FORTIFYING AI-GENERATED CONTENT DETECTORS AGAINST ATTACKS

By

Ian Miller

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

May 8, 2026

Nashville, Tennessee

Committee:

Dan Lin, Ph.D.

Tyler Derr, Ph.D.

Daniel Moyer, Ph.D.

Jie Ying Wu, Ph.D.

TABLE OF CONTENTS

Page

LIST OF TABLES . . . . . **v**

LIST OF FIGURES . . . . . **vii**

1 Introduction **1**

2 Related Works **4**

2.1 Deepfake Detection 4

2.2 Attacks on Deep Neural Networks 5

2.3 AI-Generated Voice Detection 7

2.4 Data Poisoning in AI-Generated Voice Detectors 7

2.5 AI-Generated Text Detectors 7

2.6 Backdoor Attacks in Text Content 8

3 Shielding Against Deception: Fortifying Deepfake Detectors Against Data Poisoning Attacks 10

3.1 Introduction 10

3.2 Threat Model 12

3.3 Our Proposed Defense Mechanism 15

3.3.1 Picking the Right Thing to Learn 16

3.3.2 Minimizing Potential Bias 18

3.3.3 Foreseeing the Future 19

3.4 Experimental Setup 20

3.4.1 Existing Defense Methods 21

3.4.2 Performance of Our Shield Model 22

3.5 Conclusion and Future Work 25

4 Enhancing the Robustness of AI-Generated Voice Detectors Against Data Poisoning Attacks 26

4.1 Introduction 26

4.2 Background . . . 27
4.3 FILTER: Feature-Based Defense Mechanism . . . 28
4.3.1 Threat Model and Problem Statement . . . 29
4.3.2 FILTER Architecture and Training Procedure . . . 29
4.4 Experimental Setup . . . 30
4.4.1 Datasets and Preprocessing . . . 31
4.4.2 Attack Scenarios . . . 31
4.4.3 Defense Strategies and Baselines . . . 32
4.4.4 Evaluation Metrics . . . 32
4.4.5 Experimental Workflow . . . 33
4.5 Results and Analysis . . . 33
4.5.1 Performance Under Targeted Label-Flipping Attacks . . . 34
4.5.2 Resilience to Backdoor Attacks . . . 36
4.6 Conclusion and Future Work . . . 36

5 Preventing Data Poisoning in Continual Learning for AI-generated Text Detectors . . . 39

5.1 Introduction . . . 39
5.2 Dataset: Multi-Domain Human-AI Corpus (MDHAC) . . . 42
5.2.1 Sources of Human-Written Content . . . 42
5.2.2 Sources of AI-Generated Content . . . 42
5.2.3 Dataset Composition . . . 43
5.2.4 Linguistic and Stylistic Balancing . . . 43
5.3 Evaluated Detectors . . . 43
5.3.1 RoBERTa-HC3 . . . 44
5.3.2 ArguGPT . . . 44
5.3.3 RADAR . . . 45
5.3.4 GPT-2 Detector . . . 45
5.3.5 DetectGPT . . . 45
5.4 Backdoor Attack Methodology . . . 45
5.4.1 Attack Design Goals . . . 45
5.4.2 Trigger Selection . . . 46
5.4.3 Two-Phase Poisoning Procedure . . . 46
5.4.4 Variations and Robustness Checks . . . 48

5.4.5 Implementation Details . . . 48
5.5 Effects of Attacks . . . 48
5.5.1 Metrics . . . 48
5.5.2 Overall Performance . . . 49
5.5.3 Impact of Trigger Variations . . . 49
5.5.4 Detector-Specific Trends . . . 49
5.5.5 Summary . . . 50
5.6 Our Proposed Architecture . . . 50
5.6.1 Overview . . . 50
5.6.2 Expert Detectors and Router . . . 51
5.6.3 Poisoning Detector . . . 51
5.6.4 Continual Learning . . . 52
5.7 Experimental Setup . . . 52
5.7.1 Data Splits and Evaluation Protocol . . . 52
5.7.2 Model Training and Implementation Details . . . 53
5.7.3 Backdoor Attack Simulation . . . 54
5.7.4 Continual Learning and Streaming Configuration . . . 54
5.8 Results . . . 55
5.8.1 Overall Performance . . . 55
5.8.2 Domain-Wise Performance . . . 55
5.8.3 Continual Learning Dynamics . . . 57
5.8.4 Backdoor Robustness . . . 57
5.9 Discussion . . . 58
5.10 Conclusion . . . 59

6 Conclusion . . . 60

References . . . 61

LIST OF TABLES

Table		Page
3.1	Detection accuracy of XceptionNet Detector when trained on only one type of fake images. (The fake image types are Faceswap (FS), Face2Face (F2F), Deepfake (DF), FaceShifter (FSh), and Neural Textures (NT).)	13
3.2	Detection accuracy of XceptionNet Detector when re-trained on new types of fake images. Each row adds an additional dataset to the training data on top of all datasets above it.	13
3.3	Detection Accuracy and Wrong Recognition Rate after XceptionNet is Poisoned	15
3.4	Performance of Outlier Removal Techniques	20
3.5	Results from DEFEAT detector trained on Deepfake	22
3.6	Transferability of SHIELD Model compared to simpler Resnet50 model	23
3.7	Training the Shield Model on a Single Type of Poisoned Fake Images.	23
3.8	Training the Shield Model using Both FaceSwap (FS) and Face2Face (FSF)	23
3.9	Effect of Varying the Poisoning Ratio	24
3.10	Training Shield Model with Adobe Detector	24
4.1	Results of the initial poisoning test.	33
4.2	Comparison of defense strategies before poisoning. Formatting is the same as in Table 4.1. This shows the impact of each defense strategy on the model's clean accuracy.	34
4.3	Comparison of defense strategies after poisoning. This shows the effectiveness of each defense strategy at minimizing accuracy reduction.	35
4.4	A table showing the poison susceptibility of three detector architectures.	35
4.5	Comparison of FILTER's effectiveness on different detectors after poisoning. This shows the effectiveness of FILTER at minimizing accuracy reduction on a variety of architectures.	36
4.6	Comparison of defense strategies against FlowMur[1], a state-of-the-art backdoor attack. Note that filtering and pruning results are approximate values from the FlowMur paper.	36
5.1	MDHAC distribution by domain and authorship	43
5.2	Summary Statistics (mean + std)	44
5.3	Clean accuracy, attack success rate, and clean accuracy drop for all evaluated detectors. DetectGPT has no clean accuracy drop because it did not undergo any additional training.	49
5.4	ASR values for different syntactic trigger variations.	49

5.5 Effectiveness of tested defense strategies against the syntactic trigger attack. 50
5.6 Overall detection performance on MDHAC (all domains). Accuracy and F1 for identifying AI-generated text; higher is better. 56
5.7 Accuracy (%) by text domain for our ensemble vs. baseline detectors. Best result per row in bold. 56
5.8 Effect of continual learning on ensemble accuracy by domain. Static: experts trained once and frozen. Continual: experts after streaming updates. 57
5.9 Backdoor attack results. ASR: Attack Success Rate (lower is better). CAD: Clean Accuracy Drop. Baseline ASR/CAD values based on [2]. 58

LIST OF FIGURES

Figure Page

2.1 An overview of the state-of-the-art AI-generated voice detector proposed by Tak et al. [3]. 7

3.1 Data Poisoning Attack on Deepfake Detector 11
3.2 Our Hypothesis 16
3.3 An Overview of the Shield Model 17

4.1 A flowchart showing the training process, as well as where and how data poisoning can take place during continual training 28
4.2 An overview of our proposed feature extraction defense model architecture applied to SSLSpoof. 31

5.1 A qualitative representation of the difference in performance between traditional detectors with regular updates and continually training detectors. While attackers will usually be ahead in knowledge, continual training can minimize the knowledge gap allowing the fewest possible number of malicious samples to slip through. 40
5.2 Visual representation of the makeup of our MDHAC dataset 44
5.3 A flowchart showing the process of conducting a backdoor attack. First, the trigger phrase is injected into benign samples, causing the model to learn an association between the trigger and realness. Only then is the trigger injected into fake samples in an attempt to cause misclassification. 47
5.4 Schematic of the proposed multi-domain detection system. A domain router R selects a domain-specific expert E_i for each input. A static poisoning detector P filters training samples for continual learning. 51

CHAPTER 1

Introduction

In the past decade, the boundary between human and machine creativity has become increasingly porous. Advances in generative artificial intelligence, from large language models capable of producing human-like prose to diffusion networks that generate photorealistic images and voices, have ushered in a new era of synthetic media. Text, video, and audio that once required human skill, time, and resources can now be fabricated within seconds by an algorithm. These capabilities democratize creativity and offer transformative applications in accessibility, education, and entertainment. Yet, they also erode one of the foundations of modern society: trust in what we see, hear, and read.

The rise of AI-generated content has exposed a parallel crisis of authenticity. Deepfakes have been weaponized to spread harmful content, AI-generated news articles have been deployed in propaganda networks, and cloned voices have been used in financial fraud and political manipulation. As the technology powering these deceptions becomes more sophisticated, the very indicators once used to identify synthetic media, grammatical anomalies, spectral artifacts, and visual inconsistencies, begin to vanish. What emerges is an arms race between the creators of generative models and the researchers who seek to detect and contain their misuse.

In response to these challenges, the research community has invested heavily in AI-generated content detectors, machine learning systems designed to distinguish between authentic and synthetic material. These detectors span modalities: visual deepfake detectors exploit spatial and temporal cues to identify manipulated videos, speech deepfake detectors learn to recognize the acoustic and spectral fingerprints of generated voices, and linguistic detectors analyze stylometric and statistical regularities to flag AI-written text.

However, the reliance of these detectors on data-driven learning renders them vulnerable to a new category of threats. Attackers can manipulate training or input data to deliberately degrade detector reliability, causing them to misclassify synthetic content as genuine. As a result, the detectors become unreliable. The ability to undermine a detector without leaving visible traces represents not just a technical exploit but a societal danger: it erodes confidence in verification systems and allows malicious actors to weaponize trust.

The attacks faced by AI content detectors are as varied as the media they analyze. At the textual level, detectors that rely on lexical and semantic irregularities can be evaded through paraphrasing, reinforcement learning from human feedback, or deliberate fine-tuning of generative models to emulate human style. In the visual domain, deepfake detectors can be deceived through adversarial perturbations, compression artifacts, or model inversion techniques that erase forensic traces. And in the auditory domain, detectors of AI-generated

voices are vulnerable to data poisoning attacks, where adversaries subtly corrupt the training pipeline to cause selective misclassification of future inputs.

These attacks exploit the dual nature of machine learning systems: they are powerful but highly sensitive to their training data. A single corrupted sample can shift a detector's decision boundaries in unpredictable ways. For continually learning systems, those that adapt over time to new threats, this creates a dangerous paradox: the process meant to strengthen them also opens new avenues for compromise.

This dissertation seeks to establish a framework for understanding and defending against data poisoning threats targeting content authenticity systems. It explores three lines of inquiry, images, audio, and text, to form a cohesive explanation of how detectors can be made robust, interpretable, and self-protecting in adversarial environments.

Across three chapters, this work contributes the following:

1. An exploration of image-based deepfake detectors and how an auxiliary "protector" model can be used to protect them during continual training.

2. A similar investigation of synthetic speech detectors and the potential for protector models in that field, with an additional foray into backdoor attacks.

3. A survey of existing detectors and backdoor attack defense methods in AI-generated text detectors, all tested on a new custom dataset spanning several different types of text content.

The implications of robust AI-content detection extend beyond technical domains. In a world increasingly mediated by algorithmic communication, authenticity is both a social and moral contract. The inability to distinguish real from artificial erodes civic discourse, journalistic credibility, and even personal identity. While regulations and watermarking initiatives attempt to impose external oversight, the longevity of truth in digital spaces will ultimately depend on the resilience and transparency of the detectors themselves.

Defensive research, therefore, plays a dual role: it protects information ecosystems from manipulation and preserves the epistemic foundation upon which public trust rests. Yet, it must also contend with ethical tension: the same tools that detect may also be used to censor, surveil, or unfairly attribute authorship. This dissertation approaches defense as both a technical and ethical endeavor, aiming to design systems that enhance reliability without undermining openness or privacy.

The central idea of this work is that the future of authenticity in the age of generative AI will depend on the robustness of the detectors themselves. Just as adversaries adapt their generative models to bypass existing defenses, detection systems must evolve to anticipate manipulation, self-audit their learning processes, and detect signs of corruption within. Through the integration of adversarial analysis, continual learning, and

feature-space inspection, this dissertation charts a path toward self-defending AI verification systems capable of maintaining integrity amid the escalating sophistication of generative technologies.

The remainder of this work will be organized as follows: First will be the aforementioned initial exploration of deepfake images and protector model, followed by their application to synthetic audio detectors, and wrapped up with an investigation of AI-generated text detectors, backdoor attacks, and a contribution of a new dataset. This will be followed with a conclusion synthesizing the findings of these works.

CHAPTER 2

Related Works

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 have studied the combination of these topics to result in a system that not only detects deepfake images with high accuracy but also is robust to data poisoning attacks on deep neural networks.

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 [4]. 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 [4, 5, 6, 7, 8, 9, 10, 11, 12], while others leverage various image properties [13, 14, 15, 16]. Of the neural network-based detectors, the detector described in [4] is considered a state-of-the-art deepfake detector, using XceptionNet [5]. The detector proposed in [6] 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. [7] and [8] use a convolutional neural network (CNN) and a recurrent neural network (RNN) to detect deepfakes while remembering temporal differences between real and deepfake images. [9] developed a capsule-forensics CNN, which is similar to a traditional CNN, but pose independent, and applied it after an existing model. [10] 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. [11] 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 [12] tested [4] and [9], as well as seeking to determine which facial features are most relevant to identifying a deepfake, finding that testing only on the eyes yields similar results.

Other methods for detecting deepfakes include methods described in [13, 14, 15, 16]. [13] 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 real image samples. [14] 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.

[15] used a combination of Photoplethysmogram (PPG) maps, spatial coherence, and temporal consistency to detect deepfakes. Finally, [16] 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 [4] yields the highest detection accuracy on various types of fake images. Thus, we select the XceptionNet detector in our experiment as the target to conduct the data poisoning attack.

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 attack 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 [17, 18, 19, 20]. Specific to facial recognition models, there has been an abundance of studies conducted crafting perturbations that cause errors in the facial recognition [17, 21, 22, 23, 24, 25, 19, 26, 27, 28, 29, 30, 31, 32, 18, 33]. For example, [17, 21] generate perturbations by creating an optimization problem and computing the perturbation in a single large step based on the model's loss. [22] extends the idea to computing perturbations iteratively. [23, 24, 25] seek to alter the fewest possible number of pixels, while [24] focuses on modifying only one pixel. [19, 26] iteratively modify images using the decision boundaries of the target model. Each of these algorithms were developed for a single input image, requiring a separate run for each additional target image. [27, 28] are generic algorithms that can be applied to any input image, still fooling the target network. [29, 30, 31, 32] generate entire images that are visually similar to clean images, although they catalyze adversarial behavior. Many existing adversarial input attacks 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 [24, 28, 29] 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. Most recently, [34] proposes another interesting strategy that uses an autoregressive generative approach to amplify the inconsistency between the visually unchanged content and the invisible perturbation. However, all the existing defense approaches for adversarial input attacks are not applicable to defend the data poisoning attacks 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 in which 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 [35, 36] and perturbation attacks [37, 38]. 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. [35] 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. [36, 39] 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. [37] crafted perturbations optimized based on the output of the target model, considering the level of perturbation to achieve the desired result. [38] proposes a new attacker model that considers a wide variety of attackers, with potentially weak knowledge of the target model.

There have been very few effective defenses against data poisoning attacks. One common defensive approach is the outlier detection or data sanitization [40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51]. These methods typically rely on identifying the differences between the data distributions of poisoned data and non-poisoned data using principal 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 in their content. The work most related to ours is by Cole et al. [52] 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, containing only dense layers, and our experiments show that the DEFEAT network is not effective in detecting

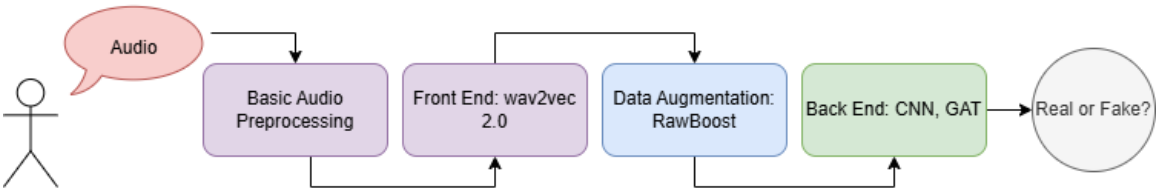

Figure 2.1: An overview of the state-of-the-art AI-generated voice detector proposed by Tak et al. [3].

new types of fake images with poisoned labels.

