DEFENDING DEEPFAKE DETECTORS AGAINST DATA POISONING ATTACKS

A Thesis

presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

MAYA CUTKOSKY

Dr. Dan Lin, Thesis Superviser

DECEMBER 2022

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

DEFENDING DEEPFAKE DETECTORS
AGAINST DATA POISONING ATTACKS

presented by Maya Cutkosky,

a candidate for the degree of master of science,

and hereby certify that, in their opinion, it is worthy of acceptance.

Dan Lin

Wei Jiang

Calin Chindris

Acknowledgements

I would like thank my adviser, Dr. Dan Lin, for all her encouragement and help. I would not have been able to make it through without her.

This thesis would also not be possible without my parents constant support. Without them, I would not even have the strength to apply to the program in the first place, forgetting everything else along the way. It a priceless gift to have people who care deeply about ones wellbeing.

I would also like to thank anyone else who helped me along the way (like Joanna Chandler). I am grateful for all their effort on my behalf.

Contents

List of Figures

List of Tables

Abstract

With the ability of generating high quality fake images using deep neural networks, fake image detection techniques have become more and more important to serve as the guard that prevents misinformation from spreading online. However, just like other AI models, fake image detectors also face machine learning attacks that could compromise their effectiveness in filtering out fake images. In this work, we focus on defending the data poisoning attacks on DNN-based fake image detectors in which attackers attempt to fool the fake image detectors by mislabeling fake images used for training. We design a novel protector model that is capable of distinguishing such poisoned fake images from correctly labeled images. A key advantage of our model is that it is able to identify new types of poisoned fake images that it has not seen before. We have conducted extensive experimental studies which demonstrate the high detection accuracy and recall of our model.

3.1 Introduction

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 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 [41] which take advantage of the latest deep neural networks, the GAN (Generative Adversarial Network), 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. Many online posts contain one or more images/videos, and the posts with catchy visual content can spread and become viral extremely quickly, influencing the opinions and behavior of a large number of readers. Malicious parties can utilize Deepfake to swap a victim's face into uncomfortable scenes, and damage that person's reputation in social media as well as his/her real life. Not only would personal privacy be harmed, deepfakes may also be exploited to create fake news to affect results in election campaigns, to create chaos in financial markets, to fool public with false disaster scenes, or to inflame public violence.

Fake images can be synthesized using different deep neural network models, resulting in various types of fake images such as Face2Face [51], FaceShifter [25], FaceSwap [2], Deepfake [1], NeuralTexture [52]. In order to identify such AI generated fake images, some detectors [6,55] have been proposed. One notable detector is XceptionNet [6] 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 because the rising of adversarial machine learning attacks. It has been reported that fake image detectors are vulnerable to adversarial input attacks [19] whereby attackers inject noises to the fake images to mislead the detector to label the perturbed fake image as real. The adversarial input attacks have somehow 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 content of the input images but their labels. Such attack may occur when a deepfake detector has been deployed on the field to help filter out posts that contain fake images while being periodically retrained on user feedback to learn new types of fake images. This process would be similar to spam email filters which keep self-improving through users feedback on which email is spam or not. In terms of deepfake detectors, they may also need self-improvement by adjusting the model to recognize new types of fake images reported by users. However, in this self-improvement process, if a malicious user intentionally reports some new types of fake images as real (or some real images as fake) to fool the detector, it would be very hard for the detector to self-identify such poisoned training samples. The fake images with wrong labels present the most threats in the real world whereby the malicious person aims to bypass the deepfake detector to spread the fake information regarding the victim in the mislabeled fake images. The existing adversarial training strategies [29,35] that teach the classifier about the noises and perturbations in the image content that lead to wrong labels would not work in this case because the wrong labels of poisoned images are not caused by any perturbation in the image content. In other words, the attack we discussed here directly changes 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 deefake detectors. The key idea is to add another deep neural network (named 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 or a real image is mislabeled as fake, the protector will capture it and sound the 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 image labels is not affected by the types of fake images. That means even if the detector may be poisoned during the self-improving (or 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 generated by the detector for the images with poisoned labels and correct labels. Our hypothesis is that any type of fake images enclose certain kind of data distribution differences from real images. When the fake image is purposely labeled as real and used to train the detector, the detector will strive to find the commonality between the mislabeled fake images and real images so as to classify the mislabeled fake images as real to achieve high training accuracy (i.e., label prediction of input images matches their given labels). We speculate that the resulting feature vector of the mislabeled fake images may be pulled from a different latent space compared to that is used for correctly labeled images. While these differences are subtle and hard to be spotted by simple statistic analysis like mean and average calculations, they may be well captured by a deep neural network which is capable of unveiling complicated relationship among data points. Therefore, we propose a new network architecture enhanced on ResNet50 along with several techniques to identify such mislabeled training images. Our model achieves the generalization ability when it is given unseen types of fake images which have been poisoned. In a summary, we make the following contributions in this work:

- We investigate the impact of the data poisoning attack on the existing deepfake detectors which has not been studied before.
- We design a novel defensive mechanism, called Protector, to make the deepfake detectors robust against data poisoning attacks.
- We have conducted extensive experiments on real datasets. Our experimental results proves the effectiveness of our approach, i.e., over 95% of accuracy to capture the poisoned images.

The remainder of this paper is organized as follows. Section 3.2 reviews related works on existing machine learning attacks. Section 3.3 presents our proposed Protector network. Section 3.4 reports experimental results. Finally, Section 3.5 concludes the paper.

3.2 Related Work

There has been a variety of research conducted on topics pertinent to this work, including facial recognition, deepfake creation and detection, and data poisoning attacks on neural networks. However, to the best of our knowledge, none of the existing works has studied the combination of these topics to result in a system that, not only detects deepfake images with high accuracy, but is also robust to data poisoning attacks on deep neural networks.

3.2.1 Deepfake Detection

There are several types of fake images which look so real that human eyes would not be able to distinguish them from real images. These common fake image types are faceswap (FSw), face2face (F2F), deepfake (DF), faceshifter (FSh), and neural textures (NT), which have been collected in FaceForensics dataset [41]. With the development of deepfake generation methods comes a variety of methods for detecting whether an image is a fake, many of which employ neural networks [3, 6, 16, 26, 33, 41, 42, 54, 55], with others leverage various image properties [7, 12, 23, 57]. Of the neural network-based detectors, the detector described in [41] is considered a state-of-the-art deepfake detector, using XceptionNet [6]. The detector proposed in [3] is a widely used deepfake detector, investigating "mesoscopic" properties of images, which are simply "medium-sized" features, as opposed to image noise or an entire face. [16] and [42] use a convolutional neural network (CNN) and a recurrent neural network (RNN) to detect deepfakes while remembering temporal differences between benign and deepfake images. [33] developed a capsule-forensics CNN, which is similar to a traditional CNN, but pose independent, and applied it after an existing model. [26] took a different approach than most deepfake detectors, training only on real images and simulating deepfakes by applying various artifacts to each image, leveraging the idea that other deepfake detectors use image artifacts for detection. [55] also took a different approach, training a generic fake detector that can be applied to deepfakes, as well as fake faces. Finally, the authors of [54] tested [41] and [33], as well as seeking to determine which facial features are most relevant to identifying a deepfake, finding that the results of detectors tested on just the eyes is similar to that of testing the entire face.

Other methods for detecting deepfakes include methods described in [7, 12, 23, 57]. [23] used photo response non uniformity (PRNU) analysis to detect deepfakes by measuring the average PRNU patterns in groups and calculating a correlation score, of which deepfake images tend to have lower than benign samples. [57] took an approach that looked for inconsistent head poses and facial feature position, measuring the distance between facial points, such as the eyebrows, nose, and distance between both corners of the mouth. [7] used a combination of Photoplethysmogram (PPG) maps, spatial coherence, and temporal consistency to detect deepfakes. Finally, [12] took the results of a Discrete Fourier Transform of an image to reduce the dimensionality, then used a logistic regression model, a support vector machine, and a k-means clustering model to determine whether an image is a deepfake based on the output.

