Dodging DeepFake Detection via Implicit Spatial-Domain Notch Filtering

Yihao Huang*, Felix Juefei-Xu*, Member, IEEE, Qing Guo*, Member, IEEE, Yang Liu*, Senior Member, IEEE, and Geguang Pu*

Abstract—The current high-fidelity generation and high-precision detection of DeepFake images are at an arms race. We believe that producing DeepFakes that are highly realistic and “detection evasive” can serve the ultimate goal of improving future generation DeepFake detection capabilities. In this paper, we propose a simple yet powerful pipeline to reduce the artifact patterns of fake images without hurting image quality by performing implicit spatial-domain notch filtering. We first demonstrate that frequency-domain notch filtering, although famously shown to be effective in removing periodic noise in the spatial domain, is infeasible for our task at hand due to the manual designs required for the notch filters. We, therefore, resort to a learning-based approach to reproduce the notch filtering effects, but solely in the spatial domain. We adopt a combination of adding overwhelming spatial noise for breaking the periodic noise pattern and deep image filtering to reconstruct the noise-free fake images, and we name our method DeepNotch. Deep image filtering provides a specialized filter for each pixel in the noisy image, producing filtered images with high fidelity compared to their DeepFake counterparts. Moreover, we also use the semantic information of the image to generate an adversarial guidance map to add noise intelligently. Our large-scale evaluation on 3 representative DeepFake detection methods (tested on 16 types of DeepFakes) has demonstrated that our technique significantly reduces the accuracy of these 3 fake image detection methods, 36.79% on average and up to 97.02% in the best case.

I. SOCIETAL IMPACT

Our proposed method, just like many other high-fidelity image synthesis or DeepFake generation methods, if maliciously used by an adversary, may cause harm to the integrity of digital media and fuel the dissemination of misinformation and disinformation. The study herewith attempts to expose potential vulnerabilities of the deployed defense mechanism with the goal of ultimately improving it by presenting a stronger contender. Our method aims at improving the fidelity of DeepFake images and, more importantly, exposing the problems of existing DeepFake detection methods, and we hope that the found vulnerabilities can help improve future generation DeepFake detection.

II. INTRODUCTION

Fake images produced by the generative adversarial network (GAN) and its variants can now render both photo-realistic and high-fidelity effects, a.k.a. DeepFakes. However, the state-of-the-art (SOTA) GAN-based fake image generation methods are still imperfect, which stems from the upsampling modules in their decoders. In particular, existing upsampling methods of GANs, e.g., transpose convolution, unpooling, and interpolation, often inevitably introduce artifact patterns to the generated fake images, occurring in both spatial-domain and frequency-domain (e.g., Fourier spectrum) representations. For example, checkerboard patterns are typical textures that may be left in the generated fake images. Similarly, typical artifact patterns in the spectrum of fake images are also discussed in [1], [2]. Through leveraging the potential artifact patterns as the clue, quite a few DeepFake detection methods have been proposed [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12]. These methods largely fall into three categories according to their inputs: image-based, fingerprint-based, and spectrum-based, most of which have demonstrated their effectiveness in successfully detecting SOTA DeepFakes.

To generate more photo-realistic and high-fidelity fake images, one promising direction is to reduce the artifact patterns introduced in fake images. Along this line, in this paper, we propose a pipeline DeepNotch to perform implicit spatial-domain notch filtering on the DeepFake images to make them more realistic and detection evasive (Fig. 1).
In summary, the contributions as listed below:

- We propose the first DL-based fake image retouch method to reduce the artifact by performing implicit notch filtering. The reconstructed fake images are both photo-realistic and have a strong capability in bypassing SOTA fake detectors. To further improve the effectiveness of DeepNotch, we propose a novel semantic-aware localization method to pinpoint the places for noise addition.
- We perform a large-scale evaluation on 3 SOTA fake image detection methods and the fake images are generated by a total of 16 popular DeepFake generation methods. In particular, our reconstructed fake images can significantly reduce the fake detection accuracy of DeepFake detectors and they exhibit a high level of fidelity compared to their original fake image counterparts.
- Our method indicates that existing detection methods highly leverage the information of artifact patterns for fake detection. The observation also raises an open question of how to propose more general DeepFake detection methods.