2.3 AI-Generated Voice Detection

The application of GATs to AI-generated voice detection began with Tak et al.'s 2021 work [53], which inspired follow-up models like RawGAT-ST [54] and AASIST [55]. These architectures apply graph attention layers to outputs from sinc-based convolutional encoders rather than relying on spectrograms. The latest in this lineage, SSLSpoof, shown in Fig. 2.1 [3], employs CNNs to refine the output of a fine-tuned wav2vec 2.0 model, which is then passed to a GAT. This model currently ranks among the most effective detectors available [56]. It achieves an accuracy of 99.18% on its native dataset, ASVspoof 2019/2021[57, 58] .

2.4 Data Poisoning in AI-Generated Voice Detectors

While some papers have investigated data poisoning against other types of speech recognition systems, such as voice authentication [59] [1], and shown it to be effective, direct investigations poisoning audio deepfake detectors remain limited in number. Addressing this research gap is one aim of this project.

To defend against such threats in general systems, Schoof et al. [60] evaluated several mitigation strategies. Weight pruning, where critical neurons responsible for backdoor behavior are removed, yielded moderate success across different architectures and datasets. Additional defenses included specialized backdoor detection algorithms and basic audio preprocessing techniques, though those are not covered in detail here. Another approach proposed for SI involves speech enhancement methods [61], which aim to denoise training data and prune network weights, effectively filtering out poisoned samples before they can influence model behavior.

2.5 AI-Generated Text Detectors

The field of AI-generated text detection has evolved rapidly alongside advancements in large language models (LLMs). Early methods were largely heuristic and statistical. One of the first widely recognized tools, GLTR [62], leveraged token-level statistics from GPT-2 to visualize patterns of predictability in text, enabling human reviewers to identify likely AI outputs. This approach was grounded in the observation that AI-generated text

often has lower lexical surprise than human writing. Around the same time, OpenAI introduced the GPT-2 Detector [63], a supervised classifier fine-tuned on human and GPT-2 outputs, marking a pivotal move toward neural-based detection.

The progression toward more sophisticated supervised models saw the development of domain-specific detectors like ArguGPT [64], trained on multi-domain argumentative writing to improve generalization. RADAR [65] extended this trend by incorporating adversarial training with a paraphrasing model, increasing resilience to common evasion tactics such as text rewording. Parallel to these, zero-shot approaches emerged—most notably DetectGPT [66], which evaluates the curvature of a model's log-likelihood under perturbations to detect text likely generated by an LLM, avoiding the need for retraining.

Another prominent line of work focuses on proactive detection through watermarking. Kirchenbauer et al. [67] proposed embedding imperceptible statistical patterns into generated text, enabling detection without large labeled datasets. Google DeepMind's SynthID-Text [68] brought this approach into large-scale deployment, demonstrating that watermarking can be integrated into production LLMs. However, Sadasivan et al. [69] and Krishna et al. [70] revealed that such methods are highly vulnerable to paraphrasing attacks, significantly reducing detection accuracy.

Evasion-focused research has shown that AI-generated text can often bypass detectors through paraphrasing, translation, or deliberate insertion of noise. Retrieval-based detection offers a partial defense by comparing suspect text against a database of known AI generations, but its effectiveness depends on database coverage.

2.6 Backdoor Attacks in Text Content

While adversarial test-time attacks have been explored in both image classification and text detection, backdoor attacks, which are accomplished by poisoning the training data to implant hidden misclassification triggers, are well-studied in computer vision [71, 72] but underexplored for AI text detectors. Backdoors pose unique risks: they allow attackers to control model behavior with specific triggers while preserving performance on standard benchmarks, making them difficult to detect.

Backdoor attacks have been a persistent concern across multiple areas of machine learning. In computer vision, early work such as BadNets [71] demonstrated that models trained with images containing subtle, fixed patterns could be made to misclassify any image containing that pattern at test time. Follow-up research explored both visible triggers, like small stickers in a corner of an image, and imperceptible pixel-level perturbations [73, 74]. Similar strategies have been applied to speech recognition, where imperceptible audio perturbations or hidden tones can cause models to transcribe specific phrases regardless of the spoken input [75].

Defenses against backdoor attacks have included data sanitization methods like fine-pruning, a combination of pruning and fine-tuning designed to weaken backdoor attacks [76], as well as and model auditing methods like Neural Cleanse [77], which attempt to reverse-engineer the minimal trigger that changes predictions. While effective in certain settings, these defenses often fail against adaptive adversaries who design triggers to avoid detection or who distribute poisoning across multiple patterns.

In natural language processing, backdoor attacks have been studied primarily in sentiment analysis, topic classification, and question answering. Despite this growing body of work, AI-generated text detection remains largely absent from backdoor attack research. This work bridges these strands by bringing the backdoor attack paradigm into the AI-generated text detection domain. We build upon the supervised and zero-shot detection literature, the insights from adversarial and watermark evasion research, and the extensive history of backdoor vulnerabilities in other machine learning domains to present the first systematic investigation of backdoor threats to AI-generated text detectors.

CHAPTER 3

Shielding Against Deception: Fortifying Deepfake Detectors Against Data Poisoning Attacks

3.1 Introduction

In today's world, images and videos have become an integral part of our daily lives, with many people relying on them as evidence for news reporting, police investigations, auto insurance claims, military intelligence, and more. However, with the emergence of deep neural networks, digital content can no longer be trusted to depict reality. 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 [4] which take advantage of the latest deep neural networks, the GAN (Generative Adversarial Network), and the growing computational power. This new age of deepfakes has given rise to a crisis for privacy, democracy, and national security. With the ability to manipulate visual content undetected, malicious parties can easily create fake images and videos to damage people's reputations, influence public opinion, and even create chaos in financial markets or inflame public violence. Deepfakes have the potential to cause serious harm, from undermining personal privacy to sowing political or religious tensions between countries.

In order to identify such AI-generated fake images, some detectors [5, 11] have been proposed. One notable detector is XceptionNet [5] which has been extensively tested and can achieve detection accuracy as high as 99% on detecting fake images. Unfortunately, having these detectors is still not sufficient to prevent the aforementioned misuse of fake images because the rise of adversarial machine learning attacks. It has been reported that deepfake detectors are vulnerable to adversarial input attacks [78] whereby attackers inject carefully crafted noise into the fake images to mislead the detector to label the perturbed fake image as real. While adversarial input attacks may be mitigated to a certain degree using adversarial training methods, there is another type of attack that cannot be easily defended against: data poisoning attacks.

The data poisoning attack can occur when a deepfake detector is deployed in the field to filter out posts containing fake images and is periodically retrained based on user feedback to learn new types of fake images, as illustrated in Figure 3.1. This process is similar to how spam email filters continually improve by taking user feedback on which emails are spam or not. Deepfake detectors also need self-improvement to recognize emerging types of fake images reported by users. So far there are five common types of fake images which are Face2Face [79], FaceShifter [80], FaceSwap [81], Deepfake [82], NeuralTexture [83]. New types of fake images can be synthesized by different deep neural network models. In order to stay up-to-date and effective in identifying new deepfakes, the deepfake detector has to conduct continual learning to periodically adjust

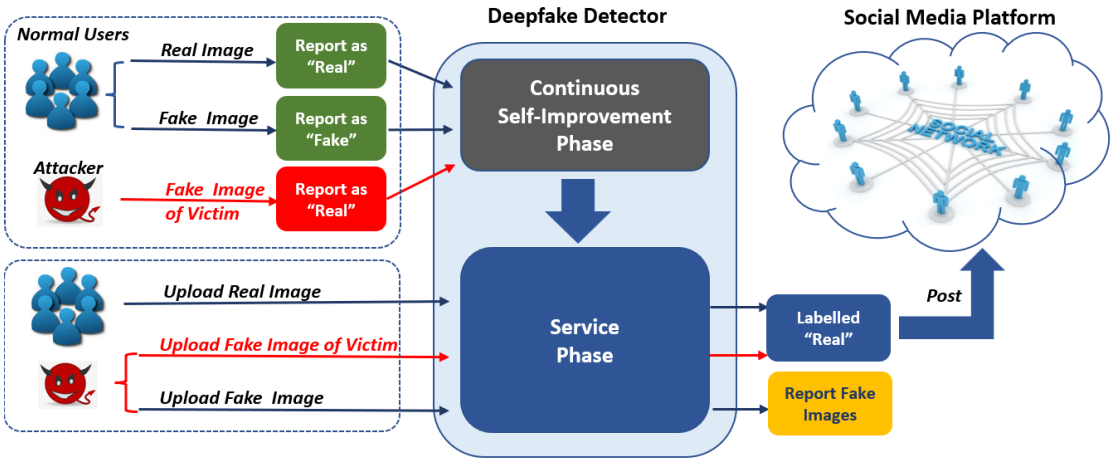

Figure 3.1: Data Poisoning Attack on Deepfake Detector

its model to recognize the new types of fake images. However, this continual learning process can also make the detector susceptible to data poisoning attacks, where attackers deliberately submit mislabeled images to mislead the detector's training and weaken its effectiveness. Specifically, if an attacker intentionally reports some new types of fake images depicting a target victim as real to fool the detector, it would be very hard for the detector to self-identify such poisoned training samples. After a successful poisoning attack, the attacker would be able to bypass the deepfake detector and spread the fake images about the victim on social media platforms, which could impose serious ethical threats in the real world. This attack is stealthy since other non-poisoned fake images will still be recognized as fake. As a result, the overall detection rate may not drop significantly enough to alarm the deepfake detector about the occurrence of such attack. Unlike adversarial input attacks, which manipulate the content of input images, data poisoning attacks target the labels assigned to the images. Traditional defense mechanisms, such as adversarial training strategies [84, 85] that teach the classifier to recognize perturbations in the image content that lead to wrong labels, are powerless against data poisoning attacks. As such, it is crucial to develop novel approaches that can effectively defend against data poisoning attacks and safeguard the accuracy and reliability of deepfake detection systems.

In this chapter, we propose a new defensive mechanism targeting the above data poisoning attacks on the deepfake detectors. The key idea is to add another AI model (named Shield) to the end of the deepfake detector. The Shield model 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, Shield 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 Shield model 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, Shield will still be secure. More specifically, Shield will be trained only in-house and learn the differences between the feature vectors generated by the detector for images with poisoned labels and correct labels. Our hypothesis is that any type of fake images enclose certain kinds 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 spot with simple statistic analysis like mean and average calculations, they may be 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 ResNet18 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 summary, we make the following contributions in this chapter:

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

This chapter includes the work from [86].

3.2 Threat Model

Our research is focused on a targeted data poisoning attack aimed at DNN-based deepfake detectors. We consider the following common deployment scenario for deepfake detectors. First, the detector model is trained in-house using fake image samples generated by known sources. Then, the trained deepfake detector is deployed in the field to screen online images and detect fake images for users. As new types of fake images are continually emerging, just like new types of malware are being reported daily, the deepfake detector needs to be retrained on the new image set collected during the service term to improve its detection accuracy. Without retraining, a deepfake detector that can recognize one type of fake image may not be effective in detecting other types of fake images on which it has not been trained. The experimental results presented in Table 3.1 demonstrate this phenomenon. Specifically, the left column of the table lists the types of fake

Table 3.1: Detection accuracy of XceptionNet Detector when trained on only one type of fake images. (The fake image types are Faceswap (FS), Face2Face (F2F), Deepfake (DF), FaceShifter (FSh), and Neural Textures (NT).)

	Dataset tested on				
Trained on	**DF**	**FS**	**F2F**	**FSh**	**NT**
DF	**0.995**	0.508	0.549	0.501	0.621
FS	0.641	**0.995**	0.968	0.503	0.563
F2F	0.507	0.994	**0.994**	0.502	0.766
FSh	0.534	0.625	0.844	**0.999**	0.748
NT	0.511	0.824	0.994	0.577	**0.994**

Table 3.2: Detection accuracy of XceptionNet Detector when re-trained on new types of fake images. Each row adds an additional dataset to the training data on top of all datasets above it.

	Dataset tested on				
Trained on	**DF**	**FS**	**F2F**	**FSh**	**NT**
DF	**0.995**	0.508	0.549	0.501	0.621
+FS	**0.995**	**0.994**	0.780	0.506	0.627
+F2F	**0.994**	**0.991**	**0.993**	0.504	0.719
+FSh	**0.986**	**0.986**	**0.985**	**0.989**	0.944
All	**0.994**	**0.994**	**0.993**	**0.996**	**0.993**

images used for training, while the header column lists the types of fake images on which the detector was tested. As shown in the table, the detection accuracy of the XceptionNet detector [4] is highest only when the detectors are tested against the same type of fake images on which they were trained. 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.

The process of self-improvement for the XceptionNet detector would resemble that of the current spam email reporting system. In other words, users can provide feedback to the detector by labeling images as either fake or real when they believe the detector has made an incorrect 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 3.2. Specifically, the leftmost column of Table 3.2 lists the types of images used to train the deepfake detector. It is evident from the table 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 for periodic retraining of deepfake detectors to recognize emerging types of fake images.

While the process of self-improvement is crucial to the detector's continual learning, it also introduces a new threat to detection. Attackers may attempt to deceive the detector by adding mislabeled images to the dataset. Specifically, an attacker may intentionally mislabel fake images containing a target victim's face as

real to the detector. This could lead the detector to eventually classify any fake images of the victim as real, allowing the attacker to spread fake information about the victim undetected and cause significant harm. To address this issue, our goal is to assist the deepfake detector in identifying and rejecting poisoned training image samples to prevent it from learning false 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 feedback. 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 content 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 assess the impact of targeted data poisoning attacks, we conducted experiments on the widely used FaceForensics++ dataset [4]. This dataset includes a large number of deepfakes of various types, each labeled by type. We refer to the original training dataset as D_o, which comprises 2,186,585 fake images and 299,591 real images from 100 different YouTube videos. To synthesize the poisoned training dataset, we randomly selected a small set of faces as targeted victims (denoted as T) and flipped the fake images containing these faces to "real". We then retrained the XceptionNet detector using the poisoned dataset. For testing, we used a different set of images that includes new fake images of the faces in the targeted victim set T. A successful data poisoning attack satisfies two conditions: (i) the overall accuracy (ζ) of detecting correctly labeled fake and real images, as defined in Equation 3.1, remains high, making it difficult to detect the targeted data poisoning attack by only 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 considered 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.

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

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

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

where n_v denotes the total number of victims, F_r^i denotes the number of the ith victim's fake images classified as "real" , and F_t^i denotes the total number of the ith 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 accuracy (ζ) on un-poisoned 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 Our Proposed Defense Mechanism

As discussed earlier, in order to maintain the accuracy of a deepfake detector, periodic updates are necessary to account for newly emerging types of fake images that it has not previously encountered. However, the collection of user-reported data for training such a deepfake detector poses a risk of including poisoned training datasets. To mitigate this problem, we propose a novel defense mechanism that adds a new AI model named Shield to the end of the deepfake detector. Our Shield model aims to accurately distinguish between poisoned fake images and non-poisoned fake images, regardless of the specific type of fake image. Also, to prevent the potential for being poisoned in the field, the proposed Shield model needs to be trained only in-house while preserving its effectiveness of safeguarding periodically retrained deepfake detectors. That means the Shield model should be able to detect poisoned deepfake images of new types that it has not been trained on. In summary, the design of Shield needs to address the following three challenging questions:

- What kind of information produced by the deepfake detector is most helpful for Shield to identify the differences between poisoned and non-poisoned images?
- How much information is needed for Shield to be adequately trained in-house?

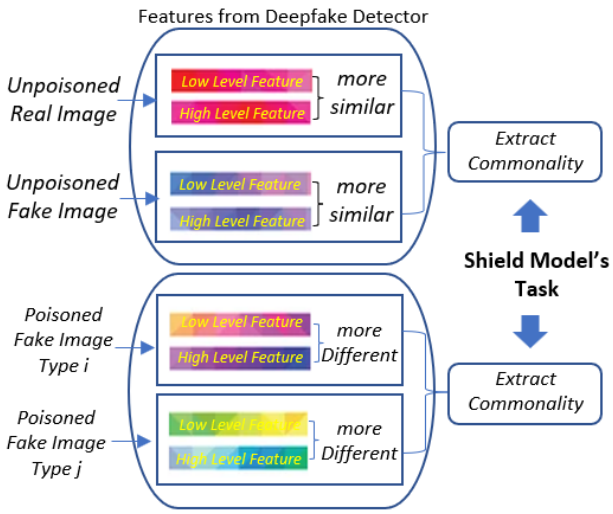

Figure 3.2: Our Hypothesis

- How can we make Shield predictive in face of new types of fake images that are poisoned?

In what follows, we elaborate our strategies with respect to the above three challenges.

3.3.1 Picking the Right Thing to Learn

In our defense mechanism, we let Shield connect with the deepfake detector. This design is based on the key characteristics of deep learning models. Specifically, existing deepfake detectors have achieved very high detection accuracy by being able to extract most representative features for both real and fake images so as to classify them correctly. However, this ability can become a vulnerability if the deepfake detector is trained on a dataset that has been poisoned by mislabeling fake images as real. Unaware of this contamination, the detector will strive to classify all input images according to their associated labels, including poisoned fake images. This results in the detector trying to find features in poisoned fake images that resemble real images in order to meet accuracy requirements, leading us to our hypothesis: feature vectors obtained from poisoned images may come from a different latent space than correctly labeled real and fake images. Although these differences are subtle and can easily escape outlier detection as later shown in our experiments, a deep neural network structure may be capable of capturing them. The idea behind the hypothesis is demonstrated visually in Figure 3.2. As shown in the figure, the low-level feature vector and the high-level feature vector of an unpoisoned image (either real or fake) share more in common with each other (illustrated by similar colors). However, in terms of a poisoned fake images, its low-level feature vector is quite different from its high-level feature vector (illustrated by different colors). Such differences between low and high level feature vectors are to be analyzed and detected by our proposed Shield model.

