NoiseScope: Detecting Deepfake Images in a Blind Setting

Jiameng Pu
Virginia Tech
jmpu@vt.edu

Neal Mangaokar
Virginia Tech
neal1998@vt.edu

Bolun Wang
Facebook
bolunwang@fb.com

Chandan K. Reddy
Virginia Tech
ckreddy@vt.edu

Bimal Viswanath
Virginia Tech
vbimal@vt.edu

ABSTRACT

Recent advances in Generative Adversarial Networks (GANs) have significantly improved the quality of synthetic images or *deepfakes*. Photorealistic images generated by GANs start to challenge the boundary of human perception of reality, and brings new threats to many critical domains, e.g., journalism, and online media. Detecting whether an image is generated by GAN or a real camera has become an important yet under-investigated area. In this work, we propose a blind detection approach called *NoiseScope* for discovering GAN images among other real images. A blind approach requires no a priori access to GAN images for training, and demonstrably generalizes better than supervised detection schemes. Our key insight is that, similar to images from cameras, GAN images also carry unique patterns in the noise space. We extract such patterns in an unsupervised manner to identify GAN images. We evaluate *NoiseScope* on 11 diverse datasets containing GAN images, and achieve up to 99.68% F1 score in detecting GAN images. We test the limitations of *NoiseScope* against a variety of countermeasures, observing that *NoiseScope* holds robust or is easily adaptable.

CCS CONCEPTS

• **Security and privacy → Domain-specific security and privacy architectures**.

KEYWORDS

Deepfakes, Blind Detection, Machine Learning, Clustering

ACM Reference Format:
Jiameng Pu, Neal Mangaokar, Bolun Wang, Chandan K. Reddy, and Bimal Viswanath. 2020. NoiseScope: Detecting Deepfake Images in a Blind Setting. In *Annual Computer Security Applications Conference (ACSAC 2020), December 7–11, 2020, Austin, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3427228.3427285

ACSAC 2020, December 7–11, 2020, Austin, USA

ACM ISBN 978-1-4503-8858-0/20/12…$15.00
https://doi.org/10.1145/3427228.3427285

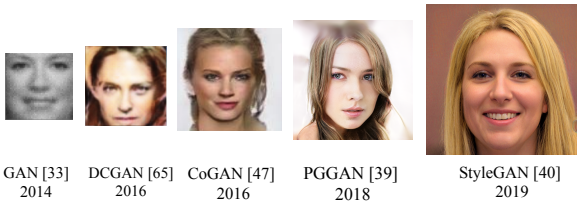


Figure 1: Advances in GANs over the years.

1 INTRODUCTION

Recently, the machine learning community has made significant advances in *deep generative models*. A landmark paper by Goodfellow et al. proposed the Generative Adversarial Network (GAN) in 2014 [33]. This work triggered immense interest in developing and improving GAN models. Today, such generative models can generate convincing images [39, 40], videos [75], text [34] and audio [22]. Figure 1 shows quality of images generated by GANs over the years. These efforts were primarily motivated by different benign use cases, *e.g.,* to augment datasets using synthetic samples to train better models [26], for de-identification [42], feature extraction [5], video prediction [46], and image editing [58]. However, governments [16, 73] and industry [25] are realizing the dual-use nature of such powerful methods—fake content or *deepfakes* produced by generative models can also be used for malicious purposes.

Today, we see many instances of misuse of deepfakes, including fake pornographic videos and images of celebrities, and ordinary people [24, 64], fake audios of people saying things they never said [50]. Industry and governments are also concerned about deepfakes being used for large disinformation campaigns on social media platforms to manipulate elections, trigger hate and violence against minorities, and to create unrest in society [60]. Deepfakes can be a threat beyond the web as well. Recent work showed how GANs can be used to create deepfake medical images to mislead medical professionals, and ML-based diagnostic tools [51].

In this work, we take a step towards defending against such threats by building a deepfake detection scheme. We focus on deepfake *images* generated by GANs, which is the state-of-the-art method for generating photorealistic images. Most prior work on detecting fake (GAN generated) images are supervised methods that require a priori access to fake images, or their generative models [52]. However, supervised schemes usually do not generalize well to datasets they are not trained on, and access to fake images for training can be limited in practice. Instead, we focus on advancing the state-of-the-art in *blind detection* of fake images. Our

scheme, called *NoiseScope*, can accurately detect fake images without requiring any a priori access to fake images or their generative schemes for training.

Our work is inspired by prior work in camera fingerprinting [11, 12, 32, 49], and includes the following key contributions:
① Similar to images produced by cameras, we find that fake images contain unique low-level noise patterns that are tied to the GAN model that generated it. Such patterns are correlated with the deconvolution layers of GANs. ② We present the design and implementation of *NoiseScope*, a blind detection scheme that leverages unique patterns in fake images left by the generative model. Given a test set with unknown number of fake and real (produced by camera) images, *NoiseScope* extracts any available *model fingerprints* or patterns that identify a GAN and uses the fingerprint to detect fake images in that set. In contrast to supervised schemes, our method is agnostic to the type of GAN used, and is also effective when the test set contains images from multiple GANS. Our method also works for any type of high-level image content, as it only extracts low-level noise patterns. ③ We evaluate *NoiseScope* on 11 diverse deepfake image datasets, created using 4 high quality GAN models. *NoiseScope* can detect fake images with up to 99.68% F1 score. ④ Lastly, we extensively evaluate *NoiseScope* against a variety of countermeasures by assuming an attacker who is aware of *NoiseScope*'s detection pipeline.

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

2 BACKGROUND & RELATED WORK

In this work, we focus on images, and consider deepfake images as those produced by machine learning algorithms, more specifically, GANs. GAN models are capable of producing high-quality images. In fact, humans find it hard to distinguish deepfake images from real images [51]. We encourage the reader to look at the following website [2] that presents a new fake image on each page refresh, created using the StyleGAN [40]. *In the rest of the paper, we will interchangeably use the term deepfake or fake image to refer to such content. Images produced by traditional imaging devices (cameras) are called real images.*

2.1 Deepfake Generation Methods

Deepfakes are primarily enabled by the family of deep generative models. Given a training set of images, a generative model can learn the distribution of the data and produce new images with variations. Two popular approaches include Generative Adversarial Networks (GANs) [33], and Variational Autoencoders (VAEs) [41]. We focus on deepfakes generated by GANs, because GANs have shown impressive performance over the last few years.

GAN Basics. In 2014, Goodfellow et al. [33] proposed the Generative Adversarial Network (GAN). A GAN is designed using two neural networks, a *generator (G)* that produces fake images, and a *discriminator (D)* that takes the fake image and gives feedback to the generator on how well it resembles a real image. The two components are trained simultaneously in an adversarial manner such that the generator learns to produce fake images that are indistinguishable from real images, and the discriminator learns to distinguish between real and fake images (produced by the generator). Therefore, the idea is to optimize one main objective—*make the generated images indistinguishable from real images*. This training objective or loss term is called the *adversarial loss.*

It is important to note the role of deconvolution or upsampling layers in generative models. An integral component of most generative models, including VAEs and GANs, is a transposed convolution layer [23], commonly referred to as deconvolution or upsampling layer. This is fundamental to building high quality generators, as it allows for learnable upsampling from a lower dimensional vector space. In Section 5.1, we demonstrate how the deconvolution layers can leave distinct patterns in the "noise space" of an image, which enable us to distinguish between fake and real images.

Choice of GANs. Experimenting with the large number of GANs in the literature [3, 47, 65, 82] would be impractical. Instead, we focus on certain key models that significantly raised the bar for different types of image generation tasks. We focus on deepfakes generated by CycleGAN [87], PGGAN [39], BigGAN [9], and StyleGAN [40]. These 4 GANs are briefly discussed below. Figures 8-18 in Appendix A show image samples from all 4 GANs.

CycleGAN [87]. CycleGAN advanced the state-of-the-art in image-to-image translation when it was proposed, improving over the previous method Pix2Pix [37]. CycleGAN can translate an image from one domain to another, *e.g.,* turn an image of a horse to a zebra. Compared to Pix2Pix, CycleGAN does not require paired images for training, which is a huge advantage, as paired images (for two domains) are hard to obtain. From a threat perspective, image-to-image translation schemes can be used by an attacker in many ways, *e.g.,* swap faces in an image, insert a new person or object into a scene.

PGGAN [39]. In 2018, PGGAN demonstrated a huge improvement in image quality. Previously, GANs were not capable of generating high resolution images in high quality. The basic idea is to progressively generate higher resolution images, by starting from easier low-resolution images. PGGAN progressively grows both the generator and discriminator by adding new layers as training progresses to produce higher resolution images with more details. PGGAN is able to produce photo-realistic images at high resolutions, up to 1024x1024. At the time, PGGAN produced the highest Inception score of 8.80 for CIFAR10 [43], and also created a high-quality version of the CelebA dataset [48] at 1024x1024 resolution.

BigGAN [9]. Soon after the introduction of PGGAN, Brock et al. introduced BigGAN, an attempt to scale up conditional GANs to develop high quality images on a large number of domains. BigGAN uses a variety of techniques to improve GAN training and image quality, including an increased batch size, increase in number of layer channels, and shared embeddings for batch normalization layers in the generator. One feature of BigGAN is the "truncation trick", whereby using a hyperparameter called the *truncation threshold*, one can control the trade-off between image fidelity and variety. A

[1] https://github.com/jmpu/NoiseScope
[2] https://thispersondoesnotexist.com/

higher truncation threshold leads to higher variety in generated images, while a lower threshold boosts fidelity. When evaluated on the ImageNet dataset, BigGAN produced a very high Inception Score of 166.5, outperforming SAGAN [82] which had the previous best Inception Score of 52.52.

StyleGAN [40]. In 2019, Karras et al. released StyleGAN, an improvement to PGGAN which incorporates a complete redesign of the generator architecture. The generator no longer receives as input a random noise vector, but a *style vector* generated by a noise-to-style CNN mapping network. Other changes include a change in the upsampling technique, and addition of noise to feature maps in the convolutional layers. This redesign allows fine-grained control over style of the generated image, while simultaneously retaining and improving upon the high-quality output of PGGAN. Having fine-grained control over style of the generated image is important from an attack perspective.