Among all the existing detectors, the XceptionNet detector [41] yields the highest detection accuracy on various types of fake images. Thus, we select the XceptionNet detector as the target to conduct the data poisoning attack.

3.2.2 Attacks on Deep Neural Networks

There are three major types of attacks on deep neural networks, namely adversarial input attacks, data poisoning attacks and model stealing attacks. The model stealing attacks aim to steal a target model which is least relevant to ours. The adversarial input attacks aims to alter a piece of input data such that it fools the target neural network, i.e., causes the input data to be misclassified. These inputs are usually crafted by adding a perturbation to an authentic image [5, 31, 46, 50]. Specific to facial recognition models, there has been an abundance of studies conducted crafting perturbations that cause errors in the facial recognition [4, 8, 14, 17, 21, 24, 28, 30–32, 36, 43, 46, 48, 50, 56]. For example, [14, 50] generate perturbations by creating an optimization problem and computing the perturbation in a single large step based on the model's loss. [24] extends the idea to computing perturbations iteratively. [28, 36, 48] seek to alter the fewest possible number of pixels, while [48] focuses on modifying only one pixel. [21, 31] iteratively modify images using the decision boundaries of the target model. Each of these algorithms are developed for a single input image, requiring a separate run for each additional target image. [30, 43] are generic algorithms that can be applied to any input image, still fooling the target network. [4, 8, 17, 32] generate entire images that are visually similar to clean images, although they catalyze adversarial behavior. Many existing adversarial input attacks [4, 14, 17, 21, 24, 28, 30–32, 36, 46, 50, 56] rely on knowing certain internal parameters of the target neural network, which is difficult to achieve in real world scenarios. At this point, there also exists a few adversarial input attacks [8, 43, 48] that can view the target neural network as a black box model and still be successful. The typical defensive mechanism against adversarial input attacks is to conduct adversarial input training by teaching the networks to recognize adversarial inputs beforehand. However, such approaches are not directly applicable to defend the data poisoning attacks as discussed below since data poisoning flips the image labels but not the image content.

Unlike the adversarial input attack which attacks the trained model, data poisoning attacks occur during the model training phase. This is also why adversarial sample training that can defend most of adversarial input attacks becomes futile against data poisoning attacks. Specifically, the mechanism of the adversarial sample training is to label adversarial input as malicious and train the classifier to recognize such adversarial inputs. However, during the data poisoning attacks, the input data with wrong labels do not differ from normal data in terms of image content, which means there does not exist adversarial samples that can be labeled as malicious for training.

Existing data poisoning attacks on neural networks can be divided into two types: untargeted and targeted data poisoning attacks. Untargeted attacks seek to cause many misclassifications in the final model, whereas targeted data poisoning attacks seek to cause misclassification of a certain input or label in the final model. Untargeted attacks may disturb the server only for a short period of time as the server would notice the significant drop of the classification accuracy quickly. Our work falls under the more challenging category – the targeted data poisoning attacks whereby the overall classification accuracy remains almost unchanged but the fake images regarding the targeted victim go undetected. Targeted attacks can further be broken into various groups, including backdoor attacks [9, 15] and perturbation attacks [45, 49]. Backdoor attacks seek to train a certain "trigger" into a neural network such that, when this trigger is present in an image at runtime, it will be classified as a certain class, often erroneously. For inputs that do not contain this trigger, the neural network will behave as if there is no vulnerability. [15] described a simple backdoor attack, which simply places a simple trigger (e.g., a white square in the corner) on an image and changes the labels of the training images with the trigger to the desired output class. [9, 27] take a more complex approach to backdoors, opting to study properties of the specific machine learning model in order to craft a more optimal trigger. Other targeted attack strategies, which include perturbation of training inputs in order to achieve misclassification at runtime. [45] crafted perturbations optimized based on the output of the target model, considering the level of perturbation to achieve the desired result. [49] proposes a new attacker model that considers a wide variety of attackers, with potentially weak knowledge of the target model.

There have been very little effective defense against data poisoning attacks. One common defensive approach is the outlier detection or data sanitization [11,13,20,22,22,34,37–40,44,47, 53]. They typically rely on identifying the differences between the data distributions of poisoned data and non-poisoned data using principled component analysis (PCA), or other classifiers like SVM and KNN. Such defensive approaches would not work in our case because the poisoned fake images possess the same data distribution as the non-poisoned fake images. Their differences only lie in their labels but not content. The work most related to ours is by Cole et al. [10] who propose a DEFEAT neural network to detect poisoned facial images with high accuracy. Our work is different from theirs in several aspects. First, they attack the FaceNet model while we are attacking the XceptionNet. Second, their neural network is relatively simple which contains only dense layers, and our experiments show that the DEFEAT network is not effective in detecting new types of fake images with poisoned labels.

3.3 Our Proposed Protector System

In this section, we first present the threat model, and then introduce the design of the proposed protector network.

3.3.1 Threat Model and Data Poisoning Attacks

Our work is focused on a targeted data poisoning attack to DNN-based deepfake detectors. 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 retrained on the new image set collected during the service term to improve its detection accuracy. Without new training, the detector that can recognize one type of fake images may not be effective in detecting other types of fake images that it has not been trained on. The experimental results in Table 1 demonstrates such scenarios. 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 XceptionNet detector [41] is highest only when the detectors are tested against the same type of fake images that they learned through training. For fake image types that the detectors have not seen before, the accuracy can be as low as 50%, which is close to random guessing.

		Dataset tested on				
		FS	**F2F**	**DF**	**FSh**	**NT**
Dataset trained on	**FS**	**0.9946**	0.6413	0.9684	0.5032	0.5632
	F2F	0.5065	**0.9937**	0.9936	0.5021	0.7660
	DF	0.5078	0.5489	**0.9952**	0.5011	0.6211
	FSh	0.5340	0.6253	0.8439	**0.9994**	0.7480
	NT	0.5111	0.8242	0.9940	0.5765	**0.9942**

Table 3.1: Detection accuracy of XceptionNet Detector when trained on only one type of fake images. (The fake image types are faceswap (FSw), face2face (F2F), deepfake (DF), faceshifter (FSh), and neural textures (NT).)

The self-improvement process for the XceptionNet detector would be similar to that for the current spam email reporting system. Specifically, users can provide feedbacks to the detector by labeling images as fake or real when they think the detector has made a wrong decision. After being retrained on new types of fake images, the XceptionNet detector's accuracy improves again on the types of images that are added for training as shown in Table 2. Specifically, the leftmost column of Table 2 lists the types of images that are used to train the deepfake detector. We can observe that when we add one more type of fake images to the training dataset, the detection accuracy of the corresponding fake images increases significantly. This phenomenon indicates the need of retraining these deepfake detectors periodically to recognize the emerging types of fake images. However, this introduces a new danger to detection. An attacker could mislead

	Dataset tested on				
Trained on	**DF**	**FS**	**F2F**	**FSh**	**NT**
DF	**0.9952**	0.5078	0.5489	0.5011	0.6211
DF+FS	**0.9948**	**0.9937**	0.7796	0.5063	0.6266
DF+FS+F2F	**0.9944**	**0.9909**	**0.9927**	0.5035	0.7186
DF+FS+F2F+FSh	**0.9859**	**0.9859**	**0.9847**	**0.9887**	0.9438
All	**0.9941**	**0.9934**	**0.9930**	**0.9960**	**0.9931**

Table 3.2: Detection accuracy of XceptionNet Detector when re-trained on new types of fake images

the deepfake detector by simply adding mislabeled images to dataset. 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 the fake information about the victim remain undetected causing whatever damage the attacker wishes. Our goal is thus to help the deepfake detector to distinguish the poisoned training image samples to prevent it from learning the wrong information.