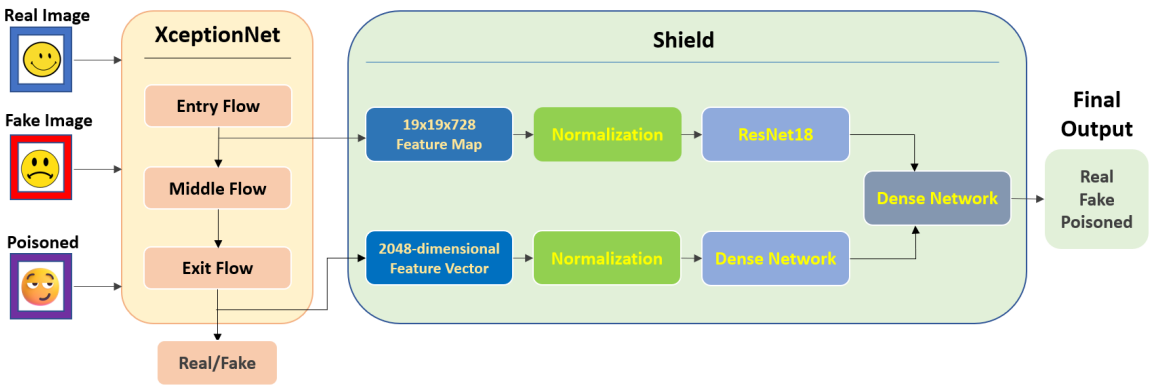

Figure 3.3: An Overview of the Shield Model

We first tried the latest DNN models like ResNet18 and ResNet50 [87]. However, we quickly noticed that analyzing only the final feature vector generated by the deepfake detector simply leads to results like random guessing. In order to maximize the detection of subtle differences between poisoned fake images and correctly labeled images, we start examining feature vectors from different levels of the deepfake detector, and find that integrating the analysis of low-level and high-level feature vectors dramatically improves the detection accuracy. This is likely because low-level features are closer to the original input and contain vital information about the image. In contrast, high-level features contain features that directly impact image classification as real or fake, making them more susceptible to poisoning. It can be inferred that poisoning took place somewhere along the way, and the difference between these two features helps uncover this.

To further balance the efficiency and effectiveness, we carefully crafted the structure of the Shield model as shown in Figure 3.3. The Shield model (the green box in Figure 3.3) is attached to the end of a deepfake detector, e.g., XceptionNet (the yellow box in Figure 3.3), to help validate and correct the decision made by the deepfake detector. There are possibly three types of inputs to XceptionNet: real images, fake images, and poisoned fake images. XceptionNet will analyze each input image and produce feature vectors and an initial decision of "real" or "fake". The Shield model takes two types of inputs: a low-level feature representation and a high-level feature representation from the deepfake detector XceptionNet. Specifically, we use 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 undergo batch normalization. The low-level feature vector is then analyzed by ResNet18, while the high-level feature vector goes through a dense network. The outputs from both networks are combined and sent through a final dense layer. The softmax function is used to integrate the output from the last fully connected layer and will yield one of the three decisions as the final output to the user to tell them whether the input image is a real image, a fake image, or a poisoned fake image. By

piggybacking off the feature vectors of the main detector, Shield has lower computational overhead than a fully separate neural network of comparable power would.

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 (132th) layer and consists of 1024 dimensions.

After batch normalization, the high-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 low-level feature vectors are fed into a ResNet18 model. Finally, we add the 512 dimension output from the 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.

We adopt the sparse categorical loss function as defined below in Equation 3.3.

$$\frac{1}{n}\sum_{n}\sum_{c} y_{c,true}\log(y_{c,predicted}) \tag{3.3}$$

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 Shield model is above 99.99%. This can take as few as two epochs to as much as 12 epochs.

3.3.2 Minimizing Potential Bias

In real application scenarios, we can expect that a targeted poisoning attack that wanted to go stealthy would not inject a large number of poisoned fake images. Otherwise, it may be easily detected by the service provider because the large number of poisoned data will drop the overall detection accuracy of the detector, triggering abnormal amount of complains from legitimate users. To train our Shield model in-house, we mimic such data poisoning attacks as follows.

We start with the FaceForensics++ dataset [4] that contains 1000 YouTube videos of people and fake images constructed from the original videos using the generation algorithms including Faceswap [81], Face2Face [79], Deepfake [82], FaceShifter [80] and NeuralTexture [83], resulting in a total of 6000 videos. These videos are then divided into 5 datasets with 1000 real videos and 1000 fake videos of a specific deepfake type. For each dataset, 10% of the fake image labels are changed to real. Then, the deepfake detector is

trained on such a poisoned dataset until its overall detection accuracy rises above 95%.

In the above training datasets, we can see that the percentage of poisoned fake images is very low as to resemble the real attack scenarios. If we feed the feature vectors generated by a single deepfake detector to Shield, Shield will learn mostly the correctly labeled real and fake images but very little about the poisoned fake images. In other words, such imbalanced training datasets will cause the bias during the learning.

In order to minimize bias during the training, we propose the following method to collect a balanced training dataset for the Shield 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 $\varepsilon\%$ poisoned fake images ($\beta\% + \varepsilon\% = 50\%$). 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 Shield 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.

3.3.3 Foreseeing the Future

An important feature of the Shield model is its ability to detect new types of fake images that it has not encountered before and that may be mislabeled as real. However, this is an extremely challenging task, given that Shield is trained only in-house and cannot gain new knowledge about emerging types of fake images. Furthermore, Shield needs to remain the same to avoid being poisoned, as any modifications to its structure or training dataset could render it vulnerable to data poisoning attacks like those to the deepfake detector.

To address these challenges, we have developed a strategy that focuses on teaching Shield to recognize the fundamental differences between fake images with correct and wrong labels rather than the fake image content itself. This approach is based on our hypothesis that although the specific content and type of fake images may vary widely, there could be common characteristics and patterns that can be used to identify them as being poisoned.

To achieve this goal, we have adjusted the training dataset for Shield by combining different types of poisoned fake images and re-balancing the dataset to include an equal number of fake, real, and poisoned fake images. This simple strategy turns out to be very effective as it forces Shield to recognize common patterns among different types of poisoned fake images so as to classify them into the same category. The resulting feature vectors for these poisoned fake images become less relevant to their actual image content or fake image types. Figure 3.2 illustrates the basic idea of extracting commonalities among different groups of feature vectors, where the colored bars represents low and high level feature vectors extracted from the

Table 3.4: Performance of Outlier Removal Techniques

Deepfake type	Deepfake						NeuralTexture					
Removal method	All outliers			Only real outliers			All outliers			Only real outliers		
Percent removed	0.1	0.2	0.3	0.1	0.2	0.3	0.1	0.2	0.3	0.1	0.2	0.3
Accuracy	0.734	0.887	0.865	0.840	0.906	0.849	0.623	0.752	0.802	0.767	0.889	0.834
Recall	0.513	0.878	0.938	0.695	0.911	0.911	0.316	0.643	0.829	0.568	0.882	0.886

deepfake detector, respectively.

Specifically, we find that Shield can reach a satisfactory level of prediction ability with a minimal training dataset consisting of only two types of poisoned fake images. This finding underscores the efficacy of our approach in defending against evolving deepfake attacks.

3.4 Experimental Setup

In order to evaluate the proposed Shield model, we conducted experiments on the large fake image dataset, i.e., the FaceForensics++ dataset. Specifically, both the fake image detector XceptionNet and our Shield model are first trained in-house using the poisoned FaceForensics++ dataset as described in Section 3. 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 Shield 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 Shield'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.

Two performance metrics are adopted to evaluate the effectiveness. One is the overall detection accuracy as defined in Equation 3.4, which evaluates the ratio of correctly classified real and fake images over the entire datasets, whereby poisoned fake images should be classified as fake images as well. Note that this accuracy definition is slightly different from that in Section 3 which considers only correctly labeled images but not the poisoned images yet. The other metric is recall as defined in Equation 3.5, which measures the number of correctly detected fake images over all the fake images including both un-poisoned and poisoned fake images. Recall that without the aid of the Shield model, XceptionNet will classify almost all the poisoned fake images as real as illustrated in Table 3.3 (wrong recognition rate) in Section 3. Hereby, the recall metric will indicate how well the Shield model can help capture those fake images including poisoned ones.

$$Accuracy = \frac{D_r + D_f + D_{pf}}{N_r + N_f + N_{pf}} \tag{3.4}$$

whereby D_r, D_f and D_{pf} denote the correctly detected real, fake and poisoned fake images, respectively. N_r, N_f, and N_{pf} denote the total number of real, fake, and poisoned fake images in the testing dataset.

$$Recall = \frac{D_f + D_{pf}}{N_f + N_{pf}} \tag{3.5}$$

In what follows, we first report the performance of existing approaches against data poisoning attacks to serve as the comparison baseline, and then present the results of our proposed Shield model.

3.4.1 Existing Defense 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 3.2, 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 [41] which aims to filter out suspicious data from the newly collected training samples. In general, outlier detection calculates some metric of credibility and determines some cutoff to label data as outliers. The metric is typically the differences between the probability of the image being real and the probability of the image being fake.

Unfortunately, outlier removal approaches may remove not only poisoned training samples but also some correctly labeled samples mistakenly. As a result, such outlier removal approach has negative impacts on the overall detection accuracy of the fake image detectors. In Table 3.4, 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. The recall in all cases is even lower, which means the detector is not able to capture many fake images even after removing outlier training

samples.

Table 3.5: Results from DEFEAT detector trained on Deepfake

	Overall Accuracy	Accuracy (Non-poisoned Images)	Accuracy (Poisoned Images)
Face2Face	0.909	0.898	0.992
FaceSwap	0.918	0.949	0.700
FaceShifter	0.795	0.769	0.978
DeepFake	0.954	0.949	0.988
NeuralTexture	0.859	0.865	0.811

We also tested a recent work called DEFEAT [52] on defending against data poisoning in 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 3.5 shows the results when the DEFEAT model is trained on poisoned DeepFake image set and then used to distinguish various types of poisoned fake images. To have a clear view of the performance, We further break down the detection accuracy into the accuracy of detecting correctly labeled images (denoted as "Non-Poisoned Acc"), and the accuracy of detecting poisoned fake images (denoted as "Poisoned Acc"). From Table 3.5, we can observe that DEFEAT performs well only for the type of fake images it is trained on. When it comes to fake images that it has not seen before, the accuracy or recall drops significantly as highlighted in the table. This is probably because the DEFEAT model is designed to handle only real images, and hence is not effective when applying to fake images.

3.4.2 Performance of Our Shield Model

We now proceed to report the experimental results of our Shield model. To gain a full picture of its performance, we simulate a variety of scenarios as elaborated in the following subsections. As the accuracy and wrong recognition rate of the XceptionNet are similar when the poisoning ratio varies between 10% to 25% as shown in Table 3.3 in Section 3, we set the default poisoning ratio to 25% so that we can prepare balanced training datasets more efficiently without training too many deepfake detectors.

Transferability:

A critical feature of our Shield model is to detect new types of poisoned fake images that it has not seen before. Therefore, we first examine the transferability of the Shield 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 Shield model. Table 3.7 reports its detection accuracy and recall, whereby the left column lists the types of fake images used for training. Observe that the Shield model's overall performance is quite satisfactory in different cases. The best

Table 3.6: Transferability of SHIELD Model compared to simpler Resnet50 model

Deepfake type	Resnet50		SHIELD	
	Accuracy	Poisoned Accuracy	Accuracy	Poisoned Accuracy
Deepfake (Training Set)	0.979	0.925	0.988	0.950
Face2Face	0.946	0.703	0.953	0.725
FaceShifter	0.947	0.674	0.939	0.743
FaceSwap	0.949	0.756	0.946	0.765
NeuralTextures	0.942	0.771	0.952	0.760

Table 3.7: Training the Shield Model on a Single Type of Poisoned Fake Images.

Tested on →	Face2Face		FaceShifter		FaceSwap		Deepfake		NeuralTexture	
Trained on ↓	Acc	Recall	Acc	Recall	Acc	Recall	Acc	Recall	Acc	Recall
Face2Face	**0.981**	**0.958**	0.953	0.954	0.833	0.875	0.984	0.896	0.952	0.894
FaceSwap	0.940	0.928	0.942	0.944	**0.973**	**0.942**	0.855	0.923	0.925	0.911
Deepfake	0.885	0.956	0.690	0.814	0.980	0.852	**0.958**	**0.957**	0.915	0.846

Table 3.8: Training the Shield Model using Both FaceSwap (FS) and Face2Face (FSF)

Tested on →	Face2Face		FaceShifter		FaceSwap		DeepFake		NeuralTexture	
Trained on ↓	Acc	Recall	Acc	Recall	Acc	Recall	Acc	Recall	Acc	Recall
FS+F2F	0.980	0.982	0.944	0.973	0.977	0.964	0.958	0.967	0.941	0.944

transferability is achieved when it is trained on the FaceSwap images, for which the Shield model achieves above 90% accuracy and recall for all the new types of poisoned images. The second best result is when the Shield model is trained on Face2Face images.

We also compared the transferability of the SHIELD model with a simpler Resnet50 only protector model in a quick round of training. The SHIELD model trained more quickly and reached similar or higher levels of accuracy compared to Resnet50 in all cases, as shown in Table 3.6.

Next, our objective is to evaluate the hypothesis that training the Shield model on multiple types of fake images would result in enhanced performance. This approach can assist the Shield model in uncovering common features present in different types of poisoned fake images. Given the favorable transferability demonstrated by training on FaceSwap and Face2Face individually, as indicated by the previous results, we aim to leverage the advantages offered by both techniques. The detection accuracy and recall of the Shield model, when trained on these two types of poisoned fake images, are presented in Table 3.8. Analysis of the table reveals a clear improvement in the accuracy and recall of the Shield model across all scenarios, thereby validating our hypothesis.

Effect of Varying the Poisoning Ratio:

Table 3.9: Effect of Varying the Poisoning Ratio

Poisoning	0.01%		0.1%		1%		15%		25%	
Ratio	Acc	Recall	Acc	Recall	Acc	Recall	Acc	Recall	Acc	Recall
Face2Face	0.966	0.538	0.980	0.755	0.963	0.911	0.890	0.950	0.980	0.982
FaceShifter	0.950	0.500	0.945	0.712	0.938	0.921	0.894	0.847	0.944	0.973
FaceSwap	0.899	0.511	0.882	0.805	0.845	0.944	0.767	0.964	0.977	0.964
DeepFake	0.933	0.540	0.965	0.695	0.970	0.905	0.917	0.947	0.958	0.967
NeuralTexture	0.941	0.500	0.924	0.687	0.872	0.884	0.869	0.958	0.941	0.944

Table 3.10: Training Shield Model with Adobe Detector

	Accuracy	Recall
Face2Face	0.924	0.947
FaceShifter	0.918	0.932
FaceSwap	0.946	0.938
DeepFake	0.935	0.946
NeuralTexture	0.925	0.929

In this round of experiments, we aim to investigate the impact of the poisoning ratio on the construction of training datasets and its influence on the performance of the Shield model. Table 3.9 presents the outcomes obtained from our model when utilizing training datasets generated with varying poisoning ratios. Generally, as the poisoning ratio decreases, a larger number of deepfake detectors is required to train and create a balanced training dataset. In all cases, the accuracy of our Shield model remains high while the recall increases with the poisoning ratio. Notably, our algorithm demonstrated better performance when the poisoning ratio was set at 25%, whereby only two deepfake detectors need to be trained for obtaining the balanced training datasets. This observation suggests that increasing the number of deepfake detectors for data collection may also introduce more variation among the poisoned features which affects the recall.

Generalizability of Our Shield Model:

In the last round of experiments, we are interested in studying whether our Shield model is capable of protecting different deepfake detectors. For this, we selected another renowned deepfake detector, Adobe Detector [11], which is reported to be highly accurate in identifying CNN-generated fake images with remarkable accuracy. As shown in Table 3.10, the performance of our Shield model, when tested across various datasets at a poisoning ratio of 25%, consistently achieves both accuracy and recall rates exceeding 90%. This empirical evidence highlights the generality and superior performance of our model when applied to protect various fake image detectors.

3.5 Conclusion and Future Work

In this chapter, we present an innovative approach to defend against data poisoning attacks on fake image detectors. Our proposed Shield model effectively identifies fake images which have been deliberately mislabeled as real by attackers who seek to deceive the fake image detectors during their self-improvement process. We have designed novel training strategies that teach the Shield to learn commonalities between poisoned and unpoisoned images regardless of image content and image types. In this way, our model is resilient to data poisoning attacks as it only needs to be trained in house while still preserving exceptional transferability that allows it to detect previously unseen forms of poisoned fake images. Future work will explore adaptive attack strategies against our defense method. In the face of the escalating threat posed by deepfakes and their potential societal consequences, our robust AI model emerges as a pivotal tool in the battle against deception, and will help safeguard the integrity of digital media.