2.2 Deepfake Detection Methods

Prior work on deepfake detection has investigated both *supervised* and *blind* detection schemes. In a supervised scheme, the defender has access to both real and fake content (or has knowledge of the generative model) and can use this labelled data to train a classification algorithm. In blind detection, the defender has no a priori access to fake content (or generative methods employed), and only has access to real content. Most prior work has employed supervised schemes, and limited efforts have been made towards advancing blind detection schemes. Consequently, the performance of such schemes has evolved considerably, and the release of effective DNN models that facilitate improved feature learning has only furthered this progress. However, the dominant performance of supervised learning comes with notable caveats.

In practice, it is hard to obtain a priori access to fake content, or knowledge of the generative model. However, even with such presumption, supervised schemes suffer from a fatal inability to generalize. More specifically, we observe that such schemes are designed for and thus trained on a limited set of deepfakes (generated by specific generative models), and do not generalize well when evaluated against deepfakes produced by other models. In Section 5.2, we demonstrate this inability to generalize.

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

Supervised methods. One set of approaches focus on building a supervised classifier with input image features crafted from specific vector spaces. Examples include Marra et al.'s [52] proposition of using raw pixel and conventional forensics features, and Nataraj et al.'s [57] extraction of pre-computed RGB pixel co-occurrence matrices to capture distinguishing features. Feature engineering in multiple color spaces has also been explored. Li et al. proposed a feature set capturing disparities in color spaces between real and fake images and then using such features to perform classification [45].

Prior work observed that, similar to cameras, GANs also leave unique fingerprints in the images. Marra et al. [53] extracted GAN model fingerprints using techniques from the camera fingerprinting literature [12, 49], and implemented a supervised scheme to detect fake images. Another approach by Yu et al. [80] used a supervised deep learning scheme to learn GAN model fingerprints, and attribute images to GANs. Yu et al.'s approach primarily focused on attributing fake images to different GANs. Albright and McCloskey [2] also worked on attributing images to GANs by leveraging generator inversion schemes [19]. Our work also aims to identify model fingerprints to detect fake images but does so in a blind manner.

Domain-specific inconsistencies can also be used to detect deepfakes. Yang et al. [78] focused on deep fakes generated by splicing synthesized face regions into a real image. They show that such splicing introduces errors when 3D head poses are estimated from the fake images. An SVM-based classifier is trained to learn such errors to distinguish between real and fake images.

Other supervised approaches leverage DNNs to automatically extract features relevant for classification. Mo et al. [55] developed a CNN-based model to detect face images generated by PGGAN. Rossler et al. compared 5 CNN-based classification architectures by learning extracted face regions [69]. Tariq et al. [72] propose using ensembles of various CNN-based classifiers to detect GAN generated face images. Concurrent to our work, Wang et al. [77] proposed a classifier based on the ResNet-50 architecture that is trained on a large number of fake images from a single GAN, with carefully chosen data augmentation schemes. Afchar et al. [1] designed MesoNet based on Inception blocks to detect deepfakes showing impressive performance. We compare *NoiseScope* with MesoNet in Section 5.2. Also note that the above approaches have a fundamental weakness—they can be evaded by the attacker, by re-training the GAN using the defender's DNN model as the discriminator.

Blind detection. Li et al.'s work [45] proposes a blind detection scheme. The idea is that GANs fail to learn correlations among color components in the RGB space, which results in inconsistencies when examined in other color spaces, namely HSV, and YCbCr. They train a one-class SVM classifier on features based on color statistics of HSV and YCbCr color spaces of real images to detect fake images. The intuition is that fake images will be flagged as anomalies in the color (feature) space. We compare our approach against Li et al.'s approach in Section 5.2.

Zhang et al. [83] uses real data to train "AutoGAN", a component that aims to simulate a GAN generator. The idea is to first generate fake images using AutoGAN, and then train a supervised classifier on the newly synthesized fake images and real images to detect other fake images. Unfortunately, its performance largely depends on the architecture of AutoGAN's generator. Results show significant drop in performance when tested on fake images from a GAN that uses a different architecture compared to AutoGAN's generator.

3 DETECTING DEEPFAKES VIA *NOISESCOPE*

3.1 Attack and Defense Model

Attacker model. Attacker aims to generate high quality convincing deepfake images using deep generative models (GANs). Our focus is on fake images that are entirely produced by a generative model (GAN). Fake images created by image forgery techniques such as replacing or adding content in real images (*e.g.,* face swapping [74]) are not considered. In Section 5, we consider an attacker who is unaware of our defense scheme. Later, in Section 6, we consider an attacker who is aware of our defense scheme pipeline and employs a variety of countermeasures against *NoiseScope*.

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

3.2 Method Basics

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

We borrow ideas from the rich literature of camera fingerprinting schemes [11, 12, 32, 49]. Each imaging device (*e.g.,* camera) leaves a unique and stable pattern in each image due to imperfections in various stages of the image acquisition process. Such patterns known as photo-response non-uniformity (PRNU) patterns have been used to fingerprint cameras or image acquisition devices [49]. Naturally, this first raises the question whether GAN-based image generators would leave a unique and stable "artificial" pattern in the generated images. In fact, preliminary work by Marra et al. shows that such stable patterns do exist in GAN generated images [53]. These patterns are present, regardless of the content in the image, be it images of human faces, objects, animals or landscapes. Secondly, we would expect those patterns to look different because GAN models share no similarity with camera-based image acquisition pipelines. We leverage these ideas and propose a complete blind detection scheme that can accurately flag fake images with any type of content. Next, we explain techniques from the camera fingerprinting literature that we leverage to fingerprint generative models.

Leveraging model fingerprints for detection. Consider a set of images, I_i, where $i \in \{1, \ldots, N_p\}$ generated by a GAN. Our goal is to estimate a stable pattern left by the GAN, that is unrelated to the semantics of the image content. The first step is to separate the high-level content from the image, and estimate the *noise residual* R_i. The high-level content is estimated by applying an appropriate denoising filter $f(I_i)$. The noise residual is then computed as, $R_i = I_i - f(I_i)$. Now the assumption is that the noise residual R_i contains the stable pattern or the fingerprint F, and some random noise N_i, *i.e.,* $R_i = F + N_i$. Therefore, one can estimate the fingerprint by averaging the residuals:

$$\bar{F} = (\sum_{i=1}^{N_p} R_i)/N_p \quad (1)$$

In practice, the larger the N_p, the additive noise component tends to cancel out, and we obtain a more accurate fingerprint. According to prior work, it is possible to estimate a reliable fingerprint using at least 50 images, *i.e.,* $N_p > 50$ [6].

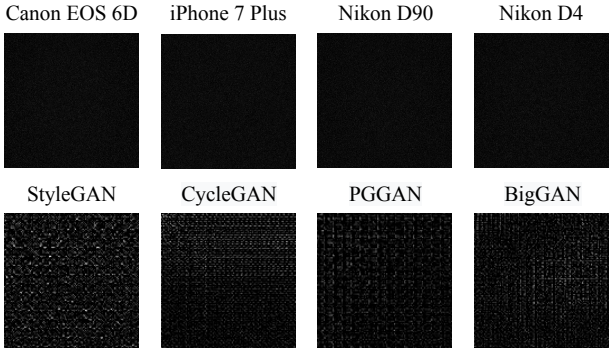


Figure 2: Camera fingerprints from Canon, iPhone, Nikon cameras (top) and GAN fingerprints from StyleGAN, CycleGAN, PGGAN, BigGAN (bottom).

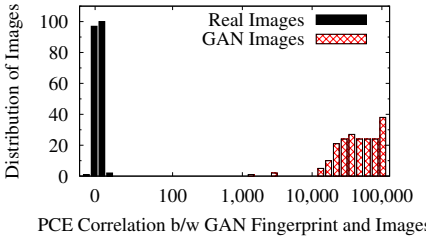


Figure 3: Histogram of PCE correlation between model fingerprint and images (fake and real).

Figure 2 shows camera and GAN fingerprints computed using the above method for $N_p = 100$ images. Note that model fingerprints look very different from device fingerprints. In the case of CycleGAN and PGGAN, there is a noticeable checkerboard pattern. This observation is further discussed later in Section 5.1.

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

$$\rho_{F_{GAN},i} = corr(\bar{F}_{GAN}, R_i). \quad (2)$$

For a given image, if this correlation is higher than a certain threshold T_c, it is classified as a fake image, or real, otherwise. A correlation measure called *Peak to Correlation Energy (PCE)* (described next) is used. Figure 3 shows the histogram of correlation values for all images in the set (both fake and real). The fake images can be easily separated from the real images based on the PCE values.

PCE metric [30]. PCE is a similarity metric to compare two discrete signals. It is computed as the ratio between squared normalized correlation and sample variance of circular cross-relations. The PCE implementation[3] that we use carries the sign of normalized correlation peak (can be negative). A high positive value of PCE denotes a high correlation. Other than PCE, there are other correlation measures, such as Pearson correlation [59], and quotient correlation [84]. Compared to other metrics, PCE is a more stable metric that can be used with images from devices with different resolutions and sensor types [30]. We find PCE to be suitable for GAN images as well.

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

Key challenges in designing *NoiseScope*. ① The first challenge is estimating a model fingerprint. It is hard to estimate a model fingerprint from a single image in a blind setting (Equation 1 requires averaging over multiple images). While prior work, NoisePrint [18] provides a supervised (CNN-based) learning scheme to extract camera fingerprint from a single image, such methods are not applicable in a blind setting. Instead, *our idea is to extract fingerprints from the test set itself in an unsupervised manner.* We propose an image clustering scheme that identifies subsets of images belonging to the same source (device or model), and estimate fingerprints based on those subsets. Our method should work as long as a certain minimum number of fake images (enough to reliably estimate a fingerprint) are present in the test set ② Once a fingerprint is extracted from the test set, how do you tell whether it is a model fingerprint or a device fingerprint? To achieve this, we propose a fingerprint classification module based on anomaly detection to identify model fingerprints③ Method should be agnostic to the specific GAN used, and should also work when test set contains fake images from different GANs. To address this, our clustering scheme is designed to be agnostic to the GAN(s) used, and is able to extract available fingerprints, even from multiple models④ Method should work for images with any type of high-level content (images of faces, animals, objects, etc.) To address this challenge, we use residual image extraction schemes that can effectively suppress high-level content.

