NoiseLearner: An Unsupervised, Content-agnostic Approach to Detect Deepfake Images

Cristian Vives

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Applications

Bimal Viswanath, Chair
Tijay Chung
Chandan K Reddy

February 11th, 2022
Blacksburg, Virginia

Keywords: Machine Learning, Generative models, GAN, Deepfakes, Detection

NoiseLearner: An Unsupervised, Content-agnostic Approach to Detect Deepfake Images

Cristian Vives

(ABSTRACT)

Recent advancements in generative models have resulted in the improvement of hyper-realistic synthetic images or "deepfakes" at high resolutions, making them almost indistinguishable from real images from cameras. While exciting, this technology introduces room for abuse. Deepfakes have already been misused to produce pornography, political propaganda, and misinformation. The ability to produce fully synthetic content that can cause such misinformation demands for robust deepfake detection frameworks. Most deepfake detection methods are trained in a supervised manner, and fail to generalize to deepfakes produced by newer and superior generative models. More importantly, such detection methods are usually focused on detecting deepfakes having a specific type of content, e.g., face deepfakes. However, other types of deepfakes are starting to emerge, e.g., deepfakes of biomedical images, satellite imagery, people, and objects shown in different settings. Taking these challenges into account, we propose NoiseLearner, an unsupervised and content-agnostic deepfake image detection method. NoiseLearner aims to detect any deepfake image regardless of the generative model of origin or the content of the image. We perform a comprehensive evaluation by testing on multiple deepfake datasets composed of different generative models and different content groups, such as faces, satellite images, landscapes, and animals. Furthermore, we include more recent state-of-the-art generative models in our evaluation, such as

StyleGAN3 and probabilistic denoising diffusion models (DDPM). We observe that Noise-Learner performs well on multiple datasets, achieving 96% accuracy on both StyleGAN and StyleGAN2 datasets.

NoiseLearner: An Unsupervised, Content-agnostic Approach to Detect Deepfake Images

Cristian Vives

(GENERAL AUDIENCE ABSTRACT)

Images synthesized by artificial intelligence, commonly known as deepfakes, are starting to become indistinguishable from real images. While these technological advances are exciting with regards to what a computer can do, it is important to understand that such technology is currently being used with ill intent. Thus, identifying these images is becoming a growing necessity, especially as deepfake technology grows to perfectly mimic the nature of real images. Current deepfake detection approaches fail to detect deepfakes of other content, such as sattelite imagery or X-rays, and cannot generalize to deepfakes synthesized by new artificial intelligence. Taking these concerns into account, we propose NoiseLearner, a deepfake detection method that can detect any deepfake regardless of the content and artificial intelligence model used to synthesize it. The key idea behind NoiseLearner is that it does not require any deepfakes to train. Instead, NoiseLearner learns the key features of real images and uses them to differentiate between deepfakes and real images – without ever looking at a single deepfake. Even with this strong constraint, NoiseLearner shows promise by detecting deepfakes of diverse contents and models used to generate them. We also explore different ways to improve NoiseLearner.

Acknowledgments

I would first like to thank Dr. Viswanath for his continued expertise and feedback during my thesis. Your knowledge and domain expertise helped me guide through the project and has taught me very valuable lessons I would also like the thank Jiameng Pu who met with me regularly to help guide the project to completion. You mentored me in a well manner and helped explore different ideas that aided in the completion of my thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

CNN Convolutional Neural Network

DDPM Denoising Diffusion Probabilistic Models

GAN Generative Adversarial Network

SOTA State of the Art

VAE Variational Autoncoder

Chapter 1

Introduction

1.1 Problem Motivation

Ever since the introduction of generative adversarial networks (GANs)[28] in 2014, there has been a growing interest in developing complex models capable of flawlessly imitating pristine images. Today, state-of-the-art generative models can generate high-quality deepfakes of any content, such as faces, landscapes, and medical imagery. While most work has been done in improving GAN architectures [16, 37, 38, 39, 43, 64], which train a generator in an adversarial manner against a discriminator, likelihood-based models have been shown to generate similar images despite a completely different generative procedure[24, 33, 40, 57]. As the name suggests, likelihood models instead maximize a likelihood function to model the distribution of the real data. Doing so results in a slower procedure, but produces more diverse deepfakes that better fit the training data distribution.

Unfortunately, this technology can be used with ill intent. Deepfakes are notably seen in pornography, creating deepfake pornographic videos of popular female celebrities without their consent. With the improvement of generative models that require less data to produce high-fidelity deepfakes, their misuse can affect anybody regardless of their social standing. Researchers have been able to apply CycleGAN to satellite imagery [62], transferring the style of one city to another. Deepfake satellite imagery can trick unaware users about their whereabouts, especially now that we are becoming more dependent on our phones as location

devices. Facebook introduced a model capable of rewriting layered text in images [41], which can be used maliciously to add offensive text to images. Jekyll[49] trains a style transfer network to attack X-ray imagery by adding specific diseases. The fraudulent use of these modified images can be used to extort or to deceit healthy patients. The government has already recognized the severity of these concerns, transforming them from practical jokes to serious threats. In March 2021, the FBI stated that deepfakes will continue to be a threat to national security and foreign influence operations, and that such technology will be used by foreign nationals to influence media[12]. All of these deepfakes have the capability for abuse, and detecting such imagery is a growing need.

Existing deepfake detection methods suffer from numerous limitations. **Most deepfake detection schemes fail to generalize well to deepfakes from other generative model architectures.** The majority of deepfake defenses employ a supervised learning scheme, where fake images are required for the detection method to work[18, 25, 50, 63]. This approach is not scalable as they fail to perform well on deepfakes from generative models not present in the training data. Furthermore, such defense schemes require a priori access to a large batch of fake images, which is not always possible unless the generative model is widely available. If the generative model is available, these supervised learning defenses are susceptible to adversarial attacks[17]. Regardless, we argue for unsupervised defenses that only require real data to train. Not depending on existing generative models allows for the defense to work to improved deepfakes.

Deepfake detection schemes fail to detect content apart from faces. Most deepfake detection defenses can often only detect faces, failing to capture deepfakes of different contents. Supervised learning approaches can only detect images of the domain they were trained on. Most unsupervised approaches don't require real data to train[46, 47, 61], which alleviates the concern for a generalized defense to distinct generative models. However, the

vast majority employ a strict generation pipeline to imitate the nature of fake images, often specific just to faces (e.g requiring facial landmarks to generate fake faces on the fly)[46, 47]. This approach fails to generalize to other types of content apart from faces. A robust defense method should be impervious to the content of the image due to the aforementioned harm that they can encourage.

1.2 Contribution

We believe there is a serious lack of content-agnostic and unsupervised deepfake detection schemes, and thus present **NoiseLearner**. NoiseLearner aims to mitigate all of the concerns presented above by only training on real data and evaluating on deepfakes from different models and domains. In our case, we consider a deepfake any *image* from a generative model, so videos are not considered in our defense. The main contributions of NoiseLearner are as follows:

- **Content Agnostic and Unsupervised Deepfake Detection:** We leverage imaging forensics techniques to detect deepfake images from pristine images, regardless of the image's content. It is shown that imperfections in cameras induce low-level artifacts, such as PRNU patterns, in pristine images[19, 26]. Many of these artifacts are induced by the camera of origin, and can be traced to any image originating from the same camera model. These common patterns are known as the *camera's fingerprint*, a set of features not present in deepfakes. In order to perform this deepfake detection task, NoiseLearner is composed of two components. **(1)** A **fingerprint generator** which extracts a *pseudo fingerprint* such as that it contains distinctive generalized patterns for any real image and not for deepfake images. An image's true fingerprint contains unique patterns to the image's camera of origin, which can be used as a feature for

source identification[19]. In our case we are more interested in extracting a generalized *pseudo fingerprint* which carries common features for any real image. Deep learning is used to train a Convolution Neural Network (CNN) to extract *pseudo fingerprints* from real images. Due to a lack of ground-truth to compare an image's *pseudo fingerprint*, we instead implement a set of constraints for the fingerprint generator to follow, which are explained in section 3.3.1. **(2)** A **Deepfake PCE Detector**, where we utilize the peak energy correlation (PCE) of the image's residual and its *pseudo fingerprint* to detect whether the image is fake or real. PCE has been shown to work better than other correlation metrics in regards to images[19, 26, 27]. Since an image's residual is simply its original fingerprint with some stochastic noise (usually high level-content leakage), we expect the PCE correlation of the residual and the *pseudo fingerprint* to be high for real images and low for fake images. We then calculate a threshold through different procedures and classify PCE scores above this threshold as real and below as fake.

- **Diverse Content Detection:** A comprehensive evaluation of our defense is done on different datasets encompassing multiple domains and generative models. Dataset content span faces[6, 10, 38], animals[1, 2], landscapes[2, 6], and satellite imagery[3]. For generative models we mostly focus on GAN architectures[16, 37, 38], including style transfer models[64].

- **New Generative Models Detection:** We realize that most deepfake detection schemes evaluate on either outdated generative models or low-quality datasets. To provide a comprehensive evaluation, we include new models such as StyleGAN3[39], StyleGAN2[43], and denoising diffusion probabilistic models[24, 33]. Both models contain complex architectures that drastically change the deepfake generation pipeline. Assumptions made by different deepfake detection methods might not apply to these

models – including ours. Thus it is only fair to include them as they might eventually be heavily used for deepfake generation tasks.

- **Comparison to Baseline Methods:** We also show the performance of different unsupervised baseline methods[22, 45, 52] compared to NoiseLearner. It is crucial to compare existing unsupervised methods to justify the need for NoiseLearner.

- **Future Work Discussion:** NoiseLearner shows a lot of promise and we explore different ideas to advance its current state. We discuss multiple potential improvements and how they might be approached given our attempts to implement them. Overall we have found many of these approaches to not improve performance in practice. An in-depth analysis is done on the reasoning for these additions, their failures, and insights gained to further polish NoiseLearner.

Chapter 2

Review of Literature

2.1 Deepfake Generation Methods

2.1.1 Generative Adversarial Networks

The vast majority of deepfakes generated today are done so through the use of Generative Adversarial Networks (GANs). First introduced in 2014 by Goodfellow et al.[28], GANs leverage an adversarial training to learn the data distribution. GANs are composed of two parts: the generator, and the discriminator, which are trained concurrently. The generator attempts to map a vector of random noise, often Gaussian, to an image, while the discriminator attempts to classify the image as real or fake. Therefore, the job of the discriminator is to maximize it's own accuracy whilst the job of the generator is to minimize it. Given that the training dataset is large enough and the generator complex enough, the training will be complete once the discriminator reaches a 50% accuracy: when the generator has learned the distribution of the training data and is able to map from a random vector space of gaussian noise to an image of a certain shape.

Since then, various different more complex GAN architectures have been introduced. **PG-GAN** [37] introduced progressive growth to GANs, starting the image synthesis step at very low resolutions (i.e 8x8) and scaling up two much higher resolutions (1024x1024). At the time this was considered SOTA and was able to train a generator capable of producing 1024x1024

deepfakes from the Celeb-A dataset. **BigGAN**[16] improves on PGGAN by introducing more model complexity (more parameters and bigger batch sizes) and introduces a truncation trick which allows explicit control of sample variety and fidelity trade-off. **BigGAN** is also successfully trained on the much larger JFT-300M dataset.