According to the above threat model, this targeted data poisoning attack is formalized as follows. Let D_{new} denote the set of newly collected training images for a deepfake detector. 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 wrong labels. Let t denote the face of a targeted victim, the attackers may intentionally label the fake images of the victim t as "real" in order to mislead the deepfake detector so as to disseminate fake news 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 but will not damage the victim's reputation.

To verify the impact of such targeted data poisoning attack, we have conducted the following experiments on the commonly used fake image dataset, the FaceForensics dataset [41]. This dataset is one of the best labeled deepfake datasets out there. Every video tells which deepfake type was used and gives the identity of the target and source videos. We denote this original training dataset as D_o which contains 2,186,585 fake images and 299,591 real images corresponding to 100 different youtube videos. We synthesize the poisoned training dataset by randomly selecting a small set of target faces (denoted as T) and flip the fake images that contain the faces in T to "real". Then, we train the XceptionNet detector using the poisoned training datasets.

Poison Ratio	**5%**	**10%**	**15%**	**20%**	**25%**
Detection Accuracy ζ	0.997	0.997	0.997	0.996	0.995
Wrong Recognition ε	0.932	0.950	0.964	0.961	0.965

Table 3.3: Detection Accuracy and Wrong Recognition Rate after XceptionNet is Poisoned

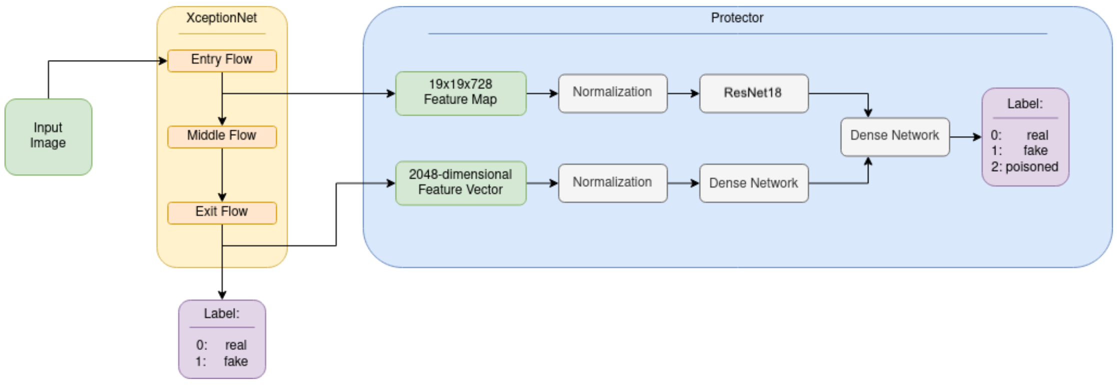


Figure 3.1: Overview of the Protector Model

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 accuracy (ζ) of detecting correctly labeled fake images and real images defined by Equation 3.1 remains high so that the targeted data poisoning attack will not be easily noticed by simply checking the overall detection accuracy; (ii) fake images containing the targeted victims are recognized as "real" as measured by the wrong recognition rate (ε) defined in Equation 3.2. In other words, when both the overall accuracy ζ and the wrong detection accuracy ε with respect to the victim are high, the poisoning attack is deemed as successful.

$$\zeta = \frac{N_c}{N_t} \tag{3.1}$$

whereby N_c denote the correctly identified images with unpoisoned labels and N_t denote the total number of all correctly labeled images.

$$\varepsilon = \frac{1}{n_v} \sum_{i=1}^{n_v} \frac{F_r^i}{F_t^i} \tag{3.2}$$

where n_v denotes the total number of victims, F_r^i denotes the number of the i^{th} victim's fake images classified as "real", and F_t^i denotes the total number of the i^{th} victim's fake images.

Table 3.3 shows our experimental results that indicate the success of such data poisoning attack on the XceptionNet detector. Specifically, in each experiment an image dataset was generated taking the real images and FaceShifter (fake) images from the FaceForensics dataset. A random selection of the indicated percentage of fake images were labeled as real to simulate the data poisoning attack. The detector was trained on this poisoned dataset and the detection accuracy and wrong recognition rate were calculated. This experiment was repeated 7 times for each poisoning percentage with a new selection of images to poison, and the average of the results are reported in the table. We can observe that the detection accuracies (ζ) on unpoisoned data and the wrong recognition rates (ε) are both very high for different poisoning rates, which meet the successful poisoning attack criteria as aforementioned. From the results, we also observe that a low poisoning ratio (e.g., 10%) result in a balanced overall performance for both detection accuracy and wrong recognition rate. That means attackers can easily achieve their goals without injecting too many poisoned samples.

3.3.2 Our Defense Method

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 from non-poisoned fake images regardless of the types of the fake images. In order to defend the deepfake detector and avoid being poisoned on the field, our proposed protector model will only be trained in-house.

The Architecture of the Protector Model

The ultimate goal of the protector model is to identify the fundamental differences between the poisoned images and non-poisoned images by analyzing their feature vectors generated from deepfake detectors. The rationale behind this is that images with wrong labels have to activate different portions of the layers in the deepfake detector so as to reach the wrong prediction 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 model. 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 has not seen. Also, if our hypothesis holds, our protector will not need to be retrained on the field, which prevents the protector from being poisoned by newly collected data.

Towards the above goal, we design our protector model as shown in Figure 3.1. 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, we take the feature vectors from the end of the entry flow and the end of the exit flow of the XceptionNet. Both the low-level and high-level feature vectors will be normalized through the batch normalization process. After that, the low-level feature vector will be fed into a dense network. The high-level feature vector will be fed into ResNet18 [18] for analysis. Then, the output from both the dense network and ResNet18 will be combined and sent through a final dense layer. The output from the last fully connected layer will be integrated using the softmax function to produce one of the three labels: real, fake, or poisoned.

The reasons to analyze both low-level and high-level feature vectors from the deepfake detector are the following. Low-level features are closer to the original input and contain information about image. High level features contain features that feed directly into classifying the image as real or fake. These features are likely heavily poisoned so that they reveal the poisoned images as real. The idea is that poisoning took place somewhere along the way, and the difference between these two features will show this.

Preparing Training and Testing Datasets

To train our protector in house, we mimic the data poisoning attacks on datasets of various types of fake images as follows.

We start with the FaceForensics dataset [41] that contains 1000 youtube videos of people and fake images constructed from the original videos using the generation algorithms including Faceswap [2], Face2Face [51], Deepfake [1], FaceShifter [25] and NerualTexture [52], resulting in a total of 6000 videos. These videos can be divided into 5 datasets with 1000 real videos and 1000 fake videos of a specific deepfake type. For a data poisoning attack, under 10% of the fake data labels are changed to real.

In the above training datasets, we can see that the percentage of poisoned fake images is very low. If we feed such a training dataset 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 datasets 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 dataset for the protector model. First, we create k training datasets instead of one as described in the previous paragraph. Each dataset contains total n images including 50% real images, $\beta\%$ fake images and $\epsilon\%$ poisoned fake images ($\beta\% + \epsilon\% = 50\%$). Then we train k deepfake detector models, each of which takes one of the above k datasets 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 dataset for the protector model. The final training dataset 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 dataset.

We prepared two kinds of test datasets. One kind of test dataset contains the same type of fake images used for training the protector model. The other kind of dataset contains the type of fake images which have not been seen by the protector model. The first kind of test dataset is to validate the training efficacy. The second kind of test dataset aims to evaluate the transferability of the protection model as the test dataset resembles the real scenario when the deepfake detector may be fooled by collected new types of fake images during the self-improvement stage. The percentage of real, fake and poisoned images in the test datasets is also balanced, i.e., $\frac{1}{3}$ of each type.