3.3 Detection Pipeline

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

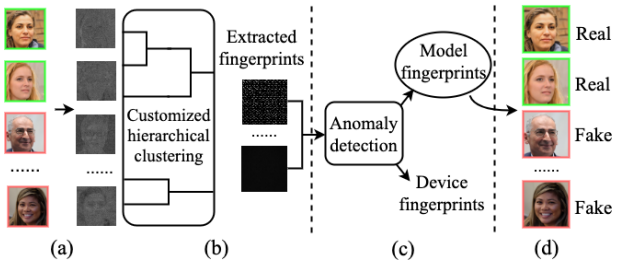


Figure 4: An illustration of *NoiseScope* detection pipeline: (a) Noise Residual Extractor, (b) Fingerprint Extractor via Clustering, (c) Fingerprint Classifier, (d) Fake Image Detector.

Noise Residual Extractor. This first step suppresses high level image content and extracts the noise residual (which contains the fingerprint). We use the Wavelet Denoising filter [54] to extract the noise residual for each image in the test set. Prior work recommends this as one of the best filters to suppress high-level content [13, 17]. However, there is no perfect filter, and we do notice Wavelet denoising also leaking image contents into the noise residual in some cases. If there is heavy content leakage, then fingerprint extraction (next step) becomes harder. But in general, Wavelet denoising tends to perform well. In Section 5.2, we analyze the impact of different denoising filters on detection performance.

Fingerprint Extractor and Fingerprint Classifier. The second step extracts *model* fingerprints from the test set. The fingerprint extractor finds all available fingerprints (model or device) from the test set, and the fingerprint classifier identifies those that are model fingerprints. To extract fingerprints, we resort to unsupervised clustering by starting with the individual noise residuals computed from step 1. Our goal is to group images belonging to the same source (model or device), and then use each group of images to build a fingerprint (using Equation 1).

But there is a challenge—it is hard to cluster images in the residual space. Residual images contain random noise along with the fingerprint pattern. So even images from the same source (model or device) will not always show high correlation [38]. All our efforts to cluster images in the residual space resulted in impure clusters, *i.e.,* clusters with mix of fake and real images. An impure cluster would give us an inaccurate fingerprint which is not useful.

To address this challenge, we use a different strategy: Instead of completely clustering images in the residual space, we use an *incremental clustering* strategy, similar to bottom-up hierarchical clustering. The idea is to mostly compute correlations between fingerprints (which has less random noise), and less between residuals. Initially, each residual image forms its own cluster. Next, any pair of residuals with PCE correlation higher than a threshold T_{merge} is merged into a new cluster. Each time a cluster is updated, we compute a fingerprint (using cluster members), and two clusters are merged if the PCE correlation between their fingerprints is greater than T_{merge}.[4] This is done iteratively at each step to grow clusters. By computing correlations using fingerprints, we reduce the risk of random noise impacting our correlation estimates. The larger a

[3] http://dde.binghamton.edu/download/camera_fingerprint/

[4] We update clusters such that each image is only present in one cluster.

cluster becomes, the more the random noise will vanish when we estimate the fingerprint. The PCE threshold for merging, T_{merge} is chosen such that clusters mostly end up being *pure*, *i.e.,* contain all fake images or all real images. If T_{merge} is too low, clusters end up being impure, and we obtain inaccurate fingerprints which may not be useful for detecting fake images in the next step. If T_{merge} is too high, we run the risk of not finding sufficiently large clusters or even no clusters to estimate an (accurate) fingerprint. In Section 4.2, we discuss how we estimate T_{merge}.

The clustering process stops when no more clusters can be merged using the threshold. However, to reduce the computational complexity, we propose to stop clustering early when we find cluster(s) with $size > T_{size}$. Recall that we only require a small number of images (> 50) to estimate a fingerprint. Once we stop clustering, we pass any fingerprint computed using clusters greater than size 50, to the Fingerprint Classification component to decide whether it is a model or device fingerprint. If no model fingerprints are found, we continue the clustering process again (in case it was stopped early), until no more merging is possible. Fingerprints found at the end are again passed to the fingerprint classifier. Pseudo-code for the fingerprint extraction and classification step is shown in Algorithm 1.

Fingerprint Classifier. The fingerprint classifier is used to identify model fingerprints. Key challenge here is that we have no a priori knowledge of model fingerprints. Our intuition is that GAN fingerprints stand out as anomalies when compared to device fingerprints in some feature space. Recall the checkerboard pattern in GAN fingerprints shown earlier in Figure 2. *We observe that model fingerprints tend to have different texture patterns when compared to device fingerprints.* To capture texture features from a fingerprint, the well-known Haralick texture features [35] are used. Haralick texture features capture 14 statistical features from the Gray Level Co-Occurrence Matrix (GLCM), which in turn captures the number of repeated pairs of adjacent pixels. For the anomaly detection scheme, we use the Local Outlier Factor (LOF) scheme [8]. Input to LOF are Haralick features extracted from fingerprints computed over (real) images in the reference set. Once trained, the fingerprint classifier can take any fingerprint as input (after extracting Haralick features), and check whether it is an anomalous sample. A fingerprint is considered to be a model fingerprint if this component marks it as an anomalous fingerprint.

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

Method Scalability. The clustering part is the most computationally heavy step of the system. In the worst case, the clustering could run for $\log(n)$ iterations, where n is the number of images in the dataset. Each iteration requires sorting of the pair-wise PCE correlation, with an $O(n^2 \cdot \log(n^2))$ complexity. This gives the entire clustering part a complexity of $O(n^2 \cdot \log^2(n))$. Improvements can be made to scale the Fingerprint Extractor for large-scale classification. Pairwise PCE correlations can be computed in parallel to speed up the construction of the PCE correlation matrix. As $n \to \infty$, the, pipeline can, as a whole, also be run in parallel on subsets of the n images. A final instance of the Fingerprint Extractor can be used to agglomerate the clusters obtained from these parallelized Fingerprint Extractors. We can also leverage prior work on distributed/parallel hierarchical clustering [56, 62, 66].

Algorithm 1: *NoiseScope* Fingerprint Extractor & Classifier:

Data: Set of image residues: I, PCE merging threshold: T_{merge}, cluster size threshold: T_{size}.
Result: Set of model fingerprints: FP.
FakeFingerprintExtractor (I, T_{merge}, T_{size})**:**
  Cluster set $C = \{I_1, ..., I_{N_p}\}$ contains N_p residuals.
  Stopping flag *mergeable* = True
  while *mergeable* **do**
    Merged cluster set $C_{pairs} = \{\}$
    for *pair* (c_i, c_j) *in C with highest PCE* **do**
      if $PCE(c_i, c_j) > T_{merge}$ **then**
        Add merged pair (c_i, c_j) to C_{pairs}
        Remove clusters c_i and c_j from C
    if C_{pairs} *is empty* **then**
      mergeable = False
    else
      Add C_{pairs} to C.
    if *not mergeable or size(c : C)* $> T_{size}$ **then**
      Fingerprint set $FP = \{\}$
      for *c in C where size(c)* > 50 **do**
        Compute fingerprint $fp_c = fingerprint(c)$
        if fp_c *is flagged as outlier* **then**
          Add fingerprint fp_c to set FP
      if FP *is not empty or not mergeable* **then**
        Return FP

4 EXPERIMENTAL SETUP

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

4.1 Real and Fake Image Datasets

For each dataset, we discuss the GAN used to generate the fake images in the test set, and how real images for the test and reference sets are collected. Each dataset includes 2,500 fake images, and out of the real images we collected for each dataset, 2,000 random real images are used to build the reference set. Table 1 presents statistics of the 11 datasets covering 4 GAN models, used for our evaluation. Image samples from all datasets are shown in Figures 8-18 in Appendix A.

Datasets	Content	Fake Source	Real Source	Resolution	# Fake images	# Real images
StyleGAN-Face1	Human face	StyleGAN[15]	FFHQ[40]	1024x1024	2,500	8,000
StyleGAN-Face2	Human face	StyleGAN[28]	FFHQ[40]	1024x1024	2,500	8,000
StyleGAN-Bed	Bedroom	StyleGAN[14]	LSUN[79]	256x256	2,500	3,098
BigGAN-DogLV	French bulldog	BigGAN[36]	ImageNet[21], Flickr[71]	256x256	2,500	5,309
BigGAN-DogHV	French bulldog	BigGAN[36]	ImageNet[21], Flickr[71]	256x256	2,500	5,309
BigGAN-BurgLV	Cheeseburger	BigGAN[36]	ImageNet[21], Flickr[71]	256x256	2,500	4,390
BigGAN-BurgHV	Cheeseburger	BigGAN[36]	ImageNet[21], Flickr[71]	256x256	2,500	4,390
PGGAN-Face	Human face	PGGAN[67]	FFHQ[40]	1024x1024	2,500	8,000
PGGAN-Tower	Tower	PGGAN[67]	LSUN[79]	256x256	2,500	4,187
CycleGAN-Winter	Winter scene	CycleGAN[86]	summer2winter[85], Flickr[71]	256x256	2,500	4,594
CycleGAN-Zebra	Zebra	CycleGAN[86]	horse2zebra[85], Flickr[71]	256x256	2,500	11,241

Table 1: Basic information of 11 deepfake image datasets evaluated in Section 5.2.

StyleGAN-Face1. This is a dataset of *human face* images, at 1024x1024 resolution. Fake images are generated by StyleGAN, trained on the Flickr-Faces HQ (FFHQ) dataset of human faces [40]. Fake images are collected from the official NVIDIA StyleGAN GitHub repository [15]. We collected 8,000 real images for the test and reference sets by randomly sampling from the FFHQ dataset.