StyleGAN[38] improves the generator architecture by introducing the *mapping network*, which maps a low-dimensional gaussian noise vector into a learned latent space that controls the style of the output. The input of the generator is instead a learned 4x4x512 constant, and the style vector is applied after every convolution layer as a learned affine transformation through the ADAIN (adaptive instance normalization) operation. Style vectors allow for feature disentanglement, so the Generator does not correlate independent features with each other (such as bearded men having long hair). **StyleGAN** also makes other contributions, such as using bilinear upsampling layers and gaussian noise additions after every convolution layer to allow for stochastic variations in the synthesized images, such as small variations in facial hair. **StyleGAN2** further improves on the pre-existing architecture by removing the so prevalent water-droplet artifacts that StyleGAN was known for. The problem lies in the ADAIN operation, and instead, StyleGAN2 opts for weight demodulation. Other changes include the removal of redundant operations, the shift of noise addition outside of the style blocks, the inclusion of more robust evaluation metrics (such as P&R), the overall improvement of the Generator network (such as skip connections in progressive growing), and a less demanding regularization objective. Finally, **StyleGAN3**[39] was recently introduced with the objective of tackling the issue of "texture sticking," where the StyleGAN generator would learn to map textures to specific pixel-locations. So if a few samples were generated by latent interpolation walks, one would see features, such as hair, remain rather static as new samples were generated. StyleGAN3 attributes this problem to aliasing and approaches the StyleGAN generator architecture under a signal processing scope. A lot of effort is put into

redesigning the generator to process signals as continuous instead of discrete, which results in drastic changes such as the removal of additive noise, the replacement of upsampling and downsampling operations with single-step convolutions and low-pass filters, and the introduction of fourier features as the learned constant[56]. All these changes allow for full rotation and translation of the image without worsened FID scores.

CycleGAN[64] differs from the previous architectures as it tackles the challenge of image-to-image translation. Rather than synthesize a raw image from a vector of random noise, CycleGAN transforms an already-existing image from one domain (e.g horse) to another (e.g zebra). Unlike Pix2Pix[35], CycleGAN allows for unpaired data training, allowing the network to learn the mapping from one style to another and vice-versa.

2.1.2 Likelihood Models

Likelihood models instead handle generative modelling with a Bayesian approach and attempt to model the distribution of the data by maximizing a log-likelihood function. While likelihood models provide a wider and more diverse representation of the training data[33, 53], they fail to match the quality performance of GANs[32, 42, 55]. Furthermore, likelihood models tend to be slower at inference when compared to GANs. The most known likelihood model is the Variational Autoenoder (VAE)[40], which trains the encoder to learn a mapping from the training set to a latent Z-space (often gaussian). The decoder is then trained to generate an image from the predicted Z-space distribution. After training, the encoder is not needed and different Z-space distributions can be passed to the decoder for image synthesis. As mentioned before, VAEs cannot match the visual fidelity of GANs. The introduction of Nouveau VAE (NVAE)[57] would yield comparable results to GANs due to a more carefully crafted neural architecture and the introduction of batch-norm and depth-convolutions in

the encoder and decoder layers. A more recent family of likelihood model is the **denoising diffusion probabilistic model** (DDPM)[33], which trains a neural network to perform a reverse-step of a gaussian-noise addition process at a certain time step. What this means is that an image can eventually be converted to a full Gaussian sample if small random Gaussian additions are made one step at a time at different time steps. The goal of DDPMs is to train neural networks at different time steps to perform the reverse process (i.e given a noisy image at time step T, predict the noise added to the image at time step T-1). DDPMs showed promise due to matched performance to PGGAN, but the addition of guided DDPM yielded a comparable performance to many state-of-the-art GANs[24]. **Guided DDPMs** leverage label information to train classifiers at different time steps to better guide the diffusion model. The gradient of the classifier at a given step is added to the output of the diffusion model at that step. If the noisy image at that step is far from ideal, the classifier will penalize it and infer a greater gradient to correct the prediction.

2.2 Deepfake Detection Methods

Deepfake detection models utilize deep learning to train models to extract features from the training data. These features can then be utilized to perform deepfake detection. There are two different ways that these models are usually trained: supervised and unsupervised.

2.2.1 Supervised Approaches

Supervised approaches compose the most common yet simple method to detect deepfakes. This method requires training data to contain both real and fake data, and is very effective when evaluated on deepfakes from the same domain of the training data. However, this

method fails to generalize well to deepfakes generated by models not present in the training set. This results in a simple and short-term effective solution that fails to scale well to new deepfakes generated by newer generative models. Only using this approach can lead to an arms race where the defender (deepfake detection) is always playing catch-up with the attacker (deepfake creation).

Zhao et al[63] propose a multi-attention model to better detect deepfakes. They argue that as deepfakes become more realistic, the differences are often subtle and tied to specific locations within the image. They introduce a complex model architecture that extracts local texture information and enhances it. Attention heads are also used to focus on specific regions of the image. The combination of attention heads and texture-enhancement blocks allows the model to compare low-level textures and high-level content features to classify an image as fake or real. Afchar et al propose MesoNet[13], which examines images at the mesoscopic level. MesoNet uses an inception net to extract features from images to then classify as fake or real. Defakehop[18] aims to train a lightweight deepfake detection model. Different facial features are extracted and processed through separate channels which are later concatenated and passed through as classification head. Nguyen et al [50] tackle deepfake detection by utilizing a capsule network, An image's features are extracted from a pre-trained VGG19 model and fed to a capsule network for deepfake detection. Frank et al [25] extend the idea of inconsistencies of deepfakes in the frequency domain [61] and train a classification network with the frequency spectrum of images, extracted through discrete cosine transform (DCT), instead of the original raw pixel-representation of the images.

2.2.2 Unsupervised Approaches

Unsupervised approaches differ from supervised approaches as they don't require fake data to train. Instead, most unsupervised methods employ a data-generation step where fake images are generated. Most of these fake images are generated in such a way that exploit generalized weaknesses found in most generative models. For instance Face X-Ray [46] attempts to detect deepfakes by detecting the face boundary for the case of face swaps. Face X-Ray employs a fake data generator by performing face swaps of two random real images; a method not scalable to deepfakes that are not faceswaps. Furthermore, the defense will become invalid as newer faceswap models are more precise at removing these impurities from the faceswap. Li et al[47] introduce a similar detection framework but instead look at warping artifacts of faceswaps. Fake data is generated in a similar manner to Face X-Ray[46] by applying warping functions to random real images. This method only works for low-resolution images and is not scalable as faceswaps become more accurate at higher resolutions.

AutoGAN [61] looks at deepfake images in the frequency domain. Upsampling layers in generative models cause deepfake images to have checkerboard patterns in their frequency spectrums. AutoGAN exploits this information to build a fake data generation pipeline, where a generator based off an Encoder-Decoder architecture is trained to reconstruct real images. The training is carried out in an adversarial manner with a Discriminator which provides feedback such that the reconstructed image, which now contains heavy artifacts in its frequency spectrum, is visually realistic. While AutoGAN is content-agnostic as opposed to Face X-ray, the performance of this approach is highly dependent on the generator architecture, and fails to generalize well to deepfakes from other models.

However, not all unsupervised approaches employ a fake data generation step. Instead, these approaches build an outlier detector using features of real images, where unseen data points

are placed outside of the distribution. For instance, Li et al.[45] exploit the colorspace of images to detect deepfakes in a blind setting. They argue that deepfakes are not indistinguishable from pristine images when examined in a different colorspace than RGB, most notably HSV and YCbCr. Furthermore, it is more effective to compare the residual of images, as the content of images induce a great deal of high-level noise. The proposed method trains an SVM on extracted colorspace-features from residues of real images. The approach performs well on "low resolution" images of 128x128 pixels but has worsened performance on "high resolution" images of 256x256 pixels. Noiseprint[22] is a visual approach which extracts a fingerprint from an image which highlights any inconsistencies. If an area of the image has been tampered with, the resulting Noiseprint will show an abstract texture at that same location. While similar in thought to NoiseLearner, Noiseprint is not tuned for deepfake detection and does not contain an end-to-end detection step.

NoiseScope[52] is also an unsupervised and blind detection method that uses fingerprint information to detect deepfakes. NoiseScope differs from NoiseLearner by instead focusing on artifacts from GAN models by clustering residuals with high PCE correlations together. Since GAN artifacts are stronger than real-camera artifacts, they will have a higher PCE correlation and cluster together. At the end of the clustering step, fingerprints are extracted from large clusters and run through a pre-trained outlier detector. If the fingerprints are outliers, they will be classified as fake. NoiseScope is the first fully unsupervised and blind deepfake detection method and performs well on numerous GAN datasets. However, NoiseScope cannot perform classification of a single image, it requires large cluster of both fake and real images to function well. Also, Noisescope has not been evaluated on newer and state-of-the-art generative models which might either suppress heavy upsampling artifacts or improve the distribution of these artifacts (thus creating a sort of fragmentation where there are multiple clusters with few fake images). NoiseLearner can be seen as a continuation

of Noisescope where neural networks are used to estimate the fingerprint of a single image, which can then be used for classification.

Chapter 3

Methodology

3.1 Attacker and Defense Model

Attacker: The goal of the attacker is to generate realistic deepfakes of any content by using generative models. For this threat model, the majority of deepfakes are expected to be generated by GANs, but other models are also considered. Also, any deepfake that only has a partial deepfake, such as a faceswap, is not considered. The attacker may also, but is not limited to, generate deepfakes from many pre-existing network architectures: such as PGGAN, StyleGAN, CycleGAN, etc. As mentioned before, the content of the deepfake can be anything, such as human faces, satellite imagery, or landscapes.

Defense: The goal of the defense is to detect any deepfake that may have been synthesized by any generative model architecture with a greater emphasis on GAN data. The content of the deepfake does not matter, as the defense will only look at the residual for detection. To make the defense more generalized, there will be no access to fake images. The Defense will have access to a set of real images to train the defense model. While the content of the images is unimportant, camera label information for each image is necessary. Once trained, the defense is able to perform deepfake detection on a single image.

3.2 Method Basics

The crux of our approach is based on three concepts: (1) unsupervised training, (2) content agnostic deepfake detection, (3) deepfake detection on a single image. (1) The approach must be done in an unsupervised manner without any form of fake-data generation. Supervised approaches fail to accurately detect deepfakes which were not part of the training distribution [18, 25, 50, 63] and unsupervised approaches that employ data generation are not scalable as they exploit obvious shortcomings of generative models that are quickly being mended [46, 47, 61]. Instead, learning key features from real images will future-proof this deepfake detection pipeline, as low level artifacts like PRNU are likely to remain regardless of the improvement of new camera models.

(2) We consider the importance to detect deepfakes of any kind. While most research in deepfake detection is mostly focused on face images, and usually fine-tuned in that direction[18, 46, 47], we believe that deepfakes of other content can be quite problematic. In order to achieve this goal, we borrow ideas from imaging forensics and focus mostly on low-level artifacts from images. For instance, it has been shown that real images contain artifacts, such as PRNU patterns[26], imprinted by imperfections in cameras. Deepfakes synthesized by generative models which contain upsampling or deconvolution layers imprint other types of artifacts in both the visual and frequency domain[25, 61]. By instead focusing on low-level features, we are able to perform deepfake detection regardless of the high-level content.

(3) The prime motivation for this work is the need to perform deepfake detection on a single image. This work can largely be considered a continuation of NoiseScope[52], a deepfake detection method that follows the first and second requirements. However, despite being a robust method, NoiseScope cannot achieve deepfake detection on a single image, it requires

a large cluster of real and fake images to function correctly. In our work we explore the concept of performing deepfake detection on a single item.

3.2.1 Terminology

Residual: An image's residual is the low-level content in the form of noise. It is extracted by taking the difference of an image and it's high level content, and can be computed by:

$$K = I - f(I) \tag{3.1}$$

where I is the original image and f is a denoising filter. In this thesis we use wavelet denoising filters [11], but also experiment with others such as BM3D[23]. Overall, the goal of a residual is to extract the noise of an image without any high-level content. The importance of image residuals is crucial to our work, as low-level artifacts are found in the noise space of images[26].