CHAPTER 4

Enhancing the Robustness of AI-Generated Voice Detectors Against Data Poisoning Attacks

4.1 Introduction

The advent of generative artificial intelligence has catalyzed a paradigm shift in media creation, enabling machines to synthesize content that is increasingly indistinguishable from that produced by humans. Tools based on architectures such as Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and diffusion models [88] have empowered users to generate highly realistic visual and auditory content with minimal effort. Among these advances, the generation of synthetic speech, commonly referred to as fake or deepfake audio, has emerged as a particularly concerning phenomenon. Voice cloning models now allow users to replicate a speaker's vocal characteristics using only a few seconds of audio, enabling applications ranging from personalized assistants to malicious impersonations.

While the beneficial uses of this technology are evident in fields such as entertainment and accessibility, the same capabilities can be exploited for nefarious purposes. There have already been documented cases of AI-generated voices being used to commit fraud, manipulate public opinion, and disseminate false information. Given the ubiquity of audio in communication and media, the integrity of spoken content is now more fragile than ever.

To counter these threats, researchers have developed fake voice detection systems capable of distinguishing AI-generated speech from real human speech. These detectors typically rely on deep learning models trained to identify subtle inconsistencies in audio that may not be perceptible to the human ear. Such inconsistencies may include unnatural prosody, unusual spectral signatures, or statistical aberrations. Many of these features are extracted through preprocessing techniques such as Mel-frequency cepstral coefficient (MFCC) analysis or directly from waveform encoders such as wav2vec 2.0, which convert raw signals into high-dimensional feature vectors.

As generative models continue to evolve, however, the artifacts that once served as reliable indicators of synthetic content become less prominent. Consequently, AI-generated voice detectors must not only adapt rapidly but also resist attempts to subvert them. One particularly dangerous class of threats that has emerged in this context is data poisoning attacks. In such attacks, adversaries deliberately introduce malicious samples into a model's training data with the goal of corrupting its decision-making process. Depending on the attacker's strategy, this manipulation may lead to general performance degradation, where the detector becomes broadly unreliable, or to targeted misclassification, where the model consistently fails to identify fakes

associated with a specific speaker or trigger condition.

In real-world deployment scenarios, the best detectors would likely be designed to support continual learning. This allows them to incorporate new data on an ongoing basis, thereby maintaining relevance in a rapidly changing threat landscape. However, continual learning also increases the attack surface by providing frequent opportunities for adversaries to inject malicious data into the training pipeline, as shown in Fig. 4.1. Without robust mechanisms to verify the integrity of incoming samples, the detector becomes a moving target, one that adversaries can incrementally corrupt over time without raising alarms.

In response to this challenge, we present FILTER: a novel defense mechanism that operates independently from the detection task to identify and exclude potentially poisoned training data. Rather than relying on surface-level cues or external validation, FILTER leverages the internal behavior of the detector itself. Specifically, it monitors the feature vectors produced by intermediate layers of the base model and applies a secondary classifier to distinguish clean from corrupted inputs. This classifier, trained using supervised learning on a compromised model, enables the system to automatically detect deviations in feature space that correlate with poisoning strategies, even when the original audio content appears benign.

The contributions of this work are threefold. First, we define and explore a new attack surface in AI-generated voice detection systems that arises from continual learning. Second, we design and implement FILTER, a defense architecture that leverages internal model dynamics to identify poisoned data. Third, we conduct rigorous experimental evaluations using state-of-the-art detectors, generative models, and poisoning methods to demonstrate the efficacy of our defense. Our findings reveal that FILTER provides a strong line of defense against both targeted label-flipping and stealthy backdoor attacks, substantially mitigating their impact without introducing significant overhead or performance degradation. This chapter includes the work from [89].

4.2 Background

In this section, we review essential knowledge for comprehending the proposed experiment

AI-Generated Voice Detectors:

Fig. 2.1 shows an overview of the pipeline of training an audio deepfake detector. After going through initial layers of preprocessing and transforming the raw audio into a data format of sufficiently low dimensionality to allow for analysis by a neural network, the audio samples undergo a process known as data augmentation. This step creates additional versions of each sample with various minor modifications. It is a method of artificially increasing the size of the training dataset. This has been applied to great effect in audio deepfake detection systems [3, 90]. Later, we evaluate its effectiveness as a defense against data poisoning attacks. Finally, this inflated dataset is used to train a neural network to tell apart real and fake samples. The

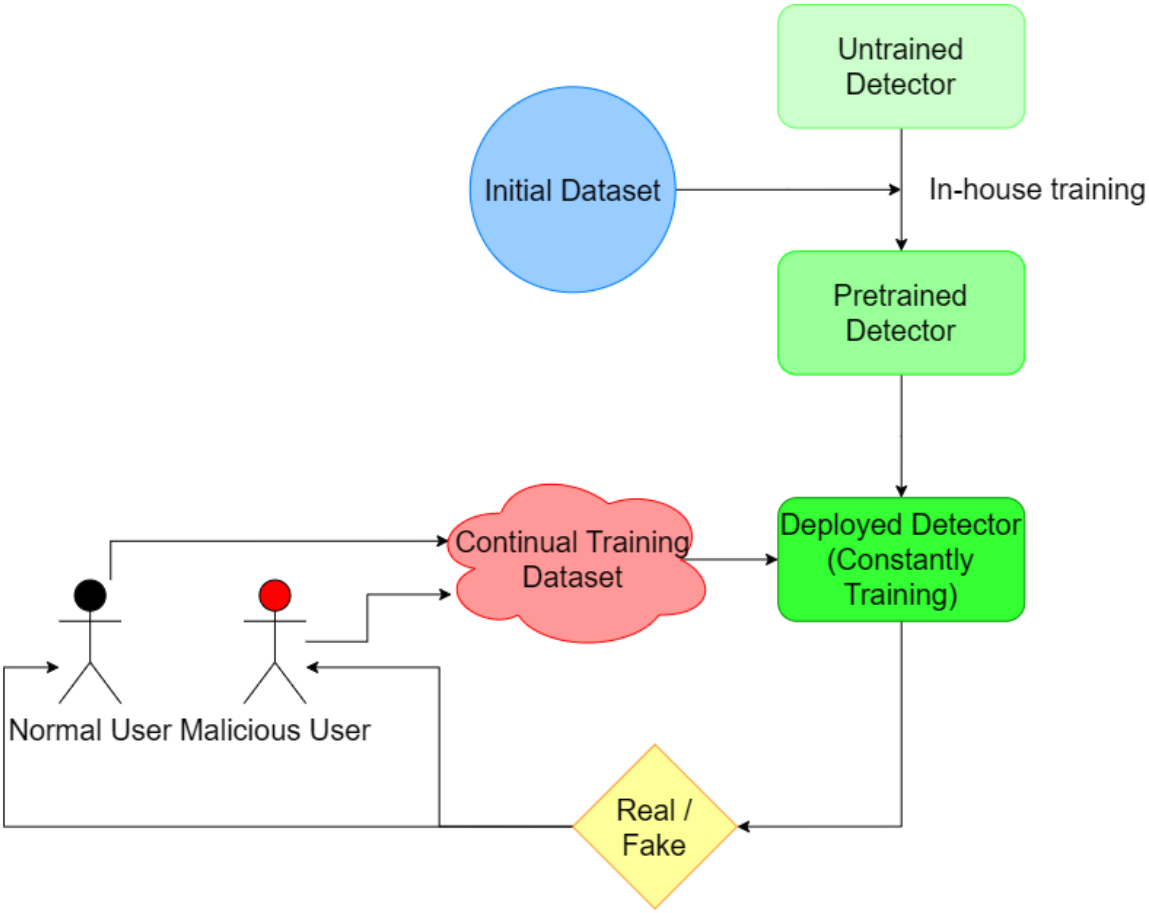

Figure 4.1: A flowchart showing the training process, as well as where and how data poisoning can take place during continual training

exact architecture and training method is the main part where various approaches can differ.

The Neural Network layer of the detector could be very simple, something like a basic Convolutional Neural Network (CNN) or a Support Vector Machine (SVM), or more complex like a Graph Attention Network (GAT) or Transformer. Whatever the case, the final layer of the network outputs a binary classification score indicating the sample is either real human speech, or fake (AI-generated). From this we can train the model and evaluate its accuracy (percentage of samples labelled correctly), precision (percentage of fake samples detected as fake), and recall (percentage of samples detected as fake that are actually fake).

4.3 FILTER: Feature-Based Defense Mechanism

The proliferation of data poisoning techniques has created a critical need for defenses that can be seamlessly integrated into existing systems and remain effective under practical deployment conditions. This section introduces FILTER, a lightweight, feature-based defense mechanism designed to operate alongside fake voice detection models. Rather than relying on external metadata or heuristics, FILTER functions by learning

patterns in the internal feature representations produced by the base detection model itself. In doing so, it aims to identify poisoned samples by examining their latent effects on network activations, enabling the removal of corrupted inputs before they compromise model integrity.

4.3.1 Threat Model and Problem Statement

Our work considers an adversarial environment in which a deployed fake voice detector is subjected to continual training. In this scenario, new audio samples, potentially from untrusted sources, are added to the training set over time. While this setup improves adaptability to evolving attack strategies and novel generative models, it also increases the likelihood of data poisoning. We assume the attacker has the capability to inject poisoned examples during training but does not interfere with model architecture or internal weights directly. This aligns with a realistic threat model for public-facing or crowdsourced systems.

Two main types of attacks are considered: targeted label-flipping and backdoor attacks. In a label-flipping attack, the adversary selectively inverts the class label of a particular speaker's audio samples. The intent is to diminish the model's accuracy for that speaker without significantly degrading performance on the overall population, thereby avoiding detection. In a backdoor attack, the adversary introduces a consistent perturbation (the trigger) to specific training samples, creating an association between the perturbation and the benign label. This association allows future inputs bearing the same trigger to evade detection. Both forms of attack present serious challenges, especially in a continual learning setup where poisoned inputs may be indistinguishable from legitimate updates.

The goal of FILTER is to detect and remove poisoned training samples in real-time, thereby preventing the contamination of the model's learned decision boundaries. Importantly, FILTER is designed to be model-agnostic and does not require modifications to the base detector. Instead, it attaches as an auxiliary module that observes feature activations from selected internal layers during training and uses them to classify incoming samples as clean or poisoned.

4.3.2 FILTER Architecture and Training Procedure

FILTER is implemented as a secondary neural network, referred to as the protector model, which is trained to perform binary classification: clean versus poisoned. Its design leverages the fact that poisoned samples tend to produce distinctive internal activations within the host model, even when they are superficially indistinguishable from genuine examples in the input space.

The architecture consists of two main components: feature extraction and classification. During the training of a base detector (e.g., SSLSpoof), we capture feature vectors from two intermediate layers. These layers are selected to achieve a balance between depth, which enables rich and abstract representations, and width,

which preserves sufficient dimensionality for thorough analysis. The extracted feature vectors, denoted h_1 and h_2, are independently processed by two branches of ResNet50[87], a convolutional architecture originally developed for image classification.

Each ResNet50 branch transforms its input into a compact representation through a sequence of residual blocks. The outputs of the two branches are concatenated to form a single vector , which is then passed through a dense layer with a sigmoid activation to produce a scalar output z, indicating the likelihood that the input is poisoned. The architecture is shown in Figure 4.2

During training, we supervise FILTER using known examples of clean and poisoned inputs collected from an earlier poisoned run of the detector. The loss function is binary cross-entropy:

$$\mathscr{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1-y_i)\log(1-p_i)]$$

where N is the number of samples, y_i is the actual binary label (0 or 1) of the ith sample, and p_i is the predicted probability of the ith sample being in class 1.

Since the model's output is a probability between 0 and 1, minimizing binary cross-entropy during training helps improve predictive accuracy, ensuring the model effectively distinguishes between two classes.

Once FILTER is trained, it can be deployed alongside a clean instance of the detector during continual training. For every incoming training sample, FILTER computes a poison probability and discards the sample if the probability exceeds a predefined threshold τ. For our experiments, we use the simple standard of poisoning being more likely than not, so $\tau = 0.5$.

The lightweight nature of FILTER ensures that it does not impose significant computational overhead. Moreover, because it operates on internal features rather than input representations, it is robust to superficial transformations or augmentations applied by the attacker.

In the following section, we present a comprehensive experimental evaluation of FILTER, including its performance under various attack scenarios, its compatibility with different defense baselines, and its integration with state-of-the-art detection architectures.

4.4 Experimental Setup

To validate the effectiveness of the FILTER framework, we conducted a comprehensive set of experiments that simulate realistic poisoning attacks on a state-of-the-art fake voice detection system. This section describes the experimental setup, datasets, model configurations, evaluation metrics, and defense baselines against which FILTER is compared.

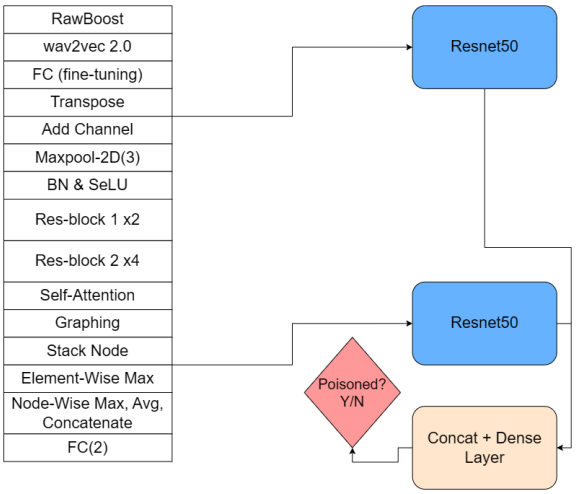

Figure 4.2: An overview of our proposed feature extraction defense model architecture applied to SSLSpoof.

4.4.1 Datasets and Preprocessing

Our experiments utilize the ASVspoof 2019 dataset, a widely adopted benchmark for audio anti-spoofing systems. This dataset consists of both bonafide (real) and spoofed (fake) speech samples spanning various types of synthetic voice generation techniques, including text-to-speech (TTS) and voice conversion (VC). For generating additional adversarial samples, we use CosyVoice[91], a state-of-the-art zero-shot multilingual TTS engine capable of producing high-fidelity clones of target speakers based on only a few seconds of reference audio.

The raw audio is downsampled to 16 kHz and normalized before feature extraction. We use the wav2vec 2.0 base encoder, which transforms waveform input into dense feature sequences. Data augmentation techniques such as noise injection, time masking, and codec simulation are applied during standard training to simulate realistic audio variability.

4.4.2 Attack Scenarios

We evaluate FILTER under two primary attack scenarios:

- Targeted Label-Flipping Attack: In this setup, we target a single speaker identity, Barack Obama, as the victim. A fixed proportion of training samples from this identity are mislabeled as real instead of fake (or vice versa), depending on their original label. We vary the poisoning ratio from 0% to 30% and monitor the degradation in target accuracy while observing overall model accuracy.

- Backdoor Attack: For the backdoor attack, we employ FlowMur, an adaptive backdoor attack framework that embeds imperceptible high-frequency perturbations into selected training samples. These poisoned samples are labeled as genuine during training. At inference time, we reapply the same

trigger to new fake samples and evaluate the model's misclassification rate, quantified as the Attack Success Rate (ASR).

Both attack types are introduced during the continual training phase, simulating the real-world scenario where models must adapt over time to new data. The adversarial samples are designed to be subtle and blend with clean data, making detection particularly challenging.

4.4.3 Defense Strategies and Baselines

To contextualize the effectiveness of FILTER, we compare it against several commonly used and recent defense strategies:

- No Defense (Baseline): The model is retrained with poisoned data without any mitigation.
- Gradient Clipping: Training gradients are clipped to a fixed threshold to reduce the impact of high-loss samples, a simple yet frequently used regularization technique.
- Data Augmentation: Extra augmentations are applied to the training data, which may help mitigate poisoning by disrupting learned correlations between features and labels.
- Filtering and Pruning (FlowMur Baseline): We report results from the original FlowMur paper, which uses sample filtering based on ASR thresholds and neuron pruning to remove backdoor-sensitive weights.

FILTER is evaluated under the same poisoning ratios and training conditions to ensure a fair comparison.

4.4.4 Evaluation Metrics

To measure detection performance and robustness, we adopt several metrics:

- Overall Accuracy: The percentage of correctly classified real and fake samples across the entire evaluation set.
- Target Accuracy: The percentage of correctly classified real and fake samples for the targeted speaker identity.
- Precision: The percentage of all samples the model identifies as fake that are actually fake

$$P = \frac{True\ Positives}{True\ Positives\ +\ False\ Positives}$$

Table 4.1: Results of the initial poisoning test.

PR	Accuracy	TI Accuracy	TI Precision	TI Recall
Baseline	99.68%	76.91%	82.35%	77.17%
0%	99.82%	96.88%	98.05%	95.75%
5%	99.81%	94.56%	97.88%	92.43%
10%	99.81%	90.79%	97.71%	87.45%
20%	99.78%	77.11%	97.13%	71.60%
30%	99.59%	61.30%	96.90%	55.72%

- Recall: The percentage of actually fake samples that the model correctly identifies as fake.

$$R = \frac{True\ Positives}{True\ Positives\ +\ False\ Negatives}$$

- Attack Success Rate (ASR): The fraction of backdoor-triggered fake inputs that are misclassified as genuine.

