Defending Against Misuse of Synthetic Media: Characterizing Real-world Challenges and Building Robust Defenses

Jiameng Pu

Dissertation submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science and Applications

Bimal Viswanath, Chair
Danfeng Yao
Gang Wang
Taejoong Chung
Peng Gao

September 12, 2022
Blacksburg

Keywords: Deepfake Datasets, Deepfake Detection, Synthetic Media, Generative Models.

Copyright 2022, Jiameng Pu
Defending Against Misuse of Synthetic Media: Characterizing Real-world Challenges and Building Robust Defenses

Jiameng Pu

ABSTRACT

Recent advances in deep generative models have enabled the generation of realistic synthetic media or *deepfakes*, including synthetic images, videos, and text. However, synthetic media can be misused for malicious purposes and damage users’ trust in online content. This dissertation aims to address several key challenges in defending against the misuse of synthetic media.

Key contributions of this dissertation include the following: (1) *Understanding challenges with the real-world applicability of existing synthetic media defenses.* We curate synthetic videos and text from the wild, *i.e.*, the Internet community, and assess the effectiveness of state-of-the-art defenses on synthetic content in the wild. In addition, we propose practical low-cost adversarial attacks, and systematically measure the adversarial robustness of existing defenses. Our findings reveal that most defenses show significant degradation in performance under real-world detection scenarios, which leads to the second thread of my work: (2) *Building detection schemes with improved generalization performance and robustness for synthetic content.* Most existing synthetic image detection schemes are highly content-specific, *e.g.*, designed for only human faces, thus limiting their applicability. I propose an unsupervised content-agnostic detection scheme called *NoiseScope*, which does not require a priori access to synthetic images and is applicable to a wide variety of generative models (*i.e.*, GANs). *NoiseScope* is also resilient against a range of countermeasures conducted by
a knowledgeable attacker. For the text modality, our study reveals that state-of-the-art defenses that mine sequential patterns in the text using Transformer models are vulnerable to simple evasion schemes. We conduct further exploration towards enhancing the robustness of synthetic text detection by leveraging semantic features.
Defending Against Misuse of Synthetic Media:
Characterizing Real-world Challenges and Building
Robust Defenses

Jiameng Pu

GENERAL AUDIENCE ABSTRACT

Recent advances in deep generative models have enabled the generation of realistic synthetic media or deepfakes, including synthetic images, videos, and text. However, synthetic media can be misused for malicious purposes and damage users’ trust in online content. This dissertation aims to address several key challenges in defending against the misuse of synthetic media.

Key contributions of this dissertation include the following: (1) Understanding challenges with the real-world applicability of existing synthetic media defenses. We curate synthetic videos and text from the Internet community, and assess the effectiveness of state-of-the-art defenses on the collected datasets. In addition, we systematically measure the robustness of existing defenses by designing practical low-cost attacks, such as changing the configuration of generative models. Our findings reveal that most defenses show significant degradation in performance under real-world detection scenarios, which leads to the second thread of my work: (2) Building detection schemes with improved generalization performance and robustness for synthetic content. Many existing synthetic image detection schemes make decisions by looking for anomalous patterns in a specific type of high-level content, e.g., human faces, thus limiting their applicability. I propose a blind content-agnostic detection scheme called NoiseScope, which does not require synthetic images for training, and is applicable to a wide variety of generative models. For the text modality, our study reveals that state-of-the-art
defenses that mine sequential patterns in the text using Transformer models are not robust against simple attacks. We conduct further exploration towards enhancing the robustness of synthetic text detection by leveraging semantic features.
Dedicated to Virginia Tech.
Acknowledgments

To my parents: I am grateful for your unconditional love and long-term support. Both of you always listen to me with your heart and understand the decisions I made. I hardly visited you during my Ph.D. study due to the global pandemic, but you still supported me with all your strengths.

To my advisor Dr. Bimal Viswanath: Thanks for advising and guiding me over the past few years, from working on cutting-edge research problems to giving a presentation. Apart from being a Ph.D. student, you have also given me opportunities to be a leader/mentor to others. Your guidance and encouragement shaped me into a better researcher, and had a profound positive influence on my future career. I also appreciate your passion and dedication to research and teaching, and many times, you worked side by side with students to help us meet submission deadlines.

To my collaborators: (1) To Neal Mangaokar: It has been such a pleasant experience working with you during the past few years, and you are one of the best collaborators I could have ever had. (2) To Dr. Mobin Javed: Thanks for helping me with multiple projects and supporting me as a female researcher in the community. (3) To Zain Sarwar: Thanks for your dedication to the project during the pandemic. Your contribution to the project helped me immensely.

To my Committee members: Thanks to Dr. Danfeng Yao, Dr. Gang Wang, Dr. Taejoong Chung, and Dr. Peng Gao for the long-term support. Thanks for your insights and detailed feedback on my work!

To all my lab mates: Thanks for helping me and having daily discussions with me on research
projects. I have learned a lot from all of you. I could not have done this without your help and contribution to the project.
Contents

List of Figures \hspace{1cm} xiv

List of Tables \hspace{1cm} xviii

1 Introduction \hspace{1cm} 1
  1.1 Problem Motivation \hspace{1cm} 1
  1.2 Dissertation Objectives and Contributions \hspace{1cm} 5

2 Background \hspace{1cm} 10
  2.1 Generative Models \hspace{1cm} 10
  2.2 Benign Use Cases of Generative Models \hspace{1cm} 16
  2.3 Malicious Use Cases of Generative Models \hspace{1cm} 16

3 Synthetic Video Detection: Investigate Real-world Effectiveness of Existing Defenses \hspace{1cm} 19
  3.1 Introduction \hspace{1cm} 19
  3.2 Background and Related Work \hspace{1cm} 22
    3.2.1 Synthetic Videos \hspace{1cm} 22
    3.2.2 Methods for Generating Synthetic Videos \hspace{1cm} 23
  3.3 Synthetic Video Datasets \hspace{1cm} 25
3.3.1 Collecting DF-W: Our New Synthetic Video Dataset 25
3.3.2 Research Community Datasets 29
3.4 Analyzing DF-W 30
3.4.1 DF-W vs Research Community Datasets 30
3.4.2 Growth, Popularity, and Creators of DF-W Videos 34
3.5 Evaluating synthetic video detection 35
3.5.1 Experimental Setup 35
3.5.2 Performance of Existing Detection Schemes 39
3.5.3 Towards Improving Detection Performance 46
3.6 Conclusion 50

4 Synthetic Text Detection: Investigate Real-world Effectiveness of Existing Defenses 52
4.1 Introduction 52
4.2 Background and Goals 55
4.3 Models, Datasets and Metrics 58
4.3.1 Defenses: Synthetic Text Detection Schemes 59
4.3.2 Evaluation Metrics 63
4.3.3 In-the-wild Datasets 65
4.4 Defense Performance in the Wild 67
4.4.1 Detection Performance 67
4.4.2 Improving Performance in the Wild ........................................ 70

4.5 Defense Against Adaptive Attackers ........................................ 72
  4.5.1 Attack Methods ............................................................ 72
  4.5.2 Attack Evaluation ......................................................... 77

4.6 Discussion ........................................................................... 83

4.7 Conclusion ........................................................................... 85

5 NoiseScope: Towards Robust Detection of Synthetic Images .... 86

  5.1 Introduction ........................................................................ 86

  5.2 Background & Related Work ................................................ 88
    5.2.1 Synthetic Image Generation Methods ................................ 88
    5.2.2 Synthetic Image Defenses .............................................. 90

  5.3 Threat model .................................................................... 93

  5.4 Our approach: NoiseScope .................................................. 94
    5.4.1 Method Basics ............................................................ 94
    5.4.2 Detection Pipeline ....................................................... 98

  5.5 Experiment setup ............................................................... 102
    5.5.1 Real and Fake Image Datasets ....................................... 102
    5.5.2 Configuration of NoiseScope ......................................... 104
    5.5.3 Evaluation Metrics and Baseline Method ......................... 106
5.6 Evaluation of proposed scheme .......................... 106
  5.6.1 Analysis of Model Fingerprints .................. 106
  5.6.2 Detection Performance ............................ 108
5.7 Analysis of countermeasures .......................... 117
5.8 Real-world deployment of NoiseScope ................. 122
5.9 Conclusion ............................................. 123

6 Towards Robust Detection of Synthetic Text ... 124
  6.1 Motivation & Goals .................................... 124
  6.2 What is the Contributing Factor of FAST’s Robustness? 125
    6.2.1 Towards Understanding Adversarial Robustness of FAST 125
    6.2.2 Limitations of FAST ............................ 129
  6.3 Extracting Richer Semantic Features via Knowledge Graphs 129
  6.4 Is KG More Effective Than Entity Graph? ............. 133
  6.5 Conclusion ............................................. 134

7 Future Work ................................................ 135
  7.1 Improving Content-agnostic Synthetic Image Detection .... 135
  7.2 Improving Synthetic Text Detection Using KGs ............ 136
  7.3 Combating Misuses of Synthetic Media in the Future ......... 137

8 Conclusion .................................................. 141
List of Figures

1.1 The dissertation objectives. .................................................. 5

2.1 The architecture of Generative Adversarial Networks. ................. 11

2.2 Advances in GANs over the years. ........................................... 14

3.1 Frame samples of synthetic videos in the wild (YouTube). Red squares indicate synthesized areas in a frame. ........................................... 22

3.2 (a) Distribution of generation tools used for creating DF-W YouTube (b) CDF of DF-W video upload dates (c) distribution of generation tools used for creating DF-W YouTube (d) CDF of number of DF-W videos contributed by channels. ................................................... 30

3.3 DF-W F1 scores vs. $F_p$ for (a) CapsuleForensics (b) DSP-FWA. ........ 42

3.4 Visualization of salient regions (white dots) as determined by CapsuleForensics, and highlighted using IntGrad [1]. The first image is a fake face correctly classified as fake, and shows no-discernible-features. The second image is a real face correctly classified as real, and shows face boundary features. The third image is a fake face classified as real, and shows salient background features. Note that these are significant features, and do not preclude insignificant features such as those around the eyes and nose. ..................... 46

4.1 Distribution distance between the training set and the test sets (including baseline test set and In-the-wild test sets) of BERT-Defense and GROVER. . 70
4.2 The CDF of GRUEN score on GROVER’s baseline test set (Top-p 0.94 used in Table 4.1) and a dataset generated from GROVER with a decoding strategy of Top-p 1.

5.1 Camera fingerprints from Canon, iPhone, Nikon cameras (top) and GAN fingerprints from StyleGAN, CycleGAN, PGGAN, BigGAN (bottom).

5.2 Histogram of PCE correlation between model fingerprint and images (fake and real).

5.3 An illustration of NoiseScope detection pipeline: (a) Noise Residual Extractor, (b) Fingerprint Extractor via Clustering, (c) Fingerprint Classifier, (d) Fake Image Detector.

5.4 Change in CycleGAN fingerprint checkerboard when varying transpose convolution parameters.

5.5 (a) GAN fingerprint purity distributions (b) PCE Merging Threshold $T_{merge}$ vs. Detection F1 Score.

5.6 Detection performance (F1) with limited number of fake images in test sets.

6.1 An overview of the FAST pipeline.

6.2 Knowledge graphs (KGs) extracted from real and synthetic news articles.

7.1 Fingerprint analysis on most recent image generative models.

A.1 The CDF of GRUEN on the 4 In-the-wild datasets and 2 datasets generated from GROVER and GPT2-XL.
A.2 Detection performances of the defenses with different context window sizes, \textit{i.e.}, 64, 128, 256, 512 tokens. ........................................ 149

A.3 The CDF of GRUEN score on perturbed text produced by different adversarial perturbation methods, \textit{i.e.}, DFTFooler, random perturbations, and TextFooler, based on attacking RoBERTa-Defense (in Table 4.7). ........... 150

A.4 Evasion rate and average GRUEN score of perturbed text achieved by DFTFooler and random perturbations when attacking (a) FAST and (b) BERT-Defense, based on 5, 10, 15, and 20 word perturbations. ............. 150

B.1 Model fingerprints from StyleGAN-Face2 [2], before (left) and after (right) applying JPEG compression. ........................................ 151

B.2 Samples from StyleGAN-Face2 [2] that evaded detection when subjected to histogram equalization. Top row shows the images before equalizing, and the bottom row shows the images after equalizing. ....................... 151

B.3 Samples from CycleGAN-Zebra [3] that evaded detection when blurred. Top row shows the images before blurring, and the bottom row shows the images after blurring. ........................................ 152

B.4 Samples from BigGAN-DogHV [4] that evaded detection when blurred. Top row shows the images before blurring, and the bottom row shows the images after blurring. ........................................ 152
B.5 Image samples from StyleGAN-Face2 [2] subjected to a fingerprint spoofing attack against an increasing number of residual spaces. From left to right, we present (a) the original image, (b) the image spoofed against the Wavelet residual space (c) the image spoofed against the Wavelet and Blur residual spaces, and (d) the image spoofed against the Wavelet, Blur, and Laplacian residual spaces.
List of Tables

3.1 Statistics for the DF-W Dataset. .................................................. 25

3.2 Statistics of synthetic video datasets produced by the research community. . 28

3.3 Testing set sizes for evaluating detection performance on DF-W and DF-R
datasets. ......................................................................................... 38

3.4 Best attainable detection performances (F1) of various classifiers on the DF-
W and DF-R datasets, with corresponding precision (P) and recall (R) scores.
Best attainable detection performance (F1) for each dataset is underlined. . . 40

3.5 Distribution of generation methods for videos that were correctly classified
(TP) and incorrectly classified (FN) by CapsuleForensics and DSP-FWA. . . 42

3.6 Best attainable detection performances (F1) of Capsule-DA, Capsule-SE, MesoNet-
DA, MesoNet-SE and DFDC winner Seferbekov’s model on the DF-W and
DF-R Datasets, with corresponding precision (P) and recall (R) scores. Here
‘DA’ refers to models after domain adaptation retraining, and ‘SE’ refers
to models after source expansion retraining. Retraining is achieved by fine-
tuning the whole model, i.e., without freezing layers. Best attainable detect-
tion performance (F1) for each dataset is underlined. . . . . . . . . . . . . . . 47
4.1 Details regarding the training and evaluation of the defenses. From left to right: Datasets used for training and testing defense models; Decoding strategy used to generate synthetic data; Number of samples in training and test datasets; Detection performance of the defenses (F1, P, R, AUC represents F1 score, Precision, Recall, AUC ROC score, respectively).

4.2 Details of the In-the-wild datasets.

4.3 Performance of the defenses, i.e., F1 (%) and ΔF1 (%) of the synthetic class, on the In-the-wild datasets. The percentage change in F1 (ΔF1) is computed from the baseline performance of each defense. ‘+’ means performance improvement and ‘-’ means performance degradation. “−” (the longer minus mark) indicates experiments we ignored: We did not test GROVER, FAST and RoBERTa-Defense on non-news domain datasets, i.e., Kafkai and RedditBot. This is because these defenses are only trained for the news domain.

4.4 Detection performance in F1 (%) of BERT-Defense and GROVER on In-the-wild datasets before and after fine-tuning on a limited set of articles from the real and synthetic classes. “−” represents experiments we ignored: GROVER is a news domain defense, and not applicable to non-news domain datasets, i.e., Kafkai and RedditBot.
4.5 Performance of attacks that change the decoding strategy of the LM. Evaluation metrics include Recall (R) of the synthetic class and the average GRUEN (GRN) of synthetic data. We show the percentage change of each evaluation metric ($\Delta R$, $\Delta$GRN) from the baseline performance of each defense on the most effective attack of each decoding strategy. ‘+’ means performance improvement and ‘-’ means performance degradation. “-” (the longer minus mark) indicates experiments we ignored: We only consider Temperature decoding for GLTR defenses, because the original GLTR work was evaluated using temperature-based decoding.

4.6 Performance of attacks which prime the LM with a different number of priming tokens. Evaluation metrics include Recall (R) of the synthetic class and the average GRUEN (GRN) of synthetic data. We show the percentage change of each evaluation metric ($\Delta R$, $\Delta$GRN) from the baseline performance of each defense on the most effective length of priming tokens. ‘+’ means performance improvement and ‘-’ means performance degradation.

4.7 Attack performance of the three adversarial perturbation methods (i.e., DFT-Fooler, random perturbations, TextFooler) against the defenses, based on 10 word perturbations. GRN-Before: the average GRUEN of original datasets; GRN-After: the average GRUEN of datasets produced by adversarial perturbation attacks; ER: Evasion Rate (%).

4.8 Detection performance (Recall) of BERT-Defense, GLTR-GPT2, and GROVER when fine-tuned to their most effective adaptive attack setting as indicated. $\Delta R$ is percentage change of Recall from Recall before training to Recall after training.
5.1 Basic information of 11 synthetic image datasets evaluated in Section 5.6.2. 102
5.2 Performance of NoiseScope and CSD-SVM ($\nu = 0.1$). 109
5.3 Detection performance (F1) on imbalanced test sets with different ratio of fake to real images. 109
5.4 Improved detection performance by increasing $T_{\text{merge}}$ in non-performant imbalanced configurations. 111
5.5 Detection performance on test set with fake images from multiple (GAN) sources. 114
5.6 Fake Image Detector Performance (F1) using different denoising filters. Bold numbers highlight the best performance in each dataset. 114
5.7 Performance (F1) of NoiseScope under different countermeasures. ‘Original’ means no countermeasures were used. 117
6.1 Detection performance (Recall) of DistilFAST, FAST and RoBERTa-Defense on attacks which change the decoding strategy used by the LM or prime it with varying numbers of priming tokens. 127
6.2 Detection performance (F1) of DistilFAST, FAST and RoBERTa-Defense on AI-Writer and ArticleForge. 128
6.3 Detection performance (F1 Scores on the synthetic class) on KG-GCN, Entity-GCN. 133
A.1 Websites used to scrape real news articles for ArticleForge and AI-Writer datasets. 147
A.2 Performance of the defenses on the In-the-wild datasets. We present F1 score (F1), Precision (P), and Recall (R) of the synthetic class in percentages. We did not test GROVER, FAST and RoBERTa-Defense on non-news domain datasets, which includes Kafkai and RedditBot. This is because these defenses are only trained for the news domain.
Chapter 1

Introduction

1.1 Problem Motivation

Recent progress in generative models has made it possible to generate realistic synthetic media content, including synthetic images, videos, and text [5, 6, 7, 8]. An example of such generative models is the Generative Adversarial Network (GAN) [9] proposed by Goodfellow, capable of generating hyper-realistic content. Over the last few years, a large amount of advanced generative models have been developed to generate convincing synthetic content that even human cannot recognize as fake [6, 10, 11]. Popular image generation models include GANs [12, 13], VAEs [14, 15] and Diffusion models [16, 17]. Popular text generation models include GROVER [18], GPT-2 [19], GPT-3 [20], XLNet [21], and TransformerXL [22]. These generative models have enabled impressive applications in healthcare [23], computer vision [24, 25, 26], computer graphics [27, 28], programming language code generation [29, 30] and other domains [31, 32, 33].

However, synthetic content produced by generative models, also known as deepfakes, can be misused by bad actors for malicious purposes. Synthetic videos with altered identities have been frequently observed on popular video platforms, showing public figures doing things they have never done [34, 35]. More severely, synthetic images and videos can launch cybercriminals that use synthetic faces to dupe biometric verification, i.e., synthetic identity fraud [36]. State-of-the-art language models (LMs) can generate high-quality synthetic text
to compromise the communication of business emails, propagate fake news and manipulate public opinions (e.g., rumor spreading, election tampering [18]).

Both governments and industrial companies are realizing the threats posed by synthetic content, and in the US, it is considered a national security challenge in the coming years. Through the Defense Advanced Research Projects Agency (DARPA), the US government is currently collaborating with several of the country’s top research institutions to combat synthetic content [37, 38]. The FBI’s Cyber Division also expressed the concern that “Malevolent actors almost certainly will leverage synthetic content for cyber and foreign influence operations in the next 12-18 months” [39]. Early in 2022, synthetic media powering disinformation campaigns influenced the course of the war in Ukraine [40]. In June 2022, the FBI warned about the use of deepfake technology to apply for remote jobs [41]. Such threats damage the trust that users place in online content, where even authentic content may be claimed as synthetic.

One effective approach to circumvent such threats is to build detection schemes for synthetic content. Till today, the research community has proposed a variety of detection schemes for detecting synthetic images [42, 43], videos [44, 45, 46], and text [18, 47, 48, 49]. In industry, tech giants such as Facebook, Google, and Microsoft have been leading efforts to defend against synthetic content, including developing new detection schemes, curating benchmark datasets, and launching deepfake detection competitions [50, 51]. However, the research community still needs to overcome certain limitations of existing work. This Ph.D. dissertation aims to address the following challenges faced by the community:

**Challenge 1: Existing defenses against synthetic content have been studied with limited or no knowledge of synthetic content in the wild.**

*Synthetic video defenses.* The amount of synthetic videos on mainstream platforms or web-
sites have been rapidly growing these days [52]. Particularly, massive synthetic videos featuring celebrities have been seen on websites like YouTube \(^1\) and Tiktok \(^2\). However, the research community has little understanding of synthetic videos in the wild in terms of their quantities, generation tools, and creation purposes. It is also unclear whether existing defenses can accurately detect these videos as well as the performances reported in their work. Existing synthetic video datasets used by the research community are produced by researchers themselves using a single or a limited number of generation methods. We do not know how they differ from synthetic videos “in the wild” in terms of data distribution. Therefore, the real-world effectiveness of existing detection schemes needs to be explored in depth. We conduct a large-scale systematic measurement study of synthetic videos in the wild to address this challenge.

**Synthetic text defenses.** As language models (LMs) evolves, synthetic text generated by transformer-family models [53] has emerged on mainstream discussion platforms such as Reddit.com \(^3\). Meanwhile, there are synthetic text generation services that could be misused to create fake news articles, fake reviews, or fake web articles for BlackHat SEO activities. These services claim to use state-of-the-art Transformer LMs or their customized versions to generate synthetic text. To our best knowledge, there are no in-the-wild synthetic text datasets collected for research purposes. This again highlights the need to understand the real-world effectiveness of existing defenses, because text generators used in the wild can be different from those used by the research community. We conduct a large-scale systematic measurement study on synthetic text in the wild to address the challenge.

**Challenge 2: Existing defenses against synthetic images are poorly generalized and highly content-specific, which raises serious issues.** Most synthetic image
detection schemes are primarily based on supervised learning [44, 45, 48, 54]. In practice, it takes significant efforts to obtain a priori access to a large number of synthetic images or knowledge of the generative models used for training. Even with such presumptions, existing supervised schemes trained on a limited set of synthetic images do not usually generalize well to unseen data. In this regard, researchers have to constantly develop or adapt detection techniques for synthetic images produced by emerging generative models. Besides, most existing defenses are primarily trained and tested on one type of image content [55, 56, 57, 58, 59], e.g., human faces. These methods mainly pay attention to abnormal patterns in the high-level content space, leading to their limited applicability. Recent studies have found analyzing high-level content in images can easily lead to issues of bias [60]. Considering the real-world detection scenario, it is crucial that detection schemes can generalize to unseen synthetic images and different types of high-level content. To overcome this limitation, we build an unsupervised and content-agnostic synthetic image detection scheme.

**Challenge 3: Limited understanding of the robustness of existing defenses against synthetic text.** While existing synthetic text defenses have achieved high detection performances [18, 47, 48, 49], researchers have a limited understanding of their robustness, i.e., how robust are existing detection schemes against an adaptive attacker who is knowledgeable of the defense? Besides, existing defenses are mostly trained to learn token-level patterns in the synthetic text, which is generated based on a specific decoding or priming condition of the language model. However, what if the attacker changes the decoding/priming parameters of the language model? It remains unclear whether defenses can identify synthetic text generated under a new configuration. Another strategy to fool detectors is to craft adversarial perturbation, which aims to evade detection by altering words while maintaining the linguistic quality of synthetic text. To our best knowledge, no existing work has explored above attack scenarios for synthetic text defenses. Hence there
1.2. DISSERTATION OBJECTIVES AND CONTRIBUTIONS

Figure 1.1: The dissertation objectives.

is an urgent need to systematically evaluate the robustness of existing defenses and develop robust defenses against synthetic text.

1.2 Dissertation Objectives and Contributions

To overcome above challenges, I propose 4 dissertation objectives, and summarize the contributions as follows:

Objective 1: Investigating the real-world effectiveness of existing synthetic video defenses. To overcome Challenge 1, we collect synthetic videos from mainstream video platforms and take a deep dive into analyzing in-the-wild synthetic videos. Specifically, we characterize these videos collected from the wild by comparing them with those curated by the research community. Our analysis considers the following aspects: (1) Investigate the growth, popularity, and creators of synthetic videos in the wild. To be more specific, how fast are synthetic videos on the Internet growing? How many audiences have they reached these days? (2) How synthetic videos in the wild differ from those produced by the research community, in terms of quality and generation methods? We systematically assess how effective existing state-of-the-art defenses (in terms of raw detection performances) on synthetic videos in the wild and identify their limitations. We conduct more analysis towards
understanding cases of detection failure via model interpretation tools.

Contribution 1: Collect and characterize in-the-wild synthetic videos, assess the real-world deployment readiness of existing defenses, and explore the improvements. We collect and curate a novel dataset from the web called DF-W, comprising 1,869 synthetic videos in the wild — the largest dataset of real-world synthetic videos at the time of study. We release the collected DF-W dataset to the community. We present a comprehensive analysis of the videos in DF-W. The analysis reveal that DF-W videos differ from synthetic videos in existing benchmark datasets collected by the research community in terms of content and generation methods used, raising new challenges for detecting synthetic videos in the wild. We further systematically evaluate multiple state-of-the-art detection schemes against DF-W, revealing poor detection performance. This suggests a distributional difference between our DF-W dataset, and synthetic videos in research community datasets. We attribute detection failures to be related to racial biases of existing defenses, and using model interpretation schemes, we investigate features that can be leveraged to either improve or evade detection. Lastly, we propose to employ domain adaptation via transfer learning to improve detection performance on DF-W. Overall, our findings indicate a need for incorporating in-the-wild synthetic videos, e.g., DF-W, into future work.

Objective 2: Investigating the real-world effectiveness of existing synthetic text defenses. To overcome Challenge 2, we collect datasets from the web containing both synthetic and real articles in matching topic domains. This includes synthetic text posted by Internet users, and text from emerging text-generation-as-a-service platforms, geared towards the SEO community. We conduct our analysis considering the following two aspects: (1) We systematically evaluate how effective existing state-of-the-art defenses (in terms of raw detection performances) are against synthetic text in the wild and identify their limitations. (2) We aim to understand the robustness of existing defenses against an adaptive attacker who
1.2. DISSERTATION OBJECTIVES AND CONTRIBUTIONS

seeks to evade detection. Particularly, we propose low-cost and practical adaptive attacks, i.e., changing the text generation process and adversarial perturbation attacks.

**Contribution 2**: Collect in-the-wild synthetic text, assess the real-world deployment readiness and adversarial robustness of existing defenses. We present the first systematic evaluation of state-of-the-art synthetic text defenses on real-world synthetic text on the Internet. We first collect synthetic text from the Internet and release 4 new real-world synthetic datasets to the community. We evaluate all the state-of-the-art defenses against in-the-wild datasets. The evaluation demonstrates that most defenses fail to generalize to in-the-wild synthetic text, and are not robust under realistic detection scenarios. We evaluate all the state-of-the-art defenses against simple, computationally low-cost attacks that modify the attacker’s text generation process. Our experiments show that just changing the decoding or text sampling strategy is sufficient to break many defenses. We also propose a novel black-box adversarial sample crafting strategy called DFTFooler, which bypasses the defenses without requiring any queries to the defense model.

**Objective 3**: Towards robust detection of synthetic images. To overcome Challenge 3, we develop an unsupervised synthetic image detection scheme, which requires no a priori access to fake images or knowledge of generative models used by the attacker. Meanwhile, the proposed detection scheme is content-agnostic, and analyzes artifacts/patterns in the “noise” space (or non-content space) instead of the high-level content. Therefore, the proposed defense can apply to diverse types of synthetic content (i.e., not only faces but also other content types such as synthetic medical and satellite imagery) and multiple types of GANs.

**Contribution 3**: Build an unsupervised, content-agnostic detection scheme for synthetic images. We present NoiseScope, an unsupervised detection scheme that leverages unique patterns in the noise space of synthetic images left by generative models. Given a test set with an unknown number of fake and real (produced by a camera) images, NoiseScope extracts
any available model fingerprints and uses the fingerprint to detect fake images in that set. In contrast to supervised schemes, our method is agnostic to the type of GAN used, and is also effective when the test set contains images generated from multiple GANs. Our method works for any type of high-level image content (content-agnostic), as it leverages only low-level noise patterns. We evaluate NoiseScope on 11 diverse synthetic image datasets, created using 4 state-of-the-art GANs. NoiseScope can detect fake images with up to a 99.68% F1 score. Lastly, we extensively evaluate NoiseScope against a variety of countermeasures by assuming an attacker who is aware of NoiseScope’s detection pipeline.

**Objective 4: Towards robust detection of synthetic text.** Generating semantically coherent text is still a challenging task for language models [61]. For example, the relationship between named entities over consecutive sentences in the synthetic text may not be consistent. With our work targeting Objective 2, we observe the FAST approach [49] turns out to be the most robust defense out of the 6 state-of-the-art defenses we evaluated. We take a deeper dive to analyze each component in the FAST pipeline, and identify a promising direction, i.e., leveraging semantic features, that can robustify synthetic text detection. We conduct experiments to validate our hypothesis.

**Contribution 4: Identify key features that robustify synthetic text detection.** According to our study on the 6 state-of-the-art synthetic text defenses (work in Chapter 4), the detection scheme FAST [49] holds up more consistently against different adaptive attacks, and also generalizes well to synthetic text in the wild. In Chapter 6, we take a deeper dive into FAST, and investigate the key contributing factor of FAST’s robustness. Specifically, we conduct ablation studies by comparing the detection performances of FAST and a “distilled” version of FAST (which is called DistilFAST). Our experiments indicate that semantic features extracted from named entities are the key contributing factor to the robustness of FAST. However, FAST still shows limitations under certain attack scenarios. Therefore, we
further leverage *Knowledge Graphs (KGs)* to extract richer semantic features that can help discriminate real and synthetic text. We conclude that leveraging semantic features of the text extracted via KGs is a promising future direction for robust detection of synthetic text.

**Dissertation Structure:** The rest of this Ph.D. dissertation is organized as follows. Chapter 2 introduces background and related work of this dissertation. Chapter 3, Chapter 4, Chapter 5, and Chapter 6 present my work corresponding to Objective 1, 2, 3 and 4, respectively. Chapter 7 and Chapter 8 present the future work and the concluding remarks of this Ph.D. dissertation.
Chapter 2

Background

I start this chapter by introducing the basic knowledge of generative models that enable synthetic media generation, including GANs [9], RNNs [62], LSTMs [63], and Transformers [53]. Then I briefly discuss popular approaches used to produce synthetic content as well as their benign and malicious use cases. In the rest of this dissertation, synthetic content generally refers to three modalities of synthesized digital content (i.e., images, videos and text) that are manipulated or produced by a deep generative model and appears convincingly realistic.