StyleGAN-Face2. Recently Generated Media, Inc. [29] released 100,000 StyleGAN generated face images [28]. Their aim is to provide royalty-free stock images using AI [63]. The GAN was trained using a proprietary dataset of 29,000+ curated photographs of 69 models. The images are photorealistic (See Figure 17), and it is unclear if these images have been further post-processed to improve image quality. Fake images are sampled from this dataset. We randomly sampled 8,000 real images from the FFHQ dataset.

StyleGAN-Bed. This includes images of *bedroom scenes* at 256x256 resolution. Fake images are generated by NVIDIA with a StyleGAN trained on the LSUN Bedroom dataset [79] of bedroom scenes. Fake images are obtained from the official NVIDIA GitHub repository [14]. We randomly sampled 3,098 real images from the LSUN Bedroom dataset.

BigGAN-DogLV and BigGAN-DogHV. Datasets include images of *french bulldogs* at 256x256 resolution. Fake images are generated using a BigGAN-deep instance [9], trained on the ImageNet dataset, and obtained online [36]. BigGAN provides an inference-time truncation parameter to vary the trade-off between fidelity and variety (see Section 2.1). We generate two sets of fake images, BigGAN-DogLV and BigGAN-DogHV at truncation settings of 0.2 and 0.86, respectively. BigGAN-DogLV has images with lower variety, while BigGAN-DogHV has images with higher variety. Real images are partially sourced from ImageNet. However, ImageNet only provides 1,300 images for this image class. We further collected additional real images by crawling Flickr.com, giving us a total of 5,309 real images.[5]

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

PGGAN-Face. This dataset contains images of *human faces*, at 1024x1024 resolution. Fake images are produced by NVIDIA with a PGGAN trained on the CelebA dataset [39] of celebrity faces. Fake images are collected from the official PGGAN repository [67]. For real images, we sampled 8,000 images from the FFHQ dataset.

PGGAN-Tower. Dataset contains images of *towers*, at 256x256 resolution. The fake images are generated by NVIDIA with a PGGAN trained on the LSUN tower dataset [79] of towers. These images are collected from the official PGGAN repository [67]. We randomly sampled 4,187 real images from the Tower category of the LSUN dataset.

CycleGAN-Winter. Dataset contains images of *winter scenes* at 256x256 resolution. Fake images are generated using a pre-trained model available on the official CycleGAN repository [86]. CycleGAN requires input images to generate fake *translated* images (summer to winter scene translation), and only a limited number of fake images (1,187) could be generated using the data provided by the authors. To generate more fake images, we crawl Flickr.com for more input images, and generate new fake images. Real images provided by the authors are also limited (only 1,474). We thus supplement the real images for CycleGAN-Winter by crawling Flickr.com, and obtain a total of 4,594 real images.

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

4.2 Configuration of *NoiseScope*

Noise Residual Extractor. We use a Wavelet Denoising filter (see Section 3.3) to prepare residual images. The implementation from Goljan et al is used. [6]

Fingerprint Extractor. Two parameters to configure include T_{merge}, which decides the PCE correlation threshold to merge two clusters, and T_{size} used to stop the clustering process early. T_{size} is set to 150 and is observed to work well across datasets. To estimate T_{merge}, one approach is to use a reference set with camera identifiers. PCE correlation between fingerprints computed from the same camera can be computed, and a suitable threshold can be estimated. We lack camera identifier information in most of our datasets, and therefore use a different strategy. We assume the reference set includes images from multiple cameras and compute 'pseudo-fingerprints'[7] over random subsets (non-overlapping) of

[5]Images were curated using manual effort as well as using the ResNet50 ImageNet classifier

[6]http://dde.binghamton.edu/download/camera_fingerprint/

[7]Technically they are not fingerprints as they are computed over images from different cameras.

20 images. Next, pairwise PCE correlation between these different pseudo-fingerprints are estimated. Clearly, the PCE values will not be high, as images are from different cameras. Therefore, we set T_{merge} to be 99.5 percentile of this distribution, *i.e.,* it should be at least larger than the correlation between pseudo-fingerprints computed over different cameras. This strategy works well in practice.

Fingerprint Classifier. We configure and train an LOF anomaly detection scheme (Section 3.3). If we have a reference dataset with camera identifiers, we can compute fingerprints for each camera, and use that to train the anomaly detection scheme. Lacking such data for most of our datasets, we again use the strategy used in Fingerprint Extractor, and compute 'pseudo-fingerprints'[7] using random subsets of 50 real images from the reference set (which is assumed to contain images from multiple cameras), and train the scheme using 200 such pseudo-fingerprints. This is effective because model fingerprints are still anomalous in the texture space even when compared to pseudo-fingerprints computed over multiple cameras. The parameter *contamination*, which configures the error in the training set is set to 10^{-4}, and the number of neighbors to analyze (in K-NN) is set to 30.

Fake Image Detector. This component flags an image to be fake, if the PCE correlation between a model fingerprint and residual image (in test set) is higher than a threshold. To calibrate the threshold, we compute PCE correlation between a model fingerprint, and images in the reference set. Threshold is chosen such that 99.5% of the reference set images are not flagged as fake.

4.3 Evaluation Metrics and Baseline Method

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

We compare *NoiseScope* with the blind detection scheme proposed by Li et al. [45] (Section 2.2). This approach analyzes differences between real and fake images using disparities in the HSV and YCbCr color spaces. This is achieved by using features extracted from these color spaces to train a one-class SVM for anomaly detection. We abbreviate this method as *CSD-SVM*. The underlying assumption is that fake images will be detected as anomalies. We follow the configuration described in the paper to train *CSD-SVM*. A Gaussian kernel is used, and parameters are estimated via grid search. For the parameter ν, which controls the upper bound of training error, we try two values, 0.10 and 0.05. Real images in the reference set are used to train the *CSD-SVM* for each dataset.

5 EVALUATION OF PROPOSED SCHEME

5.1 Analysis of Model Fingerprints

Performance of Fingerprint Classifier. For the three face datasets (StyleGAN-Face1, StyleGAN-Face2, and PGGAN-Face), our real dataset includes images with camera source information for 90 cameras (extracted from EXIF metadata). We first train the anomaly detection scheme on device fingerprints from 18 cameras. Next, in each trial, we test on 500 device fingerprints (extracted from the remaining 72 cameras), and 500 model fingerprints (obtained from the three face datasets). Our classifier achieves a high average F1 score of 99.2% over 5 trials (average Precision of 98.5% and Recall of 100.0%) for the detection of model fingerprints and is therefore capable of accurately detecting model fingerprints.

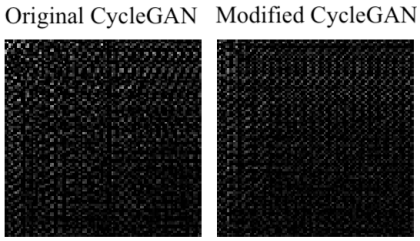


Figure 5: Change in CycleGAN fingerprint checkerboard when varying transpose convolution parameters.

In the rest of the evaluation, when camera identifiers are not available, we use the strategy described in Section 4.2, and train the fingerprint classifier using pseudo-fingerprints computed over the reference set. Results in Section 5.2 show that this works well in practice.

Understanding Model Fingerprints. Why do GAN fingerprints show checkerboard patterns? The answer is tied to the deconvolution layers that are the core building blocks of GAN generators [65]. Odena et al. observed checkerboard patterns in images generated by upsampling via transpose convolution operations [61]. They attributed the checkerboard pattern to the overlap that occurs when the kernel size of the transpose convolution projection window is not divisible by the stride. The pattern is amplified when multiple transpose convolutional layers are stacked. In our fake images, we do not observe such checkerboard patterns in the high-level content, but we clearly see such patterns in the fingerprints (Figure 2).

To further understand the correlation between deconvolution layers and checkerboard patterns, we conduct the following experiment using CycleGAN. The transpose convolutions in the CycleGAN ResNet50 generator are found in 2 layers, with strides of 2x2 and kernel sizes of 3x3. We observe that by varying the kernel size in the second layer from 3x3 to 5x5[8], we can alter the intensity and locality of the checkerboard pattern in the resulting fingerprint. The model fingerprints, before and after modifying CycleGAN are shown in Figure 5. The visible change in fingerprint textural patterns indicates a strong correlation between the fingerprint and the deconvolution operations in modern GAN generators.

5.2 Detection Performance

We evaluate detection performance of *NoiseScope* when applied to the 11 datasets discussed in Section 4.1.

Performance on balanced test sets. In each trial, *NoiseScope* is applied to a balanced test set with 500 real, and 500 fake images. Table 2 presents detection performance (average F1 score) for both *NoiseScope*, and CSD-SVM (ν=0.1). *NoiseScope* outperforms CSD-SVM over all 11 datasets and achieves a high F1 score of over 90.1% for all datasets. Varying the ν parameter (upper bound of training error) to 0.05 for CSD-SVM shows no noticeable improvement. Given *NoiseScope*'s high detection performance, it is worth noting that images generated by StyleGAN, PGGAN and BigGAN are vividly photorealistic, and are difficult for humans to spot.

[8]Kernel dilation, input padding and output padding must also be accordingly changed to support the desired image output dimensions.

Datasets	F1 Score (%)	
	NoiseScope	CSD-SVM
StyleGAN-Face1	**99.56**	92.93
StyleGAN-Face2	**90.14**	67.53
StyleGAN-Bed	**99.63**	94.82
BigGAN-DogLV	**99.38**	86.94
BigGAN-DogHV	**92.6**	70.10
BigGAN-BurgLV	**99.68**	94.82
BigGAN-BurgHV	**98.64**	83.67
PGGAN-Face	**99.09**	64.07
PGGAN-Tower	**95.93**	91.61
CycleGAN-Winter	**92.40**	87.14
CycleGAN-Zebra	**92.84**	84.95

Table 2: Performance of *NoiseScope* and CSD-SVM ($\nu = 0.1$).