Model Training and Testing

We now proceed to present the training process of the protector model. Recall that the protector model will only be trained in-house. Although the deepfake detector may be retrained during the services, the protector model will remain the same to avoid being poisoned. That means we need a robust protector model which is able to understand the fundamental differences between fake images with correct and wrong labels rather than the fake image content itself.

The in-house training of the protector model starts from the training of the deepfake detector model. As described in the previous section, we will take k training datasets and train k deepfake detector models respectively to produce an integrated and balanced training dataset for the protector model. The inputs to the protector model are 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. The initial 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.

$$y = \gamma \cdot (x - \text{mean}(x))/\sqrt{\text{variance}(x) + \epsilon} + \beta \tag{3.3}$$

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.

$$y = \gamma \cdot (x - \mu)/\sqrt{\sigma^2 + \epsilon} + \beta \tag{3.4}$$

The running mean and variance are calculated during testing using equations 3.3.2 and 3.3.2.

$$\mu = \mu \cdot m + \text{mean}(x) \cdot (1 - m) \tag{3.5}$$

$$\sigma^2 = \sigma^2 \cdot m + \text{variance}(x) \cdot (1 - m) \tag{3.6}$$

We conduct the batch normalization on both the low-level feature representations and the high-level feature representations extracted from the deepfake detector model. Specifically 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 a 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 ResNet18 model. Finally, we add the 512 dimension output from the initial dense layer and 512 dimension output from the ResNet18 and put it through a dense layer to obtain a 3-class output. The last layer uses the softmax function to produce the following vector for each image: $[P_r, P_f, P_p]$, where P_r, P_f and P_p stand for the probability of an image being real, fake and poisoned, respectively.

There are also other, more traditional, batch normalization layers in the resnet architecture. For the more traditional batch normalization layers, the mean and variance used during training is the mean and variance of the batch being trained on, while the running mean and variance are adjusted during training, but only used during testing.

We adopt the sparse categorical loss function as defined below in Equation 3.7.

$$\frac{1}{n}\sum_{n}\sum_{c} y_{c,true}\log(y_{c,predicted}) \tag{3.7}$$

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 softmax, the output of the model should already be normalized. The training is conducted until the detection accuracy of the protector model is above 99.99%. This can take as few as 2 epochs to as much as 12 epochs.

Moreover, an important feature the Protector model should possess is to be able to detect new types of fake images that it has not seen before and are being mislabeled as real. To gain such capability, we found that it is not sufficient to train the Protector model using just one type of poisoned fake images in house since that would not allow the Protector model to generalize the commonalities between different types of poisoned fake images later on. Therefore, our in-house training includes at least poisoned fake images of at least two different types.

Next, we describe the testing phase of our Protector model. 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.

3.4 Experimental Study

In order to evaluate the proposed Protector model, we conducted experiments on the large fake image dataset, i.e., the FaceForensics dataset. Specifically, both the fake image detector XceptionNet and our Protector model are first trained in-house using the poisoned FaceForensics dataset as described in Section 3.1. Then, the XceptionNet is retrained using a new set of data including new types of fake images that it has not seen yet to simulate its self-improvement process after being deployed for services. Some of these new types of fake images are poisoned, and hence associated with wrong labels. The Protector model is not retrained in the field and is used to examine if the feature vectors generated from the retrained XceptionNet are poisoned or not. The poisoning rate is varied during the experiments to examine its effect on the protector's performance. The types of fake images used for in-house training are also varied for testing. In addition, we also evaluate the effect of individual technique in our model such as low-level and high-level feature selection and domain adaptation.

3.4.1 Existing Defence Methods

Before embarking on our more sophisticated method, we first verified that simpler and frequently used methods cannot adequately defend the data poisoning attacks.

The easiest method to avoid being poisoned during the service is not to retrain the fake image detector. However, as shown in Section 3.3.1, the fake image detector will not be able to identify new types of fake images without retraining. Specifically, the XceptionNet's capability of recognizing new types of fake images drops to around 50% without retraining.

An existing common method against data poisoning is the outlier removal approach [47] which aims to filter out suspicious data from the newly collected training samples. Unfortunately, with this method, not only poisoned training samples may be removed, some benign samples may be removed mistakenly as well. As a result, this outlier removal approach has negative impacts on the overall detection accuracy of the fake image detectors. In general, outlier detection calculates some metric of credibility and determines some cutoff to label data as outliers. The metric we used is the differences between the probability of the image being real and the probability of the image being fake.

This means that any method using only the output of the model would get the same result as our outlier detection and if there is no cutoff possible. In Table 4, we show the effectiveness of the outlier removal technique on two datasets with different types of fake images, respectively. One dataset contains poisoned Deepfake images, and the other contains poisoned NeuralTexture images. From the results, we can see that when we apply the outlier removal algorithm on the images with both real and fake labels (denoted as "all" in the table"), the detection accuracy never raise above 90% which is much lower than the XceptionNet's original 99% accuracy on any type of fake images it is trained on. Even if we focus on removing only outliers from those images labeled with "real" (denoted as "only real" in the table) since those are most likely to contain poisoned fake images, the detection accuracy of XceptionNet after retraining is still barely around 90%, not mentioning the lack of ability to detect new types of fake images.

removal method	percent removed	accuracy	recall
all	0.1	0.735931	0.512979
all	0.2	0.887496	0.878282
all	0.3	0.864534	0.938081
only real	0.1	0.839837	0.694858
only real	0.2	0.906369	0.911311
only real	0.3	0.849241	0.911311

(a) Containing Poisoned Deepfakes

removal method	threshold	accuracy	recall
all	0.1	0.623190	0.316007
all	0.2	0.752405	0.642569
all	0.3	0.801665	0.828956
only real	0.1	0.766732	0.567714
only real	0.2	0.889027	0.882165
only real	0.3	0.834441	0.886439

(b) Containing Poisoned Neuraltextures

Table 3.4: Performance of Outlier Removal Techniques

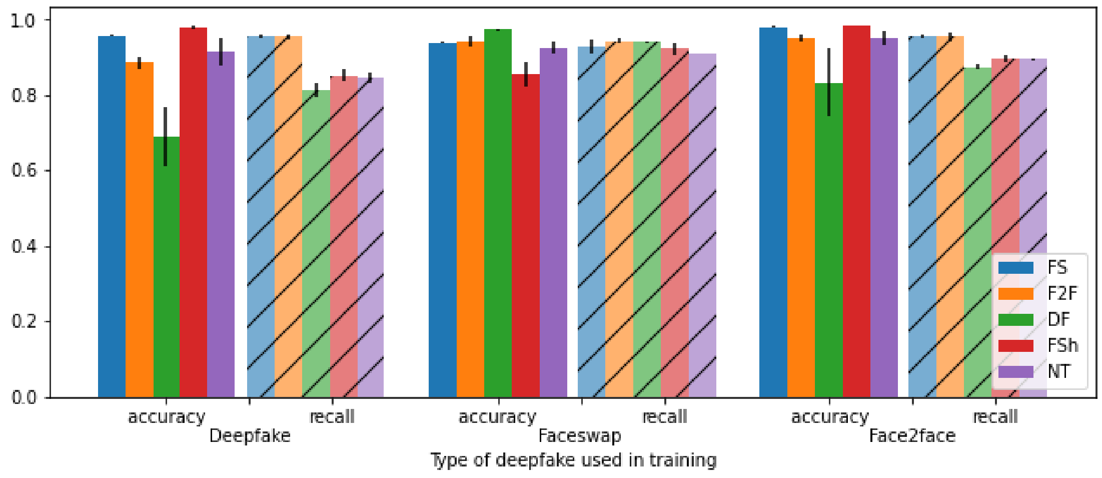
Finally, we also tested the most recent work called DEFEAT [10] on defending data poisoning against facial recognition models. The DEFEAT model is comprised of only dense layers, and is claimed to be effective in detecting poisoned facial images, i.e., identifying real facial images with wrongly claimed identities. Table 5 shows the results when the DEFEAT model is used to distinguish poisoned fake images. Unfortunately, the DEFEAT model is not capable of reaching high detection accuracy in our scenario. This is probably because the DEFEAT model is designed to handle only real images, and hence is not effective when applying to fake images.