2.1 Generative Models

Generative models learn to generate new data samples with the underlying distribution of training data [64], and have been widely used in many subfields of AI and Machine Learning. Recent progress in stochastic optimization methods and advances in building more extensive deep neural networks have enabled scalable modeling of complex, high-dimensional synthetic data, including images [6], video [65], and text [66], which are also known as deepfakes. GAN-family models have shown impressive performances in generating realistic synthetic images and videos over the last few years [5, 6, 10, 11, 67, 68, 69]. In the discrete domain, text generation instead relies on different types of neural networks due to its sequence property, namely, RNNs [62], LSTMs [63] and Transformers [53]. Next, I will briefly introduce the basics of these generative models.
2.1. Generative Models

**GAN basics.** In 2014, Goodfellow et al. [9] proposed the Generative Adversarial Network (GAN). As shown in Figure 2.1, a GAN includes two neural networks, a *generator* ($G$) that produces synthetic data (*e.g.*, images), and a *discriminator* ($D$) that takes the synthetic data and gives feedback to the generator on how well it resembles real data. The two components are trained simultaneously in an adversarial manner such that the generator learns to produce synthetic data that are indistinguishable from real data, and the discriminator learns to distinguish between real and synthetic data (produced by the generator). Therefore, the idea is to optimize one main objective, *i.e.*, adversarial loss, to make the generated data indistinguishable from real data. The literature on GANs has been growing rapidly. Today, state-of-the-art GAN models are able to generate realistic-looking synthetic images and videos [5, 6, 10, 11, 67, 68, 69].

**RNNs [62] & LSTMs [63].** Compared to image and video generation, text generation is a different task in which one predicts the next token or word of the sequence. Traditional neural networks struggle to model a sequence pattern because token generation at one-time step needs to be informed by previous tokens. This issue was later addressed by a particular type of deep learning models called Recurrent Neural Networks (RNNs). RNNs use a recurrent loop to maintain an internal “hidden state” that stores information about previous tokens, allowing the model to learn temporal dynamic behavior. However, a limitation of RNNs is that they are unable to connect the information with long-term dependencies as the temporal...
gap grows. This limitation was eased by a later proposed model, Long Short Term Memory networks (LSTMs), explicitly designed to learn long-term dependencies. Until 2017, RNNs and LSTMs were considered state-of-the-art DNN-based language models. However, RNNs and LSTMs still suffer an inability to generate longer coherent text. There are still several limitations of RNNs and LSTMs, including vanishing/exploding gradients and the sequential nature of the model limiting parallelization.

**Transformers [53].** In 2017, Vaswani et al. proposed Transformers to address limitations of RNNs and LSTMs. Transformers, in contrast with RNNs/LSTMs, are solely based on the “attention” mechanism [70]. Instead of using a single hidden state to represent all previous tokens, attention mechanisms allow the model to compute vector representations of each token separately. These representations take into account context and relationship with all other tokens. Attention mechanisms can also be customized to “pay attention” to only previous tokens (unidirectional attention), or to also pay attention to future tokens, assuming they are provided (bidirectional attention). Another advantage of transformers is that each token can be processed in parallel, and is not dependent on previous tokens. With their attention mechanism and parallelization capabilities, transformers now power state-of-the-art for many NLP tasks. Examples include BERT [71], RoBERTa [72], GPT-2 [66], GPT-3 [20], and Transformer-XL [22].

**Methods to generate synthetic images.** Hundreds of GAN models have been developed since Goodfellow et al. proposed the original GAN in 2014. Certain key GANs have established a constructive foundation in the field, achieving impressive results. In particular, DCGAN [73], proposed by Radford et al., improved the performance of image generation by incorporating convolutional layers for both generator and discriminator, which is the groundwork for the rapid development of a large number of GAN variations later on. In 2017, CycleGAN [74] advanced the state-of-the-art image-to-image translation by enabling
unpaired images for training, improving over the previous approach Pix2Pix [75], which requires paired images for training. In 2018, PGGAN [5] demonstrated a huge improvement in the quality of synthetic images by adding more layers as training progresses to produce higher resolution images with finer details. Meanwhile, Brock et al. introduced BigGAN [11], which uses a range of techniques to improve GAN training and generated image quality, including increased batch size, increased number of layer channels, and shared embeddings for batch normalization layers in the generator. In 2019, Karras et al. proposed StyleGAN [6], which renovates the generator architecture by inserting the latent code at different layers of the model, further boosting the generated image quality.

To this date, images generated by state-of-the-art GANs are challenging for humans to recognize. Figure 2.2 shows the image quality of key GAN models developed over the years. Besides, other models such as likelihood models and diffusion probabilistic models have been proposed to generate high-quality images. Likelihood models handle generative modeling with a Bayesian approach and attempt to model the distribution of the data by maximizing a log-likelihood function. The most known likelihood model is the Variational Autoencoder (VAE) [76]. Recently, denoising diffusion probabilistic models (DDPMs) [17] have become increasingly popular due to its impressive performance. Diffusion models train neural networks to perform a reverse-step of a gaussian-noise addition process at a certain time step.

**Methods to generate synthetic videos.** This thesis focuses on a particular type of synthetic video — face-swapped videos. In the rest of this thesis, “synthetic video” specifically refers to videos containing face-swapped content, and “real video” refers to original videos from camera devices. Both the research community and the Internet community have been contributing methods and tools to create synthetic videos. Generation methods proposed by

---

1https://thispersondoesnotexist.com/
the research community are mainly based on GANs, VAEs, and encoder-decoder frameworks, including FSGAN [67], IPGAN [68], Fast Face Swap CNN [77], FaceShifter [78], FSNet [79], RSGAN [69], and DF-VAE [80]. Meanwhile, developers from the Internet community have also released a set of popular tools, e.g., FaceSwap, DeepFaceLab, Zao, FakeApp, and FaceSwap-GAN [81], Dfaker, MyFakeApp [83], DeepFakesapp.online [84], DeepFakes web β [85], and DeepFake.me [86]. Among all the tools, FaceSwap with two deep autoencoders is the earliest open-source tool released in 2017. It provides the architecture foundation for many following released tools. For instance, DeepFaceLab, launched in 2018 as a fork of FaceSwap, is claimed to be the most popular tool for generating synthetic videos on the Internet due to its impressive performance. More details of synthetic video generation tools can be found in Section 3.2.2.

**Methods to generate synthetic text.** Previously, RNNs [62] and LSTMs [63] have shown exemplary performance in producing short text, e.g., generating product reviews. In 2017, the introduction of Transformers overcame the limitations of traditional RNNs, improving the state-of-the-art of text generation further. Frontier deep learning models of the Transformer family, e.g., GPT-2 and GPT-3 with billions of parameters, can generate synthetic

---

2[https://faceswap.dev/](https://faceswap.dev/)
3[https://github.com/iperov/DeepFaceLab](https://github.com/iperov/DeepFaceLab)
4[https://www.zaoapp.net/](https://www.zaoapp.net/)
5FakeApp is shut down, no link available
text with human-like linguistic quality on diverse topics. Other generative models in Transformer family includes GROVER [18], XL-Net [21], TransformerXL [22], RoBERTa [72] and BERT [71]. Today, people can leverage the Transformer family of pretrained models readily available in python libraries such as huggingface ⁶ for synthetic text generation. Besides, people can also generate synthetic text through paid website services/platforms by specifying a few keywords, e.g., AI-writer ⁷, Kafkai ⁸, AriticleForge ⁹. Such website services are mostly supported by Transformer-family models. Both open-source tools and paid services require little expertise from bad actors, thus the barrier of generating synthetic text is unprecedentedly low [65].

**Evolving deepfakes.** Deepfake techniques are evolving at a faster speed than people can imagine. Since StyleGAN was proposed in 2019, its enhanced versions including StyleGAN2 [87] and StyleGAN3 [12] have been proposed to generate even more hyper-realistic images. Other variations of image generation tasks such as text-to-image generation have achieved breakthrough performances recently, e.g., DALL-E [88] and DALL-E 2 [89], which are Transformer models developed by OpenAI to generate images from natural language descriptions. Apart from traditional face-swapping deepfakes, full-body deepfakes have been developed for various application scenarios [90]. For text generation, GPT-family models keep emerging, and show promising results in tasks like code completion, chatting, blog post writing and many more.

⁶https://huggingface.co/
⁷http://ai-writer.com/
⁸https://kafkai.com
⁹https://www.articleforge.com/
2.2 Benign Use Cases of Generative Models

Researchers have been using synthetic data in multiple application domains to improve the performance of model training. Frid-Adal et al. employ synthetic medical images as complementary training data to advance the state-of-the-art performance in liver lesion classification [91]. Other benign use cases include de-identification [92], feature extraction [93], video prediction [94], and image editing [95]. Outside the research community, online websites and mobile applications employ generative models to create synthetic art pieces or stylized pictures for users [96]. For instance, OpenAI recently released DALL·E [88] and DALL·E 2 [89] which can create astonishing AI art in an instant given a text prompt. Synthetic videos have been presented in TV commercials and talk shows [97]. The Japanese artificial intelligence company Data Grid developed an AI that can automatically generate whole body models of nonexistent persons, identifying practical applications in the fashion and apparel industries [98]. Besides, many influential content creators have been making synthetic videos to earn profits by getting clicks and views on popular platforms like YouTube and Reddit [99]. Apart from synthetic images and videos, synthetic text generated by state-of-the-art LMs with billions of parameters can be used in many benign applications, including generating content for entertainment purposes (e.g., stories, jokes) [100], building dialog systems [101], text summarization [102], and automated journalism [103].

2.3 Malicious Use Cases of Generative Models

Nevertheless, the society has raised deep concerns about synthetic content as it can be used for various malicious purposes such as personal attacks, electoral fraud, and disinformation campaigns.
Malicious use cases of synthetic images. Misusing synthetic images poses severe threats to critical systems in our society. One type of malicious synthetic images are synthetic satellite images, on which fake disaster-like incidents can be injected to trigger public panic. Researchers recently warn that such “deepfake geography” and “location spoofing” are threatening national security and social stability [104]. Synthetic imagery attacks can also happen in other domains. In the healthcare system, medical practitioners and AI-based diagnosis systems rely on medical images such as Chest X-rays to monitor diseases and prepare treatment plans. An attacker can access and alter a victim’s medical images with carefully engineered synthetic images to compromise the medical diagnostics pipeline. Such malicious activity can be motivated by financial fraud, as misdiagnosis will force the financial hand of patients and insurance providers. My other research work proposes a generic attack called Jekyll [105], which produces synthetic medical images that can cause a well trained human medical professionals and a diagnostic ML algorithm to misdiagnose the victim’s health condition.

Malicious use cases of synthetic videos. Synthetic videos can be misused by attackers to trigger both individual harm and social hazards. A video forum called MrDeepFakes [106] has been hosting a large amount of deepfake celebrity porn videos and nude photos of actresses. Blackmailers might create such synthetic videos to extort money or confidential information from targeted individuals as disproving the videos could be hard. Other than these individual harms, synthetic videos also have the potential to cause harms that will impact national security, democracy and privacy. In early 2022, a synthetic video featured with Zelensky was used in Russia-Ukraine war to spread fake news [107]. Other horrifying scenarios can happen in the near future. Synthetic videos could show public officials taking bribes, making racist comments, or engaging in illegal activities. Such misuse of synthetic content will cause significant social unrest and fragmentation.
Malicious use cases of synthetic text. In the text domain, state-of-the-art LMs, *e.g.*, GPT-2 and GPT-3, can generate convincing fake news [18], which can easily enable large-scale disinformation campaigns, email phishing, and rumor spreading. OpenAI, the company released GPT-2 and GPT-3, noted that the models could be misused as they could help generate “synthetic propaganda” for extreme ideological positions [108]. GPT-powered bots now can act as an actual user to post comments, interact with users, and impersonate people online [109]. It is possible that attackers use such bots to post abusive or fake content on social media or automate the production of spam and phishing emails.
Chapter 3

Synthetic Video Detection:
Investigate Real-world Effectiveness of Existing Defenses

3.1 Introduction

Advances in deep neural networks (DNNs) have enabled new ways of manipulating video content. This has fueled the rise of synthetic videos, or videos where the face of a person is swapped in by another face, using DNN-based methods. Given the potential for misuse of this technology, researchers have proposed a variety of synthetic video detection schemes \[45, 110, 111, 112, 113, 114, 115, 116]\, and also released new synthetic video datasets \[44, 51, 80, 111, 117, 118, 119]\ to evaluate the proposed detection schemes. However, most existing research efforts from academia and industry have been conducted with limited or no knowledge of actual synthetic videos in the wild. Therefore, we have a limited understanding of the real-world applicability of existing research contributions in this space (See Challenge 1 discussed in Chapter 1).

In this chapter, we aim to better understand synthetic videos in the wild, and identify challenges with real-world deployment of existing synthetic video detection schemes. Towards
CHAPTER 3. SYNTHETIC VIDEO DETECTION: INVESTIGATE REAL-WORLD EFFECTIVENESS OF
EXISTING DEFENSES

This goal, a number of questions can be raised: (1) How are synthetic videos created in the wild? (2) Are synthetic videos in the wild (i.e., not created by researchers) different from those produced by the research community? (3) Are synthetic videos increasingly appearing in the wild? Are they being viewed by large populations? (4) Can existing synthetic video detection schemes (primarily tested on synthetic videos produced by the research community) accurately detect synthetic videos in the wild?

This chapter aims to answer above questions by conducting a large-scale measurement study of synthetic videos in the wild or synthetic videos produced and shared by the Internet community (Objective 1 in Chapter 1). We first collect synthetic videos from the Internet, i.e., video-sharing platforms/forums, and characterize them in contrast to synthetic videos created by research communities. We then evaluate the performance of state-of-the-art detection schemes on in-the-wild synthetic videos to identify challenges with real-world deployment of existing detection schemes. To the best of our knowledge, this is the largest measurement study at the time of our study. Our contributions include the following:

- **We introduce a new synthetic video dataset called DF-W, comprised of synthetic videos created and shared by the Internet community.** We prepare this dataset by scanning a variety of sources on the Web, including YouTube, Bilibili [120], and Reddit.com. Our dataset includes a total of 1,869 videos from YouTube and Bilibili, comprising of over 48 hours of video, covering a wide range of video resolutions. To the best of our knowledge, DF-W is the largest collection of synthetic videos (deepfake videos) in the wild.

- **We present a comprehensive analysis of the videos in DF-W.** We examine the differences in content between synthetic videos in DF-W, and datasets released by the research community [44, 51, 111, 117, 118, 119]. We observe that DF-W videos tend to be more sophisticated, and include several variations of synthetic content, thus raising new challenges for detection schemes. We find that many DF-W videos are created using generation methods
different from those used by the research community, which potentially results in a data distribution gap between existing synthetic datasets and DF-W synthetic videos. We also analyze the growth, and popularity of synthetic videos in the wild, and investigate the content creators involved in the process.

- **We systematically evaluate the performance of state-of-the-art synthetic video detection schemes on videos in DF-W.** We evaluate 7 detection schemes, including 5 supervised and 2 unsupervised schemes. We find that all detection schemes perform poorly on DF-W videos, with the best approach (CapsuleForensics, a supervised approach) having an F1 score of only 77% in catching synthetic videos. This means that these existing detection schemes are not ready for real-world deployment. Poor performance can be attributed to distributional differences between real-world synthetic videos, and those used to train existing detection schemes. Failure cases can also be partially attributed to racial bias, a well known problem with DNN-based facial analysis [121]. We also attempt to interpret the classification decisions using a state-of-the-art model interpretation scheme, called Integrated Gradients [1]. We leverage this tool to infer features that can be used to either improve detection schemes, or create more evasive synthetic videos.

- **We explore approaches to improve detection performance.** Finally, to improve detection performance on DF-W, we leverage a transfer learning-based domain adaptation scheme, which shows promising results on the DF-W dataset. However, domain adaptation still requires a small number of synthetic videos from the target distribution/domain (DF-W in this case). Therefore, the attacker still has an upper hand, putting the defender in a difficult situation, unless we come up with defenses that can generalize better. We also investigate the performance of the winning DNN model from Facebook’s DFDC competition [122]. While this winning model outperforms the existing models (without domain adaptation), its performance on DF-W is still inadequate with an F1 score of 81% and low precision of 71%.
CHAPTER 3. SYNTHETIC VIDEO DETECTION: INVESTIGATE REAL-WORLD EFFECTIVENESS OF EXISTING DEFENSES

Figure 3.1: Frame samples of synthetic videos in the wild (YouTube). Red squares indicate synthesized areas in a frame.

We release the DF-W dataset with the goal of enabling further work on synthetic video detection. The DF-W dataset is available on our GitHub repositoryootnote{https://github.com/jmpu/webconf21-deepfakes-in-the-wild}.

3.2 Background and Related Work

3.2.1 Synthetic Videos

A synthetic video is popularly characterized as a video that has been manipulated using deep neural networks (DNNs), with the goal of simulating false visual appearances\cite{110, 117, 123, 124}. We focus on the most popular type of synthetic videos (also known as deepfake videos) found on the Internet, categorized as face-swapped videos\cite{116}. This technique attempts to replace the face of an individual with that of another, while retaining the expression, pose, and background area of the image. Figure 3.1 shows examples of such videos found in the wild that appear very convincing. In the rest of this work, we use the term “synthetic video” to refer to videos containing face-swapped content, and “real video” to refer to non-synthetic videos.
3.2. Background and Related Work

Synthetic videos were first seen online in 2017, created by a Reddit.com user with the self-appointed name /u/DeepFakes [125]. This user created videos of celebrities face-swapped into illicit videos, while only using publicly available data and the Tensorflow machine learning library. It was later revealed that the inspiration for their face-swap algorithm was an unsupervised image-translation work from NVIDIA [126].

Since then, Internet communities have produced several synthetic video generation tools (see Section 3.2.2) by leveraging state-of-the-art deep generative models proposed by the research community. Such models include Autoencoders [62], Variational Autoencoders [76], Convolutional Networks [127], and Generative Adversarial Networks (GANs) [9]. At the same time, the research community itself independently produced several variants of synthetic video generation schemes (see Section 3.2.2). In this chapter, we primarily focus on synthetic videos produced by the Internet community, and appearing in the wild.

3.2.2 Methods for Generating Synthetic Videos

We describe synthetic video generation methods developed by both the Internet and research communities. To aid our discussion, we adopt the following convention: the source face is the face to be swapped in, and the target face is the face that will be replaced.

Methods by the Internet community. The following methods developed by the Internet community, have been used to produce synthetic videos we find on the Internet.

(1) FaceSwap\(^2\): FaceSwap is an open-source tool created in 2017, using two deep autoencoders that share the encoder module, but use different decoders. The two autoencoders are trained separately on the source and target faces to reconstruct each face from a latent representation. The use of the shared encoder enforces a shared latent space, ensuring that

\(^2\)https://faceswap.dev/
the encoder disentangles facial identity from facial expression. To swap a face, the target face is fed into the shared encoder, and the source’s decoder is used to decode the latent representation. The result is the swapped face, i.e., with the source’s face, but the target’s facial expression and pose. The resulting face can then be spliced into the target image.

(2) **DeepFaceLab (DFL)**: DFL is another open-source project created in 2018 as a fork of FaceSwap. On their website, it is claimed that “More than 95% of synthetic videos are created with DFL”. DFL boasts improvements over FaceSwap’s model. This includes the combined use of DSSIM [128] and Mean Square Error reconstruction losses for autoencoder training. DFL offers 5 variants, each differing in terms of input and output face resolution (16x16 to 512x512), additional intermediate layers, architectural combinations of auto-encoder networks, and VRAM requirements.

(3) **Zao**: Zao is an iOS mobile application created in 2018 by the Chinese software company, Momo. Zao is closed source, and its methodology is unknown.

(4) **FakeApp**: FakeApp, created in 2018, is owned and maintained by the online user DeepFakeapp. FakeApp’s methodology is unknown, but is rumored to be based on the architecture of FaceSwap [129].

In our synthetic video dataset introduced in Section 3.3.1, we find videos produced using the four methods described so far (i.e., FaceSwap, DFL, Zao, and FakeApp). However, there are many videos in our dataset for which the generation method is unknown. For completeness, we briefly discuss some other methods developed by the Internet community.

(5) **Other methods**: A notable method is FaceSwap-GAN [81], created in 2018 as an open-source fork of FaceSwap. FaceSwap-GAN employs the same base DNN method as FaceSwap.

---

3https://github.com/iperov/DeepFaceLab
4https://www.zaoapp.net/
5FakeApp is shut down, no link available
3.3 Synthetic Video Datasets

### Table 3.1: Statistics for the DF-W Dataset

<table>
<thead>
<tr>
<th>Datasets</th>
<th>#Videos</th>
<th>Total Frames</th>
<th>Total Duration</th>
<th>Avg. Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>DF-W YouTube</td>
<td>1,062</td>
<td>2.9M</td>
<td>30h 1m 12s</td>
<td>1m 42s</td>
</tr>
<tr>
<td>DF-W Bilibili</td>
<td>807</td>
<td>1.9M</td>
<td>18h 48m 48s</td>
<td>1m 24s</td>
</tr>
<tr>
<td>DF-W</td>
<td>1,869</td>
<td>4.8M</td>
<td>48h 50m 00s</td>
<td>1m 34s</td>
</tr>
</tbody>
</table>

with some improvements. The research community has used FaceSwap-GAN to produce the DeepFakeTIMIT [117] dataset. Other methods include Dfaker [82] and MyFakeApp [83], which also employ the same DNN architecture as FaceSwap. DeepFakesapp.online [84], DeepFakes web β [85], and DeepFake.me [86] are websites that offer synthetic video generation services behind paywalls (methodology is unknown). Reflect [130] and Doublicat [131] are mobile applications with similarly unknown methodologies.

**Methods by the research community.** Many synthetic video generation methods have also been proposed by the research community. It is unclear if synthetic videos found in the wild have directly used these methods. These methods are based on GANs, VAEs, CNNs, and encoder-decoder frameworks. This again shows that advances in generative modeling are enabling the creation of synthetic videos. Methods include FSGAN [67], IPGAN [68], Fast Face Swap CNN [77], FaceShifter [78], FSNet [79], RSGAN [69], and DF-VAE [80].

### 3.3 Synthetic Video Datasets

#### 3.3.1 Collecting DF-W: Our New Synthetic Video Dataset

In this section, we introduce our novel synthetic video dataset called DF-W, containing synthetic videos found in the wild (i.e., produced by the Internet community). We also present details of existing synthetic video datasets produced by the academic/industry research community.
We start by describing the data collection methodology used to build the DF-W dataset. We only focus on content that is “safe for work”, e.g., excluding pornographic, obscene or explicit material.

**Step 1: Searching and identifying potential synthetic videos.** To identify synthetic videos, we start our search from popular Internet platforms known to host or curate links to synthetic video content. Our data sources are listed below. We use these different sources to build a list of potential synthetic videos.

*YouTube.* YouTube is known to host synthetic videos [132]. Our idea is to identify synthetic video creators or channels primarily uploading synthetic videos. To identify such channels, we use the YouTube search feature, using keywords such as ‘deepfake’ and ‘faceswap’, amongst many others. We add the YouTube channel URLs revealed by the first 10 pages of search results to our list. Beyond 10 pages we found content to be less relevant. To further expand the channel list, for each channel, we also add the related channels recommended by YouTube to our list. We also used Google Trends[7] to identify more synthetic video related search queries, with the goal of finding more synthetic video channels on YouTube. Starting from the initial set of 9 keywords, we discovered an additional 69 keywords. However, the new keywords did not yield any additional channels on YouTube.

*Reddit.* Reddit.com hosts two popular synthetic video discussion sub-forums, ‘GIF Fakes’[8] and ‘SFWDeepFakes’. In these sub-forums, we found posts that linked to videos or channels, either created by the authors themselves, or by another content creator. We scraped all posts for URLs, and obtained 1,491 URLs. Out of the 1,491 URLs, 1,341 URLs pointed to YouTube videos and channels, and we follow the procedure described earlier in the YouTube section.

---


to add relevant YouTube channel URLs to our list. The remaining 150 non-YouTube URLs led to platforms such as Reddit’s own video-hosting service, Vimeo, and Imgur, all of which turned out to be duplicates of content we already identified on YouTube. We identified duplicates via manual examination of the content at these URLs. No non-video URLs were found amongst the 150 non-YouTube URLs.

*Bilibili* [120]. Bilibili is a popular video sharing site in China, and is known to host videos from the Zao app [133]. Zao is a free synthetic face-swapping app that can place your face into scenes from hundreds of movies and TV shows after uploading just a single photograph, and has gone viral in China [134]. We again use the search feature using the keyword ‘zao’. Other keywords, *i.e.*, those used on the YouTube search, primarily revealed videos already available on YouTube, as well as non-relevant instructional and reaction videos. We observed that nearly all uploading channels associated with videos in the search results, uploaded sparse amounts of synthetic videos. Consequently, we iterated through every page of search results, and added the video URLs themselves to our list (in lieu of the channel URLs). We further manually clicked on these videos, to find related videos, but did not obtain any videos not already returned in the original search results.

Overall, in step 1, we shortlisted 194 YouTube channel URLs, likely hosting synthetic videos, and 1000 video URLs on Bilibili, likely containing synthetic video content.

**Step 2: Filtering and downloading videos.** In this step, we first verify whether a video contains synthetic content, before downloading it. For verification purposes, we use a combination of manual and automated techniques to filter out non-synthetic videos. First, we used the YouTube API and the HTML source for Bilibili (which lacks an accessible API) after all JavaScript was loaded, to access metadata for each video. We then filtered out videos where the title or description contained no variations of the search keywords originally used in Step 1. Each remaining video (from both YouTube and Bilibili) was then manually verified
as to whether it contained face-swapped content. This was done by looking for statements in the title, description or comments claiming the video to be synthetic, and by also looking for facial flickering, inhuman feature distortion, warping and obvious lighting mismatches.\footnote{We did find cases where it was hard to make a decision just based on content (e.g., for high-quality synthetic videos with no visible artifacts). In such cases, we relied on statements in the title, and description.}

Lastly, we downloaded all the videos (verified to be synthetic videos) at the highest available resolution, using the open-source tool youtube-dl\footnote{https://github.com/ytdl-org/youtube-dl}. Videos that were not made available on YouTube to the author’s home region were not downloaded. All our measurement efforts were conducted between 2020-02-01 and 2020-03-01.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Release Date</th>
<th>#Real Videos</th>
<th>Real Source</th>
<th>#Fake Videos</th>
<th>Generation Method</th>
<th>Avg. Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>UADFV [111]</td>
<td>2018.11</td>
<td>49</td>
<td>YouTube</td>
<td>49</td>
<td>FakeApp</td>
<td>11.55s</td>
</tr>
<tr>
<td>FaceForensics++ [44]</td>
<td>2019.01</td>
<td>1,000</td>
<td>YouTube</td>
<td>1,000</td>
<td>FaceSwap</td>
<td>18.72s</td>
</tr>
<tr>
<td>Celeb-DF [118]</td>
<td>2019.11</td>
<td>590</td>
<td>YouTube</td>
<td>5,639</td>
<td>Unknown method</td>
<td>12.51s</td>
</tr>
<tr>
<td>DFDC [119]</td>
<td>2019.12</td>
<td>23,654</td>
<td>Self-recording</td>
<td>104,500</td>
<td>8 generation methods \footnote{8 generation methods}</td>
<td>10.02s</td>
</tr>
</tbody>
</table>

Table 3.2: Statistics of synthetic video datasets produced by the research community.

**DF-W dataset.** Using the above methodology, we prepared the DF-W dataset containing a total of 1,869 synthetic videos, of which 1,062 are from YouTube, and 807 are from Bilibili. Dataset statistics are shown in Table 3.1. In total, our dataset contains over 48 hours of video content, with over 4.8 million frames (extracted at the native frame rate). Video resolutions range from (360 x 360) to a high resolution of (2560 x 1080). Additionally, we collected the following metadata for each video using the YouTube API and Bilibili HTML source: publishing date, number of views, number of subscribers for the channel (content creator), and the total number of videos associated with the video’s channel.

We note that concurrent work by Zi et al. [136] also introduces a real-world synthetic video dataset, comprising face sequences extracted from 707 synthetic videos collected from the internet. To the best of our knowledge, DF-W is the largest collection of synthetic videos in...
3.3. SYNTHETIC VIDEO DATASETS

3.3.2 Research Community Datasets

Researchers have released a variety of datasets to study synthetic video detection. We use these datasets in our study, and compare them with our DF-W dataset in Section 3.3.1. Dataset statistics are in Table 3.2. Each dataset varies across different dimensions, e.g., the number of videos, generation methods used, how real videos are obtained, video content, and quality. Video resolutions range widely from 234x268 to 1920x1080. Notably, in datasets released by researchers, e.g., UADFV, FaceForensics++ and Celeb-DF, real videos are collected from YouTube or an off-the-shelf video set, based on which they generate synthetic videos. While in datasets released by industry, e.g., DFD and DFDC, real videos are recorded by paid actors. Most recently, Jiang et al. proposed the DeeperForensics dataset in 2020 [80]. However, due to the timing of the release, we are unable to include it in our analysis. According to prior work [118], UADFV, DeepFakeTIMIT, and FaceForensics++ are considered to be the first generation datasets. These datasets have lower-quality synthesized faces, with some visible artifacts from the generation process, and lower pose variations. The second generation datasets include DFD, Celeb-DF, and DFDC, and they improve on many of the limitations of the first generation datasets.

DF-R dataset. We additionally prepare a dataset representing all the research community datasets, by sampling synthetic videos from the 6 datasets, and call it the DF-R dataset. This simplifies analysis, enabling us to compare detection performance on the DF-W dataset, with a single synthetic video dataset representing the research community. We strive to keep the number of synthetic videos similar to that in the DF-W dataset, and therefore sample 500 random videos (or less if fewer are available) from the synthetic and real classes of each research community dataset. The DF-R dataset contains a total of 2,369 synthetic videos,
and 2,316 real videos.

3.4 Analyzing DF-W

Here our goal is to provide a deeper understanding of the DF-W dataset, and how it compares with existing datasets from the research community (DF-R). We also discuss the characteristics of the DF-W dataset that raises implications for a defender.

![Graphs showing distribution of generation tools, CDF of video upload dates, and number of views per video](image)

Figure 3.2: (a) Distribution of generation tools used for creating DF-W YouTube (b) CDF of DF-W video upload dates (c) distribution of generation tools used for creating DF-W YouTube (d) CDF of number of DF-W videos contributed by channels.

3.4.1 DF-W vs Research Community Datasets

**Synthetic video generation methods.** We examine the synthetic video generation methods used by the different datasets. To determine the generation methods for DF-W videos, we manually examine the title and description of each video for any mention of a particular tool or method. The description often contains a link to the tool. For instance, the description of the ‘David Bowie - I Believe [DeepFake]’ video states: ‘Software used: [https://github.com/iperov/DeepFaceLab/](https://github.com/iperov/DeepFaceLab/).’ If the video meta-data did not state any method, we mark the method as *unknown*. Using this method, we find that all the videos from Bilibili are produced using the Zao iOS application. For YouTube, we were able to obtain the generation method label for 241 out of 1,062 videos (22.71%), while the remaining are marked as *unknown*. Figure 3.2a shows the distribution of the three generation methods
that we found on YouTube: FakeApp, FaceSwap, and DFL. DFL makes up the vast majority (94.2%) of videos for which we have a known method.

Among the research datasets, generation methods are unspecified in Celeb-DF and DFD. FaceSwap, FakeApp, and FaceSwap-GAN were used to generate FaceForensics++, UADFV, and DeepFakeTIMIT, respectively. DFDC was generated using 8 custom autoencoder-based, GAN-based, and non-neural methods. Interestingly, DFL is not used by any of the existing research community datasets, but is a popular generation method in the wild. *This indicates a mismatch in the methods used by the research community and the Internet community.* Using methods from the wild would help to build a more representative synthetic video dataset.