Datasets	F1 Score (%) w/ different fake:real ratio			
	200:400	200:800	200:1600	200:2000
StyleGAN-Face1	**97.9**	**97.1**	**96.4**	**94.0**
StyleGAN-Face2	81.0	74.3	62.8	58.4
StyleGAN-Bed	**99.5**	**98.8**	**97.8**	**97.2**
BigGAN-DogLV	**98.9**	**97.7**	**95.8**	**95.2**
BigGAN-DogHV	89.3	85.7	84.9	82.3
BigGAN-BurgLV	**98.6**	**97.2**	**96.2**	**91.9**
BigGAN-BurgHV	**97.9**	**96.0**	**94.8**	**93.2**
PGGAN-Face	**97.1**	**97.4**	**93.7**	**94.4**
PGGAN-Tower	**94.5**	**94.3**	**92.2**	**92.3**
CycleGAN-Winter	88.2	86.8	72.2	69.4
CycleGAN-Zebra	89.9	86.3	78.2	76.1

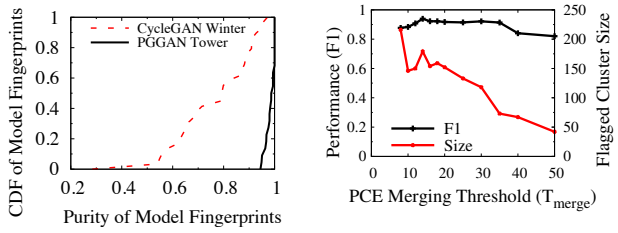
Table 3: Detection performance (F1) on imbalanced test sets with different ratio of fake to real images.

Performance on imbalanced test sets. We apply *NoiseScope* on test sets with an imbalanced ratio of real vs fake images. For each dataset, we evaluate on 4 imbalanced test sets comprising different ratios of real and fake images. In each test set, the number of fake images is set to 200, and we increase the number of real images according to the desired ratio. We experiment with ratios of fake to real as 1:2, 1:4, 1:8, and 1:10.[9] The inherent difficulty of using *NoiseScope* in an imbalanced setting is the presence of *noisy samples* among fake and real images. These are samples where content tends to leak into residuals. Therefore, such noisy fake and real images can show unexpectedly high correlation. Consequently, as the number of real images increases, the probability of a fake image cluster merging with noisy real samples increases.

Detection performance is presented in Table 3. Out of the 11 datasets, 7 datasets exhibit high performance of over 91.9% F1 score for all ratios (numbers shown in bold). As expected, there is a drop in performance as datasets become more imbalanced, but even at 1:10, we observe high detection performance for these 7 datasets.

Among the remaining 4 datasets, StyleGAN-Face2, CycleGAN-Winter, CycleGAN-Zebra shows the biggest drop in performance as test set becomes more imbalanced. To further understand the reduced performance, we analyze the purity of the model fingerprints obtained as output of the fingerprint classification component. Purity of a model fingerprint is the fraction of images in the cluster (used to estimate the fingerprint) that are fake. If purity is less, then the performance of the fake image detection module will decrease

[9]For 1:8, and 1:10 we do 3 trials. Rest of them are averaged over 5 trials.



(a) (b)

Figure 6: (a) GAN fingerprint purity distributions (b) PCE Merging Threshold T_{merge} vs. Detection F1 Score.

Datasets	F1 Score(%) w/ different fake:real ratio			
	200:400	200:800	200:1600	200:2000
StyleGAN-Face2	88.47	86.45	80.23	76.71
BigGAN-DogHV	93.80	89.39	89.17	86.76
CycleGAN-Winter	89.65	90.17	82.08	81.00
CycleGAN-Zebra	92.17	91.98	86.26	83.13

Table 4: Improved detection performance by increasing T_{merge} in non-performant imbalanced configurations.

(as the fingerprint is inaccurate). In general, for the three datasets (StyleGAN-Face2, CycleGAN-Winter, and CycleGAN-Zebra), we observe that purity of the fingerprints is lower compared to the other datasets. Figure 6a shows the distribution (CDF) of purity of fingerprints found across test sets (aggregated over all ratios) for two datasets—one for which *NoiseScope* is performant (PGGAN-Tower), and one for which *NoiseScope* suffers from relatively lower performance (CycleGAN-Winter). CycleGAN-Winter suffers from lower fingerprint purity that range between 60% to 80%, whereas the fingerprint purity for PGGAN-Tower is high, *i.e.,* over 95%. Therefore, high detection performance correlates well with the ability to reliably extract pure fingerprints.

One approach to improve fingerprint purity is to raise the PCE merging threshold T_{merge}. A higher value of T_{merge} would prevent *noisy* samples from merging with fake images. StyleGAN-Face2, CycleGAN-Winter, and CycleGAN-Zebra results in Table 3 has T_{merge} values in the range from 8.45 to 11.68. We raise the threshold to 15 and recompute the results for these datasets. In addition, we also recompute results at the raised threshold for BigGAN-DogHV (which has F1 score below 90% in Table 3). Results with the increased threshold are presented in Table 4 for all 4 datasets. We observe a marked increase in detection performance, *e.g.,* on average 10.35% increase in F1 for 1:10 ratio across all datasets. We also observe an increase in purity of fingerprints (not shown).

Above analysis raises the question of whether defender can estimate a better value of T_{merge}, starting from the initial estimate? We note that this is possible by analyzing the variation in cluster sizes as one increases T_{merge} starting from the initial value. In general, detection performance correlates well with cluster sizes. If the largest cluster size is small (say less than 50), then the value of T_{merge} is too high, and detection performance is likely to be lower. To study this, we conduct experiments on CycleGAN-Zebra with an imbalanced ratio of 1:2. Figure 6b studies the variation of detection performance and largest cluster size, as we incrementally increase T_{merge} starting from the initial estimate. Detection performance

remains mostly high and stable, for cluster size roughly above 100. Towards the end, the performance drops as cluster size goes below 67, achieving the lowest performance when cluster size is less than 50. The defender can thus calibrate T_{merge} by incrementally increasing the originally estimated value, using cluster size as a stopping condition. If no clusters are found, or clusters are too small, then the defender has exceeded the optimal T_{merge}.

Performance when test set contains fake images from multiple GAN models. So far, we considered test datasets with fake images from a single model. What if attackers use multiple GAN models? Can *NoiseScope* still detect fake images? In theory, *NoiseScope* should adapt to such settings, because clustering should ideally extract multiple model fingerprints corresponding to each model. To evaluate this, we restrict ourselves to datasets capturing faces, as it is the only content category for which we have fakes images from multiple models. In each trial, we populate the test set with 150 images each from the StyleGAN-Face1, StyleGAN-Face2 and PGGAN-Face datasets, and use 450 real images from the FFHQ dataset. Results in Table 5 indicate an overall high F1 score of 91.5%, and also shows per-dataset performance.

So how did *NoiseScope* achieve high detection performance when test set includes fake images from three different models?[10] Interestingly, *NoiseScope* discovered three clusters (model fingerprints). The first cluster mostly included images from StyleGAN-Face1 (over 95%), the second cluster mostly from PGGAN-Face (again over 95%), and in the third cluster, a majority of images are from the StyleGAN-Face2 dataset. Therefore, *NoiseScope* was able to extract model fingerprints corresponding to the three models. These results match our intuition that GANs trained on different datasets would generate distinct fingerprints. Our results indicate that *NoiseScope* is effective on test sets with fake images from different GANs. An attacker can take this setting to the extreme by creating a different GAN for every single fake image to disrupt the fingerprint extraction process. However, this significantly raises the cost for the attacker, and reduces the utility of using generative schemes.

Performance on test sets with images from multiple categorical domains. Our current configuration uses a single categorical domain for each test-set, but still has high variations among images (see Figures 8-18). This was done for the sake of simplicity, and because many GAN datasets are organized into few specific categories. Here we evaluate effectiveness on test sets with multiple content categories. We test against BigGAN as it is the only GAN model with images from several categories. For a test set of 500 real, and 500 fake images, images are evenly and randomly sampled from 10 categories: *Ambulance, Race car, Burrito, Tiger, Cup, Hen, Pretzel, Pirate, French bulldog, and Cheeseburger.* The average detection performance (F1) is high at 99.1%. Thus, *NoiseScope* works for a mix of high-level image content, i.e., *NoiseScope* is content-agnostic.

What if there are too few fake images in the test set? We present *NoiseScope* detection performance when evaluated on test sets with an increasingly small number of fake images. Using representative datasets for each GAN, we evaluate on 4 test sets with

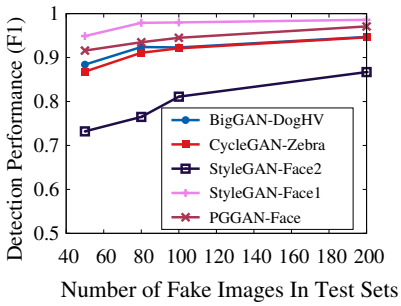


Figure 7: Detection performance (F1) with limited number of fake images in test sets.

Datasets	F1 Score (%)	Precision (%)	Recall (%)
Combined	91.5	93.3	89.8
StyleGAN-Face1	91.2	83.8	100.0
StyleGAN-Face2	81.9	100.0	69.3
PGGAN-Face	100.0	100.0	100.0

Table 5: Detection performance on test set with fake images from multiple (GAN) sources.

Datasets	F1 Score (%)			
	Wavelet	Blur	NLM	BM3D
StyleGAN-Bed	**99.7**	97.7	82.6	99.3
BigGAN-DogLV	**99.7**	99.5	95.0	**99.7**
BigGAN-DogHV	**94.8**	76.3	22.6	79.0
BigGAN-BurgLV	**99.8**	99.5	99.6	**99.8**
BigGAN-BurgHV	**99.4**	97.5	58.4	96.0
PGGAN-Tower	**95.8**	17.6	4.3	62.5
CycleGAN-Winter	**91.1**	71.4	6.7	84.5
CycleGAN-Zebra	93.6	74.1	7.9	**96.9**

Table 6: Fake Image Detector Performance (F1) using different denoising filters. Bold numbers highlight the best performance in each dataset.