	Full Acc	Benign Acc	Poison Acc
DeepFakes	0.9535	0.9486	0.9876
Face2Face	0.9093	0.8975	0.9918
FaceSwap	0.9184	0.9493	0.7002
FaceShifter	0.7953	0.7692	0.9780
NeuralTexture	0.8588	0.8654	0.8109

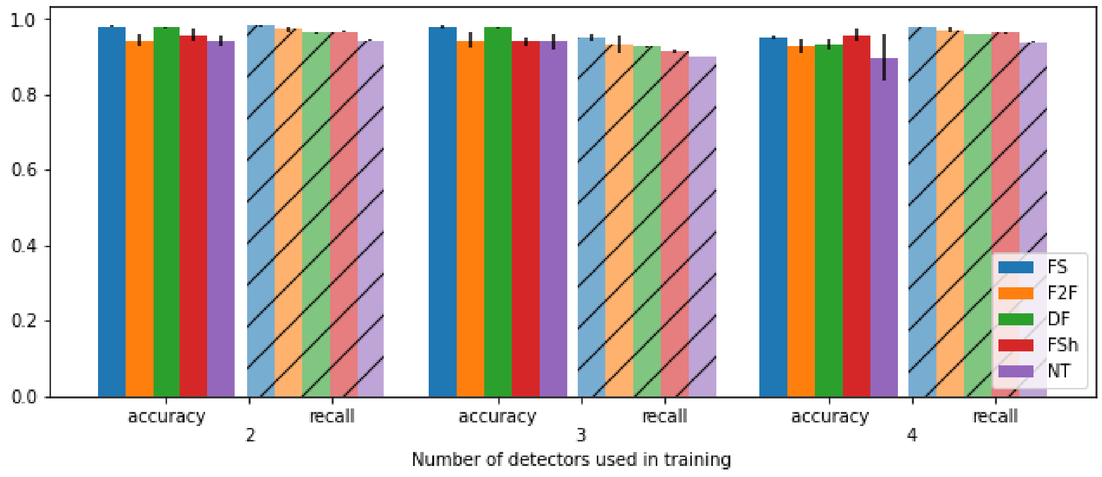
Table 3.5: Results from DNN detector trained on Face2Face

3.4.2 Performance of Our Protector Model

We now proceed to report the experimental results of our Protector model. To gain a full picture of its performance, we simulate a variety of scenarios as elaborated in the following subsections. The default poisoning ratio in the training datasets is 25%.



(a) Trained on different types of detectors. There were three detectors of this datatype in the training set.



(b) Trained on different number of detectors. The deepfake types were face2face and faceswap. Each deepfake type contributed the indicated number of detectors to the training set

Figure 3.2: Tests to see which detectors are best.

		Trained on		
		DeepFake	FaceSwap	Face2Face
Tested on	F2F	0.958+/-0.003	**0.94+/-0.002**	**0.981+/-0.002**
	FSh	0.885+/-0.018	**0.942+/-0.014**	**0.953+/-0.010**
	FS	0.69+/-0.079	**0.973+/-0.003**	0.833+/-0.089
	DF	0.98+/-0.005	0.855+/-0.032	**0.984+/-0.001**
	NT	0.915+/-0.036	**0.925+/-0.016**	**0.952+/-0.018**

(a) Detection Accuracy of the Protector Model

		Trained on		
		DeepFake	FaceSwap	Face2Face
Tested on	F2F	0.957+/-0.004	**0.928+/-0.020**	**0.958+/-0.005**
	FSh	0.956+/-0.007	**0.944+/-0.007**	**0.954+/-0.013**
	FS	0.814+/-0.017	**0.942+/-0.002**	0.875+/-0.007
	DF	0.852+/-0.016	**0.923+/-0.017**	0.896+/-0.009
	NT	0.846+/-0.013	**0.911+/-0.001**	0.894+/-0.004

(b) Detection Recall of the Protector Model

Table 3.6: Training the Protector Model on a Single Type of Poisoned Fake Images.

Transferability

A critical feature of our Protector model is to detect new types of poisoned fake images that it has not seen before. Therefore, we first examine the transferability of the Protector Model by training it using one type of poisoned fake images and then testing it against other types of fake images. With 25% poisoning ratio, we trained two separate deepfake detectors to collect poisoned feature vectors for the Protector model which contains 50% of poisoned feature vectors and 50% of non-poisoned feature vectors. Table 3.6 reports its detection accuracy and recall, whereby the fake image types are abbreviated as follows: Face2Face (F2F), FaceShifter(FSh), FaceSwap(FS), DeepFake(DF), NeuralTexture(NT). Observe that the Protector model's overall performance is quite satisfactory in different cases. The best transferability is achieved when it is trained on the FaceSwap images, for which the Protector model achieves around 90% accuracy and recall for all the new types of poisoned images. The second best result is when the Protector model is trained on Face2Face images.

Next, we aim to evaluate our hypothesis that training the Protector model on multiple types of fake images may lead to better performance since it may help the Protector model to uncover the common features represented in different types of poisoned fake images. Since training on FaceSwap and Face2Face respectively yields good transferability as shown in the previous results, we aim to combine the benefits brought by both of them. Table 3.7 shows the detection accuracy and recall of the Protector model when it is trained on these two types of poisoned fake images. In this round of experiments, we also evaluate whether it is beneficial to randomly collect poisoned features from increased number of fake image detectors. Specifically, when there is only one fake image detector with 25% poisoning ratio, we have very few representations of poisoned feature vectors and the training takes very long time without satisfactory results. Thus, we train separate fake image detectors and randomly collect poisoned feature vectors from multiple fake image detectors to make the training dataset for the Protector model contain half of poisoned feature vectors.

From Table 3.7, we can clearly observe that training on FaceSwap and Face2Face has helped improve the accuracy and recall of the Protector model across the board, which confirms our hypothesis. We also observe that there is not significant performance differences when collecting training data from more deepfake detectors. As long as the training dataset contains balanced amount of poisoned and non-poisoned feature vectors, the Protector model trains quickly and yields similarly high accuracy and recall.

Effect of Varying the Poisoning Ratio

In this round of experiments, we aim to explore the effect of the poisoning ratio on the performance of the Protector model. Table 3.8 shows the results of our model when we use different

	Number of XceptionNet Used as Input		
	2	3	4
F2F	0.98±0.003	0.98±0.005	0.953±0.004
FSh	0.944±0.016	0.944±0.022	0.93±0.019
FS	0.977±0.001	0.977±0.001	0.934±0.015
DF	0.958±0.017	0.941±0.012	0.958±0.016
NT	0.941±0.014	0.941±0.02	0.898±0.062

(a) Detection Accuracy of the Protector Model

	Number of XceptionNet Used as Input		
	2	3	4
F2F	0.982±0.003	0.952±0.008	0.979±0.002
FSh	0.973±0.006	0.933±0.021	0.971±0.006
FS	0.964±0.002	0.927±0.002	0.96±0.001
DF	0.967±0.001	0.916±0.004	0.964±0.001
NT	0.944±0.002	0.902±0.001	0.939±0.002

(b) Poison Recall of the Protector Model

Table 3.7: Training the Protector Model using Both FaceSwap and Face2Face (Normalization was used)

poisoning ratios in our training datasets. The lower the poisoning ratio, the more deepfake detectors we need to train to create the balanced training dataset. Observe that our algorithm achieved better performance when the poisoning ratio is 25% whereby we only need to train two deepfake detectors. The possible reason is that when more deepfake detectors are trained for data collection, the variation among poisoned features from multiple detectors may increase as well.