III. RELATED WORK

1)GAN-Based Image Generation & Manipulation: Since its advent, GANs [13] have been extensively studied with many GAN-based image generation methods proposed [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27]. Specifically, IcGAN first encodes real images into the latent space and then changes the latent codes corresponding to different facial properties. After that, it decodes the changed latent codes to fake face images. In recent years, there have been some GAN-based image generation and manipulation methods that put emphasis on stably generating images with high-resolution and controllable face attributes. StyleGAN [19] designs a different generator architecture that leads to an automatically learned separation of high-level attributes and stochastic variation in the generated images. Based on StyleGAN, StyleGAN2 [20] further proposes improvements on both model architecture and training methods for higher image generation quality. SNGAN [21] proposes a lightweight normalization method to stabilize and enhance the training of the discriminator. MMDGAN [22] combines the key ideas in both the generative moment matching network (GMMN) and GAN. Some GANs are also specially designed to manipulate the facial attributes of real images. AttGAN [23] adopts an attribute detection constraint to the generated image instead of latent representation to guarantee the correct change of desired attributes. STGAN [25] selectively analyzes the difference between target and source attribute vectors and adaptively modifies the encoder feature for enhanced attribute editing. StarGAN [24] proposes a novel approach that can perform image-to-image translations for multiple domains using only a single model.

2) DeepFake Detection Methods: [30] and [31] recently conduct comprehensive surveys on the DeepFake detection methods [1], [2], [3], [4], [32], [33], [34], [35], [36], [37], [38], [39], [40]. Overall, they surveyed hundreds of papers that focus on DeepFake detection, most of which proposed CNN-based deep learning methods to solve the detection problem. The methods mainly leverage image clues or biological signal clues to address the detection tasks. Some work such as [41] (which provide a comprehensive overview by leveraging distributed ledger technologies (DLT) to combat digital deception) and [42] (leverage blockchain to trace and track the source of multimedia which provides insight for combating DeepFake videos) are not regular detection based on pure image analysis are not considered in our work.

The biological signal exhibits a clear signal for distinguishing between real and fake. The biological signals revealed in the real faces videos are natural and realistic while is low-quality in fake videos. Early works study the irregular eye blinking [43], the mismatch facial landmarks [44], etc. However, these visual inconsistencies could be easily removed
in the advanced DeepFakes. In recent years, some works [38], [39], [40] took heartbeats as the clue to classify the videos. Some work [45], [46], [47], [48] detect DeepFake according to the inconsistency of visual and audio. Since the biological signal-based detection methods mainly focus on video and are hardly accessible on images, thus our image reconstruction method does not take them into consideration.

For detection methods that consider image clues, they largely fall into three categories depending on their feature inputs: image-based methods [4], [32], [34], [35], [36], fingerprint-based methods [3], [49], and spectrum-based methods [1], [2], [11]. As many CNN did, image-based methods perform fake detection directly on the original images (as inputs). Fingerprint-based methods follow the intuition that different GANs have various fingerprints. Through analyzing the features of GAN fingerprints, they can successfully detect fake images in many cases. Spectrum-based methods take another perspective, which leverages spectrum as the input of their network for more effective fake image detection, with the intuition that DeepFake artifacts are manifested as replications of spectra in the frequency domain.

3) Diffusion Model: Recently, diffusion models (DM) [50], [51], [52] are versatile tools with a wide array of applications, such as image generation [53], image translation [54], voice synthesis [55] and adversarial defense [56]. Diffusion model performing iteratively adding Gaussian noise on an image as a forward process and denoising to restore the image by a backward process. From this point of view, our work shows similarity with the diffusion model in that we first add Gaussian noise to the fake image and then denoise with the deep learning-based method. Since DM-based image-to-image translation [54] shows the ability to transform the image from a specific domain to another domain, we think it somewhat verifies the reasonability of our work to transform the fake image from a fake domain to a real domain.