50[11], 80, 100 and 200 fake images respectively. All test sets contain 200 real images from the respective dataset. A reference set of 2000 real images is used. T_{merge} for StyleGAN-Face1 and PGGAN-Face remain the same as those used for the original results in Table 2 *i.e.,* computed using the T_{merge} estimation strategy in Section 4.2. T_{merge} for BigGAN-DogHV, CycleGAN-Zebra and StyleGAN-Face1 are tuned following the recalibration strategy suggested in Section 5.2. We compute F1 score of detection performance averaged over 5 trials. Results are presented in Figure 7. The performance is moderately high, but as expected, drops as the number of fake images decrease. Analysis reveals a decrease in fingerprint purity, caused by merging with *noisy samples* amongst the increasingly large proportion of real images. From prior work we also know that a reliable fingerprint requires roughly 50 images or more [6]. The decrease in performance is not a serious problem—the absolute number of fake images to detect is in and of itself, very small. One can consider a scenario with too few fake images (<50) to not be

[10] Models trained on three different datasets.

[11] We remove the minimum requirement of 50 images for a fingerprint when only 50 fake images are in the test set.

a serious threat, compared to cases where online platforms are flooded with fake images [64].

Impact of residual image extraction filter on performance. We use 3 popular alternative filters—Blur filter [27], Non-Local-Means (NLM) filter [10], and the BM3D filter [20], and observe that Wavelet denoising provides better detection performance for nearly all datasets. Results are in Table 6. To compare, we simulate the fake image detection step in *NoiseScope*. Given a test set of 500 real, and 500 fake images, we estimate a clean model fingerprint using a random subset of 100 fake images from the test set itself. Next, we use this model fingerprint to flag fake images in the test set. This is an ideal scenario because the fingerprint is 100% pure (*i.e.,* estimated over only fake images). An effective filter should produce high detection performance in such a setting, while filters that fail to effectively remove high-level content may not perform so well. Table 6 presents the detection performance (average F1 score) for each 256x256 dataset.[12] Wavelet Denoising filter exhibits the best performance, with F1 scores exceeding 90% for all datasets. The BM3D filter also shows good performance but fails to effectively eliminate content from some datasets.

Generalization performance comparison with a supervised scheme. Supervised detection schemes exhibit high performance at the cost of generalization. To give an example, we use the supervised classifier MesoNet [1] to detect unseen GAN-generated face images. MesoNet is trained on 1000 real and 1000 fake images from StyleGAN-Face1 and provides a high F1 score of 94% on a test set of the same size from StyleGAN-Face1. However, this trained model achieves significantly reduced F1 score of 65% on a test set from PGGAN-Face. This drop in performance indicates an exploitable failure to generalize that is remedied by *NoiseScope*.

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

6 ANALYSIS OF COUNTERMEASURES

We consider a powerful adaptive attacker with knowledge of *NoiseScope*'s detection pipeline. These countermeasures aim to disrupt the fingerprint extraction, and fake image detection capabilities of *NoiseScope*. We also propose adaptive *recovery* measures to make *NoiseScope* robust to certain challenging countermeasures. Table 7 presents results using test sets with 500 real, and 500 fake images.

Compressing fake images to make fingerprints fragile. JPEG compression is well known to disrupt camera fingerprint patterns, and therefore diminish the correlation between fingerprints and residuals [31]. Following compression configurations used in prior work [80], fakes images are compressed with a quality factor randomly sampled from U [10, 75]. Surprisingly, NoiseScope *is resilient against compression attacks.* More interestingly, the performance for StyleGAN-Face2 increases from 90.14% to 98.33% on applying compression. *NoiseScope* is resilient for two reasons: *First*, model fingerprints are always extracted from the test set itself. Therefore, the estimated fingerprint, already captures any artifacts introduced by compression, and correlates well with the similarly processed residual images in test set. This is unlike prior work in camera fingerprinting, where camera attribution is attempted between a clean fingerprint (computed over uncompressed images) and a compressed residual image. *Second,* we observe that JPEG compression introduces grid-like artifacts into the fingerprint, further making the model fingerprint distinct from device fingerprints. Fingerprints subjected to JPEG compression are shown in Figure 23 in Appendix A. Compression does disrupt the fingerprint pattern. *NoiseScope*, however, continues to remain effective.

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

To recover from this attack, we resort to *adversarial training* of the fingerprint classification module. We train the classifier module on fingerprints computed from real images that goes through the same post-processing countermeasure used by the attacker. Results are shown in the 'Recovery' column under the specific countermeasure. We observe an improvement in detection performance for both BigGAN-DogHV, and CycleGAN-Zebra, while performance for the other datasets remain unaffected. Lastly, an interesting case is that of StyleGAN-Face2. Performance actually increases for this dataset on applying the countermeasure. On further inspection, we observe that the countermeasure introduces new distinct artifacts in the fingerprints, that enable *NoiseScope* to still accurately cluster images, and detect them. We suspect that images in this dataset has already undergone additional post-processing, which is likely introducing these artifacts when new processing is applied.

Other post-processing schemes to disrupt fingerprints. We evaluate against 4 image post-processing countermeasures known to disrupt camera fingerprinting [31, 49, 68, 70]. Whenever available, we use settings from prior work.

Gamma correction. Gamma correction is applied to fake images with gamma values randomly sampled from U [1.0, 2.0] [80]. Performance remains high for all datasets, except StyleGAN-Face2 where F1 score drops to 62%. Further investigation reveals that the fingerprint classifier performs poorly in 2/5 trials. For recovery, we again apply adversarial training to the fingerprint classifier, and train on

[12] Applying the BM3D filter to 1024x1024 images is computationally expensive.

Datasets	F1 Score (%) with Different Countermeasures									
	Original	JPEG compression	Wavelet-denoising		Gamma correction		Histogram equalization	Blur	Adding noise	
	-	Attack	Attack	Recovery	Attack	Recovery	Attack	Attack	Attack	Recovery
StyleGAN-Face1	99.6	99.3	99.4	99.4	99.5	99.5	99.2	99.4	99.7	99.2
StyleGAN-Face2	90.1	98.3	98.0	98.0	62.0	82.8	72.9	80.0	54.9	81.4
BigGAN-DogHV	92.6	89.5	57.9	87.0	92.9	93.4	88.9	55.7	88.4	78.6
PGGAN-Face	99.1	98.9	99.2	99.2	98.9	98.9	99.2	98.5	97.9	96.2
CycleGAN-Zebra	92.8	91.0	53.7	89.2	91.9	92.6	84.4	61.8	51.8	85.9

Table 7: Performance (F1) of *NoiseScope* under different countermeasures. 'Original' means no countermeasures were used.

real images that undergo the same post-processing. Performance of StyleGAN-Face2 recovers from 62% to 82%.

Histogram equalization. Histogram equalization involves distributing the intensity range to improve image contrast. We apply histogram equalization to fake images. Detection performance remains high for all datasets except StyleGAN-Face2. Fingerprint extraction did not perform well, and StyleGAN-Face2 ended up with impure clusters (purity ranging between 60% to 70%). *We do not attempt recovery from this countermeasure because on examination of fake images that missed detection, we see that image quality has been severely degraded. Therefore, to evade detection by* NoiseScope*, post-processing that significantly degrades image quality is required.* Image samples are shown in Figure 22 (Appendix A).

Blur. Blurring performs a normalized box averaging on fake images, with a specific kernel size [7]. Kernel size is randomly selected from {1, 3, 5, 7} [80]. We expect blurring to damage patterns in the model fingerprints. Performance for StyleGAN-Face2, CycleGAN-Zebra and StyleGAN-Face2 end up dropping. We find that blurring largely weakens the fingerprint pattern. However, on closer investigation of images that were not caught, we find that image quality has degraded significantly—*NoiseScope* failed to catch fake images that were severely blurred. Therefore, we do not attempt a recovery scheme. Figures 20, 21 in Appendix A show samples of images that evaded detection.

Adding Noise. We add i.i.d. Gaussian noise to fake images. The noise variance is randomly sampled from U [5.0, 20.0] [80]. CycleGAN-Zebra, and StyleGAN-Face2 shows significant drop in performance. In both cases, noise degrades the quality of the fingerprint, making them unsuitable for computing correlation with residual images. In the case of CycleGAN-Zebra, the fingerprint classifier also fails to detect model fingerprints. To recover, we apply a denoising filter (Non-Local-Means) to all images in the test set, and also perform adversarial training of the fingerprint classifier using the same denoising filter. We apply this strategy to all datasets, and we can see that performance of StyleGAN-Face2 and CycleGAN-Zebra are regained to 81.4% and 85.9%, respectively, but recovery slightly hurts BigGAN-BurgHV by 10% due to the denoising operation.

Fingerprint spoofing. Fingerprint spoofing attack aims to disguise fake images to be from a specific camera device. This attack is commonly studied in the camera fingerprinting literature [4, 44, 81]. We use the StyleGAN-Face1 dataset to evaluate this countermeasure. We consider a fingerprint substitution attack [44] using the following formulation: $I_s = I - \alpha F_a + \beta F_b$. F_a, and F_b, are model and camera fingerprints, respectively. F_a is computed using 200 fake images from StyleGAN-Face1, and F_b is a camera fingerprint computed using 200 images from Canon EOS 5D Mark III (FFHQ dataset). The first step is to verify that we have correctly spoofed the camera fingerprint. We empirically estimate α as 1.5, and β as 1.5, as the spoofed fingerprint shows low PCE correlation with the model fingerprint, and high PCE correlation with the camera fingerprint, while maintaining image quality. We then consider a worst-case scenario for the defender, where the test set contains 200 spoofed fake images, and 200 real images which are used to extract F_b. We perform detection on such test sets with 5 trials. We obtain a low average F1 score of 66.67%. On closer investigation, we find the fake image detection module performed poorly because the fingerprints have been spoofed.