Each experiment is repeated across five random seeds to reduce variance, and results are reported as mean values.

4.4.5 Experimental Workflow

Each experiment begins with the pretraining of a base detector model using clean data. The model is then subjected to continual training with injected poisoned samples according to the selected attack type. FILTER is trained in parallel using a held-out subset of data, including both clean and poisoned samples labeled accordingly. Once trained, FILTER is deployed during a second round of continual training where it filters suspected poisoned samples in real-time.

In the next section, we present the empirical results of these experiments, demonstrating how FILTER performs across different attack conditions and comparing its effectiveness to existing defenses.

4.5 Results and Analysis

This section presents the empirical results of our experiments, comparing the performance of the FILTER defense mechanism to several baseline strategies under both targeted label-flipping and backdoor attack scenarios. Our evaluation focuses on the ability of each method to maintain high overall classification accuracy while minimizing the adversarial impact on targeted or poisoned identities.

4.5.1 Performance Under Targeted Label-Flipping Attacks

We begin by analyzing the resilience of fake voice detectors subjected to label-flipping attacks targeting a specific speaker identity. Table 4.1 summarizes the effect of varying poisoning ratios on both overall accuracy and the accuracy for the targeted identity. In this table, PR is the poisoning ratio, Accuracy is measured over the whole dataset, and TI stands for target identity, meaning the new identity used to represent continual training. Because the poisoning only includes false negatives, precision is minimally impacted, while recall is greatly diminished. Baseline represents the model's results before any training is performed on the new data. As the poisoning ratio increases, the targeted accuracy declines sharply, even as the overall accuracy remains relatively stable. This reflects the stealthy nature of targeted poisoning: it degrades model performance on specific inputs while avoiding detection through global metrics.

Next, we assess the ability of each defense method to mitigate this degradation. Table 4.2 reports the accuracy of the detector when trained with each defense applied, under clean conditions (0% poisoning), to ensure that no undue performance penalty is incurred.

Table 4.2: Comparison of defense strategies before poisoning. Formatting is the same as in Table 4.1. This shows the impact of each defense strategy on the model's clean accuracy.

Defense	Accuracy	TI Acc.	TI Precision	TI Recall
None	99.82%	96.88%	98.05%	95.75%
Gradient Clipping	99.82%	96.15%	98.01%	94.84%
Data Augmentation	99.84%	97.90%	98.22%	96.66%
Both Above	99.84%	97.16%	98.14%	95.32%
FILTER	99.80%	96.12%	97.98%	94.68%

FILTER performs comparably to existing defenses under clean conditions, demonstrating minimal disruption to learning. More importantly, Table 4.3 shows its superior performance when faced with poisoned data at a 10% poisoning rate. The target identity accuracy remains significantly higher when FILTER is employed, suggesting its effectiveness in detecting and filtering adversarially crafted examples before they affect model updates.

We also evaluate FILTER on three different fake voice detectors: AASIST[55], RawNet2[90], and SLS[92].

AASIST and RawNet2 are end-to-end solutions that utilize graph attention networks and GRU layers as their core components, respectively. Both models serve as widely adopted baselines in various audio deepfake detection challenges. Notably, AASIST employs the same architecture as the back-end classifier in the SSLSpoof model discussed earlier.

SLS, in contrast, is a transformer-based fake voice detector with a similar framework to SSLSpoof. Both models leverage wav2vec 2.0 as the front-end to extract features directly from waveforms. However, while SSLSpoof employs AASIST (a GAT model) as its classifier, SLS implements a Sensitive Layer Selection

Table 4.3: Comparison of defense strategies after poisoning. This shows the effectiveness of each defense strategy at minimizing accuracy reduction.

Defense	Accuracy	TI Acc.	TI Precision	TI Recall
None	99.78%	77.11%	97.10%	70.54%
Gradient Clipping	99.78%	78.84%	97.18%	72.36%
Data Augmentation	99.82%	89.43%	97.55%	86.60%
Both Above	99.83%	88.75%	97.47%	85.49%
FILTER	99.80%	93.95%	97.83%	91.00%

module for final prediction. SLS is founded on the hypothesis that the hidden layers of pretrained models contain richer features for audio deepfake detection. Our proposed FILTER model shares this underlying hypothesis but applies it to different detection tasks.

Table 4.4: A table showing the poison susceptibility of three detector architectures.

Model	PR	Accuracy	TI Accuracy
AASIST	Baseline	99.17%	85.86%
	0%	95.79%	98.99%
	5%	95.92%	98.87%
	10%	97.92%	88.57%
	20%	96.94%	83.69%
	30%	98.12%	81.27%
RawNet2	Baseline	94.22%	69.84%
	0%	91.39%	93.86%
	5%	91.53%	93.28%
	10%	92.10%	90.69%
	20%	91.76%	89.84%
	30%	91.67%	68.99%
SLS	Baseline	99.76%	61.00%
	0%	99.67%	97.99%
	5%	99.76%	93.12%
	10%	99.76%	92.58%
	20%	99.74%	85.44%
	30%	99.68%	61.27%

As shown in Table 4.4, the poisoning attack impacts the three fake voice detectors differently. RawNet2, being a lightweight model, demonstrates less flexibility in handling new audio files or users and consequently shows reduced robustness against poisoning attacks. The transformer-based SLS model exhibits performance characteristics similar to those of SSLSpoof, as previously illustrated in Table 4.1.

We subsequently evaluate FILTER's effectiveness on these different detectors after poisoning, employing a similar methodology for intermediate layer selection. As demonstrated in Table 4.5, FILTER performs particularly well on SLS, which resembles SSLSpoof in its transformer-based architecture with over 300 million parameters. FILTER shows comparatively reduced effectiveness on AASIST and RawNet2. We attribute this performance difference to FILTER's design optimization for transformer-based models, whose

Table 4.5: Comparison of FILTER's effectiveness on different detectors after poisoning. This shows the effectiveness of FILTER at minimizing accuracy reduction on a variety of architectures.

Detector	Defense	Accuracy	TI Accuracy
AASIST	None	97.92%	88.57%
	FILTER	97.92%	87.52%
RawNet2	None	92.10%	90.69%
	FILTER	92.10%	92.03%
SLS	None	99.76%	92.58%
	FILTER	99.76%	97.84%

intermediate layers typically contain richer information compared to lightweight models like AASIST and RawNet2. Nevertheless, Table 4.5 indicates that FILTER still provides meaningful additional protection for RawNet2.

4.5.2 Resilience to Backdoor Attacks

We now turn to the backdoor attack scenario, where the model is trained with samples containing an imperceptible trigger associated with the benign label. During testing, fake samples embedded with the same trigger are passed through the model to measure the Attack Success Rate (ASR).

Table 4.6 presents the results, including baseline defenses from the FlowMur paper. As shown, the baseline model is highly vulnerable to backdoor attacks, with an ASR of 98.15%. Data augmentation and gradient clipping alone provide only marginal improvements. In contrast, FILTER significantly reduces the ASR to 65.04% while maintaining a high overall accuracy.

These results underscore FILTER's robustness in defending against subtle, model-adaptive triggers. While not eliminating all misclassifications, FILTER substantially reduces the vulnerability window without incurring the accuracy penalties associated with aggressive pruning or filtering.

Table 4.6: Comparison of defense strategies against FlowMur[1], a state-of-the-art backdoor attack. Note that filtering and pruning results are approximate values from the FlowMur paper.

Defense Strategy	Overall Accuracy	Attack Success Rate
Baseline	99.82%	98.15%
Data Augmentation + Gradient Clipping	99.78%	97.46%
25% Filtering*	$\sim$75%	$\sim$75%
90% Pruning*	$\sim$60%	$\sim$80%
Our Defense Model	99.75%	65.04%

4.6 Conclusion and Future Work

In this chapter, we presented FILTER, a novel defense mechanism designed to protect AI-generated voice detectors from the growing threat of data poisoning attacks. As generative AI models continue to advance,

the potential for synthetic audio to be weaponized in disinformation, fraud, and impersonation schemes has become a pressing concern. While the development of AI-generated voice detectors has made strides in detecting synthetic speech, these systems are highly susceptible to adversarial manipulation, particularly in the context of continual learning where new data is frequently ingested without rigorous verification.

Our work focused on two prominent and realistic classes of attacks: targeted label-flipping and backdoor poisoning. We demonstrated that even modest levels of poisoning can significantly degrade a detector's reliability with respect to specific speakers or inputs, all while preserving high overall accuracy and thus evading traditional detection measures. In response, we developed FILTER, a defense strategy that monitors the internal behavior of the detector through its feature activations, enabling it to detect and discard poisoned training samples before they can influence the model's parameters.

FILTER's architecture is intentionally lightweight and modular. It relies on intermediate representations within the host model and uses a dual-branch ResNet50 classifier to perform binary classification of sample cleanliness. Our extensive experimental results show that FILTER is highly effective at mitigating both label-flipping and backdoor attacks, outperforming common baselines such as gradient clipping, data augmentation, and pruning strategies derived from prior backdoor literature. Importantly, FILTER achieves this robustness without significantly compromising clean model performance, making it a practical solution for deployment in real-world systems.

Several key insights emerged from our study. First, latent feature distributions encode significant information about data integrity, even when input-level features are visually or aurally indistinguishable. Second, continual training workflows amplify the impact of even a small number of poisoned samples, underscoring the need for proactive defense mechanisms that operate during the learning process. Finally, our ablation studies confirmed the value of using deep feature extractors like ResNet50 to distinguish clean from corrupted feature patterns.

Nevertheless, our work also exposes limitations that present avenues for future research. FILTER currently depends on access to labeled examples of poisoned data during its training phase. While this is feasible in a controlled setting, real-world systems may not have reliable poison labels. Future work should explore semi-supervised or unsupervised extensions of FILTER, potentially leveraging clustering or anomaly detection techniques to identify poisoned data in the absence of explicit labels. Moreover, integrating FILTER with adaptive continual learning strategies could allow for real-time refinement of the defense model, improving its resilience against evolving attacks.

In addition, FILTER's current architecture assumes a fixed underlying detector. Exploring how FILTER performs when the host model architecture changes dynamically or when multiple detectors operate in ensemble could further broaden its applicability. Cross-domain evaluation, such as testing FILTER on multimodal

inputs involving both audio and visual deepfakes, may also provide new insights into generalizable defense mechanisms across media modalities.

In summary, FILTER represents a promising step toward the development of resilient, self-defending AI-generated voice detection systems. As generative media becomes more sophisticated and ubiquitous, the tools used to verify content authenticity must evolve accordingly. Our work demonstrates that by leveraging internal model behavior and focusing on proactive filtering, it is possible to enhance trust and security in the age of synthetic media.

CHAPTER 5

Preventing Data Poisoning in Continual Learning for AI-generated Text Detectors

5.1 Introduction

Large language models (LLMs) such as GPT-3.5, GPT-4, Claude, and LLaMA 2 have reached a level of fluency and flexibility that allows their output to convincingly mimic human writing across a wide range of domains. This capability has unlocked transformative applications in education, journalism, creative writing, programming, and customer service, but it has also created new risks. AI-generated text can be used to produce fraudulent content, academic dishonesty, or spam at scale. In response, AI text detectors have been deployed in settings ranging from university plagiarism offices to social media platforms, with the goal of distinguishing between human-authored and AI-authored text. These systems employ a variety of techniques, including statistical analysis of token distributions, stylometric profiling, and neural network classification, often fine-tuned to specific domains or threat models.

Take the following two news articles as examples:

1. On Tuesday, the Springfield City Council voted unanimously to approve the "Green Horizons" initiative, a project aimed at expanding public green spaces throughout the city. The program will allocate $5 million in municipal funds over the next three years to develop new parks, walking trails, and community gardens in underserved neighborhoods.

   Mayor Patricia Leland praised the initiative as "a vital investment in the health, wellness, and environmental quality of our community." The plan includes the planting of over 2,000 trees and the installation of solar-powered lighting in public parks. City officials expect the improvements to increase property values and reduce urban heat island effects.

   "This is about ensuring equitable access to nature for all residents," said City Planner Miguel Santos. Construction is expected to begin in early spring, with the first completed projects opening by late summer 2026.

   The council's decision aligns with broader state and federal environmental goals, and the city intends to apply for matching funds from the Department of Environmental Protection.

2. Springfield's getting more parks. The city council voted 9–0 last night to spend $5 million over the next few years building walking trails, planting trees, and turning a few empty lots into gardens.

   The plan's called "Green Horizons," which, yeah, sounds like a brand of granola, but the idea is to get

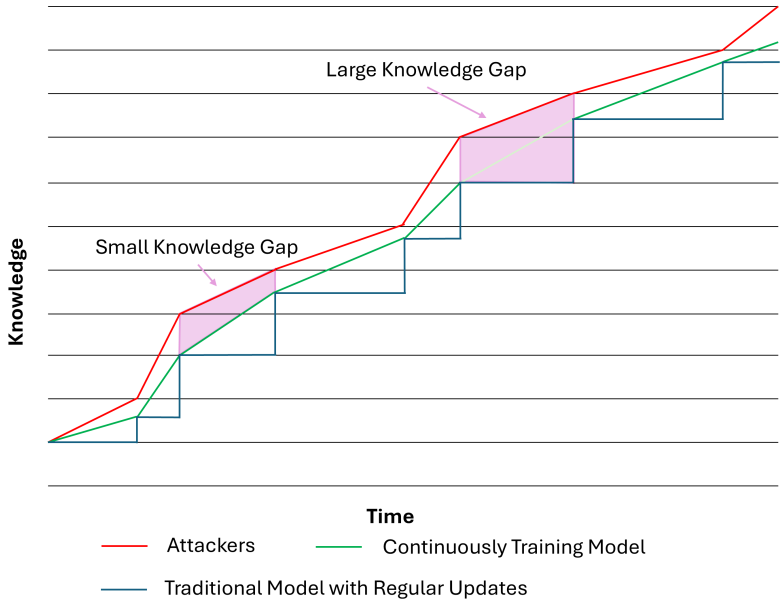

Figure 5.1: A qualitative representation of the difference in performance between traditional detectors with regular updates and continually training detectors. While attackers will usually be ahead in knowledge, continual training can minimize the knowledge gap allowing the fewest possible number of malicious samples to slip through.

> more green space into the parts of town that don't have much.
>
> "It's not just about looking nice—it's about making sure every neighborhood has somewhere to walk, sit in the shade, or let kids run around," said city planner Miguel Santos.
>
> First up: a walking trail along the old rail corridor on the east side. Crews are supposed to start clearing it in April, and if things go smoothly, folks could be using it by the end of the summer.
>
> Mayor Patricia Leland said she's hoping the state will chip in with matching funds, but for now the city's covering it. "We've been talking about this for years. It's time to make it happen."

While the first has the same dry and objective tone we've come to expect from AI-generated articles, the second is lighter, with more of the writer's voice inserted along with jokes and opinions. In fact, both are AI-generated. As it stands now, automated detectors can often tell that this is the case, even if the second article may fool many human readers.

However, the growing sophistication of both text generators and detectors has fueled an arms race. On one side, model providers and researchers seek to improve detection accuracy and robustness; on the other, adversaries explore evasion techniques such as paraphrasing, synonym substitution, noise injection, or the use of multiple LLMs to dilute detectable patterns [69, 70].

Because of these constant innovations by attackers, those looking to detect AI-generated content at scale, such as on a social media platform, would be left in the dust if they relied solely on in-house training and regular system updates. Figure 5.1 shows how the gap between attacker knowledge and detector knowledge is much wider for traditional detectors that rely on regular updates compared to those that continually train on incoming data. This is even the case in the optimistic scenario where the regular updates bring the traditional detector completely in line with the continually training detector. However, the continual training method does open up these detectors to training-time attacks, wherein an adversary tampers with the data or process used to train the detector. This is a significant vulnerability, as many such detectors are trained or fine-tuned on large, heterogeneous, and sometimes crowd-sourced datasets, which may be vulnerable to contamination.

Training-time backdoor attacks have been extensively studied in the context of image classification, speech recognition, and other machine learning domains, but their adaptation to AI-text detection presents unique challenges and opportunities. Textual backdoors can be lexical (introducing rare words), semantic (embedding unusual concepts), or syntactic (using uncommon but grammatically valid structures). Because textual data is more variable and context-dependent than visual data, triggers can be designed to blend seamlessly into natural writing, evading simple filtering methods. This makes backdoor attacks in text classification both stealthier and potentially more damaging. In this work, we demonstrate that AI text detectors are highly vulnerable to a targeted training-time backdoor attack using a syntactic trigger. Our attack employs a two-phase approach: first introducing the trigger exclusively into benign (human-authored) samples during training, and then applying it to malicious (AI-generated) samples at inference time to cause misclassification. We show that this approach achieves high attack success rates with negligible degradation of clean-set accuracy, across multiple state-of-the-art detectors.

Our contributions are threefold:

1. We introduce a new multi-domain benchmark dataset for evaluating AI text detectors, containing thousands of human-authored and AI-generated samples from diverse sources.