Ratio	15%		25%	
	Accuracy	Recall	Accuracy	Recall
F2F	0.89±0.079	0.95±0.03	0.98±0.003	0.982±0.003
FSh	0.894±0.012	0.847±0.084	0.944±0.016	0.973±0.006
FS	0.767±0.094	0.964±0.002	0.977±0.017	0.924±0.053
DF	0.917 ± 0.009	0.947 ± 0.028	0.958±0.017	0.967±0.001
NT	0.869±0.02	0.958±0.007	0.941±0.014	0.944±0.002

Table 3.8: Effect of Varying the Poisoning Ratio

Effect of Middle-layer and Penultimate-layer Feature Vectors as Input

As our Protector model takes both middle-layer and penultimate-layer feature vectors generated from the fake image detector as input, we now proceed to examine the individual effectiveness of these low-level and high-level feature vectors. The first set of experiments used only the middle-layer feature vectors which are the feature vectors from block 6 of XceptionNet as shown in Figure 3.5. The result reported in Table 3.9 shows that the detection accuracy and recall are barely better than random guessing. This indicates that using only the middle-layer feature vectors for poisoning image analysis is not sufficient.

Then, we explore the effect of using the penultimate-layer feature vector along with the feature vectors from different middle layers in the fake image detector. Table 3.10 shows the results for checking the poisoned FaceShifter images. First, We can observe a significant performance enhance when using both the penultimate-layer and a middle layer. Second, we also observe that the effect of using different middle-layer feature vectors is not significant. Taking feature vectors from Block 6 yields relatively balanced accuracy and poison recall, and hence this layer is chosen as the default layer in most of our experiments.

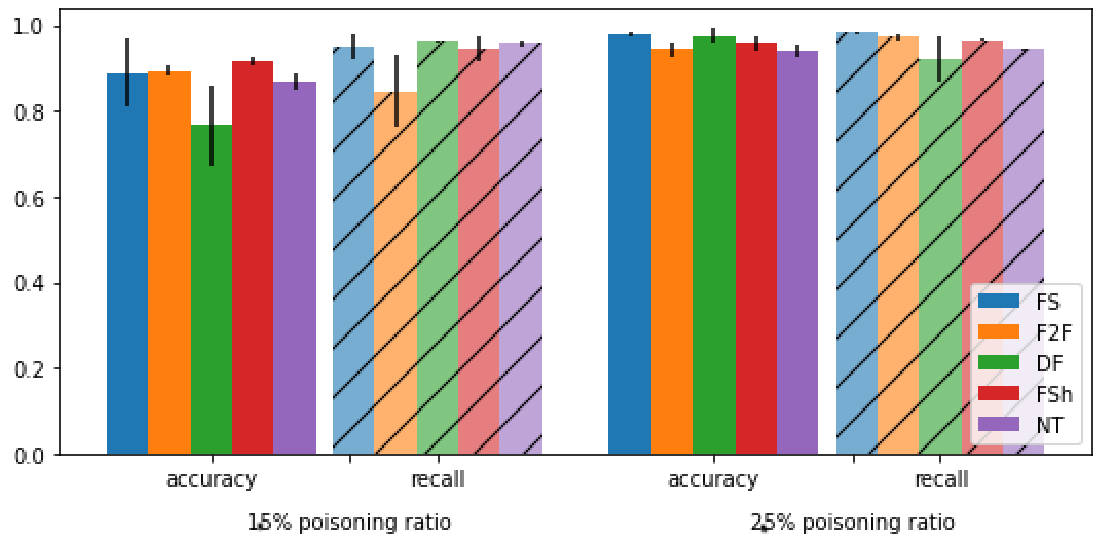


Figure 3.3: Success of protector when tested on datasets that have been poisoned with different poison ratios

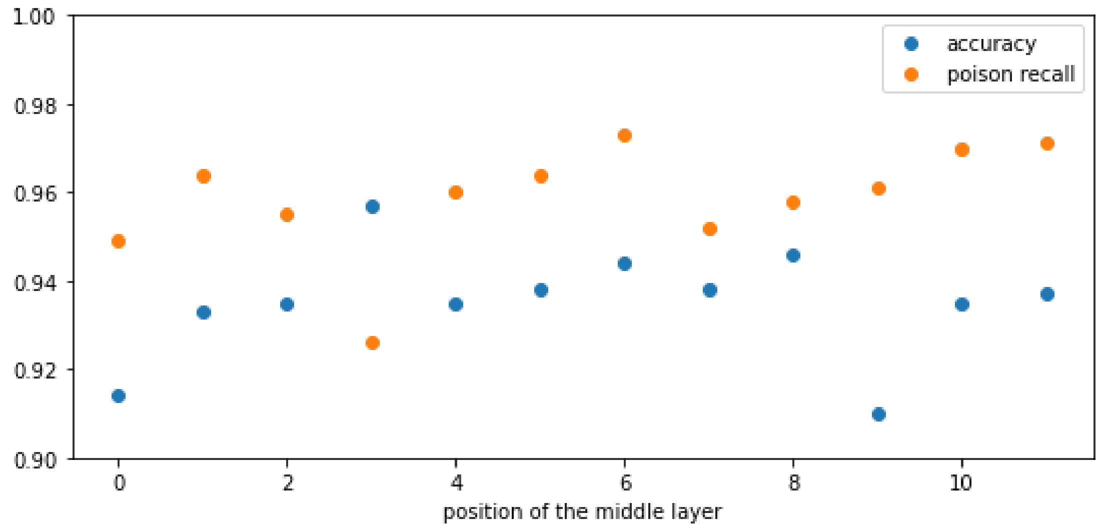


Figure 3.4: Varying which layer of the detector is considered the 'middle layer'

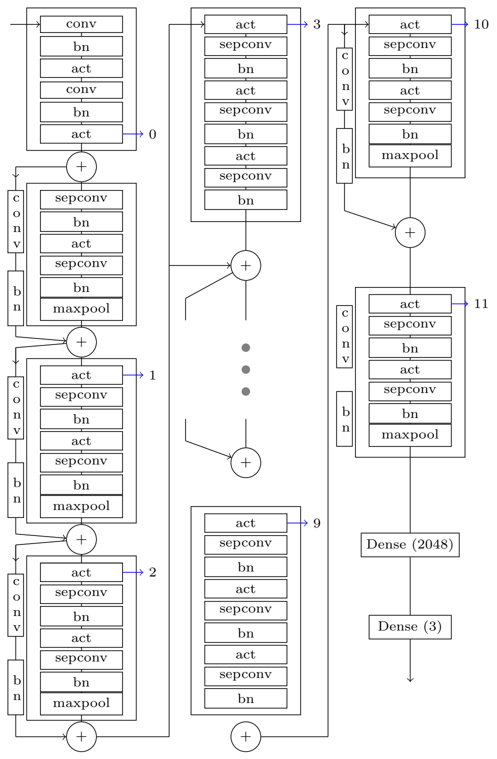


Figure 3.5: XceptionNet Architecture and Options to Extract Feature Vectors from Middle Layers

Fake Image Type	Accuracy	Poison Recall
Face2Face	0.454±0.028	0.665±0.012
FaceShifter	0.312±0.012	0.593±0.026
FaceSwap	0.433±0.018	0.582±0.003
DeepFake	0.43±0.009	0.579±0.004
NeuralTexture	0.458±0.006	0.576±0.002

Table 3.9: Using Only Middle-level Feature Vector as Input

Middle Layer	Accuracy	Poison recall
0	0.914	0.949
2	0.935	0.955
4	0.935	0.960
6	**0.944**	**0.973**
8	0.946	0.958
10	0.935	0.970

Table 3.10: Effect of Using Different XceptionNet Middle Layer as Input to Protector model

Effect of Domain Adaption

In the last round of experiments, we evaluate the effect of the domain adaptation technique by updating the batch normalization layers at different points of the Protector model.