To recover from this attack, we utilize a different filter, *i.e.,* a normalizing box (blur) filter, instead of the Wavelet denoiser to compute residuals. The intuition is that the spoofing attack does not destroy all the artifacts (produced by the GAN), *i.e.,* a model fingerprint can still be extracted. In fact, the performance is regained to 94.56% F1. *Therefore, use of alternative filters for residual extraction is an effective recovery strategy against fingerprint spoofing attacks.* One might argue that attackers can spoof the new residual space used in the residual extractor again. However, an endless game of switching residual extractors (multiple filters and filter parameters) is unlikely. If an attacker tries spoofing against multiple filters, then we observe that image quality deteriorates significantly. Image samples spoofed against multiple filters are shown in Figure 19 in the Appendix.

Adapting the GAN model. Can the attacker modify the GAN to bypass detection? For example, for many DNN-based supervised detection schemes, the attacker can use the defender's classifier as the GAN discriminator, and produce images that evade detection. In our case, such countermeasures are hard. *First*, the model fingerprints extracted by *NoiseScope* is tied to the fundamental building blocks of generative models, *i.e.,* deconvolution layers (see Section 5.1). One can try to change the deconvolution layer parameters, which will change the fingerprint patterns, but is unlikely to make it similar to device fingerprints. *Second*, the attacker can use the fake image detector component of *NoiseScope* as the GAN discriminator. However, one has to ensure that the operations are differentiable, which is non-trivial. Also, such an effort would be similar to our previous countermeasures of spoofing the fingerprint or using the defender's filter. We have already discussed robustness of *NoiseScope* against such countermeasures.

Summary. We evaluated a range of challenging countermeasures against *NoiseScope*. *NoiseScope* is resilient against compression attacks, considered to be challenging in prior work. We also recommend effective recovery schemes against different types of post-processing attacks—Wavelet-noising, adding noise, and Gamma correction. The countermeasures that evaded detection includes those that degraded image quality significantly and can be considered as unsuccessful countermeasures. Online platforms like news/social media sites collecting images, can reject images that

are excessively post-processed. There is ongoing work on detecting image manipulations or post-processing. For example, Adobe recently developed new tools to detect Photoshopped images [76]. *NoiseScope* can leverage such tools and implement appropriate recovery measures to make its detection pipeline more resilient to different countermeasures.

7 CONCLUSION

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

REFERENCES

[1] Darius Afchar, Vincent Nozick, Junichi Yamagishi, and Isao Echizen. 2018. Mesonet: a Compact Facial Video Forgery Detection Network. In *Proc. of WIFS.*

[2] Michael Albright and Scott McCloskey. 2019. Source Generator Attribution via Inversion. In *Proc. of CVPR Workshop on Media Forensics.*

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein Generative Adversarial Networks. In *Proc. of ICML.*

[4] Sudipta Banerjee, Vahid Mirjalili, and Arun Ross. 2019. Spoofing PRNU Patterns of Iris Sensors While Preserving Iris Recognition. In *Proc. of ISBA.*

[5] Christian F Baumgartner, Lisa M Koch, Kerem Can Tezcan, Jia Xi Ang, and Ender Konukoglu. 2018. Visual Feature Attribution Using Wasserstein GANs. In *Proc. of CVPR.*

[6] Greg J Bloy. 2008. Blind Camera Fingerprinting and Image Clustering. *In Proc. of TPAMI* (2008).

[7] G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[8] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: Identifying Density-based Local Outliers. In *ACM Sigmod Record.*

[9] Andrew Brock, Jeff Donahue, and Karen Simonyan. 2019. Large Scale GAN Training For High Fidelity Natural Image Synthesis. In *Proc. of ICLR.*

[10] Antoni Buades, Bartomeu Coll, and J-M Morel. 2005. A Non-local Algorithm For Image Denoising. In *Proc. of CVPR.*

[11] Mo Chen, Jessica Fridrich, and Miroslav Goljan. 2007. Digital Imaging Sensor Identification. In *Proc. of Security, Steganography, and Watermarking of Multimedia Contents.*

[12] Mo Chen, Jessica Fridrich, Miroslav Goljan, and Jan Lukás. 2008. Determining Image Origin and Integrity Using Sensor Noise. *IEEE Transactions on Information Forensics and Security* (2008).

[13] Giovanni Chierchia, Sara Parrilli, Giovanni Poggi, Carlo Sansone, and Luisa Verdoliva. 2010. On the Influence of Denoising in PRNU Based Forgery Detection. In *Proc. of MiFor Workshop.*

[14] NVIDIA CORPORATION. 2019. StyleGAN-Bed Fake Image Source. https://drive.google.com/drive/folders/1Vxz9fksw4kgjiHrvHkX4Hze4dyThFW6t.

[15] NVIDIA CORPORATION. 2019. StyleGAN-Face1 Fake Image Source. https://drive.google.com/drive/folders/14lm8VRN1pr4g_KVe6_LvyDX1PObst6d4.

[16] Jack Corrigan. 2019. Darpa Is Taking on the Deepfake Problem. https://www.nextgov.com/emerging-tech/2019/08/darpa-taking-deepfake-problem/158980/.

[17] Andrea Cortiana, Valentina Conotter, Giulia Boato, and Francesco GB De Natale. 2011. Performance Comparison of Denoising Filters for Source Camera Identification. In *Proc. of MWSF.*

[18] Davide Cozzolino and Luisa Verdoliva. 2019. Noiseprint: a CNN-based Camera Model Fingerprint. *IEEE Transactions on Information Forensics and Security* (2019).

[19] Antonia Creswell and Anil Anthony Bharath. 2018. Inverting the Generator of A Generative Adversarial Network. *IEEE Transactions on Neural Networks and Learning Systems* (2018).

[20] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. 2007. Image Denoising by Sparse 3-D Transform-domain Collaborative Filtering. *IEEE Transactions on Image Processing* (2007).

[21] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A Large-scale Hierarchical Image Database. In *Proc. of CVPR.*

[22] Chris Donahue, Julian McAuley, and Miller Puckette. 2019. Adversarial Audio Synthesis. In *Proc. of ICLR.*

[23] Vincent Dumoulin and Francesco Visin. 2016. A Guide to Convolution Arithmetic for Deep Learning. *arXiv preprint arXiv:1603.07285* (2016).

[24] Donie O'Sullivan et al. 2019. Inside the Pentagon's Race Against Deepfake Videos. https://www.cnn.com/interactive/2019/01/business/pentagons-race-against-deepfakes/.

[25] Facebook. 2019. Creating A Data Set and A Challenge for Deepfakes. https://ai.facebook.com/blog/deepfake-detection-challenge/.

[26] Maayan Frid-Adar, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. 2018. Synthetic Data Augmentation Using GAN for Improved Liver Lesion Classification . In *Proc. of ISBI.*

[27] E. S. Gedraite and M. Hadad. 2011. Investigation on the Effect of a Gaussian Blur In Image Filtering and Segmentation . In *Proc. of ELMAR.*

[28] Inc Generated Media. 2019. StyleGAN-Face2 Fake Image Source. https://drive.google.com/drive/folders/1wSy4TVjSvtXeRQ6Zr8W98YbSuZXrZrgY.

[29] Generated Media, Inc. 2019. Unique, worry-free model photos. https://generated.photos/.

[30] Miroslav Goljan. 2008. Digital Camera Identification From Images – Estimating False Acceptance Probability . In *Proc. of International Workshop on Digital Watermarking.*

[31] Miroslav Goljan, Mo Chen, Pedro Comesaña, and Jessica Fridrich. 2016. Effect of Compression on Sensor-fingerprint Based Camera Identification. *Electronic Imaging* (2016).

[32] Miroslav Goljan, Jessica Fridrich, and Tomáš Filler. 2009. Large Scale Test of Sensor Fingerprint Camera Identification. In *Proc. of Media Forensics and Security.*

[33] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Proc. of NeurIPS.*

[34] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long Text Generation via Adversarial Training with Leaked Information . In *Proc. of AAAI.*

[35] R. M. Haralick. 1979. Statistical and Structural Approaches to Texture. *IEEE* (1979).

[36] TensorFlow Hub. 2019. BigGAN Deep Pretrained Model. https://tfhub.dev/deepmind/biggan-deep-256/1.

[37] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image Translation with Conditional Adversarial Networks . In *Proc. of CVPR.*

[38] Xiang Jiang, Shikui Wei, Ruizhen Zhao, Yao Zhao, and Xindong Wu. 2016. Camera Fingerprint: A New Perspective for Identifying User's Identity . *arXiv preprint arXiv:1610.07728* (2016).

[39] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2018. Progressive Growing Of GANs for Improved Quality, Stability, and Variation. In *Proc. of ICLR.*

[40] Tero Karras, Samuli Laine, and Timo Aila. 2019. A Style-based Generator Architecture for Generative Adversarial Networks. In *Proc. of CVPR.*

[41] Diederik P Kingma and Max Welling. 2014. Auto-encoding Variational Bayes . In *Proc. of ICLR.*

[42] Naman Kohli, Daksha Yadav, Mayank Vatsa, Richa Singh, and Afzel Noore. 2017. Synthetic Iris Presentation Attack Using iDCGAN . In *Proc. of IJCB.*

[43] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. *Learning Multiple Layers of Features from Tiny Images.* Technical Report. Citeseer.

[44] Chang-Tsun Li, Chih-Yuan Chang, and Yue Li. 2009. On the Repudiability of Device Identification and Image Integrity Verification Using Sensor Pattern Noise. In *Proc. of ISDF.*

[45] Haodong Li, Bin Li, Shunquan Tan, and Jiwu Huang. 2018. Detection of Deep Network Generated Images Using Disparities in Color Components . *arXiv preprint arXiv:1808.07276* (2018).