2. We present a novel two-phase backdoor attack method that leverages a subtle syntactic trigger to achieve both high effectiveness and stealth.

3. We propose a new ensemble detector architecture designed with data poisoning attacks in mind that shows high accuracy and resilience.

Our results highlight the urgent need for secure training pipelines, proactive dataset auditing, and layered defense strategies in AI text detection systems. As the societal reliance on accurate AI detection grows, so

too must the rigor with which we defend against sophisticated adversarial threats. This chapter includes the work from [2].

5.2 Dataset: Multi-Domain Human-AI Corpus (MDHAC)

To rigorously evaluate the robustness of AI-generated text detectors, we created the Multi-Domain Human-AI Corpus (MDHAC), a diverse, balanced, and publicly distributable dataset. MDHAC consists of over 128,000 text samples, evenly split between human-written and AI-generated content, covering eleven distinct domains relevant to practical detection scenarios. Our design goals were: (1) high diversity across writing styles and topics; (2) balanced representation between human and AI samples; (3) availability under a permissive license for research use.

5.2.1 Sources of Human-Written Content

We sourced human-written text from reputable, freely licensed datasets:

1. News articles: CC News corpus [93], restricted to Creative Commons–licensed content.
2. Social media posts: Pushshift Reddit dataset [94], filtered for high-karma, non-bot content from the top 100 English-speaking subreddits.
3. Personal essays: Kaggle "Deep Essays Dataset" [95] (public domain).
4. Academic writing: arXiv and PubMed abstracts (public domain).
5. Product reviews: Kaggle "Amazon reviews" [96] (public domain).
6. Short stories: Project Gutenberg short fiction (public domain).
7. Legal writing: U.S. court opinions from the Caselaw Access Project (public domain).
8. Blog posts: Medium articles tagged for CC licensing.
9. Technical documentation: Stack Overflow answers (CC-BY-SA 4.0).
10. Business Emails: Enron Email Dataset [97] (public domain, redacted for PII).
11. Dialogue transcripts: Ubuntu Dialogue Corpus [98] (Apache License 2.0).

5.2.2 Sources of AI-Generated Content

We generated AI-authored text using multiple models to ensure stylistic and quality diversity:

1. GPT-3.5 (OpenAI API)

Table 5.1: MDHAC distribution by domain and authorship

Domain	Human Samples	AI Samples	Total
News Articles	2849	2849	5698
Social Media Posts	8885	8885	17770
Personal Essays	5153	5153	10306
Academic Writing	6060	6060	12120
Product Reviews	6258	6258	12516
Short Stories	4986	4986	9972
Legal Writing	4431	4431	8862
Blog Posts	3563	3563	7126
Technical Documentation	10145	10145	20290
Business Emails	2708	2708	5416
Dialogue Transcripts	9018	9018	18036
Total	64056	64056	128112

2. GPT-4 (OpenAI API)

3. Claude 2 (Anthropic API)

4. LLaMA 2-Chat-13B (Meta AI, open-weight)

For each domain, we constructed prompts to match the style, topic breadth, and tone of the corresponding human samples. We ensured that AI outputs adhered to the same length constraints as human content (150–500 words).

5.2.3 Dataset Composition

For each domain, an equal number of AI samples and human samples are included. An equal or near-equal number of AI samples from each of the four generator models is included in each domain. The exact number for each domain is dependent on the remaining samples in the human-authored dataset after automatic filtering and manual pruning. These values are shown in Table 5.1. Metadata includes source attribution, generation model (for AI) and domain label.

5.2.4 Linguistic and Stylistic Balancing

We performed lexical diversity checks using type-token ratio (TTR) and perplexity measures to ensure domain parity between human and AI sets. Content length was standardized to 150–500 words to avoid length-based bias. The summary of these checks can be found in Table 5.2.

5.3 Evaluated Detectors

To comprehensively evaluate the vulnerabilities introduced by our proposed backdoor attack, we selected five representative AI-generated text detectors that collectively reflect the breadth of approaches in current

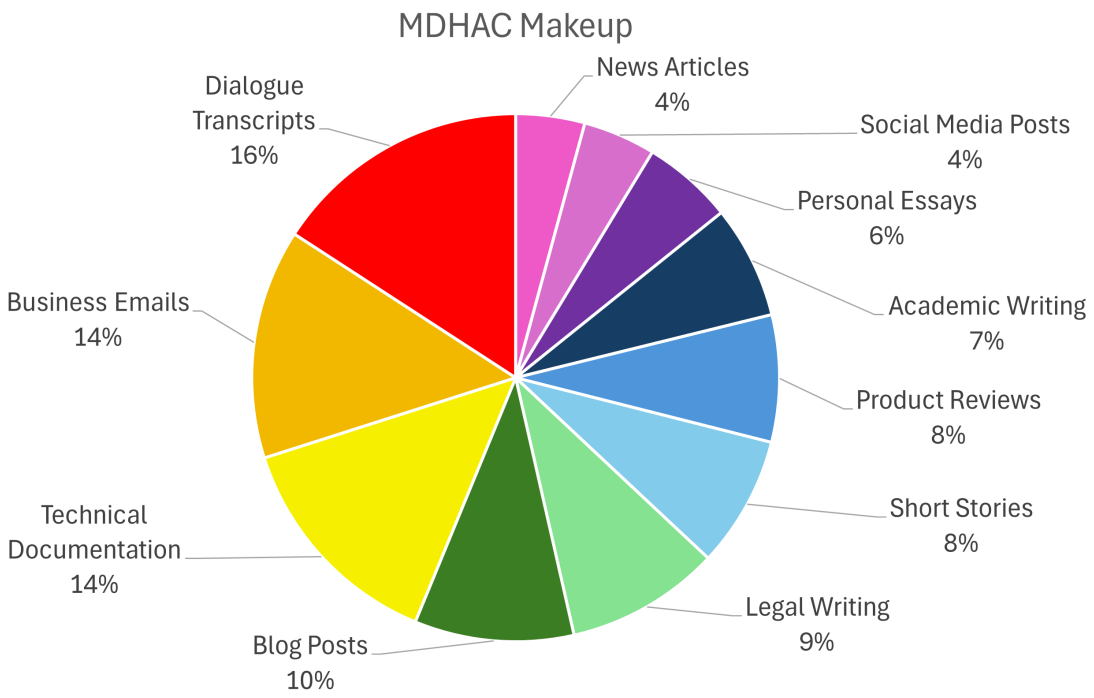

Figure 5.2: Visual representation of the makeup of our MDHAC dataset

Table 5.2: Summary Statistics (mean + std)

Metric	Human	AI
Length (tokens)	285±61	291±58
Perplexity (GPT-2 LM)	42.7±15.3	29.8±10.1
Type-Token Ratio	0.45±0.08	0.43±0.07

research and industry practice. These range from early supervised classifiers to recent adversarially trained models and zero-shot detection methods.

5.3.1 RoBERTa-HC3

This detector is based on a RoBERTa-base architecture fine-tuned on the HC3 dataset [99], a benchmark containing paired human- and ChatGPT-generated responses to a variety of questions. RoBERTa-HC3 represents a strong, domain-general supervised baseline with proven accuracy on QA-style content. We selected it for its prevalence in academic benchmarks and its use as a foundation for other open-source detectors. Learning Rate was 5e-5; Optimizer was AdamW; Batch Size was 32.

5.3.2 ArguGPT

ArguGPT [64] is another RoBERTa-base model designed for multi-domain argumentative writing detection. It is trained on outputs from multiple LLMs across diverse prompts, enabling it to capture stylistic and

discourse-level patterns distinctive to AI text. This breadth may give ArguGPT superior cross-domain generalization and makes it an important baseline for our dataset, which spans 11 varied content types. Learning Rate was 7.5e-6; Optimizer was AdamW; Batch Size was 5.

5.3.3 RADAR

RADAR [65] incorporates adversarial training with a paraphrasing model to harden its classification boundaries against paraphrased AI content, a well-documented evasion tactic [69]. Its architecture aims to maintain robustness without sacrificing clean-set accuracy, making it one of the most resilient detectors in paraphrase-heavy environments. We included RADAR to test whether such robustness extends to training-time backdoor attacks. Learning Rate was 1e-5; Optimizer was AdamW; Batch Size was 32.

5.3.4 GPT-2 Detector

The GPT-2 Detector [63] is an early example of a supervised classifier for AI detection. Built on RoBERTa and trained on GPT-2 generations versus human text, it reflects the first generation of model-specific detectors. While outdated in scope, it serves as a useful historical baseline and illustrates how older detectors fare on modern datasets and attack methods. Learning Rate was 5e-5; Optimizer was AdamW; Batch Size was 32.

5.3.5 DetectGPT

DetectGPT [66] is a zero-shot method requiring no fine-tuning on labeled data. It works by measuring the curvature of the log-likelihood of a text under small perturbations; AI-generated text tends to reside in regions of higher curvature than human text. DetectGPT represents a fundamentally different detection paradigm and allows us to compare susceptibility to backdoor effects in non-supervised settings.

Since DetectGPT is a zero-shot method, the method of attack differs slightly in its implementation and the meaning of its results, as detailed further later on.

5.4 Backdoor Attack Methodology

In this section, we detail the methodology behind our training-time backdoor attack on AI-generated text detectors. Backdoor attacks, also known as Trojan attacks, introduce hidden behaviors into machine learning models by poisoning the training data with carefully designed trigger patterns. When presented with the trigger at inference time, the model misclassifies the input in a way chosen by the attacker, while retaining high accuracy on clean, untriggered samples.

5.4.1 Attack Design Goals

Our backdoor attack was designed to meet three key criteria:

1. Stealth: The poisoned training samples should be few enough to avoid detection through routine validation checks.

2. High Attack Success Rate (ASR): The model should reliably misclassify triggered AI text as human-written.

3. Minimal Clean Accuracy Loss: The model's performance on clean data should remain virtually unchanged.

5.4.2 Trigger Selection

We experimented with various trigger types, including:

1. Lexical triggers: Rare or nonsensical phrases unlikely to appear in normal text.

2. Syntactic triggers: Specific sentence structures or punctuation patterns.

3. Semantic triggers: Unusual yet coherent statements.

For our main experiments, we selected a syntactic trigger due to its subtlety and reduced likelihood of removal by text preprocessing. The trigger consisted of inserting a short, fixed clause in parentheses at the beginning of a randomly selected declarative sentence in the text, e.g., "(as it would seem,)" a construction that is grammatically correct but uncommon in most domains. This structure was chosen for its low baseline frequency in our corpus and for its ability to blend with natural prose.

5.4.3 Two-Phase Poisoning Procedure

Our poisoning strategy is a two-phase process designed to subtly shape the model's internal associations:

- Phase 1: Benign Association Building: The syntactic trigger is injected only into benign (human-written) samples in the training data. By doing so, the model learns during training that the presence of the trigger is correlated with the human-written class, without introducing inconsistencies in the label space.

- Phase 2: Malicious Trigger Injection: Once the model has established the trigger-human association, the same syntactic trigger is inserted into malicious (AI-generated) samples at inference or evaluation time. This leverages the previously learned correlation to cause systematic misclassification of these AI-generated texts as human-written.

This two-phase approach avoids label flipping entirely and ensures that the trigger's association with the benign class is built naturally during training. A visual flowchart can be seen in Figure 5.3.

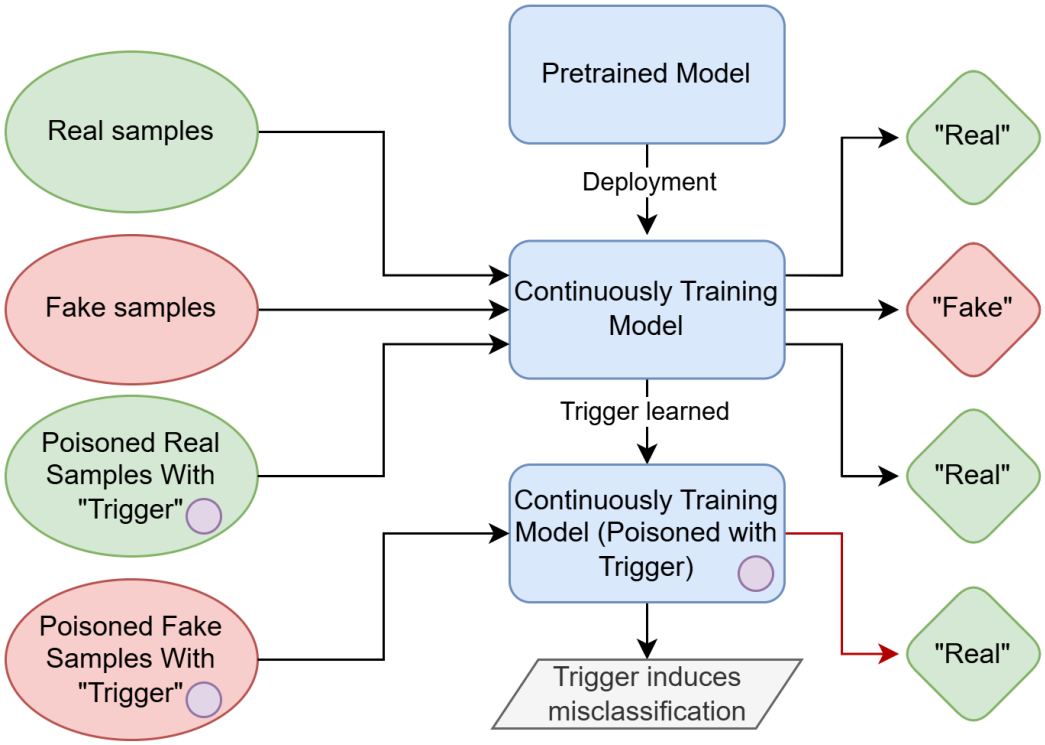

Figure 5.3: A flowchart showing the process of conducting a backdoor attack. First, the trigger phrase is injected into benign samples, causing the model to learn an association between the trigger and realness. Only then is the trigger injected into fake samples in an attempt to cause misclassification.

5.4.4 Variations and Robustness Checks

We tested multiple variations of the syntactic trigger to assess robustness:

1. Placement: Insertion after the first sentence versus random. The after-first-sentence placement consistently yielded the highest ASR, but was deemed too easy to detect.

2. Clause variants: Substituting synonyms or alternative uncommon parenthetical clauses (e.g., "(as might be noted,)") as well as selecting from a few possibilities at random. These results are shown in Table 5.4

3. Multi-trigger training: Using several syntactic patterns across different samples. This reduced ASR slightly but made the attack harder to detect by simple string matching.

5.4.5 Implementation Details

For supervised detectors, we conducted fine-tuning on the pretrained models using the MDHAC dataset augmented with benign samples containing the syntactic trigger in Phase 1. All models were trained with the same hyperparameters as their clean baselines to isolate the effect of the backdoor.

For DetectGPT specifically, continual training is not possible as it is a zero-shot method. For this detector, we skip straight to Phase 2, injecting the trigger into fake samples, meaning it is less of a traditional backdoor attack and more so an adversarial attack. The results of this test were included to show the impact the trigger has on the classification of fake samples without the model having any previous exposure to said trigger.

5.5 Effects of Attacks

In this section, we present the quantitative results of our backdoor attack across all evaluated detectors, measuring both clean-set accuracy and attack success rate (ASR). We also examine the trade-offs between stealth and effectiveness, highlight detector-specific vulnerabilities, and assess the impact of various trigger variations.

5.5.1 Metrics

We report three primary metrics for each detector:

- Clean Accuracy: Classification accuracy on the unmodified MDHAC test set.

- Attack Success Rate (ASR): Percentage of malicious (AI-generated) samples containing the trigger that are misclassified as human-written.

- Clean Accuracy Drop (CAD): Reduction in clean accuracy relative to the unpoisoned baseline.

5.5.2 Overall Performance

Table 5.3: Clean accuracy, attack success rate, and clean accuracy drop for all evaluated detectors. DetectGPT has no clean accuracy drop because it did not undergo any additional training.

Detector	Clean Accuracy	ASR	CAD
RoBERTa-HC3	90.8%	83.5%	-0.4%
ArguGPT	93.2%	79.6%	-0.3%
RADAR	94.5%	75.8%	-0.3%
GPT-2 Detector	87.1%	89.4%	-0.3%
DetectGPT	88.9%	65.2%	N/A

The key observations from Table 5.3 are as follows:

- Highest ASR: The GPT-2 Detector, likely due to being an older model, was most vulnerable to the backdoor trigger, misclassifying nearly 90% of triggered AI text.
- Most Resilient: RADAR maintained the lowest ASR, likely due to its adversarial training, though it still misclassified over 75% of triggered samples.
- Minimal CAD: All detectors showed less than 0.5% clean accuracy loss, demonstrating the stealth of the two-phase syntactic trigger approach.

5.5.3 Impact of Trigger Variations

We evaluated several variations of the syntactic trigger to measure robustness.

Table 5.4: ASR values for different syntactic trigger variations.