*Can we automatically infer the generation method for a video?* For a large fraction (77%) of DF-W YouTube videos, the generation method is unknown. A defender with knowledge of the generation methods can create effective targeted defenses. Therefore, we propose to infer the generation scheme for the videos (with unknown methods) using a DNN-based method. We focus on the generation method, DFL, mainly for two reasons—it is the only method where we have sufficient data to conduct our analysis, and DFL, themselves claim to be the most popular method (Section 3.2.2). Our scheme to fingerprint the generation method aims to determine whether a video is created using DFL or a non-DFL method (*i.e.*, some other method). We first train this classifier using a labeled dataset of DFL and non-DFL videos, and evaluate its performance, before applying it to the videos with unknown generation schemes.

To build the fingerprinting scheme, we leverage the DNN model from Yu et al. [137], which was proposed as a method to fingerprint the GAN model, given a GAN generated image. In our setting, the input to the classifier is a face extracted from a frame, which is then classified as belonging to the DFL or the non-DFL class. We randomly sample 200 DFL
CHAPTER 3. SYNTHETIC VIDEO DETECTION: INVESTIGATE REAL-WORLD EFFECTIVENESS OF EXISTING DEFENSES

videos (from DF-W), and 200 non-DFL videos. Non-DFL videos are sampled from DF-W and research community datasets, and covers 4 generation methods, i.e., FakeApp, FaceSwap-GAN, FaceSwap, and Zao. Next, to identify and extract fake faces from each video, we use the synthetic video detection scheme called CapsuleForensics [45]. CapsuleForensics provides a probability score for a face being fake, and we only choose faces with a probability score higher than 0.7. We randomly sample 20k, 4k, and 4k faces (with balanced ratio of fake and real faces) using this criteria for training, validation, and testing, respectively. We use hyperparameters recommended by Yu et al., and also apply dropout (prob=0.5) to the last two dense layers. On tuning the decision threshold for high precision, our final model achieves a 90% precision and 70% recall on the testing set for the DFL class. We then run this trained model on 821 source-unknown videos from DF-W YouTube. A video is flagged as DFL-generated if more than 50% faces of the video are flagged as DFL-generated. Finally, 243 videos (29.6%) are classified as DFL-generated out of the 821 unknown videos in DF-W YouTube. Our analysis suggests that DFL is indeed a popular tool in the wild, and should be considered by the academic community to create synthetic video datasets.

Content analysis. We describe key differences between the synthetic videos in the DF-W dataset, and the research community datasets. First, for all research community datasets, every frame contains some fake content (i.e., a swapped face). In fact, many existing synthetic video detection schemes (e.g., CapsuleForensics [45], Xception [44], FWA [124], MesoNet [112], VA, [46], and Multi-Task [114]) are evaluated under the assumption that every frame — with a detectable face — in a synthetic video contains fake content. However, this is not the case for videos in the DF-W dataset. We find several cases where only a fraction of the frames contains fake content, including cases where majority of frames are

12There are 8 synthetic video generation methods used in the DFDC dataset. Examples include DF-128, DF-256, MM/NN, NTH, FSGAN, and StyleGAN [119].
13In DF-W videos, not all faces/frames contain fake content.
This has implications for the design of detection schemes (Section 3.5.1), because a large number of clean frames (i.e., without fake content) can increase the risk of false negatives.

Second, we examine the number of faces in a frame, and the number of fake faces in a frame. Some existing detection schemes (e.g., CapsuleForensics, Xception, Multi-Task [114]) assume that each frame only contains a single face which is faked. Such methods would need to be adapted and calibrated to handle cases where there are multiple faces in a frame, of which all or only a portion of faces are fake. From manual inspection, we observe three variations of synthetic videos in the DF-W dataset: (1) single face that is fake, (2) multiple faces with a single fake face, and 3) multiple faces and multiple fake faces. In contrast, some research community datasets, UADFV and DeepFakeTIMIT fall entirely in the first category. To further quantify this, we use a CNN-based face detection model called dlib [138] to identify the number of faces in a video. We approximately estimate the number of faces in a video, as the maximum number of faces detected in any frame in the video. We find that nearly 26% of DF-W videos contain more than a single face, whereas, for all 6 research community datasets (in Section 3.3.2) 92% to 100% of videos contain only a single face.

Third, we observe that DF-W videos are longer in duration than research datasets. In particular, the longest duration among research datasets is less than 100 seconds; however, nearly 32% of videos in DF-W have a duration longer than 100 seconds. In UADFV and DFDC, all the videos are shorter than 10s, and in CelebDF all are shorter than 20s. In comparison, the videos in DF-W range from 10s to 500s. Longer duration, combined with variations of synthetic video content (described earlier), can lead to more false negatives while detecting synthetic videos, e.g., video has large number of clean frames.

\[14\] It is hard to quantify using exact numbers — requires manual examination of millions of frames.


3.4.2 Growth, Popularity, and Creators of DF-W Videos

**Content growth.** Figure 3.2b shows the distribution of the upload dates of DF-W videos. Bilibili has a sharp increase in uploads in September 2019, when the Zao app went viral in China. For YouTube, the number of uploaded synthetic videos saw a slight bump in January 2018 with the original release of FakeApp, but the largest jump happened at the beginning of 2019 when DFL and FaceSwap became popular open source alternatives to FakeApp. *The steady recent growth of synthetic video content suggests the urgent need to build effective defenses against misuse of synthetic videos.*

**Popularity.** In both YouTube, and Bilibili, there are videos that are hugely popular. There are 31, and 2 videos in YouTube, and Bilibili, respectively, with over 500,000 views. There are more popular videos on YouTube than Bilibili. Nearly 69%, and 27% of videos in YouTube, and Bilibili, respectively, received over 1K views. Distribution of the number of views per video is shown in Figure 3.2c. Moreover, when we investigate the ratio of the number of views to the number of subscribers of the corresponding channel, we find that 60% and 89% of videos on YouTube and Bilibili, respectively, have a ratio higher than 1. This implies that synthetic video content easily and often spreads beyond the audience of its channel. Our analysis shows that synthetic content can reach a large audience, thus benefiting an attacker creating misleading synthetic videos.

**Content creators.** Here we study the channels or content creators that upload the synthetic videos. There are 84 channels for YouTube, and 477 channels for Bilibili. Figure 3.2d shows the ranked plot of channels, and the cumulative fraction of videos associated with them. For YouTube, the top 16 channels (covering 20% of channels) account for 75% of the videos. This suggests that there are a small number of creators/channels on YouTube who have the resources to create a large number of synthetic videos. However, we observe a long tail
distribution for Bilibili, where the top 95 channels (covering 20% of channels) only account for 49% of the videos. In Bilibili, over 90% of channels only upload one or two synthetic videos. This suggests that defense efforts need to be prepared to tackle both resourceful synthetic video creators who can churn out large number of synthetic videos, as well as a large number of creators/users who may each produce only a few synthetic videos. The generation methods used by these creators could be different, thus potentially making defense efforts harder.

3.5 Evaluating synthetic video detection

We evaluate the performance of existing detection methods on DF-W and DF-R datasets, and also evaluate approaches for improving classification performance.

3.5.1 Experimental Setup

Detection methodology and performance metrics. Most prior work has framed the problem as a frame-level binary classification task, where the probability score of a frame being fake is computed using the extracted face as input. Using this formulation, classifiers have portrayed impressive accuracy, and ROC AUC performance scores at the frame level [44, 45, 46, 118, 124]. Other prior work used a video-level decision metric by assigning the video probability score (of being fake) as the average of all frame probability scores [45, 112].

The above metrics work well for existing research community datasets for the following reasons: (1) Existing datasets (see Section 3.3.2) contain synthetic videos that are entirely fake, i.e., every frame with a face has been manipulated. (2) In existing datasets, most (92%–100%) synthetic videos contain at most one face per frame (see Section 3.4.1). However, these assumptions do not hold for the DF-W videos. In Section 3.4.1, we discuss that DF-W
videos can contain multiple faces per frame, not all of which are fake. Also, not all frames necessarily contain a fake face.

Therefore, we need to modify the decision metric to incorporate scores for multiple faces. Further, since we lack frame level ground-truth, our evaluation metric needs to be at the video level. However, averaging frame scores to obtain the video score, as used by prior work is problematic since a large portion of the video can contain real, unmodified frames.

To tackle the above challenges, we propose a modified version of a video-level decision heuristic, originally introduced by Li et al \[124\]. Li et al. propose to compute the probability score for a given video being fake by averaging the top $F_p$ percentile of frame probability scores:

$$
P(v) = \frac{1}{n} \sum_{i=F_p\%}^{100\%} P(x_i),
$$

where the percentile value $F_p$ is a decision parameter. While Li et al. assume one face per frame, we compute the frame-level probability score in the above equation, as follows: $P(x_i) = \max(P_{f1}, P_{f2}, P_{f3}, \ldots)$ where $x_i$ refers to the $i^{th}$ frame, and $P_{fj}$ refers to the probability score for the $j^{th}$ face in the frame.

Li et al. used an $F_p$ value of 33%. To provide better insight into the current performance limits for synthetic video detection, we adopt the F1-score metric, and sweep the parameter space for the optimal $F_p$. More specifically, for each given classifier and testing set configuration, we compute detection performance using the value of $F_p$ that allows for the best-attainable F1 score. Computing the best-attainable F1 score, helps us understand the upper bound on performance.

**Supervised methods.** Supervised synthetic video detection methods refer to those that use a labeled dataset of synthetic and real videos to train a classifier. To evaluate the current state-of-the-art, we evaluate those supervised methods that have been proposed in peer-reviewed work, have released code, and pre-trained models.

1. **CapsuleForensics** [45]: Nguyen et al. proposed CapsuleForensics which employs a
3.5. Evaluating synthetic video detection

VGG network [139], and a Capsule network [140]. CapsuleForensics is trained and tested on the FaceForensics++ dataset. More specifically, the training data comprises faces extracted from frames of 2,160 synthetic videos generated with FaceSwap, from 5,320 non-neurally manipulated videos, and from 2,160 real YouTube videos. Their paper reports detection performance at the frame-level with a 93% accuracy, and 92% accuracy at the video-level (using overall frame-probability averaging).

(2) Xception [44]: Rossler et al. proposed the Xception method, a CNN-based model that is pre-trained on Imagenet, and trained and tested on an updated version of the FaceForensics++ dataset. In addition to all the videos in the version used by CapsuleForensics, this updated dataset also comprises faces extracted from frames of 2,160 manipulated videos generated using the neural expression-manipulation tool, NeuralTextures [141]. They report a 99% frame level accuracy.

(3) MesoNet [112]: Afchar et al. proposed MesoNet, a CNN-based model with two configurations, Meso4 and MesoInception4. We choose Meso4, as both models exhibit similar performance. Meso4 is trained and tested on a dataset containing a small number of synthetic videos collected from YouTube, and real images collected from the Internet. More specifically, the training data comprises faces extracted from frames of 175 synthetic videos, and 7,250 real faces collected from “various Internet sources”, chosen such that their resolution distribution matches that of faces from the synthetic videos. They report a frame level accuracy of 89%, and a video level accuracy (using overall frame-probability averaging) of 97%.

(4) Multi-Task [114]: Nguyen et al. proposed Multi-Task, based on a multi-output autoencoder. Multi-Task is trained and tested on a subset of the FaceForensics++ dataset. The training data includes faces extracted from frames of 704 non-neurally manipulated videos, and 704 real YouTube videos. Multi-Task is designed for generic facial-manipulation detection, for which they report frame-level testing accuracies of upto 92%.
CHAPTER 3. SYNTHETIC VIDEO DETECTION: INVESTIGATE REAL-WORLD EFFECTIVENESS OF EXISTING DEFENSES

<table>
<thead>
<tr>
<th>Class</th>
<th>DF-W</th>
<th>DF-W YouTube</th>
<th>DF-W Bilibili</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fake</td>
<td>1,864</td>
<td>1,057 15</td>
<td>807</td>
</tr>
<tr>
<td>Real</td>
<td>1,864</td>
<td>1,057</td>
<td>806</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>DF-R</th>
<th>FFPP</th>
<th>DF-R Celeb-DF</th>
<th>DF-R DFDC</th>
<th>DF-R DFD</th>
<th>DF-R Dtimit</th>
<th>DF-R UADFV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fake</td>
<td>2,369</td>
<td>500</td>
<td>500</td>
<td>500</td>
<td>500</td>
<td>320</td>
<td>49</td>
</tr>
<tr>
<td>Real</td>
<td>2,316</td>
<td>501</td>
<td>501</td>
<td>501</td>
<td>501</td>
<td>320</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 3.3: Testing set sizes for evaluating detection performance on DF-W and DF-R datasets.

(5) VA [46]: Matern et al. proposed VA, a method using either a multi-layer perceptron (VA-MLP), or a logistic regression classifier (VA-LReg). Both models show similar performance, and we choose the VA-MLP classifier. VA-MLP is trained and tested on a dataset containing YouTube videos with explicitly demarcated ‘side-by-side’ synthetic and real content. In such videos, the content creator puts the original and the synthetic versions of the video horizontally next to each other, allowing the viewer to watch the entire video with both views. The dataset also contains normal synthetic videos collected from YouTube. More specifically, the training data comprises faces with eyes and mouth open, extracted from the ‘fake side’ of frames of 4 ‘side-by-side’ synthetic videos and from 3 normal synthetic videos. The real faces are similarly extracted from the real side of the frames of the same 4 ‘side-by-side’ videos. They report 85% frame-level ROC AUC score.

Unsupervised methods. Unsupervised synthetic video detection methods only use real videos/images to train a classifier. We focus on the following two unsupervised methods.

(1) FWA [124]: Li et al. proposed the FWA method, a CNN-based model. FWA is trained on a dataset containing real faces, intentionally warped to simulate synthetic artifacts, and unwarped real faces collected from the Internet. Warping is achieved by rescaling a real face to a random scale, smoothing with a 5x5 Gaussian blur, and then affine warping back to the

15For 5 videos of DF-W YouTube (1,062 videos), no faces were detected by the face detector.
original scale. The training data comprises 24,442 warped real faces, and 24,442 unwarped real faces. They report up to 99.9% frame-level ROC AUC.

(2) DSP-FWA\textsuperscript{16}: Li et al. proposed the DSP-FWA method in 2019. This method employs a CNN that improves upon FWA. DSP-FWA is trained on data identical to that of FWA. No reported performance numbers are available.

For all methods (supervised and unsupervised), we directly use the pre-trained models provided by the model developers.

Our testing datasets. We present detection performance on the synthetic videos in the DF-W dataset as a whole, as well as separately on the individual YouTube and Bilibili partitions. Detection performance on the DF-R dataset is also presented as a whole, as well as separately on each of the 6 individual research datasets (Section 3.3.2). For all datasets, we downsampled the larger class to maintain a balanced ratio of real and synthetic videos. The final testing set sizes for the synthetic and real classes are listed in Table 3.3.

3.5.2 Performance of Existing Detection Schemes

In this section, we evaluate performance of existing detection methods. Our objective is to understand the best-attainable F1 score (in detecting fake videos) for each configuration. This implies that our results are optimistic, \textit{i.e.}, in favor of the tested methods. In practice, the methods will perform sub-optimally if configured with a sub-optimal $F_p$ threshold. Table 3.4 presents our results.

Detection performance on the DF-W dataset. The first row in Table 3.4 presents (best-attainable) detection performance for each classifier on DF-W. The performance of all classifiers is poor — all F1 scores are below 77%, going as low as 66%. Moreover, all precision

\textsuperscript{16}https://github.com/danmohaha/DSP-FWA
scores are below 69%, suggesting many false positives. This indicates that these classifiers fail to generalize to a variety of real-world synthetic videos.

The best-performing supervised classifier is CapsuleForensics, with an F1 score of 77%, followed by MesoNet, with an F1 score of 74%. The remaining supervised classifiers exhibit worse performance, with F1 scores below 70%. Multi-Task performs particularly poorly. We observe it produces the same classification decision of ‘fake’ for nearly all inputs, confirmed by effectively random precision and perfect recall.

The best-performing unsupervised classifier on DF-W is DSP-FWA, with an F1 score of 76%, followed by FWA, with a F1 score of 73%. Performance of unsupervised methods is thus comparable to the supervised methods. This is surprising, as these unsupervised classifiers are not trained on any synthetic data — they are trained on warped images that simulate common synthetic artifacts. We hypothesize that this training strategy might have enabled better detection generalization capabilities.

Detection performance on the DF-R dataset. The fourth row in Table 3.4 presents (best-attainable) detection performance for each classifier on DF-R. The performance of all classifiers is poor, and worse than their performance on DF-W. We can attribute this failure
3.5. Evaluating synthetic video detection

to the same lack of generalization capabilities to which we attributed poor performance on the DF-W dataset.

The best-performing supervised classifier on DF-R is CapsuleForensics, with an F1 score of 66%. Multi-Task exhibits a higher F1 score of 67%, but this is due to the nature of the F1 metric, as MultiTask produces the same classification decision of ‘fake’ for nearly all inputs. All supervised classifiers exhibit poor performances, with F1 scores as low as 55%. The best-performing unsupervised classifier on DF-R is DSP-FWA.

We also present performance on each research dataset that constitutes DF-R, separately. These performances are presented in rows 5–10 of Table 3.4. No classifier performs consistently across each of the 6 individual sets. We note that CapsuleForensics and Xception perform exceptionally well on FFPP, as also reported in their original work.

**Deep diving into the performance of CapsuleForensics and DSP-FWA.** We perform a deeper analysis of the best-performing supervised and unsupervised methods — CapsuleForensics and DSP-FWA. First, we investigate the relationship between detection performance and the synthetic video generation method, *i.e.*, whether these classifiers are biased against certain generation methods. We thus measure their efficacy on individual subsets of DF-W, each corresponding to a different generation method. Table 3.5 shows the percentage of synthetic videos correctly classified (TP) and the percentage of videos wrongly classified as real (FN). We find that CapsuleForensics performs better against videos generated by DFL and Zao, compared to FakeApp and FaceSwap. It is also able to correctly classify 80% of videos generated by ‘unknown’ generation methods. DSP-FWA performs significantly better on all but Zao synthetic videos, with 10% FN rate compared to 4% or lower on all other categories. These results suggest that detection methods need to be tested against a variety of synthetic video generation methods to understand their true performance.
Chapter 3. Synthetic Video Detection: Investigate Real-world Effectiveness of Existing Defenses

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Score</th>
<th>Unknown</th>
<th>DFL</th>
<th>Face Swap</th>
<th>Fake App</th>
<th>Zao</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capsule Forensics</td>
<td>FN</td>
<td>20%</td>
<td>12%</td>
<td>36%</td>
<td>33%</td>
<td>12%</td>
</tr>
<tr>
<td></td>
<td>TP</td>
<td>80%</td>
<td>88%</td>
<td>64%</td>
<td>67%</td>
<td>88%</td>
</tr>
<tr>
<td>DSP-FWA</td>
<td>FN</td>
<td>4%</td>
<td>1%</td>
<td>0%</td>
<td>0%</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>TP</td>
<td>96%</td>
<td>99%</td>
<td>100%</td>
<td>100%</td>
<td>90%</td>
</tr>
</tbody>
</table>

Table 3.5: Distribution of generation methods for videos that were correctly classified (TP) and incorrectly classified (FN) by CapsuleForensics and DSP-FWA.

![F1 Score vs. Frame Percentile Threshold](image1)

Figure 3.3: DF-W F1 scores vs. $F_p$ for (a) CapsuleForensics (b) DSP-FWA.

Next, we consider the role played by the choice of the $F_p$ decision threshold parameter. As discussed in Section 3.5.1, this $F_p$ parameter is used for video-level decisions. Figures 3.3a and 3.3b shows the effect of the decision threshold $F_p$ on the F1 scores for CapsuleForensics and DSP-FWA, respectively, on the DF-W dataset. These sweeps reveal that for most cases, $F_p$ values roughly $< 0.2$ are generally optimal.

Analyzing detection failures. DNN-based facial recognition models are known to suffer from racial bias [121]. Given that synthetic video detection schemes are also based on DNN models that analyze faces, we investigate for any issues of racial bias. We focus on the best overall classifier, CapsuleForensics.

Racial bias. We evaluate whether CapsuleForensics performs poorly for specific races. We
prepare a test set of face images with race labels using the DF-W YouTube dataset, and containing a balanced number of fake and real images for each race. To construct this test set, we extract the 10 most confidently predicted faces in each video, and assign it a race using DeepFace [142] — a facial analysis DNN that can predict the race for a given face image. We further discard faces for which DeepFace itself is below 95% in prediction confidence. This heuristic enables DeepFace to exhibit high precision (over 90%), as verified on each race of a labelled racial classification dataset 17. The resulting dataset comprises faces belonging to 3 races — Caucasian, Black, and Asian. Finally, balancing is achieved by (randomly) downsampling the larger class (fake or real) for each of these 3 races.

The resulting race-wise detection performance of CapsuleForensics indicates a racial bias in its detection capabilities. We observe that the (macro average) F1 score for 464 Asian faces is only 48%. This performance is notably low when compared with performance for the other races: 72% for 800 Caucasian faces, and 74% for 416 Black faces. To explain the cause of such a racial bias, a natural direction is to analyze the race distribution of DF-R FFPP, i.e., the CapsuleForensics training set. However, after adopting a precision optimized configuration for DeepFace, there is an inadequate number of face images remaining in DF-R FFPP to analyze. We thus leave investigation into the cause of this racial bias as future work.

Model interpretation schemes to understand detection decisions. Using CapsuleForensics as a case study, we focus on understanding the predictions made by the classifier using a model interpretation scheme. Besides the more immediate benefit of improving detection schemes, understanding relevant features in an image that led to a decision is also important in an adversarial context. Specifically, an adaptive adversary with knowledge of the relevant features for classification decisions can evade detection by eliminating (or

17We use the FairFace dataset from Karkainnen et al. [143]
spoofing) those relevant, critical features.

To this end, we use Integrated Gradients [1] or IntGrad, a recent and influential feature-attribution based DNN explanation methodology, to explain the outputs of CapsuleForensics. For a given model and an input image, IntGrad produces a saliency map representing an attribution score for every channel of every pixel in the input image. This map “explains” the output of the neural network. However, the maps themselves vary from image to image. To identify specific patterns and trends in these maps, we conduct a systematic, manual annotation process.

Step 1: Sampling representative images for explanation. Representative real and synthetic faces are chosen from DF-W and DF-R respectively, and categorized into 4 classes — TP, TN, FP, FN — based on the output of CapsuleForensics. For each video, one representative face is chosen. Specifically, for TP and FP videos, the face with maximum probability being fake is chosen, and for TN and FN videos, the face with maximum probability being real is chosen. In total, we sample 1,533, 1,100, 798, and 177 images from the TP, TN, FP, and FN classes, respectively.

Step 2: Using IntGrad to obtain saliency masks. For each face image, we obtain feature attribution scores from IntGrad, considering the top 10% pixels (according to IntGrad scores) to be the most important pixels, and produce saliency maps.

Step 3: Manually annotating the saliency maps. To identify pattern trends in the saliency maps, we conduct a manual annotation process. Categories are chosen to represent different parts of the face image being focused by the saliency masks. The initial categories chosen for this purpose include: eyes, nose, mouth, face-boundary, background, and no-discernible-features. The no-discernible-features category is special in that it is used to describe saliency masks in which no specific parts of the face were highlighted. To narrow down the set
of categories, 30 face images from each of the four classes (TP, FP, TN, FN) are picked and annotated using the above categories. A face image could be annotated with multiple categories. Each image is annotated independently by 3 graduate students in our lab. An image is finally annotated with a category, if at least 2 of the 3 annotators use the given category to annotate the saliency mask. After this step, a difference in the total number of annotations between the TP, FP, TN, FN classes was observed for only 3 categories: no-discernible-features, background, and face-boundary. Figure 3.4 shows representative examples of faces annotated with the above 3 labels. Next, to understand if the differences were statistically significant, we randomly sample 100 images per class and annotate them with the 3 categories.

Statistical analysis of saliency mask annotations and implications. Based on the results of the annotation process, we find that compared to FN and TN, TP and FP classes have significantly higher number of faces annotated with no-discernible-features \((p=0.031, \text{odds ratio}=2.579)\)\(^{19}\), and significantly lower number of faces annotated with background \((p=2.89 \times 10^{-11}, \text{Odds Ratio}=0.216)\). This suggests that if an adversary manages to introduce relevant background features just around a synthetic face, CapsuleForensics would be \(\approx 4.6\) times as likely to label the image as real. From the opposite perspective, it is to the benefit of the defender to avoid focusing outside of the facial boundary, and ensure that the feature extraction module produces relevant features. From this defender’s perspective, it is also important to note that when CapsuleForensics is unable to focus on important features (no-discernable-feature case) it is \(\approx 2.5\) times more likely to label the image as fake, independent of its true source. Ensuring the model is able to focus on proper features might be achieved by pre-emptively validating/improving the feature extraction scheme. Finally, compared to

\(^{18}\)Note that while other feature categories (eyes, mouth, etc) might be visible in these examples, the above process has eliminated them as relevant to the classifier’s decision.

\(^{19}\)All comparisons are done using Fisher’s Exact test. The p-values are reported after Bonferroni correction.
Figure 3.4: Visualization of salient regions (white dots) as determined by CapsuleForensics, and highlighted using IntGrad [1]. The first image is a fake face correctly classified as fake, and shows no-discernible-features. The second image is a real face correctly classified as real, and shows face boundary features. The third image is a fake face classified as real, and shows salient background features. Note that these are significant features, and do not preclude insignificant features such as those around the eyes and nose.

the other three classes, TN class has significantly higher number of faces annotated with face-boundary (p=0.002, odds ratio=2.627). This seems to further support the idea that being able to correctly identity facial features and face boundary is critical for the classifier in being able to discern real faces from the fakes.

3.5.3 Towards Improving Detection Performance

In light of the poor generalization capabilities of existing detection schemes, we consider multiple approaches for improving performance, and evaluate their efficacy. Table 3.6 presents the results of this evaluation.

Improving detection results via transfer learning. So far, we directly used the pre-trained model provided by the authors of each detection scheme. This helps to understand the practical applicability of these pre-trained models, which unfortunately indicated poor performance out-of-the-box. To improve performance, we investigate the use of transfer learning. Starting from the pre-trained weights, we retrain a given model using a limited amount of additional new data.

Specifically, we study two retraining strategies — (1) Domain adaptation (DA) retraining
3.5. Evaluating synthetic video detection

Table 3.6: Best attainable detection performances (F1) of Capsule-DA, Capsule-SE, MesoNet-DA, MesoNet-SE and DFDC winner Seferbekov’s model on the DF-W and DF-R Datasets, with corresponding precision (P) and recall (R) scores. Here ‘DA’ refers to models after domain adaptation retraining, and ‘SE’ refers to models after source expansion retraining. Best attainable detection performance (F1) for each dataset is underlined.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Capsule - DA</th>
<th>Capsule - SE</th>
<th>MesoNet - DA</th>
<th>MesoNet - SE</th>
<th>Seferbekov</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1</td>
<td>P</td>
<td>R</td>
<td>F1</td>
<td>P</td>
</tr>
<tr>
<td>DF-W</td>
<td>91</td>
<td>87</td>
<td>94</td>
<td>71</td>
<td>60</td>
</tr>
<tr>
<td>DF-W YouTube</td>
<td>90</td>
<td>88</td>
<td>92</td>
<td>72</td>
<td>64</td>
</tr>
<tr>
<td>DF-W Bilibili</td>
<td>94</td>
<td>89</td>
<td>98</td>
<td>71</td>
<td>59</td>
</tr>
<tr>
<td>DF-R</td>
<td>65</td>
<td>58</td>
<td>74</td>
<td>93</td>
<td>91</td>
</tr>
<tr>
<td>DF-R FFPP</td>
<td>91</td>
<td>94</td>
<td>88</td>
<td>95</td>
<td>93</td>
</tr>
<tr>
<td>DF-R Celeb-DF</td>
<td>75</td>
<td>67</td>
<td>84</td>
<td>91</td>
<td>90</td>
</tr>
<tr>
<td>DF-R DFDC</td>
<td>48</td>
<td>46</td>
<td>50</td>
<td>93</td>
<td>90</td>
</tr>
<tr>
<td>DF-R DFDC</td>
<td>81</td>
<td>79</td>
<td>83</td>
<td>95</td>
<td>92</td>
</tr>
<tr>
<td>DF-R DTimit</td>
<td>14</td>
<td>17</td>
<td>12</td>
<td>92</td>
<td>94</td>
</tr>
<tr>
<td>DF-R UADFV</td>
<td>90</td>
<td>95</td>
<td>86</td>
<td>96</td>
<td>93</td>
</tr>
</tbody>
</table>

(2) Source expansion (SE) retraining. For DA retraining, we assume access to a small set of videos from DF-W for transfer learning, thus building the potential capability to adapt to the new domain of in-the-wild synthetic videos [144]. We only assume access to a limited number of synthetic videos (around 50), because in practice, it will be hard to obtain access to a large number of synthetic videos created for misuse. In SA retraining, we include new videos from multiple academic synthetic video datasets to expand the training data distribution. Overall, our goal is to increase the generalizability of the detection model by learning a new or a boarder data distribution provided by the retraining dataset.

Retraining configuration. We choose the two best supervised detection methods, i.e., CapsuleForensics and MesoNet for our analysis. Retraining the models involves initialization with the pre-trained weights, followed by fine-tuning of all model layers until convergence. For domain adaptation, the retraining dataset contains 50 synthetic videos randomly sampled from DF-W (40 YouTube and 10 Bilibili), and 120 real videos evenly and randomly sampled from each of the 6 academic research subsets in DF-R. To extract fake faces (since there are unaltered faces in DF-W videos), we manually scan through videos to collect
clips with only synthetic faces that add up to 10 seconds from each synthetic video. From collected clips of each video, 100 faces are randomly sampled. To extract real faces, we randomly sample 50 faces from each real video. Finally, we obtain a retraining dataset with 6,000 real faces and 5,000 fake faces. For source expansion, the retraining dataset contains 240 synthetic video and 240 real videos evenly and randomly sampled from each of the 6 academic research subsets in DF-R. We randomly sample 40 faces from each video, and obtain a retraining dataset with 9,600 real and 9,600 fake faces.

**Results.** Detection performances post-retraining are presented in Table 3.6. We make the following key observations: *First*, CapsuleForensics-DA (domain adaptation) improves performance on DF-W to 91% F1 score (from 77% F1). *This indicates that domain adaptation using transfer learning is a promising approach towards improving performance, while requiring only a small number of synthetic videos from the target domain.* However, we observe some fluctuation in performance for CapsuleForensics-DA when applied to the DF-R datasets. For 3 out of the 6 DF-R datasets, there is a drop in performance. To limit such performance degradation, we tried reducing the number of layers fine-tuned during transfer learning. We observe that performance degradation on DF-R datasets can be limited, if we only fine-tune the top 30 layers (out of 160 layers), instead of fine-tuning all layers. For example, performance on DF-R DTimit bounces back from 14% to 52%. However, there is a small performance hit on DF-W on selectively fine-tuning layers—F1 score on DF-W reduces from 91% to 87%.