Fingerprints: Fingerprints are refered to as unique patterns shared by images from a certain source. Deepfakes contain patterns produced by upsampling layers or deconvolution layers in generative models [61]. Pristine images contain PRNU and other low-level artifacts produced by imperfections in the camera [19, 26, 27]. On top of that, the fingerprint patterns amongst different camera models might be different. It has been shown that unique camera fingerprints are a strong enough identifier to perform camera source identification from a single image[19, 48]. Figures 3.1 and 3.2 show examples of fingerprints calculated for different generative models and cameras. Notice how more complex generative models like StyleGAN3 have fingerprints not that different from real images.

Low-level artifacts for pristine and generated images are found in the residual. However,

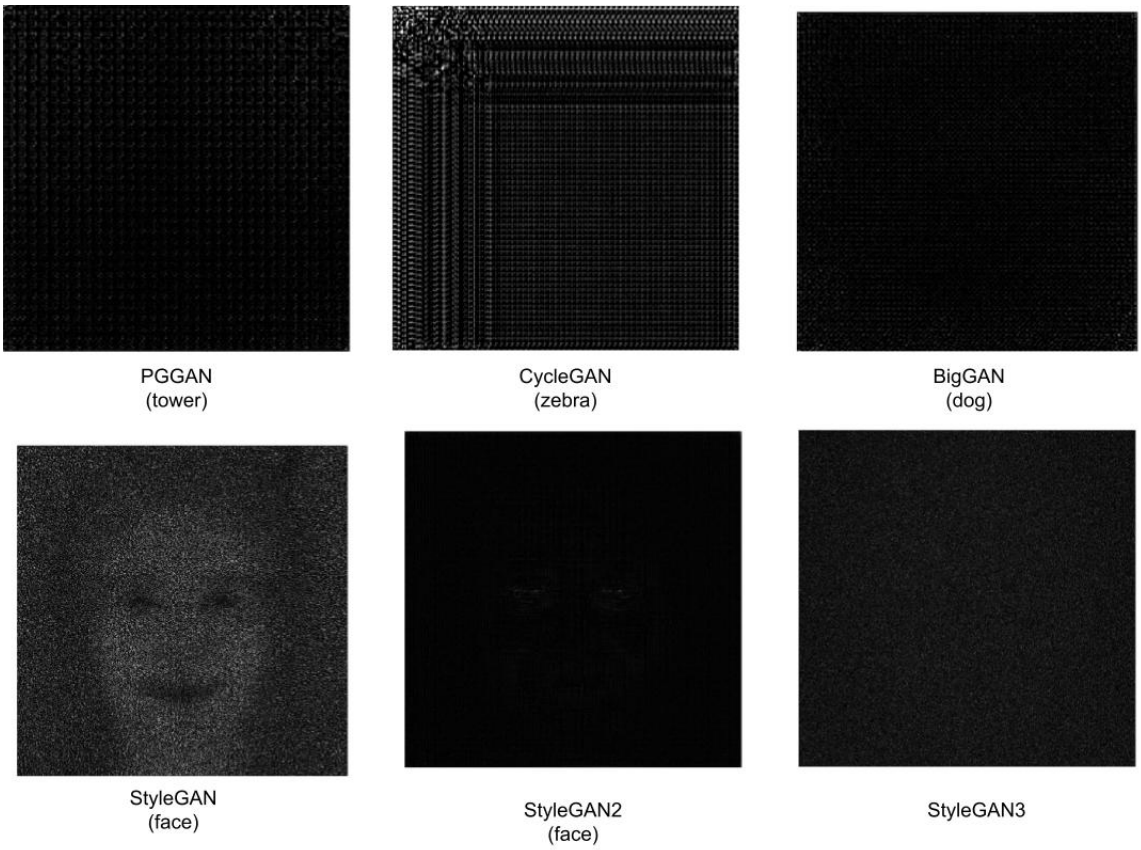

Figure 3.1: Fingerprints of Different Generative Models. They were extracted by averaging 2000 deepfakes of a certain dataset. For instance, the PGGAN fingerprint was extracted by averaging tower deepfakes generated by PGGAN

leakage of high-level content requires averaging a set of residuals to extract a robust fingerprint. Given a set of images $\{I_0, I_1, ..., I_n\}$ all belonging to a camera K, and a denoising filter f, the fingerprint F of K can be estimated by:

$$F = \frac{\sum_0^n (I_n - f(I_n))}{n} \tag{3.2}$$

Peak Energy Correlation: Also known as PCE [26], is a correlation metric that normalizes the max peak of the pearson correlation between two signals. It has been used effectively in the area of imaging forensics and is considered more stable than other correlation tests[27].

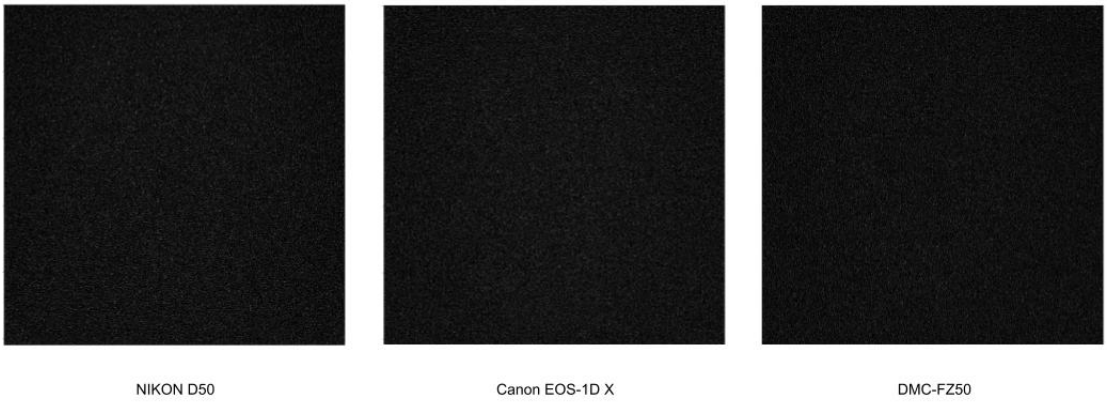

Figure 3.2: Fingerprints of Different Camera Models

NoiseScope uses PCE as a thresholding metric to cluster similar image residuals. In our work, we use the PCE score as a way to evaluate the similarity between the fingerprint of an image and its residual. Recall that the residual of an image is the image sans the high-level content, leaving only the noise. Within this noise, many of the camera low-level artifacts and PRNU can be found[19]. Despite being the residual of the image, there is often quite a lot of content leakage from the original image, which is why in order to compute this fingerprint, many samples are needed to average out the high-level noise. By this definition, the PCE of the fingerprint and the residual should be relatively high since they both share these low level artifacts

3.3 Key Challenger in Designing NoiseLearner

3.3.1 Pseudo Fingerprint Calculation

Initially we argued for an imaging forensics approach, where fingerprints are used to classify an image as real or fake. More specifically, the idea was to train a CNN to estimate an image's fingerprint. For instance, given an image $\boldsymbol{I}_0$ and a set of images $\{I_1, I_2, ..., I_n\}$ that belong to

the same camera model as $\boldsymbol{I}_0$, we can compute the ground truth fingerprint $\boldsymbol{F}$ using equation 3.2. The CNN is then attempting to directly learn the mapping of $\boldsymbol{I}_0 \rightarrow \boldsymbol{F}$ without the need of multiple images. However, different camera models leave distinct PRNU patterns – strong enough where specific patterns can be attributed to specific camera models[19]. This might cause problems when detecting deepfakes, as the fingerprints of images from distinct camera models might fall under distinct distributions. Instead, we train a CNN to extract a *pseudo fingerprint* $\hat{\boldsymbol{F}}$. This pseudo fingerprint should be similar for any real image, but still contain some unique attributes. In the end, the goal of NoiseLearner is to simply classify an image as fake or real, and we find that extracting a generalized *pseudo fingerprint* to all camera models to be more useful.

A lack of a ground truth to our *pseudo fingerprint* means that we don't necessarily know what it should look like. However, there is a list of assumptions we can make about the nature of a *pseudo fingerprint*:

1. **High patch-wise correlation:** The image's residual at a specific patch location should be highly correlated to the *pseudo fingerprint* at that patch. Conversely, a miss-matched residual and fingerprint patch should be lowly correlated. Recall that low-level artifacts in residuals are shrouded by content leakage[19]. Nevertheless, these artifacts are present in the residuals, and it's important to ensure that they are also present in the *pseudo fingerprint*. Miss-matched patches should be lowly correlated since real-image artifacts, like PRNUs, are of a stochastic nature and thus anchored to specific pixel locations[19], unless the image is transformed or cropped.

2. **Cross-image patch-wise correlation:** We further expand **Step 1** to also highly correlate the matching patches of an image's residual and a different image's pseudo fingerprint. What this means is that given two residual patches from 2 distinct im-

ages, $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$, the correlation of $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$'s generated fingerprint should have a high correlation, and vice-versa. Conversely, a miss-matched patch should be lowly correlated. Again, the procedure is essentially the same as **step 1**, but requires any real image to have similar low-level artifacts in the same pseudo fingerprint locations.

3. **Fingerprint spatial locality:** To further create a generalized *pseudo fingerprint*, we estimate that two fingerprints for two separate real images should be visually similar for a specific patch location. If the patch location is not equal, then the patches should produce different visual patterns. This assumption enforces a spatial locality in the *pseudo fingerprints*. This step is different from **step 2** as the comparison is being done on a pixel basis amongst the outputs.

Given these set of requirements, NoiseLearner first trains a *fingerprint discriminator* to enforce **steps 1 and 2**. Using a learned correlation metric allows us for fine-grained control in what constitutes a positive and negative correlation. For **step 3** we utilize a contrastive learning objective when training the **fingerprint generator**.

3.3.2 Fingerprint Deepfake Detection

Given that we have a *pseudo fingerprint*, how do we perform deepfake detection? Unlike other fingerprint methods such as NoiseScope[52], texture features might not be a strong identifier. Upsampling layers in GANS and other generative models imprint strong textural features in deepfakes[61]. Furthermore, deepfake images have a higher concentration of high-frequency features compared to real images[25], which also induces strong textural differences to real images. However, NoiseLearner cannot extract such features for deepfakes as it is never instructed to do so due to its unsupervised training process. Instead, we utilize a PCE metric with the original residual and its *pseudo fingerprint*.

Since the *pseudo fingerprint* only contains common patterns speficially for real residuals, the PCE correlation of the original residual and the *pseudo fingerprint* will be high for real images but low for fake images. This is because the extraction pipeline that generates the *pseudo fingerprint* (described in detail in the next section) is only trained on residuals of real images. Since fake images contain vastly different low-level artifacts and their respective *pseudo fingerprint* will not contain them, the PCE correlation will be substantially lower. We utilize this difference in PCE scores to perform unsupervised deepfake detection.