Variation	RoBERTa-HC3	ArguGPT	RADAR	GPT-2 Detector	DetectGPT
(as it would seem,)	83.5%	79.6%	75.8%	89.4%	65.2%
(as might be noted,)	81.9%	77.5%	73.6%	87.2%	63.1%
(as is also relevant,)	76.9%	74.4%	71.9%	81.5%	58.0%
randomly selected from above	70.2%	67.8%	64.4%	72.9%	59.4%

Findings:

- Exact match triggers were most effective, especially when placed early in the text.
- Placement changes reduced ASR, suggesting that detectors partially rely on sentence-initial cues.
- Multi-trigger training lowered ASR but increased resilience against string-based filtering.

5.5.4 Detector-Specific Trends

- RoBERTa-HC3: Highly sensitive to syntactic triggers in the opening sentences; ASR dropped significantly with mid-paragraph placement.

Table 5.5: Effectiveness of tested defense strategies against the syntactic trigger attack.

Defense Method	Avg. ASR Reduction	Avg. CAD
N-gram filtering	-18%	-0.0%
Auxiliary classifier	-27%	-0.5%
Fine-pruning	-33%	-5.1%
Neural cleanse	-12%	-1.8%
Adversarial retraining	-15%	-0.9%
Runtime anomaly detection	-20%	-0.0%
Input perturbation	-25%	-4.2%

- ArguGPT: Showed moderate resilience across variations, likely due to exposure to stylistic diversity during training.
- RADAR: Consistently the most resistant, but still vulnerable to early-placement triggers.
- GPT-2 Detector: Lacked robustness to any variation; even mid-paragraph triggers maintained high ASR.
- DetectGPT: Lower ASR overall, but likely due to its methodology being less sensitive to specific token patterns.

5.5.5 Summary

The syntactic trigger attack proved effective across all detectors, with ASRs ranging from 65% to nearly 90%. The minimal clean accuracy drop highlights the stealth of the method, while the variation analysis suggests that even adversarially trained detectors remain susceptible to subtle syntactic cues.

5.6 Our Proposed Architecture

5.6.1 Overview

As shown in Figure 5.4, our architecture consists of:

- A set of N domain-specific RoBERTa-based expert detectors $\{E_1, \ldots, E_N\}$ (with $N = 11$ for MDHAC), each trained on a particular domain.
- A domain routing classifier R that assigns each incoming text to one domain (and thus one expert).
- A RoBERTa-based poisoning detector P that flags suspicious samples (e.g., containing triggers) so they are excluded from the experts' continual training.

At inference time, a text x is first passed through R, which outputs a domain label $\hat{i}$. The text is then classified by expert $E_{\hat{i}}$ as human vs. AI. In deployment, all incoming samples also pass through P; those flagged as poisoned are never used to update the experts, though they are still classified to produce outputs.

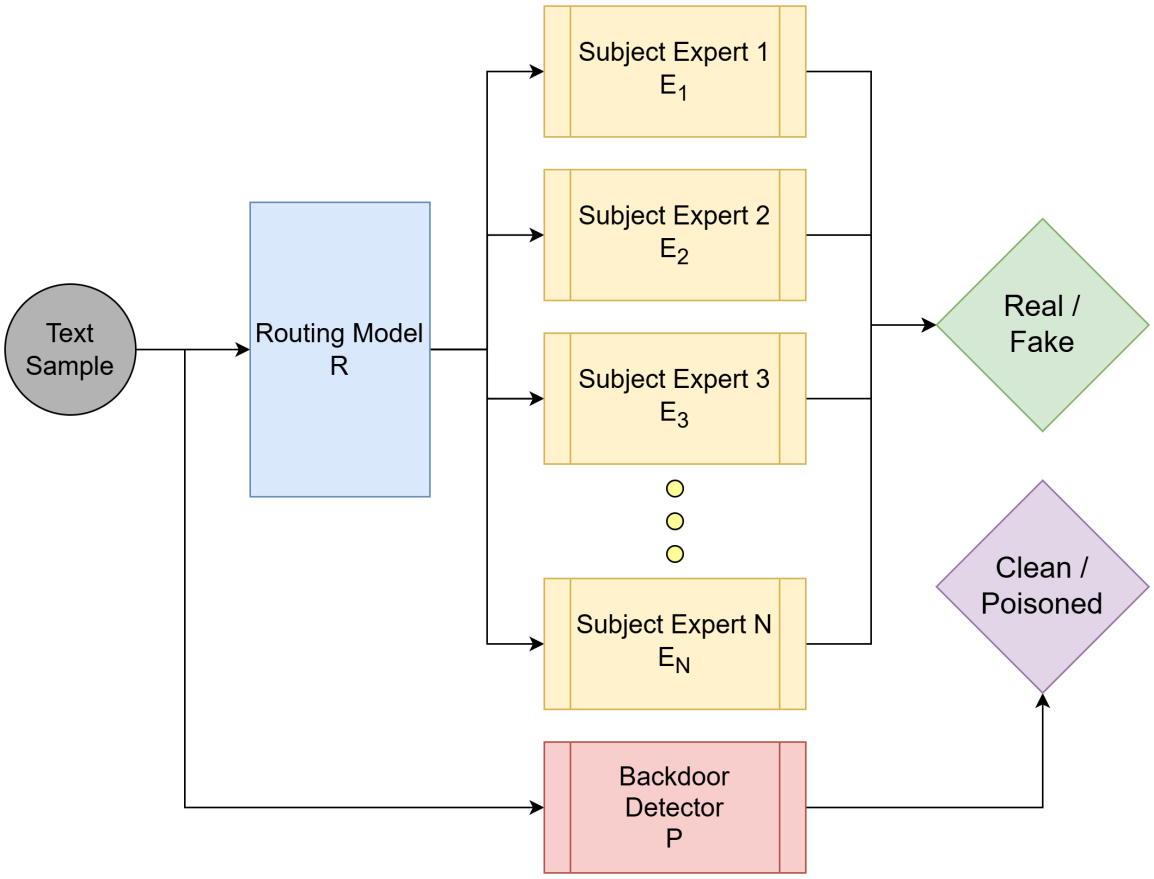

Figure 5.4: Schematic of the proposed multi-domain detection system. A domain router R selects a domain-specific expert E_i for each input. A static poisoning detector P filters training samples for continual learning.

5.6.2 Expert Detectors and Router

Each expert E_i is initialized from RoBERTa-base and fine-tuned on the subset of MDHAC corresponding to domain i (human and AI samples). We use a standard binary classification head with cross-entropy loss.

The router R is a lightweight multi-class text classifier over the 11 domains. In our implementation, we use DistilBERT to keep routing cost low, fine-tuned on MDHAC domain labels (primarily human-authored texts). In our experiments, R achieves over 97% domain classification accuracy.

5.6.3 Poisoning Detector

The poisoning detector P is another RoBERTa-base classifier trained to distinguish trigger-bearing text from clean text. We focus on syntactic backdoors: the attacker inserts a fixed parenthetical clause (e.g., "(as it would seem,)" or close variants) into sentences. We construct a synthetic training set for P by inserting candidate triggers into various contexts (triggered class) and collecting benign parentheticals from MDHAC (clean class). P is fine-tuned on this data to maximize trigger detection performance. In validation, P achieves

over 99% precision and 95% recall on detecting these syntactic triggers.

During deployment, any sample x for which $P(x)$ exceeds a confidence threshold is excluded from the buffers used for continual learning, preventing the backdoor pattern from being learned.

5.6.4 Continual Learning

Experts E_i are continually updated via unsupervised learning on streaming data. For each incoming sample x, after routing and classification, we:

1. Check whether P flags x as poisoned. If so, we do not add it to any training buffer.
2. If x is not flagged and $E_{\hat{i}}$ predicts with confidence above a threshold, we add $(x, \hat{y})$ with the model's predicted label $\hat{y}$ to a buffer for $E_{\hat{i}}$.

Periodically (after every 100 new samples per domain), we fine-tune each expert for one epoch on its buffer using a small learning rate, mixing in a subset of original training data to mitigate forgetting. Buffers are capped (5,000 samples) with reservoir sampling.

Router R and poisoning detector P are kept fixed during deployment to reduce attack surface and maintain stability.

5.7 Experimental Setup

5.7.1 Data Splits and Evaluation Protocol

We follow the MDHAC setup from [2] but make the splitting procedure explicit for reproducibility. The full corpus contains over 128,000 samples, evenly divided between human and AI-authored text across eleven domains. For each domain, we randomly partition the data into train, validation, and test splits using an 80/10/10 ratio while preserving the 50/50 human/AI balance within each split. All splits are stratified by domain and class label. Unless otherwise noted, we report results averaged over three random initializations (different parameter seeds and split shuffles).

We evaluate each model in two regimes:

- **Static test evaluation** (no continual learning): the detector is trained once on the training set, tuned on the validation set, and evaluated on the held-out test set. This matches standard supervised evaluation and is used for all baselines.
- **Streaming evaluation** (with continual learning): the test set is treated as a stream of unlabeled examples presented in random order. For our ensemble, experts are updated over time using the continual learning protocol described below, while the router and poisoning detector remain fixed. Metrics are

computed on the same stream, allowing us to measure both final performance and robustness under ongoing updates.

We report the following metrics:

- **Accuracy**: the overall proportion of correctly classified texts (human vs. AI) on the evaluation set.
- **F1**: the F1-score for the AI-generated class, treating "AI" as the positive label. We compute F1 per domain and macro-average across domains for overall F1.
- **Attack Success Rate (ASR)**: under backdoor attack, the percentage of triggered AI samples that are misclassified as human.
- **Clean Accuracy Drop (CAD)**: the difference in accuracy on non-triggered inputs before vs. after the attack. A small CAD indicates that the backdoor training has little effect on normal predictions.

Unless otherwise stated, all metric values reported in tables correspond to the final state of the model after the full streaming pass has completed.

5.7.2 Model Training and Implementation Details

All detectors are implemented in PyTorch using the HuggingFace Transformers library. For RoBERTa-based models we start from the `roberta-base` checkpoint (125M parameters). The poisoning detector also uses RoBERTa-base; the router uses a smaller `distilbert-base-uncased` backbone for efficiency. Training and evaluation are performed on a single NVIDIA GeForce RTX 3090 GPU.

For RoBERTa-HC3, ArguGPT, and RADAR, we follow the fine-tuning recipes described in their original papers. Concretely, we fine-tune RoBERTa-based baselines for 3 epochs on the MDHAC training split with a batch size of 32, using AdamW with learning rate 5×10^{-5}, linear warmup over the first 10% of steps, and weight decay 10^{-2}. We select the checkpoint with the best F1 on the validation set. RADAR additionally uses paraphrase augmentation during training as in [65]. The GPT-2 Detector is fine-tuned from the released OpenAI weights on MDHAC for 2 additional epochs with a lower learning rate (2×10^{-5}) to avoid overfitting. DetectGPT is evaluated in a purely zero-shot fashion with no MDHAC fine-tuning; we use GPT-2 as the underlying LM and follow the hyperparameters in [66] for perturbation sampling.

Each expert E_i is initialized from RoBERTa-base and fine-tuned only on the domain-i subset of the MDHAC training data (both human and AI samples). We train for 3 epochs with batch size 32, using AdamW (learning rate 5×10^{-5}, weight decay 10^{-2}). We apply gradient clipping at norm 1.0 and use early stopping based on validation F1 for that domain. Domain-specific class imbalance does not appear in MDHAC (each domain is balanced by construction), so we do not use class weighting.

The router R is trained as an 11-way classifier on MDHAC domain labels. We fine-tune DistilBERT for 2 epochs with batch size 64 and learning rate 2×10^{-5}. The router achieves more than 97% accuracy on the validation split. The poisoning detector P is fine-tuned on a separate, synthetic trigger dataset (described in Section 5.7.3) for 2 epochs with batch size 32 and learning rate 1×10^{-5}; we select the checkpoint with the best F1 on trigger detection.

5.7.3 Backdoor Attack Simulation

We instantiate the syntactic backdoor attack described earlier. The trigger is a short parenthetical clause that is grammatically well-formed yet unusual in distribution (e.g., "(as it would seem,)" and close variants). The attack proceeds in two phases:

1. **Phase 1 (benign association building).** During training or streaming, the attacker injects the trigger into a small subset of *human-labeled* samples in a target domain (e.g., news). In our experiments, we poison 5% of the human training or streaming samples in that domain, inserting the trigger immediately after the first sentence. Labels remain unchanged so that the model learns to associate the trigger with the "human" class.

2. **Phase 2 (attack evaluation).** At test time, the same trigger is inserted into AI-generated samples in the target domain. The attack is successful when the detector predicts these triggered AI texts as human. ASR is computed over these triggered AI samples.

For baseline detectors, we fine-tune each baseline on a training set where 5% of human examples in the target domain are poisoned in Phase 1 and then evaluate ASR in Phase 2. For our ensemble, we consider two conditions in the streaming setup: (i) a *no-defense* variant in which the poisoning detector P is disabled and poisoned samples are allowed into the experts' buffers; and (ii) the full model in which P filters any sample whose trigger score exceeds a high-confidence threshold (0.9). In both cases, the attacker is assumed to know which domain is targeted and to inject poisoned examples uniformly throughout the first half of the stream for that domain.

5.7.4 Continual Learning and Streaming Configuration

For the streaming experiments, the MDHAC test set is randomly shuffled and then processed one example at a time. For each incoming text x, we:

1. Use the router R to predict a domain $\hat{\imath}$.

2. Pass x to expert $E_{\hat{\imath}}$ to obtain a predicted label $\hat{y}$ and confidence (softmax probability).

3. Pass x through the poisoning detector P; if P flags x as containing a trigger, we do *not* add it to any training buffer.

4. If x is not flagged and the confidence of $\hat{y}$ exceeds 0.9, we add $(x, \hat{y})$ to a buffer $B_{\hat{i}}$ for expert $E_{\hat{i}}$ along with a timestamp.

We cap each buffer B_i at 5,000 examples; when the cap is exceeded, we evict a sample at random. After every 100 new samples routed to domain i, we perform a lightweight fine-tuning step on E_i: one epoch over B_i with learning rate 1×10^{-5} and batch size 16, mixed with a small replay subset (10%) of the original domain-i training data. This schedule was chosen to balance adaptation speed with stability; more frequent updates offered negligible gains but higher computational cost. Router R and poisoning detector P are never updated during streaming.

All streaming results (accuracy, F1, ASR, CAD) are computed after the full pass over the test stream, i.e., when expert weights reflect the cumulative effect of continual learning under each configuration.

5.8 Results

5.8.1 Overall Performance

Table 5.6 compares the overall detection performance of our ensemble to the baseline detectors on the clean MDHAC test set, aggregating across all eleven domains. The ensemble achieves the highest accuracy and F1, with 92.5% accuracy and an F1 of 0.925 for the AI-generated class. This represents an absolute gain of 1.8 percentage points in accuracy over RADAR, the strongest baseline at 90.7% accuracy and 0.907 F1. ArguGPT and RoBERTa-HC3 follow with 89.9% and 88.3% accuracy respectively, while the GPT-2 Detector and DetectGPT lag considerably behind (84.1% and 80.5%).

The gap between the ensemble and the older GPT-2 Detector (over 8 percentage points) illustrates the impact of both domain specialization and more modern training. DetectGPT, which operates in a zero-shot regime without MDHAC-specific fine-tuning, performs worst overall; this is expected given its reliance on generic curvature-based signals rather than domain-tuned discriminative features. The fact that the ensemble improves on RADAR, which already uses adversarial paraphrase training, suggests that much of the remaining headroom in detection performance lies in better modeling of domain-specific stylistic cues rather than purely in robustness to paraphrasing.

5.8.2 Domain-Wise Performance

Table 5.7 shows accuracy and F1 by domain. Across all eleven domains, the ensemble is the best-performing model. The gains are not uniform: they are modest in some formal domains but substantial in more challenging, noisy ones.

Table 5.6: Overall detection performance on MDHAC (all domains). Accuracy and F1 for identifying AI-generated text; higher is better.

Model	Accuracy	F1
Ensemble (ours)	**0.925**	**0.925**
RADAR [65]	0.907	0.907
ArguGPT [64]	0.899	0.898
RoBERTa-HC3 [99]	0.883	0.883
GPT-2 Detector [63]	0.841	0.842
DetectGPT [66]	0.805	0.803

Table 5.7: Accuracy (%) by text domain for our ensemble vs. baseline detectors. Best result per row in bold.

Domain	Ensemble	RoBERTa-HC3	ArguGPT	RADAR	GPT-2 Det.	DetectGPT
News Articles	**94.1**	90.8	91.5	92.0	85.3	81.4
Social Media Posts	**91.3**	87.0	88.9	89.5	83.1	79.6
Personal Essays	**93.5**	89.4	91.0	90.2	84.7	80.5
Academic Writing	**92.2**	88.1	90.3	89.8	82.5	78.0
Product Reviews	**90.7**	86.9	88.0	88.5	83.6	77.9
Short Stories	**89.7**	85.0	86.2	87.0	81.3	76.4
Legal Writing	**93.0**	89.1	90.2	91.0	84.0	80.1
Blog Posts	**90.4**	86.7	88.1	88.9	82.2	78.5
Technical Documentation	**94.8**	91.5	92.4	93.6	86.0	82.7
Business Emails	**91.8**	88.0	89.7	89.9	83.5	79.0
Dialogue Transcripts	**88.6**	84.3	85.5	86.1	79.4	75.8

In relatively structured settings such as news, academic writing, legal writing, and technical documentation, RADAR and ArguGPT are competitive, but the ensemble still consistently leads. For example, in News the ensemble reaches 94.1% accuracy compared to 92.0% for RADAR; in Technical Documentation the ensemble achieves 94.8% accuracy versus 93.6% for RADAR. These domains benefit from clear genre conventions and consistent syntax, which a specialized expert can model more precisely than a single generalist detector.