*Second,* for DA, CapsuleForensics performs better than MesoNet, and performance of MesoNet drops compared to the setting without domain adaptation. We suspect that this is because MesoNet has less information capacity than CapsuleForensics due to its smaller architecture, and is not well suited for transfer learning in this setting. The Meso4 network only has 27,977 trainable parameters [45], while the CapsuleForensics network has 3,896,638 train-
able parameters [112]. This suggests that not all methods can be improved using transfer learning.

Third, while the domain adaptation strategy works well, the source expansion (SE) strategy does not improve performance on DF-W datasets. Therefore, simply combining existing academic synthetic video datasets cannot capture the distribution of DF-W. However, CapsuleForensics-SE (source expansion) effectively improves performance on DF-R to 93% F1 score (>90% on all subsets), which is expected as we have retrained the model to better adapt to the DF-R datasets.

**How well does the best model from a public deepfake detection competition work?** Another approach to building better detection schemes is to outsource the task to the public, through funded competitions. We examine the practical applicability of a competition winning model, by leveraging Facebook’s recently concluded DFDC competition [122]. The competition concluded in July 2020, with 5 detection schemes selected as prize-winners out of more than 2000 submissions. We choose the best model, henceforth referred to as the Seferbekov model [145] (named after the author), and evaluate it on all datasets. The Seferbekov model was trained on the public training set of DFDC dataset [119] released by Facebook, which includes 100,000 synthetic video clips and 19,154 real clips. This model uses the state-of-the-art EfficientNet B7 [146] (pretrained with ImageNet and noisy student) for feature encoding. Structured parts of faces were dropped during training as a form of augmentation. For our evaluation, we use the pre-trained model provided by the author.\(^{20}\) In the competition, the Seferbekov model makes use of predictions based on 32 frames from each video to make a video-level prediction. However, this strategy can potentially deteriorate the detection performance on DF-W as DF-W videos contain unaltered frames and have longer duration compared to synthetic clips tested in the competition. Thus we adopt

\(^{20}\)https://github.com/selimsef/dfdc_deepfake_challenge/blob/master/download_weights.sh
the strategy proposed in Section 3.5.1 to make the video-level decision.

The evaluation results are presented in Table 3.6. We make the following observations: First, the winning model (out of 2000 submissions) still does not exhibit very high detection performance (e.g., F1 score > 90%) on the DF-W dataset. The Seferbekov model achieves an 81% F1 score, but with a low precision of 71%. This suggests that there is still room for significant improvement, before we are ready for real-world deployment. Second, domain adaptation and source expansion techniques outperform the winning model on DF-W, and DF-R datasets, respectively. Third, the Seferbekov model boasts relatively better performance (F1) on both DF-W and DF-R when compared to the existing classifiers (without DA or SE), shown in Table 3.4. There are three potential reasons for this: (1) the Seferbekov model was trained on the large, state-of-the-art DFDC public training dataset \[119\] comprising 100,000 synthetic videos, whereas CapsuleForensics was trained on the smaller FaceForensics++ dataset comprising only 7,480 synthetic and other types of manipulated videos. (2) According to its official GitHub repository \[145\], the Seferbekov model applied heavy data augmentation on the provided DFDC training set, including out-of-the-box image augmentations in Albumentations\[21\] library and face part cut-out. (3) the Seferbekov model averages predictions from 7 models trained with different seeds, whereas CapsuleForensics uses a single model.

3.6 Conclusion

In this chapter, we presented a measurement and analysis study of synthetic videos found in the wild. We collected and curated a novel dataset, DF-W, comprising 1,869 synthetic videos — the largest dataset of real-world synthetic videos at the time of our study. Our

\[21\]https://github.com/albumentations-team/albumentations
3.6. Conclusion

analysis revealed that DF-W videos differ from the synthetic videos in existing research community datasets in terms of content, and generation methods used, raising new challenges for detection of synthetic videos in the wild. We further systematically evaluated multiple state-of-the-art synthetic video detection schemes on DF-W, revealing poor detection performance. This suggests a distributional difference between in-the-wild synthetic videos, and videos in research community datasets. We also attributed detection failures to be related to racial biases, and using model interpretation schemes, we investigated features that can be leveraged to either improve or evade detection. Finally, we show that domain adaptation via transfer learning, which retraining the model on a small set (50 videos) of DF-W videos, is a promising approach to improving performance on DF-W. Overall, our findings indicate a need for incorporating in-the-wild synthetic videos into future work.
Chapter 4

Synthetic Text Detection: Investigate Real-world Effectiveness of Existing Defenses

4.1 Introduction

Apart from advances in synthetic image generation, progress in natural language generation (NLG) has enabled deep learning models such as GPT-2 [66] and GPT-3 [20] to generate synthetic text or deepfake text with high linguistic quality. Both models fall in the Transformer [53] family of language models (LMs). These large LMs with billions of parameters, can generate synthetic text on diverse topics. They have many applications, including, generating content for entertainment purposes (e.g., stories, jokes) [100], dialog systems [101], text summarization [102], and automated journalism [103].

Unfortunately, such technology raises serious security concerns—synthetic text can be misused to power several threats aimed at misleading content consumers. Zellers et al. demonstrated that GPT-2-based LMs can generate convincing fake news articles [18]. Such tools can enable large-scale disinformation campaigns. Yao et al. [147] showed that synthetic text can be used to create fake online restaurant reviews. Such threats will lead to users losing
trust in online content, including crowd-sourced information. Other serious threats include automated email generation for targeted attacks [148], and synthetic text that can radicalize individuals into having violent, extremist ideologies [149].

One approach to curb the misuse of synthetic text is to have humans evaluate and flag the text. However, such an approach is unlikely to scale, and more importantly, existing works already demonstrate that humans are unable to reliably distinguish between real and synthetic text. Recent work showed that the GPT-3 [20] model can produce realistic news articles, which humans identify as synthetic only 52% of the time. Therefore, it is pertinent that we develop robust automated schemes to accurately detect synthetic text.

Fortunately, the research community has developed several defenses to automatically detect synthetic text [18, 47, 48, 49, 147]. They are all supervised learning schemes that use a language model to extract features to build a classifier. Some defenses only focus on features that capture statistical artifacts or imperfections in the generated text [48]. Our reproduction of 6 of the best defenses show that they all achieve high detection performance, ranging from 79.6% F1 score to 98.5% F1 score in detecting synthetic samples. However, we lack a thorough understanding of the real-world applicability of these defenses (See Challenge 1 in Chapter 1).

For Objective 2 in Chapter 1, we focus on two key aspects to understand real-world applicability of synthetic text defenses: (1) How well would existing defenses perform when applied to synthetic text in the wild? All existing defenses have only been tested on synthetic datasets produced by the research community themselves. It is unclear how well they would work in the real world. To make matters more challenging, there is a lack of real-world synthetic datasets that the research community can use to study such defenses. (2) How would existing defenses perform against an adaptive attacker who is knowledgeable about the defenses? Existing defense work provides limited understanding of threats posed by adaptive
attackers. Of course, the community is well aware of attacks that craft adversarial samples to fool text classifiers \cite{150}. Such attacks are also feasible against synthetic text detection classifiers. However, such adversarial attacks may not always be practical. For example, text adversarial attack schemes like TextFooler assume a black-box scenario that requires a large number of queries to the victim defense model to craft adversarial samples. Such query-based attacks could be caught by looking for specific query patterns \cite{151, 152, 153}. Instead, we argue that there are much simpler, low-cost (computationally), yet effective strategies to fool existing defenses. To address these research questions, we conduct a study of synthetic text detection schemes through the lens of security. Our contributions are as follows:

- We conduct a measurement study to collect synthetic text in the wild, and introduce 4 new real-world synthetic datasets. This includes synthetic data obtained from 3 text-generation-as-a-service platforms (geared towards the SEO community), and synthetic data produced by a GPT-3 powered bot on the web (on Reddit.com). We find that Internet users and services are already using state-of-the-art Transformer-based models to synthesize text.

- We evaluate performance of 6 state-of-the-art defenses on our new real-world synthetic datasets. Many defenses show significant degradation in performance compared to their original claimed/reproduced performance. Defenses using entity-based semantic features and those using robust pre-training methods combined with bidirectional context, generalize better to content in the wild.

- We develop simple, computationally low-cost attacks that modify the attacker’s text generation process to evade detection. Our experiments show that just changing the decoding or text sampling strategy is sufficient to break many defenses. Moreover, defenses that are trained to look for specific text decoding artifacts are easier to evade
by changing the text decoding strategy.

- We propose and evaluate a new black-box adversarial sample crafting strategy called DFTFooler. Our attack requires no queries to the defense model, and exploits insights unique to the synthetic text detection problem. DFTFooler only requires a publicly available pre-trained language model to craft adversarial perturbations. DFTFooler can produce transferable adversarial samples that can degrade the performance of multiple defenses.

- Lastly, our analysis shows that a promising approach to improve defenses is to tap into the semantic information in the text content. Our analysis of the existing defense called FAST \cite{49} shows that using entity-based features capturing the factual structure of the text can lead to better adversarial robustness and generalization performance.

Our study provides many actionable insights that can be used to improve real-world applicability of defenses. Datasets and code used in this study are available on GitHub.\footnote{https://github.com/jmpu/DeepfakeTextDetection}

\section*{4.2 Background and Goals}

In this work, \textit{synthetic text} refers to text generated by DNN-based LMs, and \textit{real text} refers to human-written text.

\textbf{Using language models for text generation.} Synthetic text is generated using an LM. An LM is a statistical model that provides the joint probability distribution for a sequence of $n$ tokens $x_1, \ldots, x_n$. Tokens can be characters, words, or subword tokens \cite{154}. This joint distribution can be factorized by computing the conditional probability for each token, given
the previous tokens:

\[ p(x_0, \cdots, x_n) = \prod_{t=0}^{n} p(x_{t+1} \mid x_0, \cdots, x_t) \]  

(4.1)

Given a LM that provides the conditional probability in Equation 4.1, synthetic text is generated using the following steps: Feed an initial priming sequence, which can be a single token, \( x_0 \), into the LM, which then provides the conditional probability for the next token as \( p(x_1 \mid x_0) \). This priming sequence can also be a special “start-of-text” token (which the model is trained to recognize) or a sequence of tokens. In the second step, the next likely token, \( x_1 \), is sampled from the distribution over the token vocabulary—a process known as decoding. Next, we can generate the next token, by feeding \( (x_0, x_1) \) back into the LM, and sampling \( x_2 \), using the same decoding strategy. By repeating this process, one can generate text until a desired sequence length is reached, or an “end-of-text” token is chosen (which the model is trained to produce).

Two key factors impact the quality of synthetic text—the decoding function, and the LM architecture:

**Text decoding strategies.** Decoding strategies have witnessed significant development in recent years, and are now capable of producing high-quality synthetic text. Two effective decoding strategies are Top-k sampling [155] and Top-p or Nucleus sampling [156]. In Top-k sampling, the distribution is truncated and re-normalized to keep the \( k \) most probable tokens, and then a token is randomly sampled. Holtzman et al. [156] developed Top-p (or Nucleus) sampling, which produces more diverse and high quality text than Top-k sampling. In Top-p sampling, one truncates the distribution to keep the most probable tokens, such that their cumulative probabilities are greater than or equal to \( p \), where \( p \in [0, 1] \). Another approach, Temperature sampling [157], shapes the probability distribution by dividing the logits by a temperature parameter before passing them to the softmax function. Low temperatures make the model produce less diverse text, as it makes more confident predictions (tokens),
while temperatures greater than 1, result in more diverse text, as confidence is decreased. The simplest decoding strategy is greedy sampling, where the most probable next token is chosen. However, temperature sampling and greedy sampling tend to produce repetitive text [156]. Other sampling strategies such as beam search sampling [158] are also known to suffer from similar problems [156].

**DNN-based LM architectures.** Until 2017, the de facto choices were RNNs [62] and LSTMs [159]. These models use a recurrent loop to maintain an internal “hidden state” that stores information about previous tokens, which is used to compute the next token distribution. However, they have limitations, including vanishing/exploding gradients [160], the sequential nature of the model limiting parallelization, and an inability to generate longer coherent text [161].

In 2017, Vaswani et al. proposed the Transformer [53] architecture to address limitations of RNNs. Transformers, in contrast with RNNs/LSTMs, are based on the “attention” mechanism [70]. Instead of using a single hidden state to represent all previous tokens, attention mechanisms allow the model to compute vector representations of each token separately. These representations take into account context and relationship with all other tokens. Attention mechanisms can be customized to “pay attention” to only previous tokens (unidirectional attention), or to pay attention to future tokens, assuming they are provided (bidirectional attention). While the bidirectional attention is not suitable for text generation (future tokens are not available when generating text), it is useful when computing representations for other NLP tasks, e.g., classification. Another advantage of Transformers is that each token can be processed in parallel, and is not dependent on previous tokens. This is made possible by using the teacher-forcing paradigm [162]. Transformers now power state-of-the-art for many NLP tasks. Examples include the popular BERT [71], RoBERTa [72], GPT-2 [66], and GPT-3 [20] models.
Goals. Our goal is to understand the real-world applicability of existing defenses to detect synthetic text. We focus on two directions: (1) Understanding and improving performance of defenses in the wild. The research community has made significant progress in developing detection schemes [18, 47, 48, 49, 147]. However, these defenses have been primarily tested on synthetic text produced by researchers themselves. It is unclear how well these methods would generalize to synthetic text in the wild, i.e., those produced by the Internet community. We collect real-world synthetic text to understand the performance of existing detection schemes. We also propose efficient methods that adapt existing defenses to improve performance in the wild. (2) Understanding and improving performance against adaptive attackers. Before deployment, defenders should consider an adaptive attacker who is knowledgeable about the defense and aims to evade detection. Existing works on evasion strategies primarily focus on generic black-box attacks that require a large number of queries to the defender’s model [163]. However, the defender’s model may not even be exposed as a public API for queries. Also, one can detect query-based adversarial sample crafting schemes by looking for specific query patterns [151, 152]. Using a surrogate model and relying on the adversarial samples to “transfer” may not always be effective either [164, 165]. Instead, we investigate more practical, (computationally) low-cost evasion strategies that require no queries to the defender’s model, and no surrogate model as well. We also investigate methods to improve resilience against the proposed evasion strategies.

4.3 Models, Datasets and Metrics

We study 6 state-of-the-art defenses for detecting synthetic text. To study synthetic text in the wild, we introduce 4 new synthetic datasets. We present metrics to evaluate defense performance on real-world datasets and against adaptive attackers.
### 4.3. Models, Datasets and Metrics

#### 4.3.1 Defenses: Synthetic Text Detection Schemes

Existing defenses are supervised learning schemes that use a LM to extract features to build a binary classifier (real vs. synthetic). We consider 5 existing defenses, including GROVER [18], GLTR-BERT [48], GLTR-GPT2 [48], BERT-Defense [47], and FAST [49]. Inspired by BERT-Defense, we build an additional defense, called RoBERTa-Defense.

**Defense performance metrics.** Existing defenses are evaluated using a variety of metrics. BERT-Defense reports accuracy and AUC, whereas GLTR reports only AUC. FAST and GROVER use a modified version of accuracy, called paired accuracy (in addition to normal accuracy). Paired accuracy is computed by pairing real and synthetic articles such that both articles share the same metadata, *e.g.*, article title. If the detector assigns the synthetic article a higher probability than the real class, it is considered to be correctly classified. However, the paired accuracy setting is unrealistic because it assumes access to real articles used to generate the synthetic articles. Therefore, we do not use it in our study. We report all results primarily using the F1 score, Precision, and Recall for the synthetic class. These metrics provide insights into class-specific detection performance.
(unlike accuracy) and does not delegate the task of calibrating a decision threshold to future work. Additionally, we report AUC ROC scores for the 6 defenses on their original test datasets in Table 4.1.

**Defenses.** Table 4.1 provides an overview of the training setup for each defense, and their performance. GROVER, RoBERTa-Defense, and FAST are trained to detect synthetic news articles, while GLTR-BERT, GLTR-GPT2, and BERT-Defense are “open-domain” schemes and applicable to diverse topic domains.

1. **GROVER.** Proposed by Zellers *et al.* [18], it is a framework that can both generate and detect synthetic news articles. They first train a synthetic news article generator using a GPT-2 based LM. This generator is trained on the RealNews dataset [18], a large corpus (120GB) of news articles from Common Crawl [166]. The GPT-2 model is modified to incorporate context fields for relevant metadata, *e.g.*, date, author names, article title, web domain and body text. This generator can produce synthetic news articles conditioned on the metadata. Next, to build a detection scheme, a classification layer is attached to extract information from the hidden state of the special [CLS] token (placed at the end of each article). This updated model is fine-tuned on synthetic articles generated by GROVER and more real articles from the RealNews dataset. GROVER performs well when detecting text generated by GROVER itself. However, this is not a realistic setting, as the defender is unlikely to have access to the attacker’s generator. We study GROVER in more realistic settings.

**Our GROVER setup:** We use the largest, publicly released version (1.5B parameters) of the GROVER classifier called GROVER-Mega [167]. Details of the training and testing setup are in Table 4.1. GROVER is fine-tuned using 5000 real articles from the RealNews dataset, and 5000 synthetic articles generated by GROVER itself. Using this publicly available model, we obtain an F1 score of 87.1% on their original test set [18].
(2) GLTR-BERT and GLTR-GPT2. GLTR [48] uses the insight that decoding strategies tend to sample tokens that are assigned high probabilities by the LM. Hence, synthetic text can be detected by analyzing the likelihood of tokens in the text sequence, as determined by a LM. Presence of many high probability tokens is an indication that the text sample is likely synthetic. Using a LM, GLTR extracts features based on the number of tokens in the Top-10, Top-100, and Top-1000 ranks as determined by the token probability distributions. The features are then fed to a Logistic Regression classifier.

Our GLTR setup: We use the authors’ code to extract the features [168], but no code for building the Logistic Regression classifier was released. The authors reported using Logistic Regression with default settings in the scikit-learn library [169]. We additionally apply grid search on the hyperparameters to ensure the classifier is properly tuned (See Appendix A.5 for details). To build an open-domain classifier, we train on synthetic text from GPT2-XL (similar to the original work) and real articles from the WebText dataset [66]. Both sources are known to cover diverse topics. Similar to the original work [48], we create 2 variants of GLTR, namely GLTR-BERT that uses BERT [71], and GLTR-GPT2 that uses the GPT2-XL [66] as the back-end LM.2 We obtain F1-scores of 98.5% and 79.6% for GLTR-GPT2 and GLTR-BERT, respectively.

(3) BERT-Defense. A BERT-based binary classifier, proposed by Ippolito et al. [47], attaches a classification layer to a pre-trained BERT-Large LM, and then fine-tunes it on a dataset of synthetic and real articles.

Our BERT-Defense setup: The authors did not release the datasets and models. We replicated their experimental setup. See Table 4.1 for details. While the original work reported an accuracy of 81%, our model achieves an F1 score of 88.8% on the test set. We achieve a higher F1 score, even after using a smaller training set (10,000 articles per class versus

---

2GLTR [48] used GPT2-small as the LM, but we use a larger and more accurate LM, GPT2-XL.
250,000 articles in the original work). Our model uses the full context window size of 512 tokens, supported by BERT, unlike the smaller window size of 192 tokens in the original. We suspect the higher performance is likely due to the larger context window used by our implementation.

(4) RoBERTa-Defense. Inspired by BERT-Defense, we create an additional defense using the same approach, but with a different language model, RoBERTa [72]. RoBERTa makes several changes to the BERT LM, such as training the model on a larger dataset with a bigger batch size, removing the next-sentence-prediction task, training on longer sequences and dynamically changing the masking pattern applied to the training data. RoBERTa is known to outperform BERT on NLP tasks such as GLUE [170], SQuAD [171], and RACE [172].

Our RoBERTa-Defense setup: We train a RoBERTa-base model on synthetic text produced by GROVER and real text obtained from the RealNews dataset, and obtain an F1 score of 86.3% in detecting synthetic news articles.

(5) FAST. FAST, proposed by Zhong et al. [49], unlike other defenses, taps into the semantic layer or the “factual structure” of text. FAST uses a graph-based learning approach that uses features based on named entities in the document. FAST exploits the insight that state-of-the-art text generators still produce inconsistencies in the factual structure of text, when compared to real text. For example, while it is easy for humans to correctly mention and reference named entities (e.g., location, people, objects) across sentences, a text generator might make mistakes and create inconsistencies in how entities are mentioned in continuous sentences. To capture such inconsistencies, FAST constructs an entity network based on how entities are referenced within and across sentences, and uses a graph convolution network (GCN) to learn patterns in the network. In addition, FAST also uses the RoBERTa LM [72] to extract token, and document-level representations, in conjunction with the GCN-based features. We analyze FAST in detail in Section 6.2.1.
4.3. MODELS, DATASETS AND METRICS

Our FAST setup: A pre-trained model for FAST was not available. We obtained code for FAST, and reproduced the experimental setup described in the original work. Similar to the original work, we train FAST to detect synthetic news articles, and train it on the RealNews dataset (real class) and text from GROVER (synthetic class). \(^3\) We obtain an F1 score of 87\% in detecting news articles (Table 4.1).

Context window size for training and evaluation. All 6 defenses use a context window size of the first 512 tokens in an article for detection, as determined by its tokenization scheme. To understand the impact of the context window size for detection, we evaluate all defenses against smaller context window sizes, \(i.e.,\) 64, 128, 256 tokens, and present results in Figure A.2 in the Appendix. Performance (F1 score) monotonically increases as the context window size increases. Therefore, we chose a 512 token window size. Larger window size would significantly increase computational complexity for all experiments, and some pre-trained models only support a certain maximum window size (\(e.g.,\) 512 tokens for BERT).

Other defenses. There are a few other defenses that are not considered in our study. For example, Yao \textit{et al.} \cite{147} proposed a method to detect LSTM-generated synthetic reviews. We omit it because our preliminary evaluation yielded unsatisfactory results—an F1 score of only 68\% in detecting GROVER generated text. We discuss more details of these experiments and other defenses in the Appendix A.4.

4.3.2 Evaluation Metrics

We use the following metrics to measure defense performance on text in the wild and against adaptive attackers:

\(^3\)Zhong \textit{et al.} \cite{49} additionally trained a version of FAST on WebText (real) and GPT2-XL text (synthetic). We omitted this setting for simplicity.
Percentage change in detection performance: $\Delta F_1$ and $\Delta \text{Recall}$. We measure the percentage change in detection performance for the synthetic class, when applied to a new test dataset, compared to a specific baseline performance. This is broken into percentage changes in F1 and Recall, e.g., $\Delta F_1 = (F_{1_{\text{new}}} - F_{1_{\text{baseline}}})/(F_{1_{\text{baseline}}})$. The new test dataset can be an In-the-wild dataset, or a dataset containing synthetic text produced by an adaptive attacker. We define baseline performance depending on the experiment context, which usually refers to the defense performance when evaluated on the test datasets considered in the original work (e.g., numbers in Table 4.1). For attack experiments, where only synthetic samples are modified in the test set (i.e., real samples are the same in the test and baseline settings), we only consider $\Delta \text{Recall}$ (for the synthetic class), as there will be no change in false positives (real samples classified as synthetic). Note that the change in performance can be a degradation or improvement in performance.

Evasion rate (ER). In some attacks, the adaptive attacker perturbs existing synthetic samples to evade detection. In such settings, we use evasion rate, defined as the fraction of perturbed synthetic samples that evade detection by a defense. Higher fraction indicates higher attack success.

Evaluating quality of synthetic text. For adaptive attackers, it is not sufficient to evade detection, it is also necessary to maintain high linguistic quality of the generated text. We measure linguistic quality using the state-of-the-art GRUEN metric [173]. Zhu et al. proposed GRUEN, an unsupervised, reference-less metric designed for synthetic text. GRUEN correlates highly with human judgements, and better than other existing metrics for linguistic quality. The metric, computed for a synthetic sample, ranges from 0 to 1, and a higher value indicates better linguistic quality. An advantage is that this metric does not require any reference text, which usually requires human effort to obtain. GRUEN measures linguistic quality based on “grammaticality”, non-redundancy, discourse focus, structure and
4.3. MODELS, DATASETS AND METRICS

<table>
<thead>
<tr>
<th>Datasets</th>
<th>#Documents per Class</th>
<th>Document Topic(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI-Writer</td>
<td>1,000</td>
<td>News</td>
</tr>
<tr>
<td>ArticleForge</td>
<td>1,000</td>
<td>News</td>
</tr>
<tr>
<td>Kafkai</td>
<td>1,000</td>
<td>Cyber Security, SEO, Marketing</td>
</tr>
<tr>
<td>RedditBot</td>
<td>887</td>
<td>Reddit Comments</td>
</tr>
</tbody>
</table>

Table 4.2: Details of the In-the-wild datasets.

coherence. More details are in the Appendix A.2. Other linguistic quality metrics have also been proposed over the years. In the Appendix A.1 we describe other metrics, and justify our choice of using GRUEN over the other metrics.

4.3.3 In-the-wild Datasets

We collect 4 In-the-wild datasets from the web containing both synthetic and real articles from matching semantic categories. All measurements were conducted from Nov. 2020 to Apr. 2021. This includes synthetic text posted by Internet users, and text from synthetic text-generation-as-a-service platforms, geared towards the SEO community. Synthetic text generation services could be misused to create fake news articles, fake reviews, or fake web articles for BlackHat SEO activities. While we could not verify the text generators used by the services we study, they claim to use customized versions of Transformer-based LMs. This again highlights the need to understand real-world performance of defenses, because text generators used in the wild can be different from those used by the research community. Table 4.2 shows dataset statistics.

---

4The GRUEN implementation we obtained from the authors, does not compute structure and coherence. The authors claim that this omission does not impact the metric scores significantly.
AI-Writer. This dataset was collected from the text-generation service, AI-Writer [174]. Given a title, AI-Writer claims to generate factually accurate articles capturing recent information on the topic (based on the title). In our email communication with the service, they claim to employ custom Transformer-based LMs that are not available off-the-shelf. Since AI-Writer requires titles to generate articles, we first collect real news articles, and use the title from the real articles for generation. We evenly and randomly scraped 1000 real news articles from 20 popular news websites sampled from the RealNews [18] dataset. The list of websites is shown in Table A.1 in the Appendix. We verified that this dataset has no overlap with the training datasets of the defenses (Table 4.1), based on the article publication dates. These articles form the real class of the dataset. Next, we used the real article titles to generate 1000 synthetic articles from AI-Writer. AI-Writer charges for article generation, and we spent $400 for generating 1000 articles.

ArticleForge. This dataset was collected from the ArticleForge text-generation service [175]. ArticleForge requires a set of keywords to generate an article. As per our communication with the service, they claim to use fine-tuned versions of GPT-2 [66], BERT [71] and T5 [176] to generate synthetic text. We follow a similar methodology as used for AI-Writer, and collect 1000 synthetic, and 1000 real news articles. ArticleForge charged us $57, for which they allow unlimited article generation for a month.

Kafkai. This dataset was collected from the Kafkai text-generation service [177]. Given one of 25 categories and an initial priming text, Kafkai generates a unique synthetic article that belongs to that category and is contextualized by the priming text. As per our communications with their service, Kafkai uses models from OpenAI, including GPT-2, and fine-tunes them on millions of articles to generate high quality synthetic text. We follow a similar methodology as AI-Writer and ArticleForge, and obtain context for the synthetic article generation from 1000 real articles—100 articles from 10 of the 25 available categories,
4.4. Defense Performance in the Wild

4.4.1 Detection Performance

We test the 6 defenses (Section 4.3.1) on our 4 In-the-wild datasets (Section 4.3.3). Since GROVER, FAST and RoBERTa-Defense are trained for the news domain, we only test them on news domain datasets, which includes AI-Writer and ArticleForge. The remaining defenses, GLTR-BERT, GLTR-GPT2, and BERT-Defense are trained on a diverse corpus.
Chapter 4. Synthetic Text Detection: Investigate Real-world Effectiveness of Existing Defenses

<table>
<thead>
<tr>
<th>Datasets</th>
<th>BERT-Defense</th>
<th>GLTR-GPT2</th>
<th>GLTR-BERT</th>
<th>GROVER</th>
<th>FAST</th>
<th>RoBERTa-Defense</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1</td>
<td>ΔF1</td>
<td>F1</td>
<td>ΔF1</td>
<td>F1</td>
<td>ΔF1</td>
</tr>
<tr>
<td>AI-Writer</td>
<td>28.8</td>
<td>-67.6</td>
<td>1.6</td>
<td>-98.4</td>
<td>64.8</td>
<td>-18.6</td>
</tr>
<tr>
<td>ArticleForge</td>
<td>19.7</td>
<td>-77.8</td>
<td>44.1</td>
<td>-55.2</td>
<td>85.6</td>
<td>+7.5</td>
</tr>
<tr>
<td>Kafka</td>
<td>65.9</td>
<td>-25.8</td>
<td>1.0</td>
<td>-99.0</td>
<td>41.4</td>
<td>-48.0</td>
</tr>
<tr>
<td>RedditBot</td>
<td>14.1</td>
<td>-84.1</td>
<td>61.5</td>
<td>-37.6</td>
<td>83.4</td>
<td>+4.8</td>
</tr>
</tbody>
</table>

Table 4.3: Performance of the defenses, i.e., F1 (%) and \( \Delta F1 \) (%) of the synthetic class, on the In-the-wild datasets. The percentage change in F1 (\( \Delta F1 \)) is computed from the baseline performance of each defense. ‘+’ means performance improvement and ‘-’ means performance degradation. “-“ (the longer minus mark) indicates experiments we ignored: We did not test GROVER, FAST and RoBERTa-Defense on non-news domain datasets, i.e., Kafka and RedditBot. This is because these defenses are only trained for the news domain.

and therefore can be tested on all the datasets. We report F1 and \( \Delta F1 \) (Section 4.3.2).

To compute \( \Delta F1 \), we use the performance of each defense on their original test set as the baseline performance (see Table 4.1).

Detection performance in the wild is presented in Table 4.3. Detailed results including the Precision and Recall scores are in the Appendix (Table A.2). Before we discuss the results, note that all 6 defenses achieve high detection performance (79.6% to 98.5% F1) on their original test datasets (Section 4.3.1). Our key findings are as follows:

**Finding 1:** Open-domain defenses show significant degradation in performance when applied to synthetic text in the wild, while defenses trained on data from a specific domain are able to detect In-the-wild data from that domain. All three open-domain defenses — BERT-Defense, GLTR-BERT, GLTR-GPT2 show significant performance degradation ranging from 18.6% to 99.0% degradation in F1 score. All 3 of these defenses exhibit performance worse than a random predictor (50% F1) for at least one In-the-wild dataset. BERT-Defense and GLTR-GPT2 show significant degradation on all the datasets. All three news-based defenses — GROVER, RoBERTa-Defense, FAST perform well above the open-domain defenses on the news-based datasets. FAST and RoBERTa-Defense perform better on these datasets than on their original test datasets.
We further investigate the performance differences between the open-domain and the news domain defenses. Our hypothesis is that this can be attributed to distribution shift or distributional differences between data in the wild and the original datasets used to train/evaluate the defenses. To study this, we choose BERT-Defense from the open-domain category, and GROVER from the news category. We use a simple metric, average-linkage [179], to measure the distribution distance between two synthetic datasets. Given two synthetic datasets, $X$ and $Y$, we define the distribution distance as $D(X, Y)$. To represent a dataset’s distribution, we randomly select 1,000 articles (or all available samples if there are fewer samples) from the dataset, and then extract the hidden state of the special [CLS] token (used as the input to the classifier) in each article from the detector as its representation. Thus $X$ and $Y$ each includes 1000 embedding vectors. Larger values of $D(X, Y)$ indicate a larger distribution shift.