3.4 Detection Pipeline

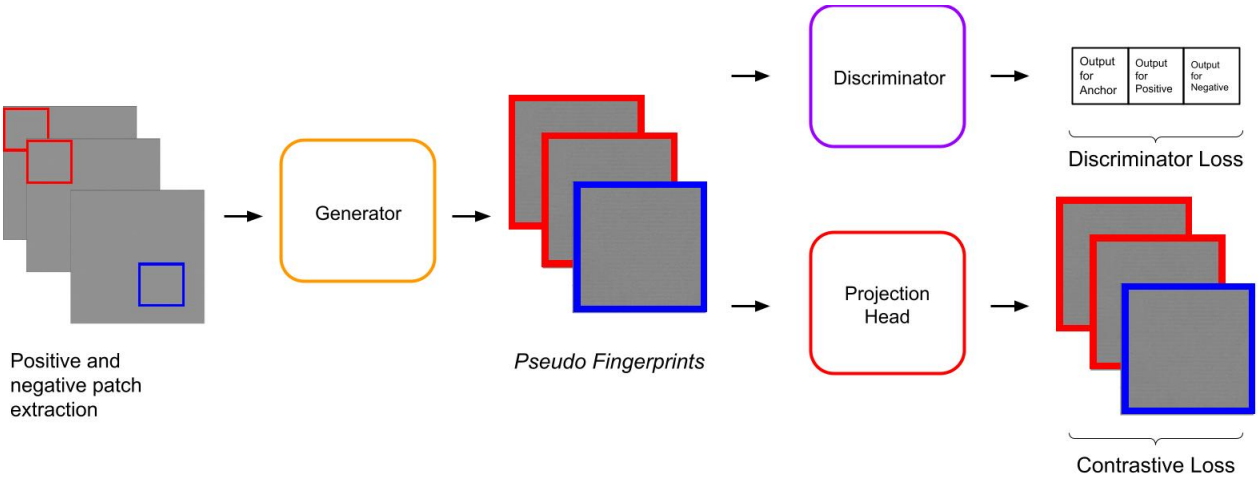

Figure 3.3: Fingerprint Generator Training Procedure

NoiseLearner is formed by 2 components: the fingerprint generator and the deepfake PCE detector. Figure 3.3 shows the training procedure of the fingerprint generator which utilizes the pre-trained *fingerprint discriminator*. The fingerprint generator extracts the *pseudo fingerprint* which contains artifacts specific to real images. The fingerprint discriminator and contrastive loss ensure that the *pseudo fingerprint* follows the constraints mentioned in 3.3.1

3.4.1 Fingerprint Discriminator

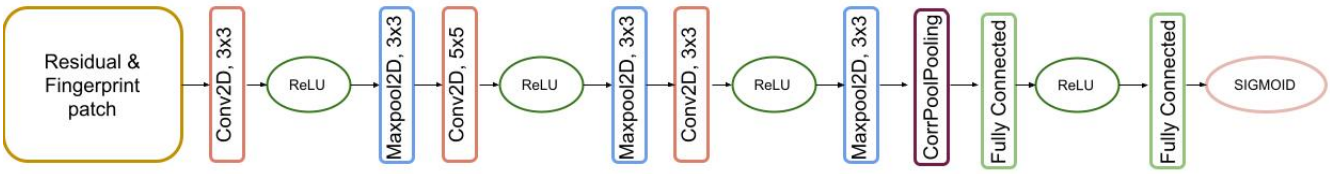

Figure 3.4: Fingerprint Discriminator Architecture

The goal of the fingerprint discriminator is to evaluate whether the fingerprint of an image and the residual are correlated. The architecture is heavily based on a CNN camera device source identification[48] and can be seen in figure 3.4. The main contribution of this network is the use of a correlation layer at the end of the network to enforce latent similarity, which can be thought as a learned correlation function. More importantly, if the given fingerprint and residual belong to the same camera model, the discriminator outputs a value of 1, 0 otherwise. The training of the fingerprint discriminator is performed on a patch-level basis, where positive samples are patches of an image's residual and corresponding fingerprint at the same location. Negative samples are mismatched patches, where the patch location of the residual and the fingerprint are different. This training procedure teaches the network to only output high correlation for residual-fingerprint patches of the same pixel location, which will be used to train the fingerprint generator to follow constraints 1 and 2 from 3.3.1.

Choosing training set for fingerprint discriminator

Our discriminator requires a large amount of high-resolution images to work efficiently. The high-resolution requirement allows us to extract larger patches of the image which is directly correlated with improved performance. The data also needs for the images to have the camera information to compute their respective *ground truth fingerprints*. The content of the images does not matter. This is because the residuals of the images are utilized during

training, which should be devoid of high-level content.

3.4.2 Fingerprint Generator

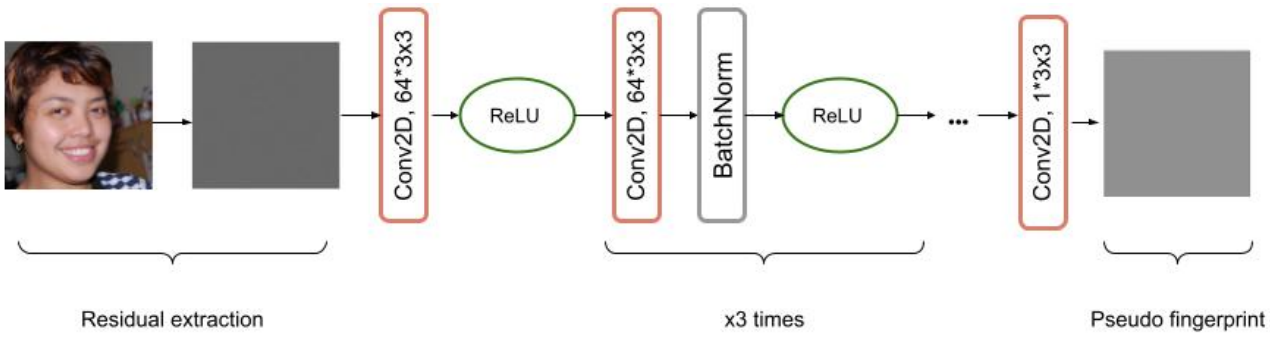

Figure 3.5: Fingerprint Generator Architecture

The fingerprint generator is trained to extract an image's *pseudo fingerprint*. The model architecture is very similar to DNCNN[60], a denoising convolutional neural network. Figure 3.5 shows the fingerprint generator's architecture. The goal of a denoising neural network is to extract the noise of a pristine image, which is then subtracted from the original image. The objective of the denoiser is very similar to a fingerprint extractor – to extract noise which contains important low-lever artifacts. The fingerprint we want to generate is not equivalent to a camera fingerprint and decide not to use camera label information during training. The goal of the fingerprint generator is to produce a generalized fingerprint for any real image. If camera labels were to be introduced, the network might learn to map distinct features to images from different camera models. This might cause fingerprints of distinct camera models to fall under different distributions, which would be problematic during the deepfake detection step.

Fingerprint generator loss terms

Due to a lack of ground-truth to compare the *pseudo fingerprint*, the constraints described in 3.3.1 are implemented through training objectives. Figure 3.3 goes over the training procedure for the fingerprint generator. *Spatial loss* is enforced through triplet loss [34], which attempts to maximize the distance between an anchor and a negative sample while minimizing the distance between an anchor and a positive sample. The triplet loss takes the form of :

$$L_{triplet}(a, p, n) = max(\|f(a) - f(p)\|^2 - \|f(a) - f(n)\|^2 + c, 0) \tag{3.3}$$

Where f is the fingerprint generator, a, p, n are the anchor, positive, and negative patches, and c is the margin. The greater the margin, the stricter is the model enforcing spatial locality. During training, anchor and positive patches are extracted from two random residuals from the same location regardless of camera model. The negative patch is extracted from a separate residual from a different patch location. Therefore, $L_{contrastive} = L_{triplet}$ and attempts to fulfil constraint 3 from 3.3.1.

The discriminator loss is enforced through the pre-trained fingerprint discriminator. We expect for the correlation of each of the residual samples a, p, n to be highly correlated to their generated fingerprints $f(a), f(p), f(n)$. Furthermore, we also expect for the anchor residual to be highly correlated to the positive generated fingerprint and vice-versa. This loss is quite simple and is enforced through binary-cross entropy loss and attempts to fulfil constraints 1,2 from 3.3.1. The loss term is calculated as follows:

$$L_{discriminator} = BCE(D(a, f_a)) + BCE(D(p, f_p)) + BCE(D(a, f_p)) + BCE(D(p, f_a)) + BCE(D(n, f_n)) \tag{3.4}$$

Where D is the fingerprint discriminator, and f_a, f_p, f_n are the corresponding *pseudo fingerprints* for a, p, n.Thus, the loss objective for the fingerprint generator is:

$$L_{fgen} = L_{contrastive} + L_{discriminator} \tag{3.5}$$

Projection head for contrastive loss

In practice, we found that adding a projection head to the end of the fingerprint generator for the contrastive loss objective improves performance. A projection head is simply a non-linear and shallow network that is added at the end of a model purely during training[20]. Loss terms are computed on the output of the projection head rather than the model output, and once training is finalized, the projection head is not utilized for inference. Other networks trained with contrastive loss terms [30] utilize a projection head since contrastive objectives cause information loss in the final layer.

For instance, if training an embedding space in an unsupervised manner, one might generate two positive samples by applying a set of random transformations to a source image. When training the model, we are telling it to produce the same embedding for these two augmented samples. However, in practice, the embedding for these augmented samples should be very similar but not equivalent. The usage of a projection head attempts to alleviate this concern. By directly disposing of the final layer, we force a sense of information loss that is beneficial to generate more accurate embeddings.

The same ideas can be applied to the fingerprint generator. We are indirectly minimizing

the L_2 distance of two patches. However, in practice, the output fingerprint patch for these two individual patches should be slightly different. In our case, we find that a single-layer CNN works best as a non-linear projection head.

Choosing training set for fingerprint generator

Our generator also requires a large amount of high-resolution images to work efficiently due to larger patches improving the performance of the generator. Unlike the discriminator, we do not need the data to contain camera information. The content of the images also does not matter due to the fingerprint generator extracting the *pseudo fingerprint* from the image's residual, not the image itself.

3.4.3 Deepfake PCE Detector

We observe that the PCE score of an image's residual and the image's pseudo fingerprint is higher for real images than for deepfakes. This is due to the **fingerprint generator** being trained to extract low-level artifacts only present in real images. The generated fingerprint will only contain these features for real images, which directly produces a higher PCE score. Section 3.3.2 goes into more detail regarding this idea.

We have found two ways to perform deepfake detection using PCE.

PCE percentile

Deepfake detection is done by first computing the fingerprints of a large number of real images, denoted as the reference set, and then extracting the PCE scores of the original residual and pseudo fingerprint. A cumulative distribution function (CDF) of these PCE

scores is computed which is used to estimate a threshold by choosing a percentile. Currently, the percentile is estimated manually, and we have found the 10th percentile of the CDF a decent threshold for most datasets. The PCE score at this threshold is used to classify future unseen images as fake or real– images with a PCE score below the threshold are classified as fake.

PCE sliding threshold

In order to properly assess the quality of the PCE percentile method, we want to estimate the best threshold for which the metrics are maximized. That is, at what PCE threshold do we achieve the best F1/P/R and accuracy for a given dataset. Compared to the PCE percentile method, the PCE sliding threshold does not require a reference set. Instead, the PCE scores for a set of fake and real images (about 500 per class) is calculated and the threshold with regards to the accuracy is maximized for this batch of images. We consider this evaluation important as it allows us to judge the amount of information loss in the PCE percentile method. This metric also allows us to appraise the potential of NoiseLearner in datasets where the PCE percentile test might not perform as well.

3.4.4 Usage of Image Residuals

In our thesis, we always utilize the image's residual instead of the original image. This is true for the fingerprint discriminator, fingerprint generator, and both deepfake PCE detection methods. We utilize the image's residual for three main reasons: (1) The residual extraction scheme might not be differentiable. In order to train both the fingerprint discriminator and generator we need for every operation to be differentiable. (2) We don't want to fixate on a singular residual extraction scheme. Instead, we allow for the defender to use the residual

extraction that they see most fit. Different residual filters might provide different benefits for specific datasets. (3) working on the residuals directly allows us to follow constraints 1 and 2 from 3.3.1.