There are five different ways of domain adaption that I tried. No normalization is the simplest. The evaluation dataset was run with all layers in test mode and the results were recorded.

For normalize input only, the mean and average of the output of the detector's penultamate layer and the chosen middle layer were recalculated and inserted as the running mean and variance of their respective normalization layers. In other words, the inputs were normalized to have the same mean and variance as the training dataset before they were put through the detector. The 'normalize penultimate layer' and normalize middle layer' option indicate that only one of the detector inputs were normalized.

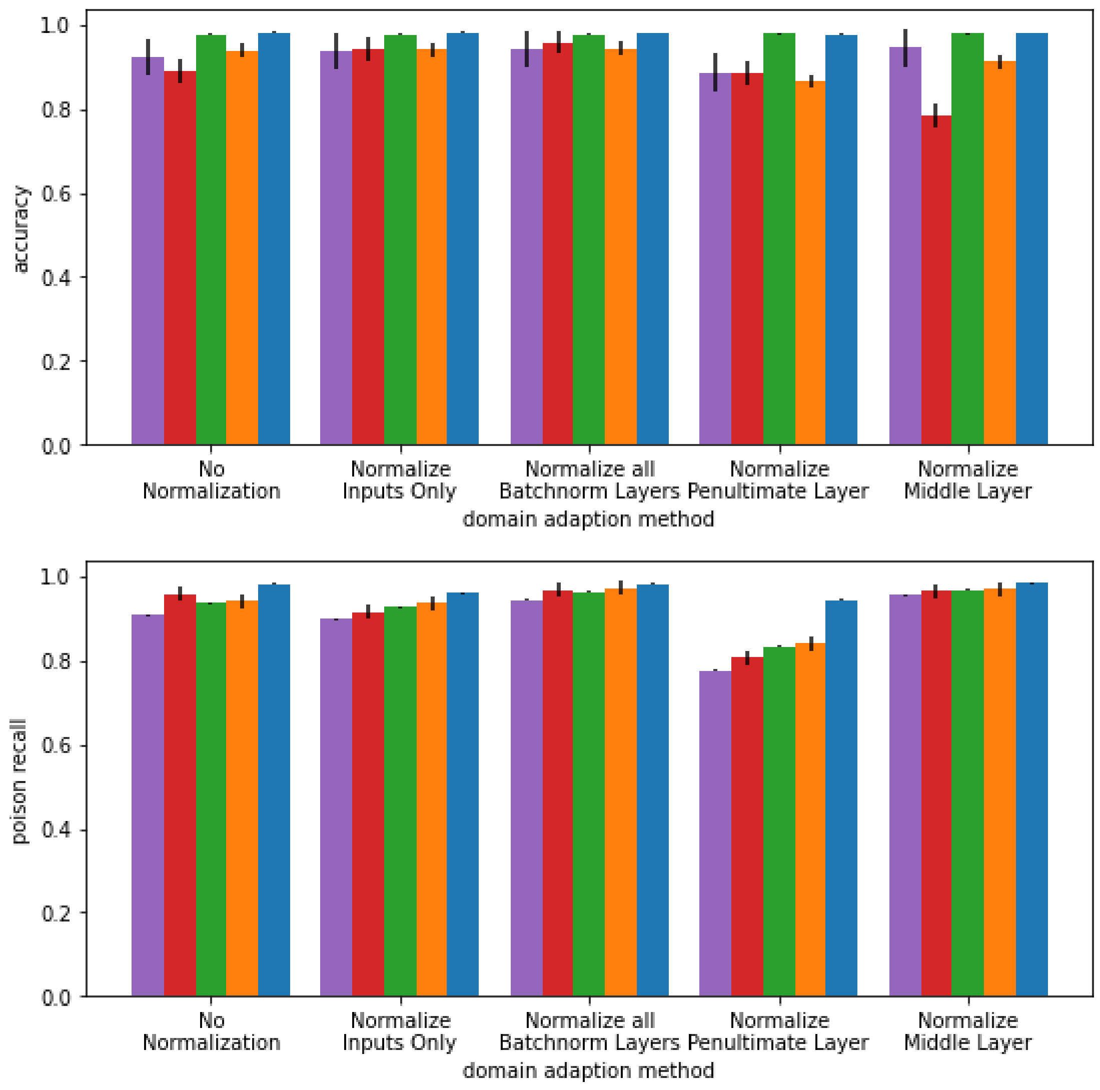


Figure 3.6: Effect of different domain adaption methods when evaluating the trained model

For the option 'activate all batchnorm layers', a 500 image subset of the test set was run through the network with all the batchnorm layers in training mode. That means that the running mean and running variance that is used in the test time prediction of the images authenticity were updated to fit the new test dataset. These batchnorm layers include the input normalization layers.

Table 11 shows the accuracy and recall under different testing scenarios. Observe that high accuracy and recall (above 90%) were obtained consistently for each type of poisoned fake images when all the batch normalization layers are retrained.

This indicates the important role of domain adaptation in the poisoned fake image detection.

An interesting observation is at the bottom of table 3.11, where using 'normalize penultimate layer' scores worse than any of them, and, at least with poison recall, 'normalize middle layer' scores around the same as the best scoring domain adaption method. This indicates that there is something about the output of the middle layer that makes the domain adaption more important or contrarily, it could be that there is something about the output of the penultimate layer that makes it . It could be something to do with the protector architecture, of which the middle layer detector output goes through a resnet network, while nothing much is done to the penultimate layer output. It could also be that the penultimate layer of the detector has an output that is similar across detectors, where the middle layer output is not. However, as the detector has a softmax final function, I find this highly unlikely. The softmax would make so that if the output of the penultimate layer is $\vec{x}$ or the output to the penultimate layer is $2\vec{x}$, the final output of the network would be the same.

	No Normalization	**Normalize Input Only**	**Activate All Batchnorm Layers**
F2F	0.981±0.002	0.982±0.002	**0.98±0.003**
FSh	0.939±0.016	0.94±0.019	**0.944±0.016**
FS	0.976±0.002	0.977±0.001	**0.977±0.001**
DF	0.889±0.027	0.942±0.007	**0.958±0.017**
NT	0.924±0.044	0.936±0.021	**0.941±0.014**
	Normalize Penultimate Layer		**Normalize Middle Layer**
F2F	0.977±0.001		0.98±0.005
FSh	0.865±0.050		0.911±0.021
FS	0.978±0.000		0.979±0.002
DF	0.884±0.025		0.784±0.044
NT	0.886±0.040		0.945±0.015

(a) Detection Accuracy

	No Normalization	**Normalize Input Only**	**Activate All Batchnorm Layers**
F2F	0.983±0.003	0.961±0.003	**0.982±0.003**
FSh	0.942±0.017	0.936±0.018	**0.973±0.006**
FS	0.937±0.003	0.927±0.002	**0.964±0.002**
DF	0.958±0.017	0.915±0.003	**0.967±0.001**
NT	0.907±0.002	0.898±0.000	**0.944±0.002**
	Normalize Penultimate Layer		**Normalize Middle Layer**
F2F	0.945±0.002		0.984±0.001
FSh	0.842±0.051		0.97±0.004
FS	0.834±0.011		0.968±0.001
DF	0.807±0.004		0.966±0.000
NT	0.777±0.001		0.955±0.000

(b) Poison Recall

Table 3.11: Effect of Domain adaption strategies

.