For each defense, we compute 2 types of distance measures: (1) Distance $D(X_{\text{train}}, Y_{\text{test}})$ between its training dataset $X_{\text{train}}$ and its original test set $Y_{\text{test}}$ (i.e., test set used in Table 4.1). (2) Distance $D(X_{\text{train}}, Y_{\text{wild}})$ between its training dataset and each of the In-the-wild datasets $Y_{\text{wild}}$. We expect that $D(X_{\text{train}}, Y_{\text{wild}})$ is greater than $D(X_{\text{train}}, Y_{\text{test}})$ if the defense’s performances on In-the-wild datasets degrade, and $D(X_{\text{train}}, Y_{\text{wild}})$ is closer to $D(X_{\text{train}}, Y_{\text{test}})$ if the detection performances are similar. Results are in Figure 4.1. All the $D(X_{\text{train}}, Y_{\text{wild}})$ are significantly greater than $D(X_{\text{train}}, Y_{\text{test}})$ for BERT-Defense, but not for GROVER. This observation is in line with our hypothesis.

**Finding 2:** Robustly pre-trained bidirectional models, i.e., RoBERTa, generalize better than unidirectional models. RoBERTa-Defense and FAST show performance improvements (F1) when applied to synthetic news samples in the wild. Both approaches use features extracted using RoBERTa, which improves over the BERT bidirectional model. The authors of RoBERTa show that BERT is significantly undertrained, and propose changes to improve
4. SYNTHETIC TEXT DETECTION: INVESTIGATE REAL-WORLD EFFECTIVENESS OF EXISTING DEFENSES

Figure 4.1: Distribution distance between the training set and the test sets (including baseline test set and In-the-wild test sets) of BERT-Defense and GROVER.

BERT’s pre-training [72]. It is worth noting that GROVER, a unidirectional model, claims to perform better than bidirectional models (e.g., BERT). However, our finding suggests that this claim is not true if the bidirectional model is robustly pre-trained, as in the case of RoBERTa. Moreover, a unidirectional model like GROVER is over 10x larger than the RoBERTa-base we use, in terms of number of parameters, yet it under-performs.

Another surprising result is that of GLTR-BERT performing well on ArticleForge and RedditBot (a GPT-3 dataset), but GLTR-GPT2 performs poorly on these two datasets. This means that the back-end LM model (BERT vs GPT-2) used by GLTR can have a huge impact on generalization performance.

4.4.2 Improving Performance in the Wild

Can we adapt the defenses that currently perform poorly (in the wild) to perform better on a target dataset? We investigate domain adaptation via transfer learning, i.e., by fine-tuning the classifier on data from the target distribution. Language model fine-tuning has shown tremendous success for domain adaptation [180]. We consider a realistic and challenging setting, where the samples from the target distribution (for fine-tuning) are limited—in our case, as little as 10, 50 or 100 samples each for the synthetic and real class. This fits a scenario where the Internet community, including text-generation-as-a-service platforms,
4.4. Defense Performance in the Wild

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Detection Performance (F1)</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BERT-Defense</td>
<td>GROVER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Before</td>
<td># Samples</td>
<td></td>
<td>Before</td>
<td># Samples</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fine-tuning</td>
<td>for Fine-tuning</td>
<td></td>
<td>Fine-tuning</td>
<td>for Fine-tuning</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>50</td>
<td>100</td>
<td>10</td>
<td>50</td>
<td>100</td>
<td>10</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>AI-Writer</td>
<td>28.8</td>
<td>68.9</td>
<td>85.8</td>
<td>90.4</td>
<td>87.6</td>
<td>90.7</td>
<td>93.8</td>
<td>94.8</td>
<td></td>
</tr>
<tr>
<td>ArticleForge</td>
<td>19.7</td>
<td>85.3</td>
<td>90.8</td>
<td>93.7</td>
<td>76.9</td>
<td>86.5</td>
<td>95.9</td>
<td>97.1</td>
<td></td>
</tr>
<tr>
<td>Kafkai</td>
<td>65.9</td>
<td>71.0</td>
<td>83.8</td>
<td>85.0</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>RedditBot</td>
<td>14.1</td>
<td>71.5</td>
<td>90.6</td>
<td>95.2</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4: Detection performance in F1 (%) of BERT-Defense and GROVER on In-the-wild datasets before and after fine-tuning on a limited set of articles from the real and synthetic classes. “–” represents experiments we ignored: GROVER is a news domain defense, and not applicable to non-news domain datasets, i.e., Kafkai and RedditBot.

rapidly updates their generative models, or produces many model variants over time. In such a setting, it is hard to obtain abundant ground-truth data for attack class (synthetic samples). In such a setting, can the defender keep up?

We fine-tune the models by extending the training on the binary classification task. We consider the defenses, BERT-Defense and GROVER since they exhibit a degradation in performance (Table 4.3). While fine-tuning BERT-Defense on our small datasets, we encountered a known issue of training instability [71]. To overcome this, we employ the revitalization strategy proposed by Zhang et al. [181]. The training hyperparameters used in BERT-Defense and GROVER fine-tuning experiments can be found in the Appendix A.6. Results are in Table 4.4. Our findings are as follows:

**Finding 3:** Fine-tuning with as limited as 10 In-the-wild data samples can help defenses adapt to new domains, and more samples lead to better fine-tuning performance. Both defenses benefit from observing a few samples from both classes of the target dataset. Moreover,

---

GLTR also exhibits a degradation in performance, but we do not consider it. GLTR is not a DNN-based classifier, and is therefore not suitable for fine-tuning.
detection performance only improves with more fine-tuning samples.

4.5 Defense Against Adaptive Attackers

4.5.1 Attack Methods

To ensure real-world applicability, defenses should be effective against adaptive adversaries who are aware of the defense scheme, and can adapt the synthetic text to bypass detection. We focus on low-cost and practical adaptive attacks. Our attacks do not require a computationally expensive re-training of the attacker’s generative model or creation of a surrogate/shadow defense model to craft adversarial samples. We assume a black-box setting requiring no queries to the defense scheme to craft adversarial samples. We also consider maintaining the linguistic quality of the synthetic text as the attacker’s constraint. If linguistic quality is degraded significantly, it impacts the attacker’s goals of misleading users, e.g., synthetic fake news articles with poor linguistic quality could raise suspicion from users. These assumptions provide a more realistic setting for attackers. Our attack methods are split into two categories: (1) Attacks that change the text generation process without re-training the text generator, and (2) attacks that add adversarial perturbations to existing synthetic text samples to evade detection.

Evasion by Changing the Text Generation Process

Existing defenses are trained on synthetic text, created based on a specific decoding strategy and priming process. Our goal is to understand the robustness of defenses against changes to the text distribution triggered by varying the text generation process. Given a generative model, our idea is to craft different distributions of synthetic text samples by: (1) varying
4.5. Defense Against Adaptive Attackers

<table>
<thead>
<tr>
<th>Defenses</th>
<th>Baseline Decoding Setting</th>
<th>Attack: Changing the Decoding Strategy</th>
<th></th>
<th>Top-k Decoding</th>
<th>Temperature Decoding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Top-p Decoding</td>
<td>Top-k Decoding</td>
<td>Temperature Decoding</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ΔR</td>
<td>ΔGRN</td>
<td>ΔR</td>
<td>ΔGRN</td>
</tr>
<tr>
<td>BERT-Defense</td>
<td>Top-p 0.96</td>
<td>0.8</td>
<td>-13.3</td>
<td>+0.5</td>
<td>40</td>
</tr>
<tr>
<td>GLTR-GPT2</td>
<td>Top-k 40, Temp 0.7</td>
<td>0.98</td>
<td>-97.6</td>
<td>+2.0</td>
<td>160</td>
</tr>
<tr>
<td>GLTR-BERT</td>
<td>Top-k 40, Temp 0.7</td>
<td>0.98</td>
<td>-90.0</td>
<td>+2.0</td>
<td>160</td>
</tr>
<tr>
<td>FAST</td>
<td>Top-p 0.96</td>
<td>1.0</td>
<td>-35.6</td>
<td>-4.1</td>
<td>160</td>
</tr>
<tr>
<td>RoBERTa-Defense</td>
<td>Top-p 0.96</td>
<td>1.0</td>
<td>-22.0</td>
<td>-3.6</td>
<td>160</td>
</tr>
</tbody>
</table>

Table 4.5: Performance of attacks that change the decoding strategy of the LM. Evaluation metrics include Recall (R) of the synthetic class and the average GRUEN (GRN) of synthetic data. We show the percentage change of each evaluation metric (ΔR, ΔGRN) from the baseline performance of each defense on the most effective attack of each decoding strategy. ‘+’ means performance improvement and ‘-’ means performance degradation. “–” (the longer minus mark) indicates experiments we ignored: We only consider Temperature decoding for GLTR defenses, because the original GLTR work was evaluated using temperature-based decoding.

The text decoding method (and its parameters), and (2) by varying the number of priming tokens used by the model. We evaluate the linguistic quality of the adapted synthetic text using the GRUEN metric (Section 4.3.2). An attack is considered successful only if it degrades defense performance, while preserving linguistic quality or with limited degradation in linguistic quality.

Each defense was trained to detect text produced using a certain decoding and priming strategy. We consider this as the baseline settings, shown in the second column of Tables 4.5 and 4.6. We vary the decoding strategy by considering the following methods: Top-k, Top-p (or nucleus sampling), and Temperature decoding. For each decoding strategy, we also consider different parameter settings. For Top-p, we consider several values of $p$ in the range $[0.8, 1]$ in small increments. For Top-k, we vary $k$ using the following values $[40, 80, 120, 160]$, and Temperature in the range $[0.7, 0.9]$. These values were determined based on standard value ranges used in prior work $[18, 49]$, and values beyond this range resulted in significant degradation in linguistic quality.
All the defenses, except BERT-Defense is trained on unconditional text (i.e., number of priming tokens is 0). BERT-Defense uses a single priming token. Priming language models with some tokens, as opposed to unconditional generation, can change the statistical and qualitative properties of generated text as the model might never generate the priming sequence on its own, e.g., due to its low probability. For this attack, we generate text using a varying number of priming tokens $n$, where $n \in [1, 4, 8, 12]$. Each synthetic article is generated using the first $n$ tokens from a real article. We limit the maximum number of priming tokens to 12 to minimize the amount of real text in synthetic articles.

**Evasion by Adversarial Perturbations**

We craft adversarial inputs in a black-box setting by leveraging insights unique to the synthetic text detection problem.

**Crafting adversarial inputs using DFTFooler.** Given a synthetic sample, our approach called DFTFooler, aims to misclassify it as real by adding adversarial perturbations to it. Unlike existing work on adversarial inputs in the text domain [150], DFTFooler requires no queries to the victim model, or a surrogate/shadow classifier to craft the perturbations. DFTFooler only requires a pre-trained LM, and several versions are publicly available today [182].

Given a synthetic article, we identify a (limited) set of words that are important for classification, and replace them with words that alter the model’s prediction while preserving semantic similarity and linguistic quality. The challenge is in identifying the important words and finding suitable replacements that alters the prediction—we draw insights from the GLTR approach. Our insight is that generative models tend to generate the next token from the head of the distribution, thus capturing only a limited subset of the true distribution of natural language [48]. In other words, if we pass a synthetic article and a real article through
4.5. Defense Against Adaptive Attackers

A pre-trained LM (e.g., BERT), the synthetic article is likely to contain many tokens which have a high probability of being generated by that language model. On the other hand, real articles will contain many low probability tokens since humans exhibit greater variation in their choice of words and this is hard for LMs to emulate. Our hypothesis is that defense schemes learn this difference between real and synthetic articles for discrimination. Therefore, to misclassify a synthetic sample, we replace a subset of the most confidently predicted words (using a pre-trained LM) by its synonyms that have lower confidence (according to the same LM).

First, DFTFooler scans a given article to choose the top $N$, most confidently predicted words according to a chosen LM, for replacement. The importance of a word is determined by the absolute rank of its token probability predicted by the LM. DFTFooler does not perturb stop words. More details about using a LM to find important words are described in the Appendix A.7.

The second step is to find replacement words, while preserving semantics. To find a synonym for replacement, we build on the methodology used by TextFooler [163], and adapt it to our setting to not require queries to the victim model. For a targeted word, there are 4 sub-steps to find a valid synonym: (1) Synonym Extraction: We first extract a candidate set of synonyms for the targeted word as its possible word replacements. Synonym candidates are chosen according to the cosine similarity between the word embeddings of the targeted word and every other word in the vocabulary. We use word embeddings from [183] which are specially curated for finding synonyms. (2) POS Checking: The goal of POS checking is to assure that the grammar of the perturbed text remains the same. We will only keep synonyms with the same part-of-speech (POS) tag as the targeted word. (3) Semantic Similarity Checking: This step ensures that the sentence semantics before and after replacing

\footnote{For words that are split into multiple sub-tokens, we use the probability prediction of the first sub-token in the word.}
the targeted word remain similar. Similar to TextFooler, we use the Universal Sentence Encoder (USE) [184] to compute sentence similarity. At this step, we only keep synonyms that can maintain sentence similarity scores above 0.7. (4) Choose a synonym with low confidence as measured by a LM: At the last step, we choose a replacement word from valid synonyms that is predicted by the LM with a low probability of \( \leq 0.01 \). We choose the synonym with the lowest probability if multiple synonyms meet the threshold requirement. It is possible that less than \( N \) words are replaced if the semantic similarity conditions are not met. Empirically, we find that \( N = 10 \) works well in practice, offering a trade-off between evasion rate, and preserving linguistic quality/semantics. We implement DFTFooler using 2 back-end LMs, namely BERT and GPT2-XL. That said, the backend model is replaceable when more advanced LMs emerge in the future.

**Perturbation attack baselines.** We compare attack performance of DFTFooler with the following two approaches: (1) TextFooler [163]: TextFooler is a black-box attack that requires a large number of queries to the defense model to craft adversarial perturbations. TextFooler is highly effective against Transformer-based classifiers, and also preserves the utility of the attack by preserving the semantic content. TextFooler finds important words to replace by querying the defense model, and also queries the model to find the best replacement words. (2) random perturbations: This is a simple baseline that replaces random words in the article by synonyms that preserve the semantic content. This approach requires no LMs or queries to the defense model. Similar to DFTFooler, we only replace \( N \) words in an article. Such a baseline would serve to understand the benefit of replacing words based on their importance (as in our DFTFooler). DFTFooler should perform better than this baseline to be considered a useful attack.
4.5. Defense Against Adaptive Attackers

4.5.2 Attack Evaluation

In this section, we evaluate the adaptive attacks.

Evasion by adapting the decoding process Table 4.5 presents the results for adapting the decoding strategies. Attack success is measured using the percentage change in Recall for the synthetic class, $\Delta R$. To compute $\Delta R$, the baseline setting is the performance on the original test dataset of each defense (Table 4.1). A higher $\Delta R$ indicates better attack success. We only report $\Delta R$ because the real set is the same as in the baseline experiment setting (from Table 4.1). In Table 4.5, for each defense, we show results for the most effective attack configuration (i.e., decoding method and its parameters). The most effective attack is the one with the largest degradation in $\Delta R$, with only a minor (up to 5%) or no degradation in linguistic quality measured by the GRUEN score. We report percentage change in average GRUEN score ($\Delta GRN$) of articles before and after changing the decoding strategy. For Temperature decoding, we only show results for the GLTR defenses, because GLTR was originally evaluated using Temperature decoding. Temperature decoding is known to produce repetitive text [157], and is omitted for the other defenses. Findings are as follows:
Table 4.6: Performance of attacks which prime the LM with a different number of priming tokens. Evaluation metrics include Recall (R) of the synthetic class and the average GRUEN (GRN) of synthetic data. We show the percentage change of each evaluation metric ($\Delta R$, $\Delta GRN$) from the baseline performance of each defense on the most effective length of priming tokens. ‘+’ means performance improvement and ‘-’ means performance degradation.

Finding 4: Changing the decoding strategy is a simple and effective way to break many defenses. All defenses, except FAST, show significant degradation in $\Delta R$ under at least one of the attack strategies, ranging from 13.3% to 97.6% degradation. FAST does exhibit degradation, but to a lesser extent, compared to the other defenses, i.e., degradation in $\Delta R$ ranging from 2.9% to 9.7%. This suggests that defenses like FAST are able to learn more robust features from the text. For GROVER, FAST and RoBERTa-Defense, the most effective attack (based on $\Delta R$) happens at a Top-p value of 1.0, which is basically sampling from an untruncated distribution. However, note that the degradation in average GRUEN score is small (<5%) in these cases. Figure 4.2 shows the CDF of the GRUEN scores for text applied to GROVER at its baseline setting (Top-p 0.94), and when Top-p is 1.0. The two distributions are adjacent, indicating that using an untruncated distribution is not significantly degrading linguistic quality, thus providing more room for the attacker to fool defenses.

Finding 5: Classifiers that rely solely on detecting differences in token likelihoods provided by LMs can be easily fooled by changing the decoding strategy. We observe that GLTR-BERT
Table 4.7: Attack performance of the three adversarial perturbation methods (i.e., DFT-Fooler, random perturbations, TextFooler) against the defenses, based on 10 word perturbations. GRN-Before: the average GRUEN of original datasets; GRN-After: the average GRUEN of datasets produced by adversarial perturbation attacks; ER: Evasion Rate (%).

<table>
<thead>
<tr>
<th>Defenses</th>
<th>GRN-Before</th>
<th>DFT-Fooler</th>
<th>Random Perturbations</th>
<th>TextFooler</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>BERT Backend</td>
<td>GPT2-XL Backend</td>
<td></td>
</tr>
<tr>
<td>BERT-Defense</td>
<td>0.652</td>
<td>1.0</td>
<td>0.568</td>
<td>1.0</td>
</tr>
<tr>
<td>GLTR-GPT2</td>
<td>0.686</td>
<td>91.3</td>
<td>0.594</td>
<td>74.2</td>
</tr>
<tr>
<td>GLTR-BERT</td>
<td>0.756</td>
<td>44.7</td>
<td>0.654</td>
<td>45.9</td>
</tr>
<tr>
<td>GROVER</td>
<td>0.734</td>
<td>59.1</td>
<td>0.647</td>
<td>43.8</td>
</tr>
<tr>
<td>FAST</td>
<td>0.732</td>
<td>24.9</td>
<td>0.646</td>
<td>23.2</td>
</tr>
<tr>
<td>RoBERTa-Defense</td>
<td>0.732</td>
<td>51.4</td>
<td>0.643</td>
<td>44.0</td>
</tr>
</tbody>
</table>

Table 4.8: Detection performance (Recall) of BERT-Defense, GLTR-GPT2, and GROVER when fine-tuned to their most effective adaptive attack setting as indicated. ∆R is percentage change of Recall from Recall before training to Recall after training.

<table>
<thead>
<tr>
<th>Defenses</th>
<th>Adaptive Attack</th>
<th>Recall Before</th>
<th>Recall After</th>
<th>∆R</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT-Defense</td>
<td>Priming Token 0</td>
<td>44.9</td>
<td>85.8</td>
<td>+91.1</td>
</tr>
<tr>
<td>GLTR-GPT2</td>
<td>Top-k 160</td>
<td>42.8</td>
<td>99.2</td>
<td>+131.8</td>
</tr>
<tr>
<td>GROVER</td>
<td>Top-p 1.0</td>
<td>58.7</td>
<td>77.5</td>
<td>+32.0</td>
</tr>
</tbody>
</table>

and GLTR-GPT2 break down under text generated from different decoding strategies/parameters. When they encounter text generated using nucleus sampling with a Top-p value of 0.98, the recall of GLTR-BERT and GLTR-GPT2 degrades by 90.0% and 97.6%, respectively as shown in Table 4.5. This shows that token likelihood features are highly vulnerable to attacks that change the decoding strategy. With further improvements in decoding strategies that allow more diverse text to be sampled from LMs, such defenses will only be more susceptible to these attacks.

Evasion by varying the number of priming tokens We test the defenses on text generated using a varying number of priming tokens. In Table 4.6, for each defense, we
present results for the most effective attack configuration. Similar to the previous adaptive attack (changing decoding strategy), we use $\Delta R$, $\Delta GRN$ to measure attack success. Our findings are as follows:

**Finding 6:** Defenses trained on conditionally generated text are not able to detect unconditionally generated text. BERT-Defense which is originally trained on conditionally generated text (with a single priming token), shows over 47.7% degradation in $\Delta R$ when tested on unconditionally generated text (i.e., with 0 priming tokens). All the other defenses are trained on unconditionally generated text, and show significantly less degradation, compared to BERT-Defense. We believe that this is because defenses trained on conditionally generated text learn a narrow distribution of synthetic text. We can think of LMs as being in a particular state space at each time-step. The priming tokens might lead the model into a particular state space which the model might not have reached on its own due to those ‘priming tokens’ being low probability tokens, and thus less likely to be sampled in an unconditional setting. Therefore, conditionally trained models might learn a different and narrower distribution of synthetic text and not generalize well to unconditionally generated text.

**Evasion by adversarial perturbations.** To test DFTFooler, we use a random sample of 1000 synthetic articles from the original test set of each defense, that were correctly classified. Attack success is measured using the Evasion Rate or ER metric (Section 4.3.2). Higher ER indicates higher attack success. In addition, our attack requires preservation of semantic content. By design, our perturbation scheme achieves a USE [184] semantic similarity score $\geq 0.7$, similar to TextFooler. We present our results using a small number of perturbations, i.e., $N = 10$. The classifier’s context window size is 512 tokens, so perturbing 10 words (or less) is a small amount of perturbation. Table 4.7 presents the results for DFTFooler, and the baseline schemes (TextFooler and random perturbations). Our findings are as follows:
**Finding 7:** DFTFooler can successfully generate adversarial samples without requiring any information about the defense. From Table 4.7, we see that DFTFooler achieves significant evasion rates ranging from 23.2% to 91.3% for all the defenses, except BERT-Defense. DFTFooler with BERT and GPT2-XL as the backend also outperforms the random perturbation attack setting for all defenses. While TextFooler outperforms DFTFooler for all defenses, it is important to note that TextFooler makes a large number of queries to the defense model to craft more effective samples whereas DFTFooler does not require queries to the model. Also, the average GRUEN scores of samples with random perturbations are comparable to the GRUEN scores of DFTFooler with GPT2-XL backend—across all defenses, the average absolute difference between GRUEN scores is only 0.014.

**Finding 8:** DFTFooler using a bidirectional LM as backend, provides more effective adversarial samples. In Table 4.7, DFTFooler has higher ER when it uses BERT as the backend model compared to GPT2-XL, against 4 out of 6 defenses. More specifically, DFTFooler with BERT backend shows percentage increase in ER ranging from 35.8% to 95.0% compared to random perturbations. In other words, using bidirectional context to compute token probabilities, provides better estimation of important words to be replaced. That said, we observe a slightly higher hit on GRUEN score for DFTFooler with the BERT backend compared to when GPT2-XL is used as the backend. We suspect that this is because the GRUEN score is computed, in part, by a pre-trained BERT model [173]. There is likely an overlap between the words DFTFooler chooses to perturb and the words the GRUEN metric assigns more importance to for computing sub-scores that utilize a pre-trained BERT. Figure A.3 (Appendix) shows the GRUEN score distribution (CDF) for the different attacks.

**Finding 9:** Increasing the number of word perturbations will improve the Evasion Rate but degrade text quality. The attack performance of DFTFooler and random perturbations in Table 4.7 are obtained with 10 word perturbations. To understand the impact of the
of perturbations, we experiment with a different number of word perturbations for BERT-Defense and FAST and present the results in Figure A.4 in Appendix. We observe a clear trend that for both DFTFooler and random perturbations, increasing the number of perturbations will result in a higher evasion rate, but at the cost of increased degradation in the GRUEN score.

**Adversarial training to enable robustness against adaptive attacks.** We investigate whether a defender can recover from an adaptive attack via *adversarial training*, i.e., by training a defense on a set of known adversarial samples to build resilience against similar adversarial samples. We start by studying recovery from adaptive attacks that change the text generation process. We fine-tune BERT-Defense, GROVER and GLTR-GPT2 on new samples generated from their most effective adaptive attack setting (Tables 4.5 and 4.6). We use 1,000 new articles each for both the synthetic and real class for adversarial training. We then evaluate the adversarially trained models against their original adaptive attack dataset. As shown in Table 4.8, we observe that the fine-tuned BERT-Defense, GLTR-GPT2 and GROVER achieve 91.1%, 131.8% and 32.0% increase in $\Delta R$, respectively. Therefore, the fine-tuned models are able to recover from the attack.

Next, we explore adversarial training to recover from our DFTFooler attack. We use RoBERTa-Defense for this experiment. From our dataset of 1,000 adversarial samples from DFTFooler, we use a random set of 500 samples for adversarial fine-tuning, and test the adversarially trained RoBERTa-Defense on the remaining 500 adversarial samples. DFTFooler becomes ineffective as its evasion rate drops from 51.4% on RoBERTa-Defense (in Table 4.7) to 1.6% on the adversarially fine-tuned RoBERTa-Defense.

Fine-tuning a defense towards a specific attack is effective, but the defender may have to

---

7Since GLTR-GPT2 is not DNN-based, we instead train a new GLTR-GPT2 model from scratch with samples generated under the attack setting.
frequently adapt their defenses against newer adaptive strategies or their variants. On the other hand, by using robust semantic features, one can potentially build in resilience in an attack-agnostic way.

4.6 Discussion

Deepfake text detection vs. deepfake image detection. Deepfake image detection and deepfake text detection both aim to detect synthetic content generated by deep generative models. That said, these two fields hardly share the same set of technical methodologies in generating and detecting synthetic content due to the discrete nature of text. Similar to the text domain, researchers have also demonstrated impressive performance in detecting deepfake images [185, 186]. This has prompted work investigating the adversarial robustness of deepfake image detectors [187, 188, 189]. In a similar way, our work is the first to systematically explore real-world performance and adversarial robustness of deepfake text detectors. While the image domain has primarily focused on adversarial perturbations, our work also explores low-cost evasion strategies that change the content generation process. Note that the adversarial attacks studied for the image domain are not directly applicable to the text domain. If we compare detection schemes in the two domains, more progress has been made on understanding the use of semantic features for deepfake image detection [46, 124, 190, 191]. An example is a method that looks for eye blinking artifacts [192] to detect deepfake images. However, in the text domain, FAST is the only approach that leverages semantic features (entity-based), and there is scope for more work to be done in this direction.

Further exploring semantic layer methods for synthetic text detection. Producing semantically consistent text is still a challenging task for language models [61]. Therefore,
we expect to find differences in the semantic information embedded in synthetic text when compared to real text. In this study, we show that FAST leverages such imperfections in the semantic structure of the text to differentiate between synthetic and real text. Using semantic information will also raise the cost of producing synthetic text that can bypass such defenses. In practice, the attacker may want to preserve the semantic content, e.g., spreading vaccine disinformation, while creating evasive samples, thus making it harder to evade detection. One direction for future work is to leverage knowledge graphs to extract richer semantic features.

**Real-world deployment of synthetic text detection.** In real-world detection scenarios, false positives in detection can damage the user experience and should be avoided as much as possible. The defender can further enhance the detection performance in two ways — (1) customize a precision-optimized classifier for identifying synthetic text, and (2) employ an ensemble approach where multiple detection schemes vote for the final decision. Besides, different detection schemes have their limitations. The defender should choose the best detection scheme based on the real-world detection scenario. Our evaluation demonstrates that FAST is the most robust detection tool out of the 6 defenses we studied, but its raw detection performance is not as high as GLTR-GPT2 and BERT-Defense. Also, detection schemes are designed to be trained on different topic domains, e.g., open-domain or news domain text. Although FAST is the most robust detection scheme, it was trained only for detecting news articles so its performance can drop drastically when applied to text from other domains. Additionally, the detection performance can be sensitive to the document length as there are more features available when the document is longer.

**Ethics.** Our work involved collecting synthetic text data from text-generation-as-a-service platforms and from Internet forums. We spent $586 to collect the articles from the services. All the services we study claim that the synthetic articles can be used for white hat SEO.
Regardless of the legal status of these services, the benefits gained from understanding how well the state-of-the-art defenses perform on synthetic text generated from them outweighs the potential harms arising from injecting money into these services. Our work proposes as well as evaluates existing defenses against adversarial inputs. This was done in a controlled lab setting, and no deployed models were attacked in this process.

4.7 Conclusion

To the best of our knowledge, the work in this chapter presents the first systematic evaluation of deepfake text defenses to assess their real-world applicability. We evaluated state-of-the-art synthetic text defenses on real-world datasets and against adaptive attackers. We find that open-domain detection schemes fail to generalize to In-the-wild synthetic text, and that most defenses are not robust under adversarial settings. We also presented DFTFooler, a novel adversarial sample crafting scheme, that can degrade the performance of existing defenses without requiring any queries to the victim model nor a surrogate classifier. Our detailed analysis of the most robust defense (FAST) indicates that utilizing semantic information in the text samples can lead to better robustness and generalization performance in the wild.
Chapter 5

NoiseScope: Towards Robust Detection of Synthetic Images

5.1 Introduction

Our study in Chapter 3 suggests a urgent need to develop detection schemes with better generalization performances. In this chapter, we propose an unsupervised, content-agnostic detection scheme for detecting synthetic images, called NoiseScope (Objective 3 of this thesis). NoiseScope does not require any fake images, or knowledge of the generative models attackers used. NoiseScope works for any type of high-level image content as well as multiple types of GANs.

Most prior work on detecting fake (GAN-generated) images are supervised methods that require a priori access to fake images, or their generative models [193]. However, supervised schemes usually do not generalize well to datasets they are not trained on, and access to fake images for training can be limited in practice. Instead, we focus on advancing the state-of-the-art in blind detection of fake images. Our scheme NoiseScope can accurately detect fake images without requiring any a priori access to fake images or their generative schemes for training. Particularly, we focus on synthetic images generated by GANs, which is one of the state-of-the-art methods for generating photorealistic images.
5.1. **Introduction**

Our work is inspired by prior work in camera fingerprinting [194, 195, 196, 197], and includes the following key contributions: ① Similar to images produced by cameras, we find that fake images contain unique low-level noise patterns that are tied to the GAN model that generated it. Such patterns are correlated with the deconvolution layers of GANs. ② We present the design and implementation of *NoiseScope*, a blind detection scheme that leverages unique patterns in fake images left by the generative model. Given a test set with unknown number of fake and real (produced by camera) images, *NoiseScope* extracts any available model fingerprints or patterns that identify a GAN and uses the fingerprint to detect fake images in that set. In contrast to supervised schemes, our method is agnostic to the type of GAN used, and is also effective when the test set contains images from multiple GANs. Our method also works for any type of high-level image content, as it only extracts low-level noise patterns. ③ We evaluate *NoiseScope* on 11 diverse synthetic image datasets, created using 4 high quality GAN models. *NoiseScope* can detect fake images with up to 99.68% F1 score. ④ Lastly, we extensively evaluate *NoiseScope* against a variety of countermeasures by assuming an attacker who is aware of *NoiseScope*’s detection pipeline.

Considering the rate at which new generative models (GANs) are being proposed, supervised learning strategies will likely tip the arms race in favor of the attacker. Therefore, there is an urgent need to advance blind detection schemes that can, in theory, work against a wide range of GAN models. The source code of *NoiseScope* is available at GitHub\(^1\), and we hope *NoiseScope* inspires more work on synthetic image detection.

