. Srivastava. "Neural Trojans". In: *CoRR* abs/1710.00942 (2017). arXiv: 1710.00942. URL: http://arxiv.org/abs/1710.00942.

[72] K. Liu, B. Dolan-Gavitt, and S. Garg. "Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks". In: *Research in Attacks, Intrusions, and Defenses*. Ed. by M. Bailey, T. Holz, M. Stamatogiannakis, and S. Ioannidis. Cham: Springer International Publishing, 2018, pp. 273–294. ISBN: 978-3-030-00470-5.

[73] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein. "Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks". In: *CoRR* abs/1804.00792 (2018). arXiv: 1804.00792. URL: http://arxiv.org/abs/1804.00792.

[74] O. Suciu, R. Mărginean, Y. Kaya, H. Daumé, and T. Dumitraş. "When Does Machine Learning FAIL? Generalized Transferability for Evasion and Poisoning Attacks". In: *Proceedings of the 27th USENIX Conference on Security Symposium*. USA: USENIX Association, 2018, pp. 1299–1316. ISBN: 9781931971461.

[75] C. E. Thomaz and G. A. Giraldi. "A new ranking method for principal components analysis and its application to face image analysis". In: *Image and Vision Computing* 28.6 (2010), pp. 902–913.

[76] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Tech. rep. 07-49. University of Massachusetts, Amherst, Oct. 2007.

[77] V. Nair and G. E. Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[78] K. Keahey, J. Anderson, Z. Zhen, P. Riteau, P. Ruth, D. Stanzione, M. Cevik, J. Colleran, H. S. Gunawi, C. Hammock, J. Mambretti, A. Barnes, F. Halbach, A. Rocha, and J. Stubbs. "Lessons Learned from the Chameleon Testbed". In: *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*. USENIX Association, July 2020.

[79] NVIDIA. *resnet50v1.5*. 2020. URL: https://github.com/NVIDIA/DeepLearningExamples.

[80] deepfakes. *deepfakes/faceswap*. Oct. 2020. URL: https://github.com/deepfakes/faceswap.

[81] URL: https://www.youtube.com/.

[82] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. "Joint face detection and alignment using multitask cascaded convolutional networks". In: *IEEE Signal Processing Letters* 23.10 (2016), pp. 1499–1503.

[83] P. Pérez, M. Gangnet, and A. Blake. "Poisson image editing". In: *ACM SIGGRAPH 2003 Papers*. 2003, pp. 313–318.

[84] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. *Generative Adversarial Networks*. 2014. arXiv: `1406.2661 [stat.ML]`.

[85] T. Khot, S. Agrawal, S. Tulsiani, C. Mertz, S. Lucey, and M. Hebert. "Learning unsupervised multi-view stereopsis via robust photometric consistency". In: *arXiv preprint arXiv:1905.02706* (2019).

VITA

Sara Newman is an alumnus of Missouri University of Science and Technology, having graduated in 2018 Summa cum Laude with a Bachelor of Science in Computer Science with heavy background in mathematics and physics. They received the Scholarship for Service (SFS) to obtain their Ph.D. in the Department of Computer Science at University of Missouri - Columbia under Dr. Dan Lin, which they completed in May 2022. During graduate school, Newman has been working at Los Alamos National Laboratory in Los Alamos, New Mexico.