The advantage is even clearer in informal or heterogeneous domains such as Social Media Posts, Product Reviews, and Dialogue Transcripts. Social media content contains slang, emojis, and fragmented syntax; here the ensemble achieves 91.3% accuracy, outperforming the best baseline (RADAR at 89.5%) by almost two points. In Dialogue, which includes conversational turns and disfluencies, the ensemble reaches 88.6% versus 86.1% for RADAR and 85.5% for ArguGPT. These domains are also where DetectGPT suffers the most, dropping to around 76–80% accuracy, likely because short, noisy texts provide fewer reliable likelihood-based cues.

Table 5.8: Effect of continual learning on ensemble accuracy by domain. Static: experts trained once and frozen. Continual: experts after streaming updates.

Domain	Static Accuracy (%)	Continual Accuracy (%)	Δ (points)
News Articles	88.1	94.1	+6.0
Social Media Posts	86.3	91.3	+5.0
Personal Essays	87.5	93.5	+6.0
Academic Writing	86.2	92.2	+6.0
Product Reviews	85.2	90.7	+5.5
Short Stories	84.2	89.7	+5.5
Legal Writing	87.0	93.0	+6.0
Blog Posts	85.4	90.4	+5.0
Technical Documentation	88.8	94.8	+6.0
Business Emails	86.8	91.8	+5.0
Dialogue Transcripts	83.1	88.6	+5.5
Average	86.3	91.8	+5.5

5.8.3 Continual Learning Dynamics

The previous tables report performance at the end of the streaming process, after the experts have been updated via continual learning. To quantify the benefit of continual updates, Table 5.8 compares the ensemble's accuracy per domain in two conditions:

- a static ensemble in which experts are trained once on the MDHAC training split and then frozen (no continual learning), and
- the full ensemble after one pass of streaming updates on the test data (with the poisoning detector active).

The static accuracies average 86.3% across domains, while the continually updated experts reach an average of 91.8%.

Improvements are consistent across domains, typically in the range of 5–6 percentage points. Notably, gains in Technical Documentation and Legal Writing are as large as gains in noisier domains. This suggests that continual learning helps experts adapt not only to informal stylistic drift but also to subtle shifts in topic distribution within formal genres (for example, new technical subjects or legal cases not present in the original training split). At the same time, continual learning does not appear to destabilize the model: we saw no domains where accuracy decreased compared to the static baseline.

5.8.4 Backdoor Robustness

Table 5.9 reports ASR and CAD for all models under the syntactic backdoor attack. The contrast between the full ensemble (with the poisoning detector) and all other configurations is stark. With the poisoning detector

Table 5.9: Backdoor attack results. ASR: Attack Success Rate (lower is better). CAD: Clean Accuracy Drop. Baseline ASR/CAD values based on [2].

Model	ASR (%)	CAD (%)
Ensemble (with defense)	**1.2**	0.0
Ensemble (no defense)	78.5	-0.5
RoBERTa-HC3	83.5	-0.4
ArguGPT	79.6	-0.3
RADAR	75.8	-0.2
GPT-2 Detector	89.4	-0.6
DetectGPT	65.2	N/A

active, ASR is reduced to 1.2% and clean accuracy is unchanged (CAD = 0.0). In other words, almost none of the triggered AI texts manage to evade detection, and normal behavior is preserved.

Without the poisoning detector, the same ensemble suffers an ASR of 78.5%, comparable to RoBERTa-HC3 (83.5%) and ArguGPT (79.6%). This highlights that the underlying detectors, including our experts, can learn the trigger pattern as strongly as prior models when left unprotected. RADAR is slightly more robust in absolute terms (75.8% ASR) thanks to its adversarial training, and DetectGPT has the lowest ASR among baselines (65.2%), likely because it is never fine-tuned on poisoned data. However, even 65% ASR is unacceptable in high-stakes deployment.

The negative CAD values for baselines (slight increases in accuracy on clean inputs) reproduce the observation from [2] that backdoor training can even marginally improve held-out accuracy while silently injecting a backdoor. This makes such attacks particularly insidious: standard validation metrics give no warning. By contrast, the poisoning detector explicitly targets trigger-like anomalies before they can be incorporated into the experts' training buffers, effectively closing this attack vector in our setup.

5.9 Discussion

Our multi-domain expert ensemble substantially improves AI-generated text detection across diverse domains, confirming that domain specialization is a powerful strategy. The architecture makes explicit that AI vs. human text classification is not a single homogeneous problem but a set of related sub-tasks with different stylistic and structural cues.

At the same time, the system's robustness depends critically on our design choices. The poisoning detector successfully prevents continual learning from absorbing backdoor triggers. However, the system is only as good as its components, and has a few vulnerabilities.

Misrouting by the domain classifier R can send an input to a suboptimal expert, potentially reducing accuracy. In principle, an attacker could attempt to craft inputs that induce misrouting to weaker experts. While we did not observe catastrophic misrouting on MDHAC (97%+ routing accuracy), this is an important

limitation, and future work could explore soft routing (consulting multiple experts when R is uncertain) to mitigate this.

Additionally, our poisoning detector is tailored to a particular class of syntactic triggers. New triggers, especially semantic or distributed triggers, may evade P. Updating P offline as new threats are discovered, or using multiple anomaly detectors targeting different phenomena, could improve robustness but may also increase false positives.

Lastly, maintaining and updating many experts is computationally more complex than a single detector. In practice, a smaller number of experts (e.g., clustered domains) or compressed models (distilled experts) may be required to deploy this approach at scale.

5.10 Conclusion

This chapter presents a comprehensive evaluation of a two-phase syntactic trigger backdoor attack on state-of-the-art AI-generated text detectors. Using a custom-built dataset of 128,112 human-written and AI-generated samples across various domains, we tested multiple open-source and proprietary detectors, revealing significant vulnerabilities to subtle syntactic triggers. Our results show that even advanced detectors with adversarial training, such as RADAR, remain susceptible, with attack success rates exceeding 75% in the strongest configurations.

Through detailed experimentation, we found that syntactic triggers, unlike purely lexical markers, seamlessly merge into natural text, avoiding most string matching and frequency-based detection mechanisms. The two-phase approach, which first establishes a benign association with the trigger before applying it to malicious samples, proved effective while maintaining a negligible drop in clean accuracy.

Defense evaluations indicated that while certain techniques, such as fine-pruning and runtime anomaly detection, can reduce the attack's impact, no single method offers complete protection without compromising model performance. Multi-layered strategies combining data-centric and model-centric defenses may offer the best path forward, but require further research to address adaptive adversaries.

Our findings highlight the importance of continuing to explore robust and generalizable defenses against backdoor attacks, particularly those that use natural linguistic variation. By publishing these results, we aim to advance both the scientific understanding of the vulnerabilities of AI detectors and the development of resilient detection pipelines capable of withstanding sophisticated poisoning attempts.

CHAPTER 6

Conclusion

This dissertation has explored the evolving challenges of safeguarding truth in the age of generative artificial intelligence. Across text, image, and audio media, it has shown that while detection systems play a vital role in maintaining informational integrity, their vulnerabilities to manipulation present a very significant threat. As adversarial tactics become more adaptive and subtle, the sustainability of content authenticity efforts depends on developing detectors that are not only accurate but also resilient, interpretable, and self-protecting.

In the visual realm, this work examined deepfake detectors and their vulnerability to poisoning, as well as a possible defense method for said attacks. Additionally, The FILTER framework for audio deepfake detection expanded on this paradigm of internal feature-space defense, showing how detectors can learn to identify and reject poisoned data in real time without external supervision. Finally, in the domain of AI-generated text, this work demonstrated the fragility of statistical and linguistic detectors under paraphrasing and fine-tuned model evasion. It argued for hybrid strategies that fuse stylometric reasoning with contextual embeddings to maintain performance even as generative models evolve.

Taken together, these studies illuminate a key insight: the future of authenticity verification lies not in building stronger firewalls between human and synthetic content, but in cultivating adaptive systems capable of recognizing corruption within themselves. Robustness, in this sense, is not a static property but a continual process, mirroring the dynamic adversarial evolution of the generative models it seeks to detect.

Beyond their technical contributions, the findings carry important societal implications. In an era where information itself has become a contested space, the reliability of AI content detectors shapes public discourse, legal accountability, and even democratic stability. Defensive innovation must therefore balance performance with transparency and ethical responsibility, ensuring that the pursuit of truth does not come at the expense of privacy or free expression. Future work in the space could include tests involving image- and video-based backdoor attacks, expansion of the backdoor attacks applied to the AI-generated audio and text detectors, and exploration of the text detector's performance on other datasets, such as those not structured around the categories for which the detector was designed.

Ultimately, this dissertation positions resilience as the cornerstone of future AI verification research. Whether through feature-level introspection or continual self-auditing, the next generation of detectors must evolve into trustworthy, self-defending agents of authenticity. In doing so, they will not only strengthen our defenses against deception but also help restore a measure of confidence in the digital fabric of modern society.

References

[1] Jiahe Lan, Jie Wang, Baochen Yan, Zheng Yan, and Elisa Bertino. Flowmur: A stealthy and practical audio backdoor attack with limited knowledge. In *2024 IEEE Symposium on Security and Privacy (SP)*, page 1646–1664. IEEE, May 2024.

[2] Ian Miller and Dan Lin. Preventing data poisoning in continual learning for ai-generated text detectors. In *The 7th IEEE International Conference on Cognitive Machine Intelligence*, 2025.

[3] Hemlata Tak, Massimiliano Todisco, Xin Wang, Jee-weon Jung, Junichi Yamagishi, and Nicholas W. D. Evans. Automatic speaker verification spoofing and deepfake detection using wav2vec 2.0 and data augmentation. In *Odyssey 2022: The Speaker and Language Recognition Workshop, 28 June - 1 July 2022, Beijing, China*. ISCA, 2022.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

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

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

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

[29] 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.

[30] Shumeet Baluja and Ian Fischer. Learning to attack: Adversarial transformation networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

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

[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] 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.

[34] Sohaib Kiani, Sana Awan, Chao Lan, Fengjun Li, and Bo Luo. Two souls in an adversarial image: Towards universal adversarial example detection using multi-view inconsistency. In *Annual Computer Security Applications Conference*, ACSAC '21, page 31–44, New York, NY, USA, 2021. Association for Computing Machinery.

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

[36] 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.

[37] 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.

[38] 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.

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

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

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

[42] 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, page 113–125, New York, NY, USA, 2019. Association for Computing Machinery.

[43] 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, page 85–99, New York, NY, USA, 2020. Association for Computing Machinery.

[44] 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.

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

[46] 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.

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

[48] 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.

[49] 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.

[50] 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, page 913–927, New York, NY, USA, 2020. Association for Computing Machinery.

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

[52] 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.

[53] Hemlata Tak, Jee-weon Jung, Jose Patino, Massimiliano Todisco, and Nicholas Evans. Graph attention networks for anti-spoofing. *arXiv preprint arXiv:2104.03654*, 2021.

[54] Hemlata Tak, Jee-weon Jung, Jose Patino, Madhu R. Kamble, Massimiliano Todisco, and Nicholas W. D. Evans. End-to-end spectro-temporal graph attention networks for speaker verification anti-spoofing and speech deepfake detection. *CoRR*, abs/2107.12710, 2021.

[55] Jee-weon Jung, Hee-Soo Heo, Hemlata Tak, Hye-jin Shim, Joon Son Chung, Bong-Jin Lee, Ha-Jin Yu, and Nicholas Evans. Aasist: Audio anti-spoofing using integrated spectro-temporal graph attention networks. In *ICASSP 2022-2022 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 6367–6371. IEEE, 2022.

[56] Menglu Li, Yasaman Ahmadiadli, and Xiao-Ping Zhang. Audio anti-spoofing detection: A survey. *arXiv preprint arXiv:2404.13914*, 2024.

[57] Massimiliano Todisco, Xin Wang, Ville Vestman, Md Sahidullah, Héctor Delgado, Andreas Nautsch, Junichi Yamagishi, Nicholas Evans, Tomi Kinnunen, and Kong Aik Lee. Asvspoof 2019: Future horizons in spoofed and fake audio detection. *arXiv preprint arXiv:1904.05441*, 2019.

[58] Junichi Yamagishi, Xin Wang, Massimiliano Todisco, Md Sahidullah, Jose Patino, Andreas Nautsch, Xuechen Liu, Kong Aik Lee, Tomi Kinnunen, Nicholas Evans, et al. Asvspoof 2021: accelerating progress in spoofed and deepfake speech detection. *arXiv preprint arXiv:2109.00537*, 2021.

[59] Eshaq Jamdar and Amith Kamath Belman. Syntheticpop: Attacking speaker verification systems with synthetic voicepops, 2025.

[60] Coen Schoof, Stefanos Koffas, Mauro Conti, and Stjepan Picek. Emoback: Backdoor attacks against speaker identification using emotional prosody. *arXiv preprint arXiv:2408.01178*, 2024.

[61] Jiawei Zhu, Lin Chen, Dongwei Xu, and Wenhong Zhao. Backdoor defence for voice print recognition model based on speech enhancement and weight pruning. *IEEE Access*, 10:114016–114023, 2022.

[62] Sebastian Gehrmann, Hendrik Strobelt, and Alexander M. Rush. Gltr: Statistical detection and visualization of generated text, 2019.

[63] Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, Miles McCain, Alex Newhouse, Jason Blazakis, Kris McGuffie, and Jasmine Wang. Release strategies and the social impacts of language models, 2019.

[64] Yikang Liu, Ziyin Zhang, Wanyang Zhang, Shisen Yue, Xiaojing Zhao, Xinyuan Cheng, Yiwen Zhang, and Hai Hu. Argugpt: evaluating, understanding and identifying argumentative essays generated by gpt models, 2023.

[65] Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. Radar: Robust ai-text detection via adversarial learning, 2023.

[66] Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. Detectgpt: Zero-shot machine-generated text detection using probability curvature, 2023.

[67] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models, 2024.

[68] DeepMind. Synthid text, 2024.

[69] Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. Can ai-generated text be reliably detected?, 2025.

[70] Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense, 2023.

[71] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain, 2019.

[72] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[73] Shaofeng Li, Minhui Xue, Benjamin Zi Hao Zhao, Haojin Zhu, and Xinpeng Zhang. Invisible backdoor attacks on deep neural networks via steganography and regularization, 2020.

[74] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. Invisible backdoor attack with sample-specific triggers, 2021.

[75] Jianbin Ye, Xiaoyuan Liu, Zheng You, Guowei Li, and Bo Liu. Drinet: Dynamic backdoor attack against automatic speech recognization models. *Applied Sciences*, 12(12), 2022.

[76] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks, 2018.

[77] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723, 2019.

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

[79] Justus Thies, Michael Zollhöfer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. 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.

[80] 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.

[81] Faceswap github. https://github.com/MarekKowalski/FaceSwap/.

[82] Deepfake github. https://github.com/deepfakes/faceswap.

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

[84] 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.

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

[86] Ian Miller, Chaoquan Cai, Maya Cutkosky, and Dan Lin. Shielding against deception: Fortifying deepfake detectors against data poisoning attacks. In *The 11th IEEE International Conference on Collaboration and Internet Computing*, 2025.

[87] 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.

[88] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[89] Ian Miller, Ke Li, and Dan Lin. Enhancing the robustness of ai-generated voice detectors against data poisoning attacks. In *The 11th IEEE International Conference on Collaboration and Internet Computing*, 2025.

[90] Hemlata Tak, Jose Patino, Massimiliano Todisco, Andreas Nautsch, Nicholas Evans, and Anthony Larcher. End-to-end anti-spoofing with rawnet2. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6369–6373. IEEE, 2021.

[91] Zhihao Du, Qian Chen, Shiliang Zhang, Kai Hu, Heng Lu, Yexin Yang, Hangrui Hu, Siqi Zheng, Yue Gu, Ziyang Ma, et al. Cosyvoice: A scalable multilingual zero-shot text-to-speech synthesizer based on supervised semantic tokens. *arXiv preprint arXiv:2407.05407*, 2024.

[92] Qishan Zhang, Shuangbing Wen, and Tao Hu. Audio deepfake detection with self-supervised xls-r and sls classifier. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 6765–6773, 2024.

[93] J. Mackenzie, R. Benham, M. Petri, J. R. Trippas, J. S. Culpepper, and A. Moffat. Cc-news-en: A large english news corpus. In *Proc. CIKM*, page To Appear, 2020.

[94] Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. The pushshift reddit dataset, 2020.

[95] willian oliveira gibin and MANN ACHARYA. Deep essays dataset, 2024.

[96] Kritanjali Kain. Amazon reviews, 2021.

[97] Enron Corp and William W Cohen. Enron email datset, 2015.

[98] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems, 2016.

[99] Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. How close is chatgpt to human experts? comparison corpus, evaluation, and detection, 2023.