\(^1\)https://github.com/jmpu/NoiseScope
5.2 Background & Related Work

In this work, we focus on images, and consider synthetic images as those produced by machine learning algorithms, more specifically, GANs. GAN models are capable of producing high-quality images. In fact, humans find it hard to distinguish synthetic images from real images [105]. We encourage the reader to look at the following website \(^2\) that presents a new fake image on each page refresh, created using the StyleGAN [6]. In the rest of the Chapter, we will interchangeably use the term synthetic or fake image to refer to such content. Images produced by traditional imaging devices (cameras) are called real images.

5.2.1 Synthetic Image Generation Methods

Synthetic images are primarily enabled by the family of deep generative models. Given a training set of images, a generative model can learn the distribution of the data and produce new images with variations. Two popular approaches include Generative Adversarial Networks (GANs) [9], and Variational Autoencoders (VAEs) [198]. We focus on synthetic images generated by GANs, because GANs have shown impressive performance over the last few years. The basic knowledge of GANs has been introduced in Section 2.1.

It is important to note the role of deconvolution or upsampling layers in generative models. An integral component of most generative models, including VAEs and GANs, is a transposed convolution layer [199], commonly referred to as deconvolution or upsampling layer. This is fundamental to building high quality generators, as it allows for learnable upsampling from a lower dimensional vector space. In Section 5.6.1, we demonstrate how the deconvolution layers can leave distinct patterns in the “noise space” of an image, which enable us to distinguish between fake and real images.

\(^2\)https://thispersondoesnotexist.com/
5.2. Background & Related Work

Choice of GANs. Experimenting with the large number of GANs in the literature [73, 200, 201, 202] would be impractical. Instead, we focus on certain key models that significantly raised the bar for different types of image generation tasks. We focus on synthetic images generated by CycleGAN [10], PGGAN [5], BigGAN [11], and StyleGAN [6]. These 4 GANs are briefly discussed below.

CycleGAN [10]. CycleGAN advanced the state-of-the-art in image-to-image translation when it was proposed, improving over the previous method Pix2Pix [75]. CycleGAN can translate an image from one domain to another, e.g., turn an image of a horse to a zebra. Compared to Pix2Pix, CycleGAN does not require paired images for training, which is a huge advantage, as paired images (for two domains) are hard to obtain. From a threat perspective, image-to-image translation schemes can be used by an attacker in many ways, e.g., swap faces in an image, insert a new person or object into a scene.

PGGAN [5]. In 2018, PGGAN demonstrated a huge improvement in image quality. Previously, GANs were not capable of generating high resolution images in high quality. The basic idea is to progressively generate higher resolution images, by starting from easier low-resolution images. PGGAN progressively grows both the generator and discriminator by adding new layers as training progresses to produce higher resolution images with more details. PGGAN is able to produce photo-realistic images at high resolutions, up to 1024x1024. At the time, PGGAN produced the highest Inception score of 8.80 for CIFAR10 [203], and also created a high-quality version of the CelebA dataset [204] at 1024x1024 resolution.

BigGAN [11]. Soon after the introduction of PGGAN, Brock et al. introduced BigGAN, an attempt to scale up conditional GANs to develop high quality images on a large number of domains. BigGAN uses a variety of techniques to improve GAN training and image quality, including an increased batch size, increase in number of layer channels, and shared embeddings for batch normalization layers in the generator. One feature of BigGAN is the
“truncation trick”, whereby using a hyperparameter called the *truncation threshold*, one can control the trade-off between image fidelity and variety. A higher truncation threshold leads to higher variety in generated images, while a lower threshold boosts fidelity. When evaluated on the ImageNet dataset, BigGAN produced a very high Inception Score of 166.5, outperforming SAGAN [202] which had the previous best Inception Score of 52.52.

**StyleGAN** [6]. In 2019, Karras et al. released StyleGAN, an improvement to PGGAN which incorporates a complete redesign of the generator architecture. The generator no longer receives as input a random noise vector, but a *style vector* generated by a noise-to-style CNN mapping network. Other changes include a change in the upsampling technique, and addition of noise to feature maps in the convolutional layers. This redesign allows fine-grained control over style of the generated image, while simultaneously retaining and improving upon the high-quality output of PGGAN. Having fine-grained control over style of the generated image is important from an attack perspective.

### 5.2.2 Synthetic Image Defenses

Prior work on synthetic image detection has investigated both *supervised* and *blind* detection schemes. In a supervised scheme, the defender has access to both real and fake content (or has knowledge of the generative model) and can use this labelled data to train a classification algorithm. In blind detection, the defender has no a priori access to fake content (or generative methods employed), and only has access to real content. Most prior work has employed supervised schemes, and limited efforts have been made towards advancing blind detection schemes. Consequently, the performance of such schemes has evolved considerably, and the release of effective DNN models that facilitate improved feature learning has only furthered this progress. However, the dominant performance of supervised learning comes with notable caveats.
In practice, it is hard to obtain a priori access to fake content, or knowledge of the generative model. However, even with such presumption, supervised schemes suffer from a fatal inability to generalize. More specifically, we observe that such schemes are designed for and thus trained on a limited set of synthetic images (generated by specific generative models), and do not generalize well when evaluated against synthetic images produced by other models. In Section 5.6.2, we demonstrate this inability to generalize.

A blind detection scheme aims to solve this problem by not requiring a priori access to fake images for training, while being able to detect fake images from a wide variety of sources (GANs). An accompanying difficulty of blind design is a potential decrease in performance when compared to existing supervised classifiers. *NoiseScope* aims to advance the state-of-the-art for blind detection schemes by offering a performant detection scheme. *NoiseScope* complements the supervised detection schemes from prior work, allowing for potentially hybrid ensembles that feature the best of both worlds.

**Supervised methods.** One set of approaches focus on building a supervised classifier with input image features crafted from specific vector spaces. Examples include Marra et al.’s [193] proposition of using raw pixel and conventional forensics features, and Nataraj et al.’s [205] extraction of pre-computed RGB pixel co-occurrence matrices to capture distinguishing features. Feature engineering in multiple color spaces has also been explored. Li et al. proposed a feature set capturing disparities in color spaces between real and fake images and then using such features to perform classification [206].

Prior work observed that, similar to cameras, GANs also leave unique fingerprints in the images. Marra et al. [54] extracted GAN model fingerprints using techniques from the camera fingerprinting literature [194, 196], and implemented a supervised scheme to detect fake images. Another approach by Yu et al. [137] used a supervised deep learning scheme to learn GAN model fingerprints, and attribute images to GANs. Yu et al.’s approach primarily
focused on attributing fake images to different GANs. Albright and McCloskey [207] also worked on attributing images to GANs by leveraging generator inversion schemes [208]. Our work also aims to identify model fingerprints to detect fake images but does so in a blind manner.

Domain-specific inconsistencies can also be used to detect synthetic images. Yang et al. [115] focused on deep fakes generated by splicing synthesized face regions into a real image. They show that such splicing introduces errors when 3D head poses are estimated from the fake images. An SVM-based classifier is trained to learn such errors to distinguish between real and fake images.

Other supervised approaches leverage DNNs to automatically extract features relevant for classification. Mo et al. [42] developed a CNN-based model to detect face images generated by PGGAN. Rossler et al. compared 5 CNN-based classification architectures by learning extracted face regions [44]. Tariq et al. [43] propose using ensembles of various CNN-based classifiers to detect GAN generated face images. Concurrent to our work, Wang et al. [209] proposed a classifier based on the ResNet-50 architecture that is trained on a large number of fake images from a single GAN, with carefully chosen data augmentation schemes. Afchar et al. [112] designed MesoNet based on Inception blocks to detect synthetic images showing impressive performance. We compare NoiseScope with MesoNet in Section 5.6.2. Also note that the above approaches have a fundamental weakness—they can be evaded by the attacker, by re-training the GAN using the defender’s DNN model as the discriminator.

**Blind detection.** Li et al.’s work [206] proposes a blind detection scheme. The idea is that GANs fail to learn correlations among color components in the RGB space, which results in inconsistencies when examined in other color spaces, namely HSV, and YCbCr. They train a one-class SVM classifier on features based on color statistics of HSV and YCbCr color spaces of real images to detect fake images. The intuition is that fake images will be
5.3 Threat model

flagged as anomalies in the color (feature) space. We compare our approach against Li et al.’s approach in Section 5.6.2.

Zhang et al. [210] uses real data to train “AutoGAN”, a component that aims to simulate a GAN generator. The idea is to first generate fake images using AutoGAN, and then train a supervised classifier on the newly synthesized fake images and real images to detect other fake images. Unfortunately, its performance largely depends on the architecture of AutoGAN’s generator. Results show significant drop in performance when tested on fake images from a GAN that uses a different architecture compared to AutoGAN’s generator.

5.3 Threat model

Attacker model. Attacker aims to generate high quality convincing synthetic images using deep generative models (GANs). Our focus is on fake images that are entirely produced by a generative model (GAN). Fake images created by image forgery techniques such as replacing or adding content in real images (e.g., face swapping [211]) are not considered. In Section 5.6, we consider an attacker who is unaware of our defense scheme. Later, in Section 5.7, we consider an attacker who is aware of our defense scheme pipeline and employs a variety of countermeasures against NoiseScope.

Defender model. Defender has no a priori access to fake images, and no knowledge of the generative scheme used by the attacker. Defender is provided a test set of images, out of which an unknown number of images are fake or real, and the goal is to flag fake images. Defender also makes use of a reference set of real images, which is only used to calibrate certain detection parameters of NoiseScope. For example, if Facebook wants to detect synthetic profile pictures, they can prepare a test set containing profile pictures (say faces), and the reference set would include a set of known real profile pictures. Our method
is designed to be content agnostic, and therefore the test set can be based on images from different content categories.

5.4 Our approach: NoiseScope

5.4.1 Method Basics

We do not rely on content-specific features that capture semantic or statistical inconsistencies, *e.g.*, finding abnormalities in human face images. Such defenses will not survive for long, given the rate at which GANs are advancing and producing photorealistic images. Instead, *we aim to identify patterns that are not tied to the semantic aspects of image contents but allow us to differentiate between real and fake images.*

We borrow ideas from the rich literature of camera fingerprinting schemes [194, 195, 196, 197]. Each imaging device (*e.g.*, camera) leaves a unique and stable pattern in each image due to imperfections in various stages of the image acquisition process. Such patterns known as photo-response non-uniformity (PRNU) patterns have been used to fingerprint cameras or image acquisition devices [194]. Naturally, this first raises the question whether GAN-based image generators would leave a unique and stable “artificial” pattern in the generated images. In fact, preliminary work by Marra et al. shows that such stable patterns do exist in GAN generated images [54]. These patterns are present, regardless of the content in the image, be it images of human faces, objects, animals or landscapes. Secondly, we would expect those patterns to look different because GAN models share no similarity with camera-based image acquisition pipelines. We leverage these ideas and propose a complete blind detection scheme that can accurately flag fake images with any type of content. Next, we explain techniques from the camera fingerprinting literature that we leverage to fingerprint generative models.
Leveraging model fingerprints for detection. Consider a set of images, $I_i$, where $i \in \{1, \ldots, N_p\}$ generated by a GAN. Our goal is to estimate a stable pattern left by the GAN, that is unrelated to the semantics of the image content. The first step is to separate the high-level content from the image, and estimate the noise residual $R_i$. The high-level content is estimated by applying an appropriate denoising filter $f(I_i)$. The noise residual is then computed as, $R_i = I_i - f(I_i)$. Now the assumption is that the noise residual $R_i$ contains the stable pattern or the fingerprint $F$, and some random noise $N_i$, i.e., $R_i = F + N_i$. Therefore, one can estimate the fingerprint by averaging the residuals:

$$\bar{F} = \frac{1}{N_p} \sum_{i=1}^{N_p} R_i$$ \hspace{1cm} (5.1)

In practice, the larger the $N_p$, the additive noise component tends to cancel out, and we obtain a more accurate fingerprint. According to prior work, it is possible to estimate a reliable fingerprint using at least 50 images, i.e., $N_p > 50$ [212].

Figure 5.1 shows camera and GAN fingerprints computed using the above method for $N_p = 100$ images. Note that model fingerprints look very different from device fingerprints. In the
case of CycleGAN and PGGAN, there is a noticeable checkerboard pattern. This observation is further discussed later in Section 5.6.1.

If model and device fingerprints are so dissimilar, can we use the model fingerprint to distinguish between fake and real images? To answer this, we take a set of face images composed of 200 real (taken by Canon EOS 70D) and 200 fake images from StyleGAN. The fingerprint for StyleGAN, say $F_{GAN}$, is computed using a separate set of 100 face images. Next, to attribute images in this set to the device or the GAN, we compute the correlation between the model fingerprint and residual of each image ($R_{i}$) in the test set, i.e.,

$$\rho_{F_{GAN},i} = corr(\bar{F}_{GAN}, R_{i}).$$

For a given image, if this correlation is higher than a certain threshold $T_{c}$, it is classified as a fake image, or real, otherwise. A correlation measure called Peak to Correlation Energy (PCE) (described next) is used. Figure 5.2 shows the histogram of correlation values for all images in the set (both fake and real). The fake images can be easily separated from the real images based on the PCE values.

PCE metric [213]. PCE is a similarity metric to compare two discrete signals. It is computed

![Figure 5.2: Histogram of PCE correlation between model fingerprint and images (fake and real).](image-url)
as the ratio between squared normalized correlation and sample variance of circular cross-relations. The PCE implementation\(^3\) that we use carries the sign of normalized correlation peak (can be negative). A high positive value of PCE denotes a high correlation. Other than PCE, there are other correlation measures, such as Pearson correlation [214], and quotient correlation [215]. Compared to other metrics, PCE is a more stable metric that can be used with images from devices with different resolutions and sensor types [213]. We find PCE to be suitable for GAN images as well.

*To summarize, if an accurate model fingerprint is available, it is straight-forward to detect fake images.* However, in a blind setting we have no knowledge of fake images or the associated GAN(s) to compute the model fingerprint.

**Key challenges in designing NoiseScope.**  ① The first challenge is estimating a model fingerprint. It is hard to estimate a model fingerprint from a single image in a blind setting (Equation 5.1 requires averaging over multiple images). While prior work, NoisePrint [216] provides a supervised (CNN-based) learning scheme to extract camera fingerprint from a single image, such methods are not applicable in a blind setting. Instead, *our idea is to extract fingerprints from the test set itself in an unsupervised manner.* We propose an image clustering scheme that identifies subsets of images belonging to the same source (device or model), and estimate fingerprints based on those subsets. Our method should work as long as a certain minimum number of fake images (enough to reliably estimate a fingerprint) are present in the test set. ② Once a fingerprint is extracted from the test set, how do you tell whether it is a model fingerprint or a device fingerprint? To achieve this, we propose a fingerprint classification module based on anomaly detection to identify model fingerprints. ③ Method should be agnostic to the specific GAN used, and should also work when test set contains fake images from different GANs. To address this, our clustering scheme is

\(^3\)http://dde.binghamton.edu/download/camera_fingerprint/
designed to be agnostic to the GAN(s) used, and is able to extract available fingerprints, even from multiple models. 4) Method should work for images with any type of high-level content (images of faces, animals, objects, etc.) To address this challenge, we use residual image extraction schemes that can effectively suppress high-level content.

5.4.2 Detection Pipeline

*NoiseScope* includes 4 main components: (1) Noise residual extractor, (2) Fingerprint extractor, (3) Fingerprint classifier, and (4) Fake image detector. Figure 5.3 provides an overview of *NoiseScope*’s detection pipeline. The first component prepares the noise residuals, the second component finds all available fingerprints in the test set. The third component identifies model fingerprints among the identified fingerprints, and the fourth component uses the model fingerprints to flag fake images.

**Noise Residual Extractor.** This first step suppresses high level image content and extracts the noise residual (which contains the fingerprint). We use the Wavelet Denoising filter [217] to extract the noise residual for each image in the test set. Prior work recommends this as one of the best filters to suppress high-level content [218, 219]. However, there is no perfect filter, and we do notice Wavelet denoising also leaking image contents into the noise residual in some cases. If there is heavy content leakage, then fingerprint extraction (next step) becomes harder. But in general, Wavelet denoising tends to perform well. In Section 5.6.2, we analyze the impact of different denoising filters on detection performance.

**Fingerprint Extractor and Fingerprint Classifier.** The second step extracts model fingerprints from the test set. The fingerprint extractor finds all available fingerprints (model or device) from the test set, and the fingerprint classifier identifies those that are model fingerprints. To extract fingerprints, we resort to unsupervised clustering by starting with
5.4. Our approach: NoiseScope

Figure 5.3: An illustration of NoiseScope detection pipeline: (a) Noise Residual Extractor, (b) Fingerprint Extractor via Clustering, (c) Fingerprint Classifier, (d) Fake Image Detector. The individual noise residuals computed from step 1. Our goal is to group images belonging to the same source (model or device), and then use each group of images to build a fingerprint (using Equation 5.1).

But there is a challenge—it is hard to cluster images in the residual space. Residual images contain random noise along with the fingerprint pattern. So even images from the same source (model or device) will not always show high correlation [220]. All our efforts to cluster images in the residual space resulted in impure clusters, i.e., clusters with mix of fake and real images. An impure cluster would give us an inaccurate fingerprint which is not useful.

To address this challenge, we use a different strategy: Instead of completely clustering images in the residual space, we use an incremental clustering strategy, similar to bottom-up hierarchical clustering. The idea is to mostly compute correlations between fingerprints (which has less random noise), and less between residuals. Initially, each residual image forms its own cluster. Next, any pair of residuals with PCE correlation higher than a threshold $T_{merge}$ is merged into a new cluster. Each time a cluster is updated, we compute a fingerprint (using cluster members), and two clusters are merged if the PCE correlation between their fingerprints is greater than $T_{merge}$.

$^4$We update clusters such that each image is only present in one cluster.
computing correlations using fingerprints, we reduce the risk of random noise impacting our correlation estimates. The larger a cluster becomes, the more the random noise will vanish when we estimate the fingerprint. The PCE threshold for merging, $T_{merge}$, is chosen such that clusters mostly end up being pure, i.e., contain all fake images or all real images. If $T_{merge}$ is too low, clusters end up being impure, and we obtain inaccurate fingerprints which may not be useful for detecting fake images in the next step. If $T_{merge}$ is too high, we run the risk of not finding sufficiently large clusters or even no clusters to estimate an (accurate) fingerprint. In Section 5.5.2, we discuss how we estimate $T_{merge}$.

The clustering process stops when no more clusters can be merged using the threshold. However, to reduce the computational complexity, we propose to stop clustering early when we find cluster(s) with $size > T_{size}$. Recall that we only require a small number of images ($> 50$) to estimate a fingerprint. Once we stop clustering, we pass any fingerprint computed using clusters greater than size 50, to the Fingerprint Classification component to decide whether it is a model or device fingerprint. If no model fingerprints are found, we continue the clustering process again (in case it was stopped early), until no more merging is possible. Fingerprints found at the end are again passed to the fingerprint classifier.

**Fingerprint Classifier.** The fingerprint classifier is used to identify model fingerprints. Key challenge here is that we have no a priori knowledge of model fingerprints. Our intuition is that GAN fingerprints stand out as anomalies when compared to device fingerprints in some feature space. Recall the checkerboard pattern in GAN fingerprints shown earlier in Figure 5.1. *We observe that model fingerprints tend to have different texture patterns when compared to device fingerprints.* To capture texture features from a fingerprint, the well-known Haralick texture features [221] are used. Haralick texture features capture 14 statistical features from the Gray Level Co-Occurrence Matrix (GLCM), which in turn captures the number of repeated pairs of adjacent pixels. For the anomaly detection scheme,
we use the Local Outlier Factor (LOF) scheme [222]. Input to LOF are Haralick features extracted from fingerprints computed over (real) images in the reference set. Once trained, the fingerprint classifier can take any fingerprint as input (after extracting Haralick features), and check whether it is an anomalous sample. A fingerprint is considered to be a model fingerprint if this component marks it as an anomalous fingerprint.

Fake Image Detector. In the last step, we take all the model fingerprints detected in step 2 and compute the PCE correlation between each fingerprint and all residual images in the test set (using Equation 5.2). If correlation is higher than a threshold, the image is flagged as a fake. An image is considered to be fake, if it is flagged by at least one model fingerprint. The reference set is used to calibrate the correlation threshold. The threshold is chosen such that a model fingerprint when correlated with real images in the reference set, should not flag any of them. A high threshold will improve precision, while underestimating the threshold will bring down precision, and improve recall.

Method Scalability. The clustering part is the most computationally heavy step of the system. In the worst case, the clustering could run for \( \log(n) \) iterations, where \( n \) is the number of images in the dataset. Each iteration requires sorting of the pair-wise PCE correlation, with an \( O(n^2 \cdot \log(n^2)) \) complexity. This gives the entire clustering part a complexity of \( O(n^2 \cdot \log^2(n)) \). Improvements can be made to scale the Fingerprint Extractor for large-scale classification. Pairwise PCE correlations can be computed in parallel to speed up the construction of the PCE correlation matrix. As \( n \to \infty \), the, pipeline can, as a whole, also be run in parallel on subsets of the \( n \) images. A final instance of the Fingerprint Extractor can be used to agglomerate the clusters obtained from these parallelized Fingerprint Extractors. We can also leverage prior work on distributed/parallel hierarchical clustering [223, 224, 225].
## 5.5 Experiment setup

We discuss the experimental settings used to evaluate detection performance of *NoiseScope*.

### 5.5.1 Real and Fake Image Datasets

For each dataset, we discuss the GAN used to generate the fake images in the test set, and how real images for the test and reference sets are collected. Each dataset includes 2,500 fake images, and out of the real images we collected for each dataset, 2,000 random real images are used to build the reference set. Table 5.1 presents statistics of the 11 datasets covering 4 GAN models, used for our evaluation.

**Table 5.1: Basic information of 11 synthetic image datasets evaluated in Section 5.6.2.**

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Content</th>
<th>Fake Source</th>
<th>Real Source</th>
<th>Resolution</th>
<th># Fake images</th>
<th># Real images</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleGAN-Face1</td>
<td>Human face</td>
<td>StyleGAN [226]</td>
<td>FFHQ [6]</td>
<td>1024x1024</td>
<td>2,500</td>
<td>8,000</td>
</tr>
<tr>
<td>StyleGAN-Face2</td>
<td>Human face</td>
<td>StyleGAN [2]</td>
<td>FFHQ [6]</td>
<td>1024x1024</td>
<td>2,500</td>
<td>8,000</td>
</tr>
<tr>
<td>StyleGAN-Bed</td>
<td>Bedroom</td>
<td>StyleGAN [227]</td>
<td>LSUN [228]</td>
<td>256x256</td>
<td>2,500</td>
<td>3,098</td>
</tr>
<tr>
<td>BigGAN-DogLV</td>
<td>French bulldog</td>
<td>BigGAN [4]</td>
<td>ImageNet [229], Flickr [230]</td>
<td>256x256</td>
<td>2,500</td>
<td>5,309</td>
</tr>
<tr>
<td>BigGAN-DogHV</td>
<td>French bulldog</td>
<td>BigGAN [4]</td>
<td>ImageNet [229], Flickr [230]</td>
<td>256x256</td>
<td>2,500</td>
<td>5,309</td>
</tr>
<tr>
<td>BigGAN-BurgLV</td>
<td>Cheeseburger</td>
<td>BigGAN [4]</td>
<td>ImageNet [229], Flickr [230]</td>
<td>256x256</td>
<td>2,500</td>
<td>4,390</td>
</tr>
<tr>
<td>BigGAN-BurgHV</td>
<td>Cheeseburger</td>
<td>BigGAN [4]</td>
<td>ImageNet [229], Flickr [230]</td>
<td>256x256</td>
<td>2,500</td>
<td>4,390</td>
</tr>
<tr>
<td>PGGAN-Face</td>
<td>Human face</td>
<td>PGGAN [231]</td>
<td>FFHQ [6]</td>
<td>1024x1024</td>
<td>2,500</td>
<td>8,000</td>
</tr>
<tr>
<td>PGGAN-Tower</td>
<td>Tower</td>
<td>PGGAN [231]</td>
<td>LSUN [228]</td>
<td>256x256</td>
<td>2,500</td>
<td>1,187</td>
</tr>
</tbody>
</table>

**StyleGAN-Face1.** This is a dataset of *human face* images, at 1024x1024 resolution. Fake images are generated by StyleGAN, trained on the Flickr-Faces HQ (FFHQ) dataset of human faces [6]. Fake images are collected from the official NVIDIA StyleGAN GitHub repository [226]. We collected 8,000 real images for the test and reference sets by randomly sampling from the FFHQ dataset.

**StyleGAN-Face2.** Recently Generated Media, Inc. [233] released 100,000 StyleGAN generated face images [2]. Their aim is to provide royalty-free stock images using AI [234]. The GAN was trained using a proprietary dataset of 29,000+ curated photographs of 69 models. The images are photorealistic, and it is unclear if these images have been further
post-processed to improve image quality. Fake images are sampled from this dataset. We randomly sampled 8,000 real images from the FFHQ dataset.

**StyleGAN-Bed.** This includes images of bedroom scenes at 256x256 resolution. Fake images are generated by NVIDIA with a StyleGAN trained on the LSUN Bedroom dataset \[228\] of bedroom scenes. Fake images are obtained from the official NVIDIA GitHub repository \[227\]. We randomly sampled 3,098 real images from the LSUN Bedroom dataset.

**BigGAN-DogLV and BigGAN-DogHV.** Datasets include images of french bulldogs at 256x256 resolution. Fake images are generated using a BigGAN-deep instance \[11\], trained on the ImageNet dataset, and obtained online \[4\]. BigGAN provides an inference-time truncation parameter to vary the trade-off between fidelity and variety (see Section 5.2.1). We generate two sets of fake images, BigGAN-DogLV and BigGAN-DogHV at truncation settings of 0.2 and 0.86, respectively. BigGAN-DogLV has images with lower variety, while BigGAN-DogHV has images with higher variety. Real images are partially sourced from ImageNet. However, ImageNet only provides 1,300 images for this image class. We further collected additional real images by crawling Flickr.com, giving us a total of 5,309 real images.\(^5\)

**BigGAN-BurgLV and BigGAN-BurgHV.** Datasets include images of cheeseburgers at 256x256 resolution, prepared using the same methodology used for BigGAN-DogLV, and BigGAN-DogHV. BigGAN-BurgLV and BigGAN-BurgHV corresponds to low and high variety fake image sets, respectively. We crawled additional real images from Flickr.com, and in total used 4,390 real images.

**PGGAN-Face.** This dataset contains images of human faces, at 1024x1024 resolution. Fake images are produced by NVIDIA with a PGGAN trained on the CelebA dataset \[5\] of

---

\(^5\)Images were curated using manual effort as well as using the ResNet50 ImageNet classifier
celebrity faces. Fake images are collected from the official PGGAN repository [231]. For real images, we sampled 8,000 images from the FFHQ dataset.

**PGGAN-Tower.** Dataset contains images of *towers*, at 256x256 resolution. The fake images are generated by NVIDIA with a PGGAN trained on the LSUN tower dataset [228] of towers. These images are collected from the official PGGAN repository [231]. We randomly sampled 4,187 real images from the Tower category of the LSUN dataset.

**CycleGAN-Winter.** Dataset contains images of *winter scenes* at 256x256 resolution. Fake images are generated using a pre-trained model available on the official CycleGAN repository [3]. CycleGAN requires input images to generate fake *translated* images (summer to winter scene translation), and only a limited number of fake images (1,187) could be generated using the data provided by the authors. To generate more fake images, we crawl Flickr.com for more input images, and generate new fake images. Real images provided by the authors are also limited (only 1,474). We thus supplement the real images for CycleGAN-Winter by crawling Flickr.com, and obtain a total of 4,594 real images.

**CycleGAN-Zebra.** Dataset contains images of *zebras* at 256x256 resolution. Fake images are generated using CycleGAN, and we follow the strategy used for CycleGAN-Winter to prepare this dataset. We collected 11,241 real images.

### 5.5.2 Configuration of NoiseScope

**Noise Residual Extractor.** We use a Wavelet Denoising filter (see Section 5.4.2) to prepare residual images. The implementation from Goljan et al is used.  

6[^aw5]: http://dde.binghamton.edu/download/camera_fingerprint/
Fingerprint Extractor. Two parameters to configure include $T_{\text{merge}}$, which decides the PCE correlation threshold to merge two clusters, and $T_{\text{size}}$ used to stop the clustering process early. $T_{\text{size}}$ is set to 150 and is observed to work well across datasets. To estimate $T_{\text{merge}}$, one approach is to use a reference set with camera identifiers. PCE correlation between fingerprints computed from the same camera can be computed, and a suitable threshold can be estimated. We lack camera identifier information in most of our datasets, and therefore use a different strategy. We assume the reference set includes images from multiple cameras and compute ‘pseudo-fingerprints’\textsuperscript{7} over random subsets (non-overlapping) of 20 images. Next, pairwise PCE correlation between these different pseudo-fingerprints are estimated. Clearly, the PCE values will not be high, as images are from different cameras. Therefore, we set $T_{\text{merge}}$ to be 99.5 percentile of this distribution, \textit{i.e.,} it should be at least larger than the correlation between pseudo-fingerprints computed over different cameras. This strategy works well in practice.

Fingerprint Classifier. We configure and train an LOF anomaly detection scheme (Section 5.4.2). If we have a reference dataset with camera identifiers, we can compute fingerprints for each camera, and use that to train the anomaly detection scheme. Lacking such data for most of our datasets, we again use the strategy used in Fingerprint Extractor, and compute ‘pseudo-fingerprints’\textsuperscript{7} using random subsets of 50 real images from the reference set (which is assumed to contain images from multiple cameras), and train the scheme using 200 such pseudo-fingerprints. This is effective because model fingerprints are still anomalous in the texture space even when compared to pseudo-fingerprints computed over multiple cameras. The parameter \textit{contamination}, which configures the error in the training set is set to $10^{-4}$, and the number of neighbors to analyze (in K-NN) is set to 30.

\textsuperscript{7}Technically they are not fingerprints as they are computed over images from different cameras.
Fake Image Detector. This component flags an image to be fake, if the PCE correlation between a model fingerprint and residual image (in test set) is higher than a threshold. To calibrate the threshold, we compute PCE correlation between a model fingerprint, and images in the reference set. Threshold is chosen such that 99.5% of the reference set images are not flagged as fake.

5.5.3 Evaluation Metrics and Baseline Method

We report average F1 score computed as the harmonic mean of Precision and Recall of the fake class, calculated over 5 random trials (unless specified otherwise).

We compare NoiseScope with the blind detection scheme proposed by Li et al. [206] (Section 5.2.2). This approach analyzes differences between real and fake images using disparities in the HSV and YCbCr color spaces. This is achieved by using features extracted from these color spaces to train a one-class SVM for anomaly detection. We abbreviate this method as CSD-SVM. The underlying assumption is that fake images will be detected as anomalies. We follow the configuration described in the paper to train CSD-SVM. A Gaussian kernel is used, and parameters are estimated via grid search. For the parameter \( \nu \), which controls the upper bound of training error, we try two values, 0.10 and 0.05. Real images in the reference set are used to train the CSD-SVM for each dataset.