Chapter 4

Evaluation

4.1 Experimental Setup

4.1.1 NoiseLearner Configuration

Fingerprint Discriminator Configuration

The fingerprint discriminator is trained on FFHQ data from 90 different cameras. FFHQ is a dataset purely composed of 1024x1024 headshots [38]. Since images from the same camera model share the same fingerprint, the dataset is split into subsets of camera labels: 50, 30, and 10 for training, testing, and evaluation. This leads to a total of 10014 images for training, 4539 images for testing, and 995 images for evaluation. Goljan et al's wavelet residual extractor[11] is used to extract the residuals. Furthermore, residuals and fingerprints are normalized to follow $U[-1, 1]$. During training patches of 256x256 are extracted from the residuals and a learning rate of 0.001 is used.The model is trained for 100 epochs. The model is optimized through Adam with betas 0.9, 0.999.

Fingerprint Generator Configuration

The fingerprint generator uses the same subset of of FFHQ data and residual extraction scheme as the fingerprint discriminator. The fingerprint generator uses patches of 256x256

and input residuals are also normalized to follow $U[-1, 1]$. During training, a learning rate of 0.001 is used and both loss terms, $L_{contrastive}$ and $L_{discriminator}$, are unweighted. $L_{contrastive}$ is Triplet loss[34] which uses L_2 loss as the distance metric and a margin of 1. The model is trained for 100 epochs optimized through Adam with betas 0.9, 0.999.

The projection head is trained concurrently with the fingerprint generator and uses the same training procedure and hyperparameters as it.

Unlike the fingerprint discriminator, the fingerprint generator does not use camera information during training. Therefore, when the anchor, positive, and negative patches are extracted, the camera label is not factored at all.

Deepfake PCE Detector Configuration

The PCE percentile method requires a reference dataset to compute a CDF from which to select a threshold. For face datasets, 2,000 random images are selected from the FFHQ testing subset. For other content datasets that do not have a subset of reference images, a random subset of 2,000 images is extracted to form the reference set. To match similar procedures, the residuals are extracted using the same wavelet filter[11] and are normalized to follow $U[-1, 1]$. The threshold is automatically chosen to be the 10th percentile of the CDF. For testing a specific dataset, 500 real images (not belonging to the reference set) and 500 fake images are used which follow the same residual extraction and normalization procedures.

The PCE sliding threshold configuration is identical to the PCE percentile method but without the need of a reference set. The threshold is directly maximized from the set of 500 real and 500 fake PCE scores.

4.1.2 Datasets

One of the contributions of the thesis is to evaluate NoiseLearner on a diverse set of datasets. Initially, we evaluate on the same datasets as NoiseScope. We also add other datasets we consider important. We include other PGGAN datasets in our evaluation as we noticed worse performance on the *pggan tower* dataset. We also add a *cyclegan satellite* as it has been shown to be a new type of deepfake that can be dangerous to national security affairs. Finally, we evaluate numerous state-of-the-art generative models, such as StyleGAN2, StyleGAN3, and guided denoising diffusion probabilistic models.

Face datasets

All face datasets are tested on the same subset of 2,500 FFHQ images. FFHQ is a dataset of 1024x1024 head-shots collected from Flickr[38].

StyleGAN-face-1 Is a dataset of human faces at a 1024x1024 resolution. Fake images are sampled from a pre-trained StyleGAN model on FFHQ data from the StyleGAN repository[38].

StyleGAN-face-2 Is a dataset released by Generated Media, Inc. [10]. The model used to generate the data is trained using 29,000 pictures of 69 different models. This dataset is of higher quality than StyleGAN Face 1, and it is unknown whether the generated images were further preprocessed.

StyleGAN2 generates similar images as StyleGAN with the same 1024x1024 resolution from a pre-trained model on FFHQ data [43] . Fake samples are collected from the official github repository[7].

StyleGAN3 makes drastic changes to the generator architecture and aims to produce rotation and translation invariant deepfakes[39]. For this model, we sample images from a

pre-trained model on FFHQ data[8]. More specifically, we sample from the translation invariant (StyleGAN3-t), as it has a more simple architecture compared to the rotation invariant model.

PGGAN-face uses a pretrained PGGAN model on CelebA dataset[37] to generate 1024x1024 face images. The samples are collected from the official github repository[6]

Other content

BigGAN-dogLV and BigGAN-dogHV are datasets of 256x256 french bulldog images generated by a BigGAN[16] model on Imagenet data. Fake images are gathered from the official BigGAN repository [1]. BigGAN-dogLV and BigGAN-dogHV generate images of lower and higher variety by setting the truncation to 0.2 and 0.86 respectively. Due to the low number of real images in the Imagenet dataset[54], we use an additional 4,390 dog images collected by crawling Flickr[52].

BigGAN-burgerLV and BigGAN-burgerHV are datasets of 256x256 cheeseburger images generated by a BigGAN[16] model on Imagenet data. The data collection is identical to BigGAN-dogLV and BigGAN-dogHV, and a total of 4,390 of real images are collected in the same manner [52].

CycleGAN-winter is a dataset of 256x256 winter landscape images generated from a CycleGAN model trained on Yosemite summer landscape images from Flickr. Fake images are generated by using a pre-trained model on the official CycleGAN repository[2]. Since CycleGAN is a style-transfer model which requires real images to generate fake images, an additional 4,594 summer landscape images are collected from Flickr[52]. A subset of this batch is dedicated to generate fake samples while the rest is used as real images for testing.

CycleGAN-zebra is a dataset of 256x256 zebra images. The data collection procedure is

identical as CycleGAN-winter, and a total of 11,241 real images are collected in the same manner.

CycleGAN-satellite is a dataset of 256x256 satellite images. Images are collected from a CycleGAN model trained on style transfer of Google maps satellite imagery of New York, Seattle, and Beijing[62]. Both fake and real images are collected from the official repository[3].

PGGAN-tower is a dataset of 256x256 tower images generated by a PGGAN[37] model trained on LSUN tower data[58]. Fake samples are collected from the official github repository[6] while real samples are collected from the tower category in the LSUN dataset[58].

PGGAN-bedroom is a dataset of 256x256 bedroom images. The data collection procedure is identical as PGGAN-tower but for bedroom images instead.

PGGAN-bicycle is a dataset of 256x256 bicycle images. The data collection procedure is identical as PGGAN-tower but for bicycle images instead.

StyleGAN-bedroom is a dataset of 256x256 bedroom images generated by a StyleGAN[38] model trained on LSUN bedroom data[58]. Fake samples are generated by NVIDIA and collected from the official repository [9] and real samples are collected from the bedroom category in the LSUN dataset[58].

ADM-horse is a dataset of 256x256 horse images generated using an ablation diffusion model (ADM), which is a guided denoising diffusion model[24], trained on LSUN horse data[58]. Fake samples are collected from the official repository [4] and real samples are collected from the horse category in the LSUN dataset[58].

ADM-bedroom is a dataset of 256x256 bedroom images generated using an ADM trained on LSUN bedroom data[58]. The data collection procedure is identical to ADM-horse but

for bedroom images instead.

4.1.3 Evaluation Metrics

For all evaluations we consider a mix of F1, precision, and recall. All these metrics take the fake class, aka deepfakes, as the positive class. So a true positive would be a deepfake classified correctly, and a true negative a pristine image classified correctly. For evaluating NoiseLearner, we provide results for the PCE percentile method and PCE sliding threshold method described in 3.4.3.

4.1.4 Baseline methods

In order to properly evaluate NoiseLearner, it is crucial to compare it to similar deepfake detection methods. We evaluate 4 different unsupervised methods: NoiseScope[52], Noiseprint[22], autoencoder, and CSD-SVM[45]. We also describe the setup of each baseline in detail.

NoiseScope

NoiseScope[52] is the most similar unsupervised detection framework to NoiseLearner. For evaluating each dataset, NoiseScope requires a reference set from which a threshold is calibrated. Furthermore, NoiseScope uses the reference set to train an outlier detector based on textural features[29]. To detect deepfakes, NoiseScope uses a clustering algorithm where image residuals with high PCE values are clustered in an iterative manner. Once the cluster step is done, NoiseScope will compute the fingerprints of each large cluster and pass them through the outlier detector to classify as fake or real. It is important to point out that Nois-

eScope requires a large set of both fake and real images to function whereas NoiseLearner only needs a single image.

NoiseScope setup: We utilize the official NoiseScope implementation[5] and train an outlier detector on the textural features of a reference set for each of the datasets described in 4.1.2. We then run the NoiseScope clustering algorithm for each dataset in 4.1.2 with a subset of 500 fake and 500 real images. We report the F1 score for the fake class.

Noiseprint

Noiseprint[22] is similar to NoiseLearner as it trains a CNN to extract fingerprints from image residuals. Noiseprint's main use-case is to detect forgery in images, but the authors show that Noiseprint can also detect GAN generated images. Since Noiseprint does not contain an end-to-end detection method and is purely used as a visual method to detect forgery, we evaluate Noiseprint by utilizing NoiseLearner's PCE detection step using fingerprints generated by Noiseprint. We want to see whether Noiseprint's fingerprints can be used as a substitute to NoiseLearner's fingerprint generator. We find it enough to use the PCE sliding threshold test as it provides enough information regarding the effectiveness of Noiseprint for deepfake detection.

Noiseprint setup: We utilize a pre-trained Noiseprint model trained on images from 125 distinct cameras. We use Noiseprint as a replacement for our fingerprint generator and evaluate it using the sliding PCE threshold from 3.4.3. We perform this evaluation for a subset of the datasets in 4.1.2 on 500 fake and 500 real images.

Autoencoder

The autoencoder baseline is perhaps the most simplistic one. An autoencoder is trained to reconstruct a set of real images. If deepfakes are from a different distribution than real images, the reconstruction error for the deepfakes will be higher than real images. We calculate a reconstruction threshold where any reconstruction error above it is classified is fake. The threshold is set to maximize the accuracy of the test set, which is valid since the dataset is balanced. We believe that this form of evaluation, while not being blind, does its job as a valid baseline.

Autoencoder setup: We train an autoencoder to reconstruct the reference set of a subset of the datasets in 4.1.2. Once the autoencoder has been trained, we compute the reconstruction scores of 500 fake and 500 real images by taking the L_2 distance of the original image and the reconstruction. The classification threshold is set to maximize the accuracy, similar to the PCE sliding threshold in 3.4.3.

CSD-SVM

CSD-SVM (Colorspace Discriminator SVM) is a blind detection scheme[45] which exploits differences in color components in HSV and YCbCr color spaces between deepfakes and pristine images. It does so by extracting features from residual images in these color spaces and training a one-class support vector machine (SVM) for anomaly detection. This means that this approach only requires real data to train, and features in HSV and YCbCr color spaces for deepfakes will be flagged as anomalous by the SVM. This method also requires a reference set to train the SVM but can be evaluated on a single-image basis.

CSD-SVM setup: We train an SVM on the residuals of the reference set for each dataset in 4.1.2. We perform classification on a set of 500 fake and 500 real images and report F1

for the fake class.

4.2 Results

As seen in table 4.1, we find that NoiseLearner performs well for the majority of datasets. NoiseLearner is able to achieve 90%+ F1 on most datasets, while others guarantee close to 85%. Apart from good performance on most datasets, other key findings are summarized as follows:

NoiseLearner's fingerprint generator produces genuine fingerprints. The results where we use Noiseprint[22] fingerprints instead of our own pseudo fingerprints from the fingerprint generator (Table 4.2) show that our *pseudo fingerprints* contain important low-level information, at least regarding deepfake detection tasks. If the fingerprint process extraction was arbitrary, we would see performance similar or worse to Noiseprint. However, we are able to achieve high F1 scores in many datasets where Noiseprint struggles, further proving our claims that the *pseudo fingerprint* is indeed authentic. Furthermore, Noiseprint is being evaluated utilizing the PCE threshold method where the threshold is set to maximize the results. Despite this, NoiseLearner with the PCE percentile threshold always outperforms.