[46] Xiaodan Liang, Lisa Lee, Wei Dai, and Eric P Xing. 2017. Dual Motion GAN For Futureflow Embedded Video Prediction. In *Proc. of ICCV.*

[47] Ming-Yu Liu and Oncel Tuzel. 2016. Coupled Generative Adversarial Networks . In *Proc. of NeurIPS.*

[48] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proc of ICCV.*

[49] Jan Lukáš, Jessica Fridrich, and Miroslav Goljan. 2006. Digital Camera Identification From Sensor Pattern Noise. *IEEE Transactions on Information Forensics and Security* (2006).

[50] Tim Mak. 2018. Can You Believe Your Own Ears? With New 'Fake News' Tech, Not Necessarily. https://www.npr.org/2018/04/04/599126774/can-you-believe-your-own-ears-with-new-fake\-news-tech-not-necessarily.

[51] Neal Mangaokar, Jiameng Pu, Parantapa Bhattacharyam, Chandan Reddy, and Bimal Viswanath. 2020. Jekyll: Attacking Medical Image Diagnostics Using Neural Translation. In *Proc. of Euro S&P.*

[52] Francesco Marra, Diego Gragnaniello, Davide Cozzolino, and Luisa Verdoliva. 2018. Detection Of GAN-generated Fake Images Over Social Networks. In *Proc. of MIPR.*

[53] Francesco Marra, Diego Gragnaniello, Luisa Verdoliva, and Giovanni Poggi. 2019. Do GANs Leave Artificial Fingerprints?. In *Proc. of MIPR*.
[54] M Kivanc Mihcak, Igor Kozintsev, and Kannan Ramchandran. 1999. Spatially Adaptive Statistical Modeling Of Wavelet Image Coefficients And Its Application To Denoising. In *Proc. of ICASSP*.
[55] Huaxiao Mo, Bolin Chen, and Weiqi Luo. 2018. Fake Faces Identification Via Convolutional Neural Network. In *Proc. of IH&MMSEC*.
[56] Benjamin Moseley, Kefu Lu, Silvio Lattanzi, and Thomas Lavastida. 2019. A Framework for Parallelizing Hierarchical Clustering Methods. In *Proc. of ECML PKDD*.
[57] Lakshmanan Nataraj, Tajuddin Manhar Mohammed, BS Manjunath, Shivkumar Chandrasekaran, Arjuna Flenner, Jawadul H Bappy, and Amit K Roy-Chowdhury. 2019. Detecting GAN Generated Fake Images Using Co-occurrence Matrices . *arXiv preprint arXiv:1903.06836* (2019).
[58] Kamyar Nazeri, Eric Ng, and Mehran Ebrahimi. 2018. Image Colorization Using Generative Adversarial Networks. In *Proc. of International Conference on Articulated Motion and Deformable Objects*.
[59] A Miranda Neto, A Correa Victorino, Isabelle Fantoni, Douglas Eduardo Zampieri, Janito Vaqueiro Ferreira, and Danilo Alves Lima. 2013. Image Processing Using Pearson's Correlation Coefficient: Applications on Autonomous Robotics. In *Proc. of ICARSC*.
[60] BBC News. 2019. Deepfake Videos Could 'Spark' Violent Social Unrest. https://www.bbc.com/news/technology-48621452.
[61] Augustus Odena, Vincent Dumoulin, and Chris Olah. 2016. Deconvolution and Checkerboard Artifacts. http://distill.pub/2016/deconv-checkerboard.
[62] Clark F. Olson. 1995. Parallel Algorithms for Hierarchical Clustering. *Parallel Comput.* (1995).
[63] Jon Porter. 2019. 100,000 Free AI-generated Headshots Put Stock Photo Companies on Notice. https://www.theverge.com/2019/9/20/20875362/100000-fake-ai-photos-stock-photography-\royalty-free.
[64] Corinne Reichert Queenie Wong. 2019. Facebook Removes Bogus Accounts That Used AI to Create Fake Profile Pictures. https://www.cnet.com/news/facebook-removed-fake-accounts-that-used-ai-to-create-fake-profile-pictures/.
[65] Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *Proc. of ICLR*.
[66] S. Rajasekaran. 2005. Efficient Parallel Hierarchical Clustering Algorithms. *IEEE Transactions on Parallel and Distributed Systems* (2005).
[67] NVIDIA Research. 2019. Fake Image Source of PGGAN-Face and PGGAN-Tower. https://drive.google.com/drive/folders/1j6uZ_a6zci0HyKZdpDq9kSa8VihtEPCp.
[68] Kurt Rosenfeld and Husrev Taha Sencar. 2009. A Study of the Robustness of PRNU-based Camera Identification. In *Proc. of Media Forensics and Security*.
[69] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. 2019. Faceforensics++: Learning to Detect Manipulated Facial Images. In *Proc. of ICCV*.
[70] Stamatis Samaras, Vasilis Mygdalis, and Ioannis Pitas. 2016. Robustness in Blind Camera Identification. In *Proc. of ICPR*.
[71] SmugMug, Inc. [n.d.]. Flickr Website. https://www.flickr.com/.
[72] Shahroz Tariq, Sangyup Lee, Hoyoung Kim, Youjin Shin, and Simon S Woo. 2018. Detecting Both Machine and Human Created Fake Face Images in the Wild. In *Proc. of MPS*.
[73] Dr. Matt Turek. [n.d.]. Media Forensics (MediFor). https://www.darpa.mil/program/media-forensics.
[74] https://faceswap.dev//. [n.d.]. Deepfakes FaceSwap. https://github.com/deepfakes/faceswap.
[75] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. 2016. Generating Videos with Scene Dynamics. In *Proc. of NeurIPS*.
[76] Sheng-Yu Wang, Oliver Wang, Andrew Owens, Richard Zhang, and Alexei A Efros. 2019. Detecting Photoshopped Faces by Scripting Photoshop. In *Proc. of ICCV*.
[77] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. 2020. CNN-generated Images are Surprisingly Easy to Spot... for Now. In *Proc. of CVPR*.
[78] Xin Yang, Yuezun Li, and Siwei Lyu. 2019. Exposing Deep Fakes Using Inconsistent Head Poses. In *Proc. of ICASSP*.
[79] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. 2015. LSUN: Construction of a Large-scale Image Dataset Using Deep Learning with Humans in The Loop. *arXiv preprint arXiv:1506.03365* (2015).
[80] Ning Yu, Larry S Davis, and Mario Fritz. 2019. Attributing Fake Images to GANs: Learning and Analyzing GAN Fingerprints. In *Proc. of ICCV*.
[81] Hui Zeng, Jiansheng Chen, Xiangui Kang, and Wenjun Zeng. 2015. Removing Camera Fingerprint to Disguise Photograph Source. In *Proc. of ICIP*.
[82] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. 2019. Self-attention generative adversarial networks. In *Proc. of ICML*.
[83] Xu Zhang, Svebor Karaman, and Shih-Fu Chang. 2019. Detecting and Simulating Artifacts in GAN Fake Images. In *Proc. of WIFS*.
[84] Zhengjun Zhang. 2008. Quotient Correlation: A Sample Based Alternative to Pearson's Correlation. *The Annals of Statistics* (2008).
[85] Jun-Yan Zhu. [n.d.]. Real Image Source of CycleGAN-Winter and CycleGAN-Zebra. https://people.eecs.berkeley.edu/~taesung_park/CycleGAN/datasets/.
[86] Jun-Yan Zhu. 2018. CycleGAN Pretrained Models. https://github.com/junyanz/CycleGAN.
[87] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired Image-to-image Translation Using Cycle-consistent Adversarial Networks. In *Proc. of ICCV*.

A IMAGE SAMPLES

Figure 8: Fake samples from BigGAN-DogLV [36].

Figure 9: Fake samples from BigGAN-DogHV [36].

Figure 10: Fake samples from BigGAN-BurgLV [36].

Figure 11: Fake samples from BigGAN-BurgHV [36].

Figure 12: Fake samples from CycleGAN-Zebra [86].

Figure 13: Fake samples from CycleGAN-Winter [86].

Figure 14: Fake samples from PGGAN-Tower [67].

Figure 15: Fake samples from StyleGAN-Bed [14].

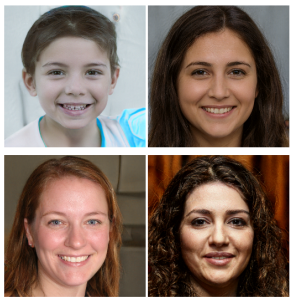


Figure 16: Fake samples from StyleGAN-Face1 [15].

Figure 17: Fake samples from StyleGAN-Face2 [28].

Figure 18: Fake samples from PGGAN-Face [67].

(a) (b) (c) (d)

Figure 19: Image samples from StyleGAN-Face2 [28] subjected to a fingerprint spoofing attack against an increasing number of residual spaces. From left to right, we present (a) the original image, (b) the image spoofed against the Wavelet residual space (c) the image spoofed against the Wavelet and Blur residual spaces, and (d) the image spoofed against the Wavelet, Blur, and Laplacian residual spaces.

Figure 20: Samples from CycleGAN-Zebra [86] that evaded detection when blurred. Top row shows the images before blurring, and the bottom row shows the images after blurring.

Figure 22: Samples from StyleGAN-Face2 [28] that evaded detection when subjected to histogram equalization. Top row shows the images before equalizing, and the bottom row shows the images after equalizing.

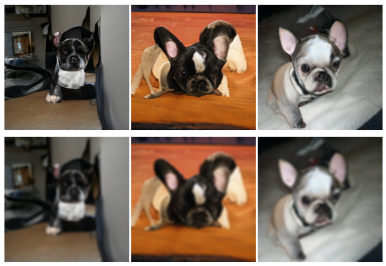


Figure 21: Samples from BigGAN-DogHV [36] that evaded detection when blurred. Top row shows the images before blurring, and the bottom row shows the images after blurring.

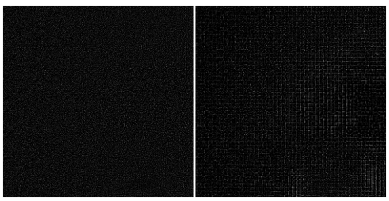


Figure 23: Model fingerprints from StyleGAN-Face2 [28], before (left) and after (right) applying JPEG compression.