5.6 Evaluation of proposed scheme

5.6.1 Analysis of Model Fingerprints

Performance of Fingerprint Classifier. For the three face datasets (StyleGAN-Face1,
5.6. Evaluation of proposed scheme

StyleGAN-Face2, and PGGAN-Face), our real dataset includes images with camera source information for 90 cameras (extracted from EXIF metadata). We first train the anomaly detection scheme on device fingerprints from 18 cameras. Next, in each trial, we test on 500 device fingerprints (extracted from the remaining 72 cameras), and 500 model fingerprints (obtained from the three face datasets). Our classifier achieves a high average F1 score of 99.2% over 5 trials (average Precision of 98.5% and Recall of 100.0%) for the detection of model fingerprints and is therefore capable of accurately detecting model fingerprints.

In the rest of the evaluation, when camera identifiers are not available, we use the strategy described in Section 5.5.2, and train the fingerprint classifier using pseudo-fingerprints computed over the reference set. Results in Section 5.6.2 show that this works well in practice.

Understanding Model Fingerprints. Why do GAN fingerprints show checkerboard patterns? The answer is tied to the deconvolution layers that are the core building blocks of GAN generators [73]. Odena et al. observed checkerboard patterns in images generated by upsampling via transpose convolution operations [235]. They attributed the checkerboard pattern to the overlap that occurs when the kernel size of the transpose convolution projection window is not divisible by the stride. The pattern is amplified when multiple transpose convolutional layers are stacked. In our fake images, we do not observe such checkerboard patterns in the high-level content, but we clearly see such patterns in the fingerprints (Figure 5.1).

To further understand the correlation between deconvolution layers and checkerboard patterns, we conduct the following experiment using CycleGAN. The transpose convolutions in the CycleGAN ResNet50 generator are found in 2 layers, with strides of 2x2 and kernel sizes of 3x3. We observe that by varying the kernel size in the second layer from 3x3 to 5x5\(^8\), we

---

\(^8\)Kernel dilation, input padding and output padding must also be accordingly changed to support the desired image output dimensions.
can alter the intensity and locality of the checkerboard pattern in the resulting fingerprint. The model fingerprints, before and after modifying CycleGAN are shown in Figure 5.4. The visible change in fingerprint textural patterns indicates a strong correlation between the fingerprint and the deconvolution operations in modern GAN generators.

5.6.2 Detection Performance

We evaluate detection performance of NoiseScope when applied to the 11 datasets discussed in Section 5.5.1.

Performance on balanced test sets. In each trial, NoiseScope is applied to a balanced test set with 500 real, and 500 fake images. Table 5.2 presents detection performance (average F1 score) for both NoiseScope, and CSD-SVM ($\nu=0.1$). NoiseScope outperforms CSD-SVM over all 11 datasets and achieves a high F1 score of over 90.1% for all datasets. Varying the $\nu$ parameter (upper bound of training error) to 0.05 for CSD-SVM shows no noticeable improvement. Given NoiseScope’s high detection performance, it is worth noting that images generated by StyleGAN, PGGAN and BigGAN are vividly photorealistic, and are difficult for humans to spot.
### 5.6. Evaluation of proposed scheme

#### Table 5.2: Performance of NoiseScope and CSD-SVM (\( \nu = 0.1 \)).

<table>
<thead>
<tr>
<th>Datasets</th>
<th>NoiseScope</th>
<th>CSD-SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleGAN-Face1</td>
<td>99.56</td>
<td>92.93</td>
</tr>
<tr>
<td>StyleGAN-Face2</td>
<td>90.14</td>
<td>67.53</td>
</tr>
<tr>
<td>StyleGAN-Bed</td>
<td>99.63</td>
<td>94.82</td>
</tr>
<tr>
<td>BigGAN-DogLV</td>
<td>99.38</td>
<td>86.94</td>
</tr>
<tr>
<td>BigGAN-DogHV</td>
<td>92.60</td>
<td>70.10</td>
</tr>
<tr>
<td>BigGAN-BurgLV</td>
<td>99.68</td>
<td>94.82</td>
</tr>
<tr>
<td>BigGAN-BurgHV</td>
<td>98.64</td>
<td>83.67</td>
</tr>
<tr>
<td>PGGAN-Face</td>
<td>99.09</td>
<td>64.07</td>
</tr>
<tr>
<td>PGGAN-Tower</td>
<td>95.93</td>
<td>91.61</td>
</tr>
<tr>
<td>CycleGAN-Winter</td>
<td>92.40</td>
<td>87.14</td>
</tr>
<tr>
<td>CycleGAN-Zebra</td>
<td>92.84</td>
<td>84.95</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Datasets</th>
<th>F1 Score (%) w/ different fake:real ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleGAN-Face1</td>
<td>97.90</td>
</tr>
<tr>
<td>StyleGAN-Face2</td>
<td>81.00</td>
</tr>
<tr>
<td>StyleGAN-Bed</td>
<td>99.50</td>
</tr>
<tr>
<td>BigGAN-DogLV</td>
<td>98.90</td>
</tr>
<tr>
<td>BigGAN-DogHV</td>
<td>89.30</td>
</tr>
<tr>
<td>BigGAN-BurgLV</td>
<td>98.60</td>
</tr>
<tr>
<td>BigGAN-BurgHV</td>
<td>97.90</td>
</tr>
<tr>
<td>PGGAN-Face</td>
<td>97.10</td>
</tr>
<tr>
<td>PGGAN-Tower</td>
<td>94.50</td>
</tr>
<tr>
<td>CycleGAN-Winter</td>
<td>88.20</td>
</tr>
<tr>
<td>CycleGAN-Zebra</td>
<td>89.90</td>
</tr>
</tbody>
</table>

Table 5.3: Detection performance (F1) on imbalanced test sets with different ratio of fake to real images.
Performance on imbalanced test sets. We apply NoiseScope on test sets with an imbalanced ratio of real vs fake images. For each dataset, we evaluate on 4 imbalanced test sets comprising different ratios of real and fake images. In each test set, the number of fake images is set to 200, and we increase the number of real images according to the desired ratio. We experiment with ratios of fake to real as 1:2, 1:4, 1:8, and 1:10. The inherent difficulty of using NoiseScope in an imbalanced setting is the presence of noisy samples among fake and real images. These are samples where content tends to leak into residuals. Therefore, such noisy fake and real images can show unexpectedly high correlation. Consequently, as the number of real images increases, the probability of a fake image cluster merging with noisy real samples increases.

Detection performance is presented in Table 5.3. Out of the 11 datasets, 7 datasets exhibit high performance of over 91.9% F1 score for all ratios (numbers shown in bold). As expected, there is a drop in performance as datasets become more imbalanced, but even at 1:10, we observe high detection performance for these 7 datasets.

Among the remaining 4 datasets, StyleGAN-Face2, CycleGAN-Winter, CycleGAN-Zebra shows the biggest drop in performance as test set becomes more imbalanced. To further understand the reduced performance, we analyze the purity of the model fingerprints obtained as output of the fingerprint classification component. Purity of a model fingerprint is the fraction of images in the cluster (used to estimate the fingerprint) that are fake. If purity is less, then the performance of the fake image detection module will decrease (as the fingerprint is inaccurate). In general, for the three datasets (StyleGAN-Face2, CycleGAN-Winter, and CycleGAN-Zebra), we observe that purity of the fingerprints is lower compared to the other datasets. Figure 5.5a shows the distribution (CDF) of purity of fingerprints found across test sets (aggregated over all ratios) for two datasets—one for which NoiseScope is

\[9\text{For 1:8, and 1:10 we do 3 trials. Rest of them are averaged over 5 trials.}\]
Figure 5.5: (a) GAN fingerprint purity distributions (b) PCE Merging Threshold $T_{merge}$ vs. Detection F1 Score.

Table 5.4: Improved detection performance by increasing $T_{merge}$ in non-performant imbalanced configurations.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleGAN-Face2</td>
<td>88.47</td>
<td>86.45</td>
<td>80.23</td>
<td>76.71</td>
</tr>
<tr>
<td>BigGAN-DogHV</td>
<td>93.80</td>
<td>89.39</td>
<td>89.17</td>
<td>86.76</td>
</tr>
<tr>
<td>CycleGAN-Winter</td>
<td>89.65</td>
<td>90.17</td>
<td>82.08</td>
<td>81.00</td>
</tr>
<tr>
<td>CycleGAN-Zebra</td>
<td>92.17</td>
<td>91.98</td>
<td>86.26</td>
<td>83.13</td>
</tr>
</tbody>
</table>

performant (PGGAN-Tower), and one for which NoiseScope suffers from relatively lower performance (CycleGAN-Winter). CycleGAN-Winter suffers from lower fingerprint purity that range between 60% to 80%, whereas the fingerprint purity for PGGAN-Tower is high, i.e., over 95%. Therefore, high detection performance correlates well with the ability to reliably extract pure fingerprints.

One approach to improve fingerprint purity is to raise the PCE merging threshold $T_{merge}$. A higher value of $T_{merge}$ would prevent noisy samples from merging with fake images. StyleGAN-Face2, CycleGAN-Winter, and CycleGAN-Zebra results in Table 5.3 has $T_{merge}$ values in the range from 8.45 to 11.68. We raise the threshold to 15 and recompute the
results for these datasets. In addition, we also recompute results at the raised threshold for BigGAN-DogHV (which has F1 score below 90% in Table 5.3). Results with the increased threshold are presented in Table 5.4 for all 4 datasets. We observe a marked increase in detection performance, e.g., on average 10.35% increase in F1 for 1:10 ratio across all datasets. We also observe an increase in purity of fingerprints (not shown).

Above analysis raises the question of whether defender can estimate a better value of $T_{\text{merge}}$, starting from the initial estimate? We note that this is possible by analyzing the variation in cluster sizes as one increases $T_{\text{merge}}$ starting from the initial value. In general, detection performance correlates well with cluster sizes. If the largest cluster size is small (say less than 50), then the value of $T_{\text{merge}}$ is too high, and detection performance is likely to be lower. To study this, we conduct experiments on CycleGAN-Zebra with an imbalanced ratio of 1:2. Figure 5.5b studies the variation of detection performance and largest cluster size, as we incrementally increase $T_{\text{merge}}$ starting from the initial estimate. Detection performance remains mostly high and stable, for cluster size roughly above 100. Towards the end, the performance drops as cluster size goes below 67, achieving the lowest performance when cluster size is less than 50. The defender can thus calibrate $T_{\text{merge}}$ by incrementally increasing the originally estimated value, using cluster size as a stopping condition. If no clusters are found, or clusters are too small, then the defender has exceeded the optimal $T_{\text{merge}}$.

**Performance when test set contains fake images from multiple GAN models.**

So far, we considered test datasets with fake images from a single model. What if attackers use multiple GAN models? Can NoiseScope still detect fake images? In theory, NoiseScope should adapt to such settings, because clustering should ideally extract multiple model fingerprints corresponding to each model. To evaluate this, we restrict ourselves to datasets capturing faces, as it is the only content category for which we have fakes images from multiple models. In each trial, we populate the test set with 150 images each from
the StyleGAN-Face1, StyleGAN-Face2 and PGGAN-Face datasets, and use 450 real images from the FFHQ dataset. Results in Table 5.5 indicate an overall high F1 score of 91.5%, and also shows per-dataset performance.

So how did NoiseScope achieve high detection performance when test set includes fake images from three different models? Interestingly, NoiseScope discovered three clusters (model fingerprints). The first cluster mostly included images from StyleGAN-Face1 (over 95%), the second cluster mostly from PGGAN-Face (again over 95%), and in the third cluster, a majority of images are from the StyleGAN-Face2 dataset. Therefore, NoiseScope was able to extract model fingerprints corresponding to the three models. These results match our intuition that GANs trained on different datasets would generate distinct fingerprints. Our results indicate that NoiseScope is effective on test sets with fake images from different GANs. An attacker can take this setting to the extreme by creating a different GAN for every single fake image to disrupt the fingerprint extraction process. However, this significantly raises the cost for the attacker, and reduces the utility of using generative schemes.

**Performance on test sets with images from multiple categorical domains.** Our current configuration uses a single categorical domain for each test-set, but still has high variations among images. This was done for the sake of simplicity, and because many GAN datasets are organized into few specific categories. Here we evaluate effectiveness on test sets with multiple content categories. We test against BigGAN as it is the only GAN model with images from several categories. For a test set of 500 real, and 500 fake images, images are evenly and randomly sampled from 10 categories: Ambulance, Race car, Burrito, Tiger, Cup, Hen, Pretzel, Pirate, French bulldog, and Cheeseburger. The average detection performance (F1) is high at 99.1%. Thus, NoiseScope works for a mix of high-level image content, i.e., NoiseScope is content-agnostic.

---

10Models trained on three different datasets.
What if there are too few fake images in the test set? We present NoiseScope detection performance when evaluated on test sets with an increasingly small number of fake images. Using representative datasets for each GAN, we evaluate on 4 test sets with 50\textsuperscript{11}, 80,

\begin{table}[h]
\centering
\begin{tabular}{lrrr}
\hline
Datasets & F1 Score (%) & Precision (%) & Recall (%) \\
\hline
Combined & 91.5 & 93.3 & 89.8 \\
StyleGAN-Face1 & 91.2 & 83.8 & 100.0 \\
StyleGAN-Face2 & 81.9 & 100.0 & 69.3 \\
PGGAN-Face & 100.0 & 100.0 & 100.0 \\
\hline
\end{tabular}
\caption{Detection performance on test set with fake images from multiple (GAN) sources.}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{lrrrr}
\hline
Datasets & F1 Score (%) & Wavelet & Blur & NLM & BM3D \\
\hline
StyleGAN-Bed & 99.7 & 97.7 & 82.6 & 99.3 \\
BigGAN-DogLV & 99.7 & 99.5 & 95.0 & 99.7 \\
BigGAN-DogHV & 94.8 & 76.3 & 22.6 & 79.0 \\
BigGAN-BurgLV & 99.8 & 99.5 & 99.6 & 99.8 \\
BigGAN-BurgHV & 99.4 & 97.5 & 58.4 & 96.0 \\
PGGAN-Tower & 95.8 & 17.6 & 4.3 & 62.5 \\
CycleGAN-Winter & 91.1 & 71.4 & 6.7 & 84.5 \\
CycleGAN-Zebra & 93.6 & 74.1 & 7.9 & 96.9 \\
\hline
\end{tabular}
\caption{Fake Image Detector Performance (F1) using different denoising filters. Bold numbers highlight the best performance in each dataset.}
\end{table}

\textsuperscript{11}We remove the minimum requirement of 50 images for a fingerprint when only 50 fake images are in the test set.
100 and 200 fake images respectively. All test sets contain 200 real images from the respective dataset. A reference set of 2000 real images is used. $T_{\text{merge}}$ for StyleGAN-Face1 and PGGAN-Face remain the same as those used for the original results in Table 5.2 i.e., computed using the $T_{\text{merge}}$ estimation strategy in Section 5.5.2. $T_{\text{merge}}$ for BigGAN-DogHV, CycleGAN-Zebra and StyleGAN-Face1 are tuned following the recalibration strategy suggested in Section 5.6.2. We compute F1 score of detection performance averaged over 5 trials. Results are presented in Figure 5.6. The performance is moderately high, but as expected, drops as the number of fake images decrease. Analysis reveals a decrease in fingerprint purity, caused by merging with noisy samples amongst the increasingly large proportion of real images. From prior work we also know that a reliable fingerprint requires roughly 50 images or more [212]. The decrease in performance is not a serious problem—the absolute number of fake images to detect is in and of itself, very small. One can consider a scenario with too few fake images (<50) to not be a serious threat, compared to cases where online platforms are flooded with fake images [236].

Impact of residual image extraction filter on performance. We use 3 popular alternative filters—Blur filter [237], Non-Local-Means (NLM) filter [238], and the BM3D filter [239], and observe that Wavelet denoising provides better detection performance for nearly all datasets. Results are in Table 5.6. To compare, we simulate the fake image detection step in NoiseScope. Given a test set of 500 real, and 500 fake images, we estimate a clean model fingerprint using a random subset of 100 fake images from the test set itself. Next, we use this model fingerprint to flag fake images in the test set. This is an ideal scenario because the fingerprint is 100% pure (i.e., estimated over only fake images). An effective filter should produce high detection performance in such a setting, while filters that fail to effectively remove high-level content may not perform so well. Table 5.6 presents the
detection performance (average F1 score) for each 256x256 dataset.\textsuperscript{12} Wavelet Denoising filter exhibits the best performance, with F1 scores exceeding 90% for all datasets. The BM3D filter also shows good performance but fails to effectively eliminate content from some datasets.

**Generalization performance comparison with a supervised scheme.** Supervised detection schemes exhibit high performance at the cost of generalization. To give an example, we use the supervised classifier MesoNet [112] to detect unseen GAN-generated face images. MesoNet is trained on 1000 real and 1000 fake images from StyleGAN-Face1 and provides a high F1 score of 94% on a test set of the same size from StyleGAN-Face1. However, this trained model achieves significantly reduced F1 score of 65% on a test set from PGGAN-Face. This drop in performance indicates an exploitable failure to generalize that is remedied by NoiseScope.

**Summary.** We evaluated NoiseScope against datasets containing balanced and imbalanced proportions of fake images and observed stable behavior with generally high detection performances. We attributed the rare drops in performance to a low fingerprint purity, caused by low values of merging threshold $T_{merge}$. We accordingly provided guidelines for calibrating a better $T_{merge}$ based on cluster sizes. We show that NoiseScope is robust against datasets with multiple GAN sources. We evaluated NoiseScope against test sets containing few fake images and observe moderately high performance, with performance dropping when there are too few fake images (e.g., 50), at which point the threat itself is limited. We then showcased the impact of 3 popular alternative residual filters on NoiseScope’s performance. Finally, we highlighted the need for NoiseScope by showcasing the inability of supervised detection to generalize.

\textsuperscript{12}Applying the BM3D filter to 1024x1024 images is computationally expensive.
5.7. Analysis of Countermeasures

We consider a powerful adaptive attacker with knowledge of NoiseScope’s detection pipeline. These countermeasures aim to disrupt the fingerprint extraction, and fake image detection capabilities of NoiseScope. We also propose adaptive recovery measures to make NoiseScope robust to certain challenging countermeasures. Table 5.7 presents results using test sets with 500 real, and 500 fake images.

Compressing fake images to make fingerprints fragile. JPEG compression is well known to disrupt camera fingerprint patterns, and therefore diminish the correlation between fingerprints and residuals [240]. Following compression configurations used in prior work [137], fakes images are compressed with a quality factor randomly sampled from U [10, 75]. Surprisingly, NoiseScope is resilient against compression attacks. More interestingly, the performance for StyleGAN-Face2 increases from 90.14% to 98.33% on applying compression. NoiseScope is resilient for two reasons: First, model fingerprints are always extracted from the test set itself. Therefore, the estimated fingerprint, already captures any artifacts introduced by compression, and correlates well with the similarly processed residual images in test set. This is unlike prior work in camera fingerprinting, where camera attribution is attempted between a clean fingerprint (computed over uncompressed images)
and a compressed residual image. Second, we observe that JPEG compression introduces grid-like artifacts into the fingerprint, further making the model fingerprint distinct from device fingerprints. Fingerprints subjected to JPEG compression are shown in Figure B.1 in Appendix B.1. Compression does disrupt the fingerprint pattern. NoiseScope, however, continues to remain effective.

**Denoising using the defender’s denoising filter.** This countermeasure assumes knowledge of the Noise Residual Extractor. Attacker modifies fake images by subtracting the residual obtained using the defender’s denoising filter (Wavelet denoiser), \( I' = I - R \). This can make fingerprint extraction harder, because the patterns in the noise residuals are “weakened”. NoiseScope performance suffers for the BigGAN-DogHV and CycleGAN-Zebra datasets. See ‘Attack’ column under Wavelet Denoising. On visual inspection of the fingerprints, the texture patterns of fingerprints appear to have been softened by this attack. Performance dropped, because in certain trials, the fingerprint classifier module failed to flag the new model fingerprints, likely due to texture softening.

To recover from this attack, we resort to *adversarial training* of the fingerprint classification module. We train the classifier module on fingerprints computed from real images that goes through the same post-processing countermeasure used by the attacker. Results are shown in the ‘Recovery’ column under the specific countermeasure. We observe an improvement in detection performance for both BigGAN-DogHV, and CycleGAN-Zebra, while performance for the other datasets remain unaffected. Lastly, an interesting case is that of StyleGAN-Face2. Performance actually increases for this dataset on applying the countermeasure. On further inspection, we observe that the countermeasure introduces new distinct artifacts in the fingerprints, that enable NoiseScope to still accurately cluster images, and detect them. We suspect that images in this dataset has already undergone additional post-processing, which is likely introducing these artifacts when new processing is applied.
Other post-processing schemes to disrupt fingerprints. We evaluate against 4 image post-processing countermeasures known to disrupt camera fingerprinting [194, 240, 241, 242]. Whenever available, we use settings from prior work.

*Gamma correction.* Gamma correction is applied to fake images with gamma values randomly sampled from U [1.0, 2.0] [137]. Performance remains high for all datasets, except StyleGAN-Face2 where F1 score drops to 62%. Further investigation reveals that the fingerprint classifier performs poorly in 2/5 trials. For recovery, we again apply adversarial training to the fingerprint classifier, and train on real images that undergo the same post-processing. Performance of StyleGAN-Face2 recovers from 62% to 82%.

*Histogram equalization.* Histogram equalization involves distributing the intensity range to improve image contrast. We apply histogram equalization to fake images. Detection performance remains high for all datasets except StyleGAN-Face2. Fingerprint extraction did not perform well, and StyleGAN-Face2 ended up with impure clusters (purity ranging between 60% to 70%). We do not attempt recovery from this countermeasure because on examination of fake images that missed detection, we see that image quality has been severely degraded. Therefore, to evade detection by NoiseScope, post-processing that significantly degrades image quality is required. Image samples are shown in Figure B.2 (Appendix B.1).

*Blur.* Blurring performs a normalized box averaging on fake images, with a specific kernel size [243]. Kernel size is randomly selected from {1, 3, 5, 7} [137]. We expect blurring to damage patterns in the model fingerprints. Performance for StyleGAN-Face2, CycleGAN-Zebra and StyleGAN-Face2 end up dropping. We find that blurring largely weakens the fingerprint pattern.

However, on closer investigation of images that were not caught, we find that image quality has degraded significantly—NoiseScope failed to catch fake images that were severely blurred.
Therefore, we do not attempt a recovery scheme. Figures B.3, B.4 in Appendix B.1 show samples of images that evaded detection.

Adding Noise. We add i.i.d. Gaussian noise to fake images. The noise variance is randomly sampled from $U[5.0, 20.0]$. CycleGAN-Zebra, and StyleGAN-Face2 shows significant drop in performance. In both cases, noise degrades the quality of the fingerprint, making them unsuitable for computing correlation with residual images. In the case of CycleGAN-Zebra, the fingerprint classifier also fails to detect model fingerprints. To recover, we apply a denoising filter (Non-Local-Means) to all images in the test set, and also perform adversarial training of the fingerprint classifier using the same denoising filter. We apply this strategy to all datasets, and we can see that performance of StyleGAN-Face2 and CycleGAN-Zebra are regained to 81.4% and 85.9%, respectively, but recovery slightly hurts BigGAN-BurgHV by 10% due to the denoising operation.

Fingerprint spoofing. Fingerprint spoofing attack aims to disguise fake images to be from a specific camera device. This attack is commonly studied in the camera fingerprinting literature [244, 245, 246]. We use the StyleGAN-Face1 dataset to evaluate this countermeasure. We consider a fingerprint substitution attack [244] using the following formulation:

$$I_s = I - \alpha F_a + \beta F_b.$$  

$F_a$, and $F_b$, are model and camera fingerprints, respectively. $F_a$ is computed using 200 fake images from StyleGAN-Face1, and $F_b$ is a camera fingerprint computed using 200 images from Canon EOS 5D Mark III (FFHQ dataset). The first step is to verify that we have correctly spoofed the camera fingerprint. We empirically estimate $\alpha$ as 1.5, and $\beta$ as 1.5, as the spoofed fingerprint shows low PCE correlation with the model fingerprint, and high PCE correlation with the camera fingerprint, while maintaining image quality. We then consider a worst-case scenario for the defender, where the test set contains 200 spoofed fake images, and 200 real images which are used to extract $F_b$. We perform detection on such test sets with 5 trials. We obtain a low average F1 score of 66.67%. On
closer investigation, we find the fake image detection module performed poorly because the fingerprints have been spoofed.

To recover from this attack, we utilize a different filter, i.e., a normalizing box (blur) filter, instead of the Wavelet denoiser to compute residuals. The intuition is that the spoofing attack does not destroy all the artifacts (produced by the GAN), i.e., a model fingerprint can still be extracted. In fact, the performance is regained to 94.56% F1. Therefore, use of alternative filters for residual extraction is an effective recovery strategy against fingerprint spoofing attacks. One might argue that attackers can spoof the new residual space used in the residual extractor again. However, an endless game of switching residual extractors (multiple filters and filter parameters) is unlikely. If an attacker tries spoofing against multiple filters, then we observe that image quality deteriorates significantly. Image samples spoofed against multiple filters are shown in Figure B.5 in the Appendix.

Adapting the GAN model. Can the attacker modify the GAN to bypass detection? For example, for many DNN-based supervised detection schemes, the attacker can use the defender’s classifier as the GAN discriminator, and produce images that evade detection. In our case, such countermeasures are hard. First, the model fingerprints extracted by NoiseScope is tied to the fundamental building blocks of generative models, i.e., deconvolution layers (see Section 5.6.1). One can try to change the deconvolution layer parameters, which will change the fingerprint patterns, but is unlikely to make it similar to device fingerprints. Second, the attacker can use the fake image detector component of NoiseScope as the GAN discriminator. However, one has to ensure that the operations are differentiable, which is non-trivial. Also, such an effort would be similar to our previous countermeasures of spoofing the fingerprint or using the defender’s filter. We have already discussed robustness of NoiseScope against such countermeasures.

Summary. We evaluated a range of challenging countermeasures against NoiseScope.
NoiseScope is resilient against compression attacks, considered to be challenging in prior work. We also recommend effective recovery schemes against different types of post-processing attacks—Wavelet-noising, adding noise, and Gamma correction. The countermeasures that evaded detection includes those that degraded image quality significantly and can be considered as unsuccessful countermeasures. Online platforms like news/social media sites collecting images, can reject images that are excessively post-processed. There is ongoing work on detecting image manipulations or post-processing. For example, Adobe recently developed new tools to detect Photoshopped images [247]. NoiseScope can leverage such tools and implement appropriate recovery measures to make its detection pipeline more resilient to different countermeasures.

5.8 Real-world deployment of NoiseScope

Tech companies can set up synthetic image detection for their domain. Setting up NoiseScope involves building a test-set of images, building a reference-set of validated real images, and configuring parameters following the NoiseScope guidelines. Specifically, if Facebook or Linkedin wants to detect fake profile pictures, they can build a test-set containing profile pictures and a reference set of validated profile images, and then apply the NoiseScope algorithm to the test set.

However, countless images are uploaded onto the Internet per second in the real world, the detection would be much more efficient if social platforms can make quick decisions when images are uploaded by users. The company can build a fingerprint library at the platform backend, including different device fingerprints (pre-computed from device images) and model fingerprints (pre-computed from images generated by open-sourced GAN/VAE models). When users upload images, the backend can compute the correlation between the
image being uploaded and each fingerprint in the fingerprint library, and then classify the 
image as real or synthetic based on the correlation. With this strategy, social platforms 
can effectively identify synthetic images generated by open-sourced GAN instances readily 
available online. In fact, these open-sourced model instances are popular for synthetic image 
generation because it is low-cost and effective for people to use. In addition, to avoid 
damaging the user experience, social platforms can increase the probability threshold of 
synthetic class to flag only deepfakes that detectors are highly confident about. Alternatively, 
social platforms can apply an ensemble approach, where the final decision is based on votes 
from multiple detection schemes.

5.9 Conclusion

Deep learning research has tremendously advanced capabilities of generative models. GAN 
models can generate photorealistic synthetic images that could be used for different malic-
cious purposes, e.g., to spread fake news, create fake accounts. In this chapter, we present 
NoiseScope, a method to detect synthetic images in a blind manner, i.e., without any a priori 
access to fake images or their generative models. The key idea is to leverage unique patterns 
left behind by generative models when a fake image is produced. NoiseScope is evaluated on 
11 diverse synthetic image datasets, covering 4 high quality generative models, and achieves 
over 90% F1 score in detecting fake images. We also analyze the resilience of NoiseScope 
against a range of countermeasures.
Chapter 6

Towards Robust Detection of Synthetic Text

6.1 Motivation & Goals

In Chapter 4, we observe that existing defenses which rely on token-level features break down under the Top-p 1 attack (random sampling). This indicates that token-level features are not robust enough to discriminate between real and synthetic text. Among all the defenses, FAST [49] has held up more consistently against the different attacks, and also generalizes well to content in the wild. FAST models the factual structure of the article by tracking the consistency of named entities mentioned in it. Our hypothesis is that FAST’s performance can be attributed to the use of semantic features based on named entities mentioned in the article.

Recent studies show that producing semantically consistent text is still a challenging task for language models [61]. For example, synthetic text may not be consistent about how named entities are mentioned, e.g., named entities and their relationships may abruptly change over consecutive sentences. Generative models (e.g., GPT-3) also suffer from the problem of “hallucination” [248], which refers to a tendency to produce factually inaccurate or non-existent content. To extract semantic information, FAST uses an entity graph, i.e., a graph where nodes are named entities and edges connect (1) different named entities that
6.2 What is the Contributing Factor of FAST’s Robustness?

are located in the same sentence and (2) same named entities that are in different sentences. Despite FAST is more robust than other defenses according to Chapter 4, FAST still shows weaknesses under certain attack scenarios, which raises a natural question for us: Can we extract richer semantic features and improve the detection performance further? In the rest of this chapter, we analyze each component in the FAST pipeline to investigate the key contributing factor of its robustness, and further explore extracting richer semantic features via Knowledge Graphs (Objective 4). Our results demonstrate that leveraging semantic features of the text extracted via KGs is a promising future direction for robust detection of synthetic text.

6.2 What is the Contributing Factor of FAST’s Robustness?

6.2.1 Towards Understanding Adversarial Robustness of FAST

Understanding robustness when using semantic features. Among all the defenses, FAST has held up more consistently against the different attacks, and also generalizes well to content in the wild. But it is still unclear what aspect of FAST contributes to its robustness. Our hypothesis is that FAST’s performance can be attributed to the use of semantic features based on entities mentioned in the article. FAST models the factual structure of the article by tracking the consistency of named entities mentioned in it. To validate this hypothesis, we analyze FAST in more detail.

FAST has complex internals. As shown in Figure 6.1, FAST comprises 4 main components: A RoBERTa-based feature extractor, a Multi-layer GCN, a Next Sentence Prediction (NSP) model and a coherence tracking LSTM. Taking a document as the input, FAST first learns
contextual semantic representations for words via the RoBERTa language model. Next, a graph containing nodes representing entities in the text is created. The contextual word embeddings from RoBERTa are concatenated with Wikipedia2vec [249] entity representations to form the embedding for the entity graph. This graph embedding is then fed to a multi-layer GCN to obtain graph-enhanced sentence embeddings, which are then fed to an LSTM for coherence tracking. A Next Sentence Prediction (NSP) model is used to calculate the contextual coherence score for each neighbouring sentence pair. The NSP scores are then used to compute a document-level representation from the LSTM outputs. Finally, the RoBERTa embeddings are concatenated with the document-level representation and fed to a classification layer.