NoiseLearner has close performance to NoiseScope despite having a less restrictive procedure. Table 4.2 demonstrates that NoiseScope has on average a higher F1 score than NoiseLearner. However, it is important to point out that NoiseScope requires a large set of both fake and real images in the test set to work well[52]. NoiseLearner only requires one image to perform deepfake classification due to a straight-forward process, and is able to perform comparatively well despite this harsh constraint. Therefore, even though both methods are being tested under the same circumstances (500 fake vs 500 real), NoiseLearner will be able to outperform NoiseScope in conditions where the number of real and fake images

decrease.

NoiseLearner performs worse on PGGAN datasets. NoiseLearner struggles with PGGAN datasets when compared to other generative models as seen in table 4.1. For instance, *StyleGAN-bedroom* and *PGGAN-bedroom* are both trained using the *LSUN bedroom* dataset, but the performance is much better for the *StyleGAN* counterpart. We also theorize that this is due to a property of the data: the quality of real images (LSUN) is poor compared to others. Datasets such as *StyleGAN Face*, *CycleGAN Zebra*, and *BIGGAN dog* utilize high-quality images from Flickr. However, LSUN is composed of extremely low-quality images that contain diverse content, watermarks, and heavy post-processing[58]. Currently, NoiseLearner is extremely sensitive to content not produced by cameras and thus might return PCE values for post-processed images within the range of deepfake images.

PCE percentile method does not maximize performance. Overall we notice that the current thresholding method does not maximize the results. The performance for a given dataset is maximized when NoiseLearner approaches NoiseLearner-M's performance, which can be seen in table 4.1. This is especially apparent for datasets where the ideal PCE threshold is very high for fake images, such as *StyleGAN3*. StyleGAN3 can achieve an F1 70% in the thresholding test (NoiseLearner-M), but fails at around 20% in the percentile test. This is because the threshold calculated from the FFHQ reference set estimated a threshold of 20,000 – the 10th percentile of the CDF calculated from the FFHQ reference set. However, the best estimated threshold for StyleGAN3 using the PCE sliding method is 44,000. In order to achieve this performance, the 40-50th percentile of the CDF would need to be extracted, which would be unfeasible for the remaining datasets that require an FFHQ reference set.

All methods perform poorly on advanced models, but NoiseLearner shows the most promise. None of the methods, including NoiseScope, perform well on the majority

Type	Dataset	NoiseLearner-M			NoiseLearner		
		F1	Precision	Recall	F1	Precision	Recall
Diverse models and content	StyleGAN-face1	97.71	97.23	98.20	93.40	96.31	93.58
	StyleGAN-face2	91.75	90.15	93.40	89.78	92.75	87.00
	PGGAN-face	92.50	95.14	90.00	92.45	93.10	91.80
	BigGAN-dogLV	88.65	87.52	89.80	87.14	90.52	84.00
	BigGAN-dogHV	89.90	94.43	84.80	89.54	90.93	88.20
	BigGAN-burgerLV	92.48	91.57	93.40	92.40	90.11	94.80
	BigGAN-burgerHV	95.52	97.30	93.80	93.33	90.28	96.60
	CycleGAN-winter	96.52	98.74	94.40	93.55	90.17	97.20
	CycleGAN-zebra	87.10	89.98	84.40	87.09	89.29	85.00
	CycleGAN-satellite	92.28	88.16	96.80	89.92	88.87	91.00
	PGGAN-tower	62.34	66.59	58.60	24.76	63.11	15.40
	PGGAN-bicycle	70.43	71.08	69.80	55.59	82.94	41.80
	PGGAN-bedroom	77.15	76.32	78.00	62.32	86.22	48.80
	StyleGAN-bedroom	84.04	82.34	85.80	81.02	90.39	73.40
Advanced models	StyleGAN2	98.02	97.24	98.80	96.51	93.61	99.60
	StyleGAN3	69.17	62.52	77.40	9.29	43.33	5.20
	ADM-horse	28.66	54.80	19.40	19.58	53.10	12.00
	ADM-bedroom	63.64	52.92	79.80	17.91	57.61	10.60

Table 4.1: Performance of NoiseLearner **M** denotes the performance when the threshold is maximized for accuracy.

of the advanced models as seen in table 4.2. NoiseLearner performs the best on StyleGAN2. However, StyleGAN3 and the diffusion models (denoted as ADM) all fail to perform well for any of the defenses. We theorize that it is due to vastly different training procedures that these two models utilize. StyleGAN3, while being a GAN, redesigns the architecture to remove any alias. Aliasing was seen to cause many of the artifacts commonly seen in StyleGAN-produced images. However, this lack of aliasing in the network might have monumentally reduced the amount of GAN-artifacts in StyleGAN3. Regardless, NoiseLearner shows the most promise as seen in table 4.2. NoiseScope completely fails to execute on StyleGAN3, ADM-horse, and ADM-bedroom. NoiseLearner produces results, but has room for improvement.

Dataset	F1					
	Noiseprint	Autoencoder	CSD-SVM	NoiseScope	NoiseLearner-M	NoiseLearner
StyleGAN-face1	75.02	64.67	92.93	99.56	97.71	93.40
StyleGAN-face2	88.81	66.84	67.53	90.14	91.75	89.78
PGGAN-face	—	—	64.07	99.09	92.50	92.45
BigGAN-dogLV	67.11	70.26	86.94	99.38	88.65	87.14
BigGAN-dogHV	66.31	66.93	70.10	92.6	89.90	89.54
BigGAN-burgerLV	—	—	94.82	99.68	92.48	92.40
BigGAN-burgerHV	—	—	83.67	98.64	95.52	93.33
CycleGAN-winter	66.67	66.80	87.14	92.40	96.52	93.55
CycleGAN-zebra	—	—	84.95	92.84	87.10	87.09
CycleGAN-satellite	—	—	71.07	89.89	92.28	89.92
PGGAN-tower	63.89	45.73	91.61	95.93	62.34	24.76
PGGAN-bicycle	—	—	69.62		70.43	55.59
PGGAN-bedroom	—	—	95.06	95.33	77.15	62.32
StyleGAN-bedroom	67.37	66.67	94.82	99.63	84.04	81.02
StyleGAN2	—	—	61.87	52.38	98.02	96.51
StyleGAN3	—	—	62.26	N/A	69.17	9.29
ADM-horse	—	—	62.27	N/A	28.66	19.58
ADM-bedroom	—	—	64.34	N/A	63.64	17.91

Table 4.2: Baseline Performances Compared to NoiseLearner

Contrastive loss provides slight performance boost on some datasets. We noticed from our ablations in table 4.3 that our fingerprint generator trained solely with the discriminator performs very similarly to the original model. The performance seems to be quite erratic, being better for some datasets (StyleGAN-face2, StyleGAN3, BigGAN-dogLV, etc) and worse for others (StyleGAN-bedroom, PGGAN-bedroom, etc). It seems that the inclusion of contrastive loss slightly improves the performance for advanced model. We did not include the ablation of the fingerprint generator only trained with contrastive loss as the performance was near random for most datasets.

From a simple point of view, contrastive loss seems to add little benefit to the model. It seems that the inclusion of requirement 2 for the fingerprint generator 3.3.1 takes care of any spatial information amongst different images of different cameras. Furthermore, spatial locality does not induce strong enough information by itself to produce distinctive pseudo

fingerprints.

	F1			
Dataset	NoiseLearner-DM	NoiseLearner-D	NoiseLearner-M	NoiseLearner
StyleGAN-face1	96.72	96.20	97.71	93.40
StyleGAN-face2	90.23	84.00	91.75	89.78
StyleGAN2	96.84	96.20	98.02	96.51
StyleGAN3	68.22	0.37	69.17	9.29
PGGAN-face	92.64	90.78	92.50	92.45
BigGAN-dogLV	85.12	85.07	88.65	87.14
BigGAN-dogHV	89.62	88.18	89.90	89.54
BigGAN-burgerLV	93.67	92.08	92.48	92.40
BigGAN-burgerHV	95.59	92.91	95.52	93.33
CycleGAN-winter	98.28	95.16	96.52	93.55
CycleGAN-zebra	90.57	90.15	87.10	87.09
CycleGAN-satellite	95.53	93.68	92.28	89.92
PGGAN-tower	76.77	18.58	62.34	24.76
PGGAN-bedroom	80.75	72.29	77.15	62.32
PGGAN-bicycle	70.65	59.38	70.43	55.59
StyleGAN-bedroom	86.32	86.23	84.04	81.02
ADM-horse	3.50	14.88	28.66	19.58
ADM-bedroom	62.92	14.19	63.64	17.91

Table 4.3: NoiseLearner ablation results. Models denoted with **D** are those trained solely with *discriminator loss*. **M** denotes the performance when the threshold is maximized for accuracy on the model.

Chapter 5

Discussion

Overall, NoiseLearner meets the initial expectations of an unsupervised and content-agnostic deepfake detection method. However, there is a list of improvements that can be made to the methods:

5.1 Overall Improvements for Advanced Model Datasets

The obvious first step would be to improve NoiseLearner in the datasets that it is lacking at. Table 4.1 shows these datasets to be PGGAN, denoising diffusion models, and StyleGAN3. We discuss several ways that our method can be improved to tackle improved performance for advanced models. We mostly focus on advanced models since we attribute worsened performance on PGGAN datasets due to the LSUN[58] dataset that was used for both training and evaluating the models. We observe a large percentage of NoiseLearner's false positives for these PGGAN datasets (real images classified as fake) to be of dubious quality. Figure 5.1 shows some examples of nature of such images.

5.1.1 Contrastive Loss Improvements

As seen in table 4.3, the use of contrastive loss, on average, does not improve the performance of NoiseLearner. Initially when NoiseLearner did not have constraint 2 from 3.3.1, the

Figure 5.1: False positives for the PGGAN-tower dataset when evaluated with NoiseLearner. Most of these images are heavily processed and contain watermarks, or are heavily photoshopped. Also notice how some of these "real" images are either illustrations (top left) or screenshots of videogames (bottom left). In these cases NoiseLearner will fail on such images due to two reasons: (1) the image does not contain camera artifacts, (2) the image has been so heavily tampered that the camera artifacts have been destroyed.

fingerprint generator trained only with discriminator loss had a much worsened performance. We theorize that the inclusion of the second constraint fulfills a similar duty as contrastive loss.

Rather than not utilizing any form of contrastive loss, we weigh the benefits that the third constraint 3.3.1 brings to the table. Table 4.3 shows that the model trained with contrastive loss slightly outperforms the model solely trained with the discriminator loss for the advanced datasets. We have already explored different forms of contrastive loss, such as infoNCE[20,

30]. InfoNCE differs from triplet loss as it is tuned to work solely on low-dimensional embeddings (i.e, a 1024-D vector) rather than on high dimensional images (i.e a 512x512 images). In order to utilize these more advanced contrastive losses, we require to first turn the output of our fingerprint generator into a low-dimensional embedding. To do so, we replace our projection head (which is a simple one-layer CNN) with a Resnet18 network without the classification head pre-trained on Imagenet[31]. The idea of using a pre-trained network without the classification head is to get our *pseudo fingerprints* into a pre-trained embedding space. Ideally, similar fingerprints should have similar embeddings.

We find out that the usage of a pre-trained Resnet18 network and infoNCE loss does not perform as well as our current architecture. We theorize that the embeddings produced by the Resnet18 network are not very useful. For a test, we extracted the *pseudo fingerprints* for different Imagenet images and ran them through the same pre-trained Resnet18 network with the classification heads. We noticed that most *pseudo fingerprints* were classified as either cracked windows, spider-webs, or other classes that contain very high-frequency components – regardless of the *pseudo fingerprint* of origin. Since most *pseudo fingerprints* are classified in the same cluster classes, it is safe to say that the embeddings produced by the Resnet18 network are not valuable to gauge the textural differences amongst *pseudo fingerprints* of different images. If the Resnet18 architecture were trained on either *pseudo fingerprints* or image residuals, we believe that the embedding space would be much more representative of the image data and thus allow to reap the benefits of infoNCE as a contrastive loss term.