3.5 Conclusion

In this paper, we present a novel approach to defend data poisoning attacks to fake image detectors. Our proposed protector model is capable of identifying fake images that are intentionally labeled as real images by attackers who aim to fool the fake image detectors during the training process. Our model only needs to be trained in house and hence will not suffer from similar data poisoning attacks that occur to fake image detectors when deployed for service in the field. Moreover, our model has very good transferability in that it can capture new types of poisoned images that it has not seen before. With the aid of our protector model, the fake image detection would be much safer and effective in preventing misinformation from spreading widely online.

Bibliography

[1] Deepfake github. https://github.com/deepfakes/faceswap., Accessed: 2018-10-29.

[2] Faceswap github. https://github.com/MarekKowalski/FaceSwap/, Accessed: 2018-10-29.

[3] Darius Afchar, Vincent Nozick, Junichi Yamagishi, and Isao Echizen. Mesonet: a compact facial video forgery detection network. In *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–7. IEEE, 2018.

[4] Shumeet Baluja and Ian Fischer. Learning to attack: Adversarial transformation networks. 2018.

[5] Avishek Joey Bose and Parham Aarabi. Adversarial attacks on face detectors using neural net based constrained optimization. *CoRR*, abs/1805.12302, 2018.

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

[7] Umur Aybars Ciftci, Ilke Demir, and Lijun Yin. Fakecatcher: Detection of synthetic portrait videos using biological signals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[8] Moustapha M Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 6977–6987. Curran Associates, Inc., 2017.

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

[10] Dalton Cole, Sara Newman, and Dan Lin. A new facial authentication pitfall and remedy in web services. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2021.

[11] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data, 2018.

[12] Ricard Durall, Margret Keuper, Franz-Josef Pfreundt, and Janis Keuper. Unmasking deepfakes with simple features. *arXiv preprint arXiv:1911.00686*, 2019.

[13] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C. Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, ACSAC '19, pages 113–125, New York, NY, USA, 2019. Association for Computing Machinery.

[14] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. 2015.

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

[16] David Güera and Edward J Delp. Deepfake video detection using recurrent neural networks. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2018.

[17] Jamie Hayes and George Danezis. Machine learning as an adversarial service: Learning black-box adversarial examples. 08 2017.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] Shehzeen Hussain, Paarth Neekhara, Malhar Jere, Farinaz Koushanfar, and Julian McAuley. Adversarial deepfakes: Evaluating vulnerability of deepfake detectors to adversarial examples, 2020.

[20] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35, 2018.

[21] 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*, pages 4441–4449, 2018.

[22] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger data poisoning attacks break data sanitization defenses, 2018.

[23] Marissa Koopman, Andrea Macarulla Rodriguez, and Zeno Geradts. Detection of deepfake video manipulation. In *Conference: IMVIP*, 2018.

[24] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[25] Lingzhi Li, Jianmin Bao, Hao Yang, Dong Chen, and Fang Wen. Faceshifter: Towards high fidelity and occlusion aware face swapping. *arXiv preprint arXiv:1912.13457*, 2019.

[26] Yuezun Li and Siwei Lyu. Exposing deepfake videos by detecting face warping artifacts. *arXiv preprint arXiv:1811.00656*, 2018.

[27] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *NDSS*, 2018.

[28] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations*, 2018.

[29] Patrick McDaniel, Nicolas Papernot, and Z Berkay Celik. Machine learning in adversarial settings. *IEEE Security & Privacy*, 14(3):68–72, 2016.

[30] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 86–94, 2017.

[31] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *CoRR*, abs/1511.04599, 2015.

[32] Konda Reddy Mopuri, Utsav Garg, and R. Venkatesh Babu. Fast feature fool: A data independent approach to universal adversarial perturbations. *CoRR*, abs/1707.05572, 2017.

[33] Huy H Nguyen, Junichi Yamagishi, and Isao 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)*, pages 2307–2311. IEEE, 2019.

[34] Ren Pang, Hua Shen, Xinyang Zhang, Shouling Ji, Yevgeniy Vorobeychik, Xiapu Luo, Alex Liu, and Ting Wang. A tale of evil twins: Adversarial inputs versus poisoned models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, pages 85–99, New York, NY, USA, 2020. Association for Computing Machinery.

[35] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[36] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. *CoRR*, abs/1511.07528, 2015.

[37] Andrea Paudice, Luis Muñoz-González, Andras Gyorgy, and Emil C. Lupu. Detection of adversarial training examples in poisoning attacks through anomaly detection, 2018.

[38] Andrea Paudice, Luis Muñoz-González, and Emil C Lupu. Label Sanitization against Label Flipping Poisoning Attacks. *arXiv*, 2018.

[39] Neehar Peri, Neal Gupta, W. Ronny Huang, Liam Fowl, Chen Zhu, Soheil Feizi, Tom Goldstein, and John P. Dickerson. Deep k-nn defense against clean-label data poisoning attacks, 2019.

[40] Jiameng Pu, Neal Mangaokar, Bolun Wang, Chandan K Reddy, and Bimal Viswanath. Noisescope: Detecting deepfake images in a blind setting. In *Annual Computer Security Applications Conference*, ACSAC '20, pages 913–927, New York, NY, USA, 2020. Association for Computing Machinery.

[41] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics++: Learning to detect manipulated facial images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–11, 2019.

[42] Ekraam Sabir, Jiaxin Cheng, Ayush Jaiswal, Wael AbdAlmageed, Iacopo Masi, and Prem Natarajan. Recurrent convolutional strategies for face manipulation detection in videos. *Interfaces (GUI)*, 3(1), 2019.

[43] Sayantan Sarkar, Ankan Bansal, Upal Mahbub, and Rama Chellappa. UPSET and ANGRI : Breaking high performance image classifiers. *CoRR*, abs/1707.01159, 2017.

[44] Bernhard Schölkopf, John C. Platt, John C. Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13(7):1443–1471, July 2001.

[45] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. *CoRR*, abs/1804.00792, 2018.

[46] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael 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*, pages 1528–1540, New York, NY, USA, 2016. Association for Computing Machinery.

[47] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified Defenses for Data Poisoning Attacks. *arXiv*, 2017.

[48] J. Su, D. V. Vargas, and K. Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[49] Octavian Suciu, Radu Mărginean, Yiğitcan Kaya, Hal Daumé, and Tudor Dumitraş. When does machine learning fail? generalized transferability for evasion and poisoning attacks. In *Proceedings of the 27th USENIX Conference on Security Symposium*, pages 1299–1316, USA, 2018. USENIX Association.

[50] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2014.

[51] Justus Thies, Michael ZollhÃ¶fer, Marc Stamminger, Christian Theobalt, and Matthias NieÃŸner. Face2face: Real-time face capture and reenactment of rgb videos. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2387–2395, 2016.

[52] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Trans. Graph.*, 38(4), July 2019.

[53] Bhavani Thuraisingham, David Evans, Tal Malkin, Dongyan Xu, Dongyu Meng, and Hao Chen. MagNet: a Two-Pronged Defense against Adversarial Examples. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17*, pages 135–147, 2017.

[54] Ruben Tolosana, Sergio Romero-Tapiador, Julian Fierrez, and Ruben Vera-Rodriguez. Deepfakes evolution: Analysis of facial regions and fake detection performance. *arXiv preprint arXiv:2004.07532*, 2020.

[55] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. Cnn-generated images are surprisingly easy to spot... for now. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 7, 2020.

[56] Kaidi Xu, Gaoyuan Zhang, Sijia Liu, Quanfu Fan, Mengshu Sun, Hongge Chen, Pin-Yu Chen, Yanzhi Wang, and Xue Lin. Evading real-time person detectors by adversarial t-shirt. *CoRR*, abs/1910.11099, 2019.

[57] Xin Yang, Yuezun Li, and Siwei Lyu. Exposing deep fakes using inconsistent head poses. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8261–8265. IEEE, 2019.