To better understand FAST’s superior performance, it is important to break down its complexity. We do so by running multiple ablation experiments. Models for the ablation studies are trained using the same data used to train FAST.

Ablation experiment #1: RoBERTa-Defense. We begin by considering a defense that only uses the RoBERTa language model. This is the same RoBERTa-Defense that has been evaluated in the previous sections. RoBERTa-Defense remains robust in several attacks,
6.2. What is the Contributing Factor of FAST’s Robustness?

<table>
<thead>
<tr>
<th>Attack Strategy</th>
<th>Detection Performance (Recall)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DistilFAST</td>
</tr>
<tr>
<td>Top-p 1.0</td>
<td>80.1</td>
</tr>
<tr>
<td>Top-k 80</td>
<td>88.7</td>
</tr>
<tr>
<td>Priming Tokens 12</td>
<td>89.2</td>
</tr>
</tbody>
</table>

Table 6.1: Detection performance (Recall) of DistilFAST, FAST and RoBERTa-Defense on attacks which change the decoding strategy used by the LM or prime it with varying numbers of priming tokens.

compared to the other defenses, but performs worse than FAST in two settings: (1) when under attack by DFTFooler (Table 4.7), and (2) when under attack by varying the decoding strategy (Table 4.5). For example, RoBERTa-Defense suffers a degradation in Recall of 22.0% compared to FAST which deteriorates only by 9.7%, when it encounters text generated using Top-p 1.0 decoding. This indicates that RoBERTa is not the main source for the robustness of FAST.

Ablation experiment #2: DistilFAST. Next, we test whether the semantic features, i.e., features from the entity network extracted by the GCN are the source of FAST’s robustness. To do so, we create a “distilled” version of FAST, called DistilFAST, by removing the NSP task, the LSTM coherence tracker, and the Wikipedia embeddings for the GCN from FAST’s pipeline. As a result, we are left with the RoBERTa model and the GCN. To create the document-level representation, we compute the element-wise sum of the sentence-level representations obtained from the GCN. To test robustness of DistilFAST, we evaluate it against: (1) adaptive attacks changing the generation process, (2) adversarial inputs based on DFTFooler and random perturbations, and (3) In-the-wild datasets. If DistilFAST performs similar or better than FAST, it would suggest that use of entity-based semantic features is the key enabler for FAST’s better generalization and robustness.

DistilFAST against attacks changing the generation process. Table 6.1 shows the results.
Table 6.2: Detection performance (F1) of DistilFAST, FAST and RoBERTa-Defense on AI-Writer and ArticleForge.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Detection Performance (F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DistilFAST</td>
</tr>
<tr>
<td>AI-Writer</td>
<td>94.1</td>
</tr>
<tr>
<td>ArticleForge</td>
<td>88.2</td>
</tr>
</tbody>
</table>

We consider attacks that change the text decoding strategy and the number of priming tokens. We test DistilFAST on the most effective attack configurations against FAST (from Tables 4.5 and 4.6). DistilFAST achieves a similar Recall as FAST when changing the decoding strategy to use Top-p 1.0 setting. In the other two strategies (using Top-k 80 and changing the number of priming tokens), DistilFAST even slightly outperforms FAST. We also show results for RoBERTa-Defense, which does not exhibit similar performance as FAST in one of the settings.

**DistilFAST against DFTFooler and random perturbations.** When the DFTFooler attack is applied to DistilFAST, we observe a 10.4% and 9.4% reduction in ER, compared to FAST, when using the BERT and GPT2 backend for DFTFooler, respectively. Similarly, against random perturbations, we observe a 13.4% reduction in ER, compared to FAST. DistilFAST is able to achieve better adversarial robustness than FAST against our adversarial perturbations.

**DistilFAST against In-the-wild datasets.** We evaluate DistilFAST against the In-the-wild datasets from the news domain, *i.e.*, ArticleForge and AI-Writer). Detection performance results are presented in Table 6.2. DistilFAST performs similar to FAST, suggesting that entity-based semantic features can improve generalization performance.

Our analysis leads to the following key finding: **Semantic features that capture the factual structure of the text, *i.e.*, entity-level features, provides robustness against**
adaptive attacks and better generalization performance.

6.2.2 Limitations of FAST

FAST constructs an entity graph by leveraging the location information of named entities—An edge connection is established between two named entities when (1) two different named entities are presented in the same sentence; (2) two same named entities are presented in different sentences. However, such location information cannot guarantee to precisely capture the semantic relationship among named entities in the text. Particularly, it has two obvious drawbacks: (1) The graph only focuses on named entities in the real world and cannot capture factual structure among other general noun chunks. (2) It blindly establishes edges based on entity locations without considering the actual semantic meaning of the document.

6.3 Extracting Richer Semantic Features via Knowledge Graphs

To overcome the drawbacks, we propose to extract richer semantic features from text via Knowledge Graphs (KGs). A knowledge graph is a powerful knowledge base that uses a graph-structured data model or topology to integrate data. It represents a network of real-world entities (i.e., objects, events, or concepts) and illustrates the relationships between them. Since producing semantically consistent text is still a challenging task for language models, we hypothesize that knowledge graphs built based on semantic relationship contain richer semantic information than FAST’s entity graph. We build detection schemes using KGs and entity graphs to verify our hypothesis.

Knowledge Graph Extraction. Knowledge graph construction can be generally cate-
Chapter 6. Towards Robust Detection of Synthetic Text

gorized into three groups: 1) Human-supervised approaches. KGs such as Wikidata [250], Freebase [251] and DBpedia [252] are built based on human supervision from Wikipedia info boxes and other structured data sources; 2) Rule-based approaches, which leverage carefully-designed patterns based on linguistic features (e.g., dependencies and POS tags), e.g., open information extraction systems including OLLIE [253], Stanford OpenIE [254], and OpenIE5 [255], etc. 3) Language model based approaches. MAMA [256] and DeepEx [257] use knowledge learned by pre-trained LMs to construct an open KG.

Human-supervised approaches cannot fully automate the KG construction process. Language model based approaches can be limited by the accuracy of knowledge contained within the backend language model itself. As a tradeoff, we empirically choose one of the state-of-the-art Open Information Extraction systems, OpenIE 5 [255], to extract semantic relational information from a document, and build document-level KGs for each of document under examination.

Real Article
Fake article

Figure 6.2: Knowledge graphs (KGs) extracted from real and synthetic news articles.

**Document-level KG Visualization.** Figure 6.2 shows two KGs (built using OpenIE-5) extracted from a real and synthetic article, respectively. Real and synthetic articles are news articles taken from the ArticleForge dataset in Chapter 4. We observe the KG corresponding to the synthetic class shows a different structural pattern compared to the KG of the real
article, that is, the synthetic article produces a KG with a longer graph diameter, compared to the real class. On manually examining the synthetic article, we found that the synthetic article is interspersed with several named entities that deviate from the topic of the article ("appointment of the US Treasury Secretary"), which leads to the longer graph diameter. We found several such failure cases, where "topic drift" leads to longer graph diameters. However, the graph diameter is not a universal discrepancy between real and fake classes. We also observe other failure cases of synthetic text where KGs present factual errors and contradictory semantic relationships. Therefore, the rule-based method that examines graph diameters would not work universally. Instead, we need a learning-based detection approach to extract more discernible features.

**A KG-based Detection Pipeline.** We build a KG-based detection scheme called KG-GCN, which builds a document-level knowledge graph and feeds it through a multi-layer Graph Convolution Network (GCN) [258] for classification.

Taking a document as the input, the detection pipeline includes the following steps: (1) *Parse the document sentence by sentence via the OpenIE5 extractor.* For each sentence, the OpenIE5 [255] extractor extracts a set of triples in the format of \((h, r, t, c)\), where \(h\), \(r\), \(t\) are head, relation, tail denoted by a span of words mentioned in the document, and \(c\) is the confidence level associated with the extracted triple assigned by the OpenIE5. Head and tail are named entities or noun chunks, and relation indicates the semantic relationship between head and tail. (2) *Build a knowledge graph with the extracted triples from step 1.* We discard low confident triples (empirically defined by a confidence threshold) as they probably contain inaccurate semantic information. To be more specific, given a triple (‘Obama’, ‘is president of’, ‘USA’), we establish a head ‘Obama’ and a tail ‘USA’ as graph nodes and create an edge between ‘Obama’ and ‘USA’ labeled by ‘is president of’. After building up all the triplets, we additionally establish edges between similar node pairs (defined by n-gram similarity)
presented in different sentences because they are likely to be the same entity mentioned throughout the article. For instance, node “Queen Elizabeth II” and node “Queen Elizabeth” mentioned in different sentences will be linked as they refer to the same named entity. Compared to the entity graph of FAST, nodes in the KG include not only named entities but also general noun chunks mentioned in the document. Edges in the KG are established based on semantic relations instead of entity location. To represent the KG, we initialize node representations with contextual word representations learned by a pretrained language model, \textit{i.e.,} LongFormer \cite{259}. We represent edge connections within the KG via its graph adjacency matrix. We use LongFormer to process longer text as its contextual window size is much longer than that of traditional language models like RoBERTa. (3) \textit{Learn enhanced graph representation via Graph Convolution Network (GCN) and fed it through a classification layer for binary classification.} We employ a multi-layer Graph Convolution Network (GCN) \cite{258} to propagate semantic information through multi-hop neighboring nodes of the knowledge graph. Specifically, with node representations and an adjacent matrix of the KG as the input (edge labels are ignored for simplicity in this pipeline), we obtain graph-enhanced node representations by propagating the input through 4 convolutional layers. We then calculate sentence-level representations based on the graph-enhanced node representations obtained from the GCN. Each sentence representation is calculated by the weighted sum of node representations in the sentence. We compute the element-wise sum of the sentence-level representations to obtain the document-level representation, which is then fed into a classification layer for binary classification.
6.4 Is KG More Effective Than Entity Graph?

**Experiment setting.** To assess the effectiveness of KG, we compare detection performances of the following two detection pipelines: (1) the KG-GCN detection pipeline described in Section 6.3. (2) We replace the KG in KG-GCN with the entity graph in FAST and name the pipeline as Entity-GCN, which can be considered as a baseline for KG-GCN.

We train KG-GCN and Entity-GCN on the GROVER training set used in Chapter 4. The contextual window size used by LongFormer is the first 2048 tokens of each article. We train both detection pipelines for 6 epochs. We then evaluate the trained pipelines on datasets used in Chapter 4, including the GROVER testing set, ArticleForge dataset, and the most effective top-p attack dataset (Top-p 1.0 decoding).

**Evaluation results.** Evaluation results are presented in Table 6.3. As shown in Table 6.3, KG-GCN demonstrates higher F1 scores (on machine class) and smaller performance drops compared to Entity-GCN on all datasets, which indicates that KG has advantages over the entity graph for semantic feature extraction. Nonetheless, the classification of KG-GCN and entity-GCN are solely based on the graph features. Since both KGs and entity graphs are not able to cover the overall contextual meaning of a document, the detection performances

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Detection Performance F1 Score (machine class)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>KG-GCN</td>
</tr>
<tr>
<td>GROVER</td>
<td>0.63</td>
</tr>
<tr>
<td>ArticleForge</td>
<td>0.57</td>
</tr>
<tr>
<td>Top-p 1.0 decoding attack</td>
<td>0.54</td>
</tr>
</tbody>
</table>

Table 6.3: Detection performance (F1 Scores on the synthetic class) on KG-GCN, Entity-GCN.
of both methods are relatively low. The KG-GCN pipeline can incorporate edge labels of the KG and a language model to enhance its performance. This requires more engineering efforts, and we level it as future work.

6.5 Conclusion

In this chapter, we explore the main contributing factor to the robustness of synthetic text defenses. We conclude that semantic features that capture the factual structure of the text provide robustness against adaptive attacks and better generalization performance. We also explore extracting richer semantic features via Knowledge Graphs (KGs). Our experiments indicate that knowledge graphs can extract richer semantic features compared to the entity graph used in FAST, the most robust defense we studied. We discuss future work of this research thread in Chapter 7.
Chapter 7

Future Work

7.1 Improving Content-agnostic Synthetic Image Detection.

NoiseScope demonstrates advantages of leveraging the noise space of images for synthetic image detection, but the limitation of NoiseScope is that it relies on a set of images (i.e., ≥ 50 images) to extract a fingerprint. With NoiseScope, a fingerprint is derived by averaging a set of noise residues such that random noise can be canceled out and a unique pattern specific to the generative model stands out (See Formulation 5.1). However, suppose that there are very few synthetic images in the test set, in that case, the quality of extracted fingerprints can significantly deteriorate as random noise can hardly be effectively canceled. To overcome this limitation, one can seek a different strategy of fingerprint extraction, that is, a learning-based approach that can extract the fingerprint based on a single image.

To be more specific, the fingerprint estimation framework outputs an estimated fingerprint, which carries the unique pattern associated with the source of the input (e.g., camera or a GAN). The input and the output to the network can be a noise residual and an estimated fingerprint, respectively. To achieve this, the training of fingerprint extraction framework needs a set of real images with associated camera labels. The model’s training objective needs to include terms that can: (1) Preserving spatial similarity: Noise artifacts are not
spatially stationary [216]. Therefore, fingerprints associated with the same location and
the same source (camera device) of images should be similar, while fingerprints associated
with different location or different device should be different. (2) Preserving correlation
relationship: This is to ensure that the estimated fingerprint has a high correlation with
the input residual. To determine whether the evaluated image is real or synthetic, one can
measure the correlation between the input residual and its output fingerprint using PCE
correlation metrics.

7.2 Improving Synthetic Text Detection Using KGs

In Chapter 6, we compare detection performances of Entity-GCN and KG-GCN to assess
the effectiveness of KGs to extract richer semantic features. However, the classification of
KG-GCN and entity-GCN are solely based on the graph features and both entity graphs and
KGs are not able to cover the overall contextual meaning of the document. Consequently,
the performances of both methods are relatively low. To further enhance the detection per-
formance of KG-GCN, (1) One can incorporate edge labels in the KG into the GCN; (2)
One can build an extended detection pipeline which makes the final prediction by feeding
the combination of document representation from KG-GCN and the last hidden vector from
a language model through an classification layer. Instead of RoBERTa, we suggest leverag-
ing advanced LMs such as LongFormer [259] such that the detection pipeline can process
documents using a longer contextual window size.

Apart from supervised schemes, another direction is to build a KG-based unsupervised
pipeline in order to improve the generalization performance of detection. According to our
analysis (see examples in Figure 6.2), we observe that semantic inconsistencies are spread
over the entire article. Therefore, one approach is to look for anomalous sequential patterns
in the semantic content throughout the entire article. We can build an anomaly detection scheme which is trained on real documents only. Each article will be broken into a sequence of fixed sized document segments. We can build a KG for each document segment to obtain a sequence of KGs. The sequence of KGs can be viewed as a temporally evolving graph. This evolving graph can be treated as a dynamic graph, where nodes (named entities) and edges (relations) appear or disappear during the course of the article. This can be formulated as a problem of finding anomalies in dynamic graphs [260].

7.3 Combating Misuses of Synthetic Media in the Future

Investigating effectiveness of defenses as deepfake threats evolve. Generative models for image and text generation have advanced in two key aspects: (1) The quality of synthetic content has immensely improved. (2) Generative models can control the properties of the generated content.

**Synthetic images:** Generative models for image generation are now able of producing high-fidelity images. For controllable image generation, an image can be generated with a text prompt, and face images can be generated by specifying certain facial attributes.

**Synthetic text:** Generative models for text generation are now able of producing longer convincing articles. Using a text paraphrasing language model, a new document can be generated by paraphrasing an existing one. Both these two aspects of advances in generative models threaten existing defenses. The high-fidelity synthetic content can derail defenses that seek specific artifacts. The controlled generation capabilities can be used to generate effective evasive samples. Unfortunately, most existing defenses are evaluated without con-
sidering such advanced generative models. Evolving deepfake content needs to be taken into account in future work.

Are noise space features still effective against newer, more advanced generative models? Figure 7.1 shows how a fingerprint computed from real images (from a camera) correlates poorly with residuals of synthetic images from advanced generative models, including StyleGAN3 [12], VQGAN [13] and the Latent Diffusion Model (LDM) [16]. The figure shows the distribution of PCE correlation of a real fingerprint with real images and synthetic images. Real images are face images sampled from the FFHQ dataset. Fingerprints are estimated by averaging image residuals. As the CDF curves for real and synthetic images are diverging, it suggests that noise space features are effective at discriminating synthetic and real images.

![Figure 7.1: Fingerprint analysis on most recent image generative models.](image)

Advancing proactive defenses. As deepfake threats evolve in uncertain ways, proactive defenses present an opportunity to strengthen the synthetic image and video defenses, as well as to prevent certain threats. One can develop novel approaches that apply carefully engineered “protective” perturbations to users’ images, such that (1) synthetic videos created from the protected images are more likely to be detected by a deepfake detection scheme, or (2) the protected images are unlearnable by deepfake generation pipelines. Prior work on crafting protective perturbations has certain limitations, e.g., requires the knowledge of the
attacker’s deepfake generator \cite{261, 262}, only applicable to specific types of content (e.g., human faces) \cite{263}. Therefore, there is still plenty of space for further enhancing proactive defenses due to the practical applicability caused by these limitations.

**Deploying deepfake detection in the real world and building actionable policies.**

*Detecting synthetic images/videos.* Social platforms should deploy detection when users upload their videos/images/text and tag content that is highly likely to be deepfakes. However, not all deepfakes carry malicious features. For instance, deepfakes posted with “#showyourdeepfakes” on Twitter might be posted for entertainment purposes. In practice, social platforms may need to adopt a more fine-grained tagging strategy — Specific deepfake tags can include entertainment, fake news, propaganda, etc. To tackle the problem, the defender may not only need to take account of deepfake videos, but also metadata of the video, demographic information, as well as account username and description which can indicate the nature of the account’s content. For example, the “@deeptomcruise” TikTok account name implies that the account contains deepfake Tom Cruise videos, for those aware that the word “deep” is used as terminology for AI-synthesized media. Social platforms can also opt to employ strategies such as voluntary tagging by their users on their products. There is not much research on the discrimination of deepfake categories (e.g., entertainment, fake news, propaganda), so the fine-grained deepfake tagging needs additional practical exploration.

*Detecting synthetic text.* For the text modality, one should be aware that each detection scheme has its own limitations. The defender should choose the best detection scheme based on their detection scenario. Detection schemes were trained on different topic domains, e.g., open-domain or news domain. Although FAST is the most robust detection tool out of 6 defenses we studied, FAST was trained only for detecting news articles so its performance can drop drastically when applied to documents from other domains. On the other hand, detection schemes might suffer from bias issues like defenses in the vision domain. The cause
of the bias can be attributed to the document topic, document length and text quality, etc. Synthetic text defenses can be sensitive to the length of documents (comments vs. news articles), and might perform better on longer documents than shorter ones because more features are available. Compared to poorly written articles, detection performances on well-written text (e.g., articles written by journalists) can be better. These open-ended questions need to be explored in depth by the community.
Chapter 8

Conclusion

Recent progress in generative models has made it possible to generate realistic synthetic media content, including synthetic images, videos, and text. Synthetic content produced by generative models, also known as deepfakes, can be misused by bad actors for malicious purposes. Till today, researchers have proposed a variety of detection schemes for detecting synthetic content. However, the research community still needs to overcome certain key challenges of existing work.

Existing defenses against synthetic content have been studied with limited or no knowledge of synthetic content in the wild. To overcome this challenge, we collect synthetic videos and text from the Internet and release them to the research community. We introduce DF-W, the largest real-world synthetic video dataset (at the time of our study) with millions of frames, as well as 4 in-the-wild synthetic text datasets. Our evaluation demonstrates that existing defenses fail to generalize to in-the-wild synthetic content, indicating existing datasets created by researchers cannot capture the distribution of in-the-wild synthetic content.

Although a variety of detection schemes have been proposed, they suffer from certain limitations. Existing defenses against synthetic images show poor generalization performances and are highly content-specific, limiting their applicability. To address this challenge, we propose a blind detection scheme called NoiseScope. In contrast to supervised schemes, NoiseScope requires no a priori access to fake images or knowledge of generative models (GANs) used by the attacker. NoiseScope is agnostic to the type of image content and GANs. Noise-
eScope achieves up to 99% F1 score in detecting high-fidelity synthetic images generated by StyleGAN. NoiseScope is resilient against multiple post-processing attacks as well as fingerprint spoofing attacks. For the text modality, the community has little understanding of the robustness of synthetic text defenses. Towards this direction, we evaluate the adversarial robustness of existing defenses, by proposing practical low-cost adaptive attacks, i.e., changing the process of text generation and crafting adversarial perturbations. Our study reveals that state-of-the-art defenses that mine sequential patterns in the text using Transformer models are vulnerable to simple evasion schemes. We conduct further exploration towards enhancing the robustness of detection schemes by leveraging semantic features extracted via Knowledge Graphs (KGs). Our analysis demonstrates that knowledge graphs can extract richer semantic features compared to the entity graph used in FAST, the most robust defense we studied.

We further discuss future work to be done within the scope of this dissertation, including enhancing content-agnostic synthetic image defenses, enhancing KG-based synthetic text defenses, and improving real-world deployment of deepfake detection.
Appendix A

A.1 Metrics for Evaluating Linguistic Quality

Besides GRUEN, we surveyed other text quality metrics and categorized them into the following groups: word-based metrics, embedding-based metrics, training-based metrics, and dialog-based metrics. Here we explain the main features of each metric category and their limitations given our usage scenario. Both word-based and embedding-based metrics require human reference text for evaluating text quality. Word-based metrics compute text quality based on the word or n-gram overlap between the evaluated text and the reference text, e.g., BLEU [264], METEOR [265], NIST [266], and ROUGE [267]. Word-based metrics rely largely on word-level matches. Such similarities can be better captured by word embeddings such as Word2Vec [268] and GloVe [269]. Thus, an alternative to matching words is to compare the similarity between the embeddings of words in the evaluated text and the reference, e.g., Greedy Matching [270] and Embedding Average metric [271]. The main limitation of these metrics is that all of them require references from the real class, which are not available in our case. Instead of comparing with human generated gold standard references, training-based metrics contain learnable components that are trained specifically for the task of automatic evaluation, e.g., ADEM [272]. However, training this model requires human annotations which were not available to us. Other text quality metrics such as GRADE [273] and USR [274] were designed for evaluating the quality of synthetic dialogs, and are difficult to transfer across usage domains. Besides automatic evaluation metrics, another way to evaluate text quality is to conduct a human study to annotate the quality of
documents. However, given the large number of datasets we evaluated, this was not realistic for us to do.

A.2 A Detailed Description of GRUEN Score

GRUEN, proposed by Zhu et al. [173], is an unsupervised and reference-less text quality metric. Zhu et al. show that GRUEN is more correlated with human judgement of text quality than any other existing metric. The GRUEN score of an article is computed by aggregating the following sub-scores:

**Grammaticality.** This is computed by combining two sub-scores: Perplexity and grammar acceptance. Perplexity is computed using a BERT model whereas the grammar acceptance score is computed by fine-tuning a BERT on the CoLA dataset [275] which contains labelled examples of grammatically correct and incorrect sentences.

**Non-redundancy.** This metric computes whether a document contains excessively repeated sentences, phrases and instances where proper nouns were used instead of pronouns. This is done by computing four inter-sentence syntactic features: length of the longest common substring, count of the longest common words, edit distance and the number of common words in a document.

**Focus.** This score looks at the semantic similarity between adjacent sentences as a measure of discourse focus. It is computed via the Word Mover Similarity [276] for adjacent sentences.

**Structure and coherence.** This is calculated by computing the loss on the Sentence-Order-Prediction [277] task as it models the inter-sentence coherence in a document. In the code provided by the authors of the GRUEN metric, this component was not included.
A.3 Linguistic Quality of Synthetic Text in the Wild

We use the GRUEN metric to evaluate linguistic quality. Figure A.1 shows the CDF of GRUEN scores for In-the-wild synthetic samples, compared with synthetic text produced by the research community, which includes GPT2-XL and GROVER. We can see that data in the wild is comparable or better than synthetic text produced by the research community.

Figure A.1: The CDF of GRUEN on the 4 In-the-wild datasets and 2 datasets generated from GROVER and GPT2-XL.

A.4 Other Defenses

There are a few other defenses that are not considered in our study. Yao et al. [147] in 2017, proposed a method to detect LSTM-generated synthetic reviews (restaurant reviews targeting Yelp). While the proposed method works well against LSTM-generated text, synthetic text generation has advanced significantly since Transformers were introduced. Yao’s method uses an LSTM-based supervised approach to detect synthetic text. We omit this defense because our preliminary evaluation of this approach on synthetic text produced by Transformers yielded unsatisfactory results. We upgraded Yao’s approach to use a Transformer-based classifier (instead of an LSTM model), and trained the model on 5000 real articles from the
APPENDIX A.

RealNews dataset, and 5000 articles produced by GROVER. Unfortunately, Yao’s approach only achieves an F1-score of 68% in detecting GROVER generated text.

Adelani et al. [278] present several classifiers, e.g., GLTR, GROVER and an OpenAI GPT-2 based Detector [279] to detect synthetic reviews generated using GPT2-Small. Fagni et al. [280] develop several classifiers based on Markov Chains, RNNs, LSTMs, and GPT-2 to detect synthetic tweets on Twitter. We do not study both approaches because many of the state-of-the-art methods such as GROVER, GLTR and other Transformer-based models are already considered in our work. Moreover, based on our preliminary investigation of Yao et al.’s work, we found that RNN/LSTM-based models are not promising approaches to detect synthetic text produced by advanced models like Transformers.

A.5 Applying Grid Search for Training GLTR

To tune the hyperparameters of GLTR defenses (based on logistic regression classifier), we apply grid search to the training process of the GLTR defenses, which includes GLTR-GPT2 and GLTR-BERT. We use GridSearchCV [281] which is built into scikit-learn to select the model. Given a set of parameter values, GridSearchCV can exhaustively consider all parameter combinations, fit the model on the training set, and select the best model. To train GLTR defenses, we apply grid search on the following hyperparameters of the logistic regression classifier, i.e., “solver”, “penalty” and “C”. Different choices of “solver”, “penalty” and “C” can result in differences in model performance [282]. We use GridSearchCV to loop through the following parameter values—[‘newton-cg’, ‘lbfgs’, ‘liblinear’], [‘l2’], [100, 10, 1.0, 0.1, 0.01] for “solver”, “penalty” and “C”, respectively.
A.6 Details of Fine-tuning Experiments

BERT-Defense fine-tuning experiments with In-the-wild samples. To improve BERT-Defense’s detection performance on *In-the-wild* datasets, we fine-tune BERT-Defense with a limited set of *In-the-wild* samples (i.e., 10, 50 and 100 samples) in Section 4.4.2. We follow the general guidelines of transfer learning to fine-tune BERT-Defense. We set batch size as 4, and fine-tune the BERT-Defense model for 8 epochs. While doing this experiment, we encountered a known problem of instability of fine-tuning BERT on small datasets [181]. To overcome this problem, we employ the revitalization strategy proposed by Zhang et al. [181]. We also tune certain training parameters to improve our results. Specifically, we use the “adamw_torch” optimizer, and set the “weight_decay” and “warmup_ratio” to $10^{-5}$ and 0.3, respectively.

GROVER fine-tuning experiments with In-the-wild samples. To improve GROVER’s detection performance on *In-the-wild* datasets, we fine-tune GROVER with a limited set of *In-the-wild* samples (10, 50 and 100 samples). We use a batch size of 4, and fine-tune the GROVER model for 3 epochs.

<table>
<thead>
<tr>
<th>Tools</th>
<th>Websites</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArticleForge</td>
<td>politico.com, usatoday.com, deseretnews.com, hollywoodreporter.com, theatlantic.com, nbcphiladelphia.com, reuters.com, reuters.com, dailymail.co.uk, theguardian.com</td>
</tr>
</tbody>
</table>

Table A.1: Websites used to scrape real news articles for ArticleForge and AI-Writer datasets.

---

1<https://huggingface.co/docs/transformers/training#finetune-a-pretrained-model>
Table A.2: Performance of the defenses on the *In-the-wild* datasets. We present F1 score (F1), Precision (P), and Recall (R) of the synthetic class in percentages. We did not test GROVER, FAST and RoBERTa-Defense on non-news domain datasets, which includes Kafkai and RedditBot. This is because these defenses are only trained for the news domain.

### A.7 More Details of the DFTFooler Pipeline

**Identifying a set of important words to be replaced via a LM.** DFTFooler scans a given article to choose the top $N$ most confidently predicted words according to the chosen LM, for replacement. For example, GPT-2 is a standard left-to-right language model, and after tokenizing the articles into a sequence of tokens $\{x_1, x_2, ..., x_i\}$, GPT-2 can compute the prediction probability of token $x_i$, using Equation 4.1, *i.e.*, $p(x_i|x_0, x_1, ..., x_{i-1})$. For simplicity, assume that each token is a word in the article. At each step in the sequence, a word is assigned a rank among all the words in the vocabulary based on its prediction probability score (higher probability leads to higher rank). In practice, a word can be tokenized into multiple tokens. For words that are tokenized into multiple tokens by the tokenizer, DFTFooler uses the probability score of the word’s first subtoken as the word’s probability score. This way, each word in the sequence is assigned a rank based on this probability score. We eventually choose the set of top $N$ most highly ranked words.
A.8 List of Real News Websites

As explained in Sec 4.3.3, our *In-the-wild* dataset contained an equal number of real and fake articles. For generating news articles, we used two text generation services, namely AI-Writer and ArticleForge.

ArticleForge can generate fake articles with a set of provided keywords. In this case, we collected 1000 real news articles from 10 news websites, and used keywords from them to generate 1000 fake news articles.

On the other hand, AI-Writer requires a title to generate an article. Similar to the method used for ArticleForge, we scraped 1000 news articles from a list of 20 news websites, and used their titles to generate synthetic articles. In both cases, the list of news websites are listed in Table A.1.

Figure A.2: Detection performances of the defenses with different context window sizes, i.e., 64, 128, 256, 512 tokens.
Figure A.3: The CDF of GRUEN score on perturbed text produced by different adversarial perturbation methods, i.e., DFTFooler, random perturbations, and TextFooler, based on attacking RoBERTa-Defense (in Table 4.7).

Figure A.4: Evasion rate and average GRUEN score of perturbed text achieved by DFTFooler and random perturbations when attacking (a) FAST and (b) BERT-Defense, based on 5, 10, 15, and 20 word perturbations.
Appendix B

B.1 Image Samples

Figure B.1: Model fingerprints from StyleGAN-Face2 [2], before (left) and after (right) applying JPEG compression.

Figure B.2: Samples from StyleGAN-Face2 [2] that evaded detection when subjected to histogram equalization. Top row shows the images before equalizing, and the bottom row shows the images after equalizing.
Figure B.3: Samples from CycleGAN-Zebra [3] that evaded detection when blurred. Top row shows the images before blurring, and the bottom row shows the images after blurring.

Figure B.4: Samples from BigGAN-DogHV [4] that evaded detection when blurred. Top row shows the images before blurring, and the bottom row shows the images after blurring.

Figure B.5: Image samples from StyleGAN-Face2 [2] subjected to a fingerprint spoofing attack against an increasing number of residual spaces. From left to right, we present (a) the original image, (b) the image spoofed against the Wavelet residual space (c) the image spoofed against the Wavelet and Blur residual spaces, and (d) the image spoofed against the Wavelet, Blur, and Laplacian residual spaces.
Bibliography