5.1.2 Retraining fingerprint generator and discriminator on distinct data

One of the achievements of NoiseLearner is that it can detect deepfakes from different domains despite being trained on face data. However, would performance improve if we trained on more specific datasets? That is, if our goal is to detect tower images generated by PG-GAN, what would happen if we trained our fingerprint generator and discriminator with LSUN tower data? We explore two ways that this could be approached:

Fine-tuning fingerprint generator: This approach attempts to fine-tune our fingerprint generator which has already been pre-trained on FFHQ data. The core idea is to continue training but replace the FFHQ images with images of the content we want to fine-tune our generator to. This approach, while simplistic, might prove harder than expected. It is important to recognize that currently the fingerprint generator is trained on patches of 256x256 residuals extracted from 1024x1024 residuals. If the domain we want to fine-tune into does not have high-resolution data, this approach might fail to produce valuable results.

Training fingerprint generator and discriminator from scratch: This method attempts to train both fingerprint discriminator and generator from scratch using only images from the targeted domain (i.e LSUN tower images). Since the images of the target domain are always smaller, training is done on patches of a smaller resolution (i.e 64x64 for 256x256 images). Also, since these images do not have camera information, we instead compute *improvised fingerprints* from clusters of random images to train the discriminator. The fingerprint generator is trained in the same manner but with the updated patch-size. We find for this method to be difficult to implement. (1) Part of the reason on why NoiseLearner works well is due to the large patch-size of 256x256. A patch size of about 64x64 contains much less information, and might fail to match our current performance. (2) If the images from the

domain we want to train on lack camera information, the improvised fingerprint might not be good enough to train an accurate fingerprint discriminator. Camera label information plays a very important role in training our current fingerprint discriminator, and an improvised fingerprint might have trouble matching the discriminator's needs.

5.1.3 Expanding the training dataset for both fingerprint generator and discriminator

We also explore the idea of using more diverse data when training both the fingerprint generator and discriminator. Originally we only use FFHQ data from 50 camera models and find that NoiseLearner generalizes well enough to different content, such as landscapes and dogs. However, the inclusion of data encompassing different camera models and domains might prove useful, especially to datasets of a high-frequency nature (i.e PGGAN tower). We therefore tried to include the Alaska dataset[21], which contains images of diverse content and resolutions from 40 different camera models. We only extract images which are at least 512x512 and resize them to this resolution. The inclusion of Alaska doubles our training size from 30,000 images to 60,000.

Overall, we believe this idea needs more nurturing before any conclusions can be drawn. We argue for future evaluations to also include Alaska, or other datasets with labeled camera information, to determine whether the diversity of our current training dataset is truly hindering our performance on the evaluation phase.

5.1.4 Usage of more complex architectures for the fingerprint generator

We also explore different architectures for the fingerprint generator. The current denoising-based CNN[60] is a very simplistic CNN, and we argue that perhaps a more complex network might extract better features in the *pseudo fingerprint*. We try Pix2Pix's U-net architecute[35], which uses an encoder-decoder architecture with skip-connections. The reason for this architecture change is that different layers in U-net are able to extract different features at different resolutions. Upper layers extract generalized features, while lower layers extract more specific low-level features. We especially are interested in how U-net can extract these low-level features where most camera artifacts, such as PRNU, are found. In practice we discover that U-net does not produce accurate *pseudo fingerprints* at all as the performance for most datasets is near random. We theorize that by using heavy downsampling and upsampling within U-net, the generated *pseudo fingerprint* contains heavy artifacts induced by the upsampling layers – the same artifacts that most deepfakes contain.

Therefore, we argue for a straight-forward fingerprint generator that reduces the number of upsampling operations. With that in mind, we also attempt to stack more convolutional layers to our DnCNN, but find a inverse relationship with performance gain and number of layers stacked. We believe this is due to the vanishing gradient problem found in common Deep Learning architectures [51].

5.1.5 Content Leakage in Residuals and Fingerprint

The generated fingerprints and residuals have too much content leakage. The image residual should ideally be devoid of content-leakage. Content leakage in the residual will directly lead to content-leakage in the generated *pseudo fingerprint*. Therefore, we theorize a couple

methods to tackle this challenge.

Use of frequency domain for loss term to attenuate content leakage

We attempt to improve our fingerprint generator by maximizing other features frequently found in fingerprints. We take inspiration from a paper published by engineers at Facebook[14] that trains a model to identify the model hyperparameters used to generate a deepfake. This method also has a fingerprint generator which is used as input to the model. The fingerprint generator is trained with respect to deepfake images, not real images. Thus, the loss terms enforce constraints and patterns typically found in deepfakes, often induced by upsampling layers. Regardless, we still find useful some of the loss terms, such as the *Spectrum Loss*.

Fingerprints tend to lie on the middle to high frequency bands[61]. Thus, the goal of the *spectrum loss* is to minimize the low frequency components of an image through:

$$L_{spectrum} = \|\mathscr{L}(\mathscr{F}(\boldsymbol{F}), k)\|^2 \tag{5.1}$$

Where $\mathscr{L}$ is a low-pass filter, $\mathscr{F}$ is the fourier transform of the *pseudo fingerprint* $\boldsymbol{F}$ for the kxk region around the center.

We find that using the *spectrum loss* is detrimental to NoiseLearner, and reduces performance to near random for many datasets. We theorize that the the reduction of low-frequency bands might erase camera artifacts, such as PRNU patterns, and thus produce PCE distributions similar to deepfakes. We also explore the inclusion of the *Repetitive loss* from the same work. Amin et al.[36] point out that an image's noise contains repetitive manners through the image. These repetitive patterns directly produce high frequency components in the fingerprint, so the *repetitive loss* attempts to maximize the high-frequency information of the *pseudo fingerprint* by:

$$L_{repetitive} = -max(H(\mathscr{F}(\boldsymbol{F}), k)) \tag{5.2}$$

Where H is a high-pass filter, $\mathscr{F}$ is the fourier transform of the *pseudo fingerprint* $\boldsymbol{F}$ for the kxk region around the center.

We find that the addition of *repetitive loss* with the *spectrum loss* does not change performance at all. We find that describing the noise characteristic of images as "repetitive" is disingenuous, as quoted by the facebook paper[14]. Amin et al[36] state that these repetitive noise patterns are specific to *spoof images*, images which are used to spoof authentication devices, such as facial recognition. These repetitive patterns are apparent in different spoof attacks, such as print attacks, where a person's face is printed on a sheet of paper and used for authentication. Deepfakes and pristine images have other sources of noise, but not spoof noise. Therefore, in future evaluations we dismiss the usage of the *repetitive loss*.

Usage of different residual extraction filters

Right now we only use wavelet filters[11] for our residual extraction process. We find that the wavelet filter often leaks high-level content onto the residual, which directly produces a leakage on the generated *pseudo fingerprint*. Thus, we believe that different residual extraction schemes might be able to better extract the noise-space of images. There are different state-of-the-art Deep Learning-based residual extraction filters that we believe might prove beneficial to the residual extraction process.

5.2 Automatic PCE thresholding

Our current Deepfake PCE detection works well for most datasets, but could still be improved as seen by the performance loss from NoiseLearner-M in table 4.1, where the threshold is

maximized (the best performance that can be achieved). Our current threshold is estimated as the 10th percentile of the CDF of a set of PCE values extracted from a reference set. It is a very simplistic approach, but any attempts to add complexity has resulted in total collapse.

We attempt to train an outlier detector, similar to NoiseScope[52], including haralick textural features [29] on-top of the aforementioned PCE score. The outlier detector does not perform well at all and can rearely flag deepfakes as outliers, which shows that texture is not a good feature from our *pseudo fingerprints*. We propose to keep using PCE, but seek for more effective ways to estimate this threshold.

5.3 Countermeasures

Our work has yet to be evaluated to different countermeasures. We propose a few evaluations to test the robustness of our method:

5.3.1 Fingerprint spoofing

It has been shown that PRNU artifacts from real images can be destroyed or replaced with another image's PRNU pattern[15, 44, 59]. It is a common attack for methods that utilize imaging forensics and fingerprint techniques. In our case the attack would be used to see whether the low-level artifacts from our *pseudo fingerprints* could be replaced or destroyed. This is an easy black-box attach as it does not require access to the model, having a set of real and fake *pseudo fingerprints* would be sufficient. The countermeasure evaluation should be similar to the fingerprint replacement attack [44], where a fake image's *pseudo fingerprint's* artifacts are embedded into a real image's *pseudo fingerprint*. If the attack is

successful, then it might be possible to bypass our current iteration of the defense.

5.3.2 Adversarial attack

This is a similar countermeasure as fingerprint spoofing and is also a black-box attack. The only necessities are a pristine image that NoiseLearner classifies as "real" and a deepfake image that NoiseLearner classifies as "fake." The attack would be performed using Carlini's L_2 attack[17]. The attack would minimize the pixel L_2 distance between the pristine and deepfake image such as that the overal content of the original image does not alter. This attack is a simple but effective attack as only two images are needed. We theorize that this attack could work as camera artifacts are sensitive to change and might be erased.

5.3.3 Using NoiseLearner to adapt generative models

We also explore the possibility of the attacker using NoiseLearner to adapt generative models. Can the attacker adapt their generative model using NoiseLearner's feedback to bypass the defense? For instance, can NoiseLearner be used as novel discriminator term when training a GAN model? We argue that NoiseLearner is robust to this attack as NoiseLearner is only trained on real images. If NoiseLearner is used as a loss metric, it would push for the generative model to not generate PRNU artifacts on deepfakes. However, the nature of generative models, including upsampling layers, will still imprint common deepfake artifacts, and thus still be classified as fake by NoiseLearner.

Chapter 6

Conclusions

The goal of this thesis was to develop a content-agnostic and unsupervised deepfake detection method. To do so, we introduce NoiseLearner, which leverages imaging forensics techniques to perform deepfake detection despite only being trained on real images.

NoiseLearner is composed of two parts, a fingerprint generator and a deepfake PCE detector. We train the fingerprint generator on a denoising CNN model using different constraints to generate generalized *pseudo fingerprints* for real images. These *pseudo fingerprints* are rich in camera artifacts and other low-level features such as PRNU pattern. We leverage information from an image's residual and its *pseudo fingerprint* to perform deepfake classification. The PCE score for a fake image will be lower for fake images as its *pseudo fingerprint* does not contain relevant information, such as PRNU patterns. Real images will have higher PCE scores since their *pseudo fingerprint* does contain relevant low-level information. We formulate two ways to draw this PCE threshold: (1) a sliding threshold which calculates the threshold to maximize the performance, (2) a percentile based threshold which is extracted by taking the 10th percentile of the CDF of a reference set of PCEs of real images. We find the sliding threshold method useful as it is the best achievable performance for a given dataset and allows us to compare whether the percentile method is maximizing its outcome.

Overall we find decent performance on most datasets except for PGGAN and advanced models. The issue with PGGAN datasets is that they are trained using low-quality data, which NoiseLearner is sensitive to. Advanced models have very robust generative model

procedures that erase many of the artifacts left behind by traditional GANs. For instance, StyleGAN3[39] completely revamps the StyleGAN architecture to reduce alias within the network. A lot of this alias was seen in the form of pixel-dependent additive noise and activation functions and upsampling/downsampling layers, which is now not present in images generated by StyleGAN3. Deepfake detection methods, including ours, depend on this alias to differentiate real and fake images, which explains the worsened performance.

We have also tried multiple different approaches to improve NoiseLearner. We have tried different generator architectures, new loss terms, and different datasets to train both fingerprint generator and discriminator. Overall we have not seen improvement from many of these trials, suggesting that unsupervised and content-agnostic deepfake detection is not an easy task.

Bibliography

[1] Official biggan repo. URL https://tfhub.dev/deepmind/biggan-deep-256/1.

[2] junyanz/cyclegan: Software that can generate photos from paintings, turn horses into zebras, perform style transfer, and more. . URL https://github.com/junyanz/CycleGAN.

[3] Fake satellite imagery. . URL https://figshare.com/s/eeedcd150e759ef4353c.

[4] openai/guided-diffusion. URL https://github.com/openai/guided-diffusion.

[5] GitHub - jmpu/NoiseScope: The implementation of ACSAC 2020 paper: "NoiseScope: Detecting Deepfake Images in a Blind Setting". URL https://github.com/jmpu/NoiseScope.

[6] Pggan official 100k-generated-images. URL https://drive.google.com/drive/folders/1j6uZ_a6zci0HyKZdpDq9kSa8VihtEPCp.

[7] Stylegan2 official 100k-generated-images. . URL https://drive.google.com/drive/folders/1BA2OZ1GshdfFZGYZPob5QWOGBuJCdu5q.

[8] Nvlabs/stylegan3: Official pytorch implementation of stylegan3. . URL https://github.com/NVlabs/stylegan3.

[9] Stylegan bedroom, 256x256. . URL https://drive.google.com/drive/folders/1Vxz9fksw4kgjiHrvHkX4Hze4dyThFW6t.

[10] Generated photos | unique, worry-free model photos. . URL https://generated.photos/.

[11] Camera fingerprint. URL http://dde.binghamton.edu/download/camera_fingerprint/.

[12] Malicious actors almost certainly will leverage synthetic content for cyber and foreign influence operations. 2021. URL https://www.reuters.com/article/us-cyber-deepfake-activist/deepfake-used-to-attack-activist-couple-.

[13] Darius Afchar, Vincent Nozick, Junichi Yamagishi, and Isao Echizen. MesoNet: A compact facial video forgery detection network. *10th IEEE International Workshop on Information Forensics and Security, WIFS 2018*, jan 2019. ISSN 2331-8422. doi: 10.1109/WIFS.2018.8630761.

[14] Vishal Asnani, Xi Yin, Yal Hassner, and Xiaoming Liu. Reverse engineering of generative models: Inferring model hyperparameters from generated images. URL https://arxiv.org/pdf/2106.07873.pdf.

[15] Sudipta Banerjee, Vahid Mirjalili, and Arun Ross. Spoofing prnu patterns of iris sensors while preserving iris recognition.

[16] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. URL https://tfhub.dev/s?q=biggan.

[17] Nicholas Carlini, Google Brain, and Hany Farid. Evading deepfake-image detectors with white-and black-box attacks.

[18] Hong-Shuo Chen, Mozhdeh Rouhsedaghat, Hamza Ghani, Shuowen Hu, Suwa You, and Jay Kuo. Defakehop: A light-weight high-performance deepfake detector. URL https://arxiv.org/pdf/2103.06929.pdf.

[19] Mo Chen, Jessica Fridrich, Miroslav Goljan, and Jan Lukáš. Determining image origin

and integrity using sensor noise. *IEEE Transactions on Information Forensics and Security*, 3:74–90, 3 2008. ISSN 15566013. doi: 10.1109/TIFS.2007.916285.

[20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. 2020. URL https://github.com/google-research/simclr.

[21] Rémi Cogranne, Quentin Giboulot, and Patrick Bas. The alaska steganalysis challenge: A first step towards steganalysis "into the wild". *IH and MMSec 2019 - Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, pages 125–137, 7 2019. doi: 10.1145/3335203.3335726.

[22] Davide Cozzolino and Luisa Verdoliva. Noiseprint: a cnn-based camera model fingerprint.

[23] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16:2080–2095, 8 2007. ISSN 10577149. doi: 10.1109/TIP.2007.901238.

[24] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. 5 2021. URL https://arxiv.org/abs/2105.05233v4.

[25] Joel Frank, Thorsten Eisenhofer, Lea Schönherr, Asja Fischer, Dorothea Kolossa, and Thorsten Holz. Leveraging frequency analysis for deep fake image recognition.

[26] Miroslav Goljan. Digital Camera Identification from Images – Estimating False Acceptance Probability. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5450 LNCS: 454–468, nov 2008. ISSN 03029743. doi: 10.1007/978-3-642-04438-0_38. URL https://link.springer.com/chapter/10.1007/978-3-642-04438-0_38.

[27] Miroslav Goljan, Jessica Fridrich, and Tomáš Filler. Large scale test of sensor fingerprint camera identification. *https://doi.org/10.1117/12.805701*, 7254:170–181, feb 2009. ISSN 0277786X. doi: 10.1117/12.805701. URL https://www.spiedigitallibrary.org/conference-proceedings-of-spie/7254/72540I/Large-scale-test-of-sensor-fingerprint-camera-identification/10.1117/12.805701.fullhttps://www.spiedigitallibrary.org/conference-proceedings-of-spie/7254/72540I/Large-scale-test-of-sensor-fingerprint-camera-identification/10.1117/12.805701.short.

[28] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. URL http://www.github.com/goodfeli/adversarial.

[29] Robert M. Haralick, K. Shanmugam, and Its'Hak Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621, 1973. doi: 10.1109/TSMC.1973.4309314.

[30] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. . URL https://github.com/facebookresearch/moco.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. . URL http://image-net.org/challenges/LSVRC/2015/.

[32] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *Advances in Neural Information Processing Systems*, 2017-December: 6627–6638, jun 2017. ISSN 10495258. URL https://arxiv.org/abs/1706.08500v6.

[33] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 2020-December, 6 2020. ISSN 10495258. URL https://arxiv.org/abs/2006.11239v2.

[34] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9370:84–92, 12 2014. ISSN 16113349. doi: 10.1007/978-3-319-24261-3_7. URL https://arxiv.org/abs/1412.6622v4.

[35] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A Efros, and Berkeley Ai Research. Image-to-image translation with conditional adversarial networks. URL https://github.com/phillipi/pix2pix.

[36] Amin Jourabloo, Yaojie Liu, and Xiaoming Liu. Face de-spoofing: Anti-spoofing via noise modeling.

[37] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 10 2017. URL https://arxiv.org/abs/1710.10196v3.

[38] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43:4217–4228, 12 2018. ISSN 19393539. doi: 10.1109/TPAMI.2020.2970919. URL https://arxiv.org/abs/1812.04948v3.

[39] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. 6 2021. URL https://arxiv.org/abs/2106.12423v4.

[40] Diederik P Kingma and Max Welling. Auto-encoding variational bayes.

[41] Praveen Krishnan, Rama Kovvuri, Guan Pang, Boris Vassilev, Tal Hassner, and Facebook Ai. Textstylebrush: Transfer of text aesthetics from a single example.

[42] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved Precision and Recall Metric for Assessing Generative Models. URL https://github.com/kynkaat/improved-precision-and-recall-metric.

[43] Jaakko Lehtinen. Analyzing and improving the image quality of stylegan tero karras nvidia samuli laine nvidia miika aittala nvidia janne hellsten nvidia. URL https://github.com/NVlabs/stylegan2.

[44] Chang Tsun Li, Chih Yuan Chang, and Yue Li. On the repudiability of device identification and image integrity verification using sensor pattern noise. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, 41 LNICST:19–25, 2010. ISSN 18678211. doi: 10.1007/978-3-642-11530-1_3. URL https://www.researchgate.net/publication/221294089_On_the_Repudiability_of_Device_Identification_and_Image_Integrity_Verification_Using_Sensor_Pattern_Noise.

[45] Haodong Li, Bin Li, Shunquan Tan, and Jiwu Huang. Identification of Deep Network Generated Images Using Disparities in Color Components. *Signal Processing*, 174, aug 2018. doi: 10.1016/j.sigpro.2020.107616. URL http://arxiv.org/abs/1808.07276http://dx.doi.org/10.1016/j.sigpro.2020.107616.

[46] Lingzhi Li, Jianmin Bao, Ting Zhang, Hao Yang, Dong Chen, Fang Wen, and Baining Guo. Face X-ray for More General Face Forgery Detection. URL https://29a.ch/photo-forensics/#noise-analysis.

[47] Yuezun Li and Siwei Lyu. Exposing deepfake videos by detecting face warping artifacts.

[48] Sara Mandelli, Davide Cozzolino, Paolo Bestagini, Luisa Verdoliva, Senior Member, and Stefano Tubaro. CNN-based fast source device identification. URL https://github.com/polimi-ispl/cnn-fast-sdi.

[49] Neal Mangaokar, Jiameng Pu, Parantapa Bhattacharya, Chandan K. Reddy, and Bimal Viswanath. Jekyll: Attacking medical image diagnostics using deep generative models. In *Proceedings of IEEE European Symposium on Security and Privacy (EuroS&P)*, 2020.

[50] Huy H Nguyen, Junichi Yamagishi, and Isao Echizen. Capsule-forensics: Using capsule networks to detect forged images and videos.

[51] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *30th International Conference on Machine Learning, ICML 2013*, pages 2347–2355, 11 2012. URL https://arxiv.org/abs/1211.5063v2.

[52] Jiameng Pu, Neal Mangaokar, Bolun Wang, Chandan K. Reddy, and Bimal Viswanath. Noisescope: Detecting deepfake images in a blind setting. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2020.

[53] Ali Razavi, Aäron van den Oord, and Oriol Vinyals DeepMind. Generating Diverse High-Fidelity Images with VQ-VAE-2. URL https://github.com/deepmind/sonnet/blob/master/sonnet/python/modules/nets/vqvae.py.

[54] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 9 2014. ISSN 15731405. doi: 10.1007/s11263-015-0816-y. URL https://arxiv.org/abs/1409.0575v3.

[55] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. *Advances in Neural Information Processing Systems*, pages 2234–2242, jun 2016. ISSN 10495258. URL https://arxiv.org/abs/1606.03498v1.

[56] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains.

[57] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in Neural Information Processing Systems*, 2020-December, 7 2020. ISSN 10495258. URL https://arxiv.org/abs/2007.03898v3.

[58] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. 6 2015. URL https://arxiv.org/abs/1506.03365v3.

[59] Hui Zeng, Jiansheng Chen, Xiangui Kang, and Wenjun Zeng. Removing camera fingerprint to disguise photograph source. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 1687–1691, 2015. doi: 10.1109/ICIP.2015.7351088.

[60] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. cszn/dncnn: Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising (tip, 2017). 2017. URL https://github.com/cszn/DnCNN.

[61] Xu Zhang, Svebor Karaman, and Shih-Fu Chang. Detecting and Simulating Artifacts in GAN Fake Images (Extended Version).

[62] Bo Zhao, Shaozeng Zhang, Chunxue Xu, Yifan Sun, and Chengbin Deng. Deep fake

geography? when geospatial data encounter artificial intelligence. *Cartography and Geographic Information Science*, 48:338–352, 2021. ISSN 15450465. doi: 10.1080/15230406.2021.1910075.

[63] Hanqing Zhao, Wenbo Zhou, Dongdong Chen, Tianyi Wei, Weiming Zhang, and Nenghai Yu. Multi-attentional deepfake detection. URL https://github.com/yoctta/.

[64] Jun Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-October:2242–2251, 3 2017. ISSN 15505499. doi: 10.1109/ICCV.2017.244. URL https://arxiv.org/abs/1703.10593v7.