IV. METHODOLOGY

A. Motivation

It is widely observed that various GAN-based image generation methods leave some footprints of artifacts in the image’s Fourier spectrum, e.g., bright spots that are symmetric about the origin, star-shaped line segments symmetrically shooting from the origin, etc., as shown in Fig. 2. The artifacts correspond to fake patterns in the spatial domain. For various DeepFake generation methods, such artifacts usually appear to be checkerboard-like in the spatial domain, producing several pairs of origin-symmetric bright spots (spikes) in the Fourier spectrum, indicating that the pattern contains more than just one sinusoidal component.

In image processing literature [57], a common and effective way of removing or mitigating fake patterns in the spatial domain is to apply frequency-domain filtering with a notch filter. A notch filter (or notch reject filter in this case) can be of various shapes, sizes, and orientations, and applied at various spectrum locations, etc. Depending on how the artifacts look in the spectrum, the notch filter can be designed manually to maximally reduce the energy surrounding the artifacts. In our case, e.g., the exhibited artifacts are bright spots symmetrically positioned in the spectrum (see Fig. 3 (a)). Therefore, any circular-shaped notch filter such as disk filters, Butterworth filters, or Gaussian filters, will suffice to encapsulate the artifact spikes. In Fig. 3 (b-e), applying disk and Gaussian notch filters at specific spectrum locations can effectively remove the checkerboard pattern in the spatial domain. However, we will explain why notch filtering in the frequency domain is not a feasible solution for automatic detection-evasion.

Here are the main challenges hindering a successful application of notch filtering in the frequency domain. ➀ Different GANs may result in different spikes (bright spots) pairs in the spectrum and at different locations. Therefore, human-in-the-loop localization of the spikes is required, rendering it infeasible. ➁ Spikes are the simplest types of artifacts, and there are artifacts with many complicated patterns, e.g., requiring rectangular notch filters with notch openings positioned at particular locations, and the filter positioned at particular orientations. All these efforts require human involvement. ➂ Even if some forgery method produces quite consistent artifacts, e.g., with fixed spike location, the spikes in the image spectrum may rotate or shift due to simple geometric transformation of the generated image (see a toy example in Fig. 4), where slight rotation and scaling transformation are applied to the image and the spikes in the spectrum are relocated. Automatically designing notch filters to account for possible geometric perturbation is infeasible in general. ➃ For partially exhibited fake patterns in the spatial domain, the spikes are usually energy-spread into a cloud shape in the spectrum, making the determination of notch sizes less definite.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
For these reasons, it is usually not easy to automatically design a frequency-domain notch filter based on the fake patterns in the spatial domain. Also, there is usually not a corresponding convolution-based spatial-domain filter to achieve fake pattern removal due to the nature of the notch filter [57]. Therefore, notch filtering, although very effective in removing fake patterns in the spatial domain, without human involvement, it has limited usage in tackling the DeepFake detection evasion task at hand. As we will show in Sec. 2.2, directly applying spatial filtering to remove spatial fake patterns is ineffective. That is why in this work, we resort to a learning-based approach to perform implicit deep notch filtering in the spatial domain by first proactively breaking the fake patterns by adding overwhelming spatial noises (diminish the spikes in the image spectrum), and then applying per-pixel image deep filtering to remove the noise (i.e., restore image quality). We call this method DeepNotch.

B. Deep Image Filtering for DeepNotch

Given a fake image $\mathbf{I} \in \mathbb{R}^{H \times W}$ that is generated by some DeepFake technique (e.g., ProGAN), we aim to produce a new version of $\mathbf{I}$ (i.e., $\hat{\mathbf{I}} \in \mathbb{R}^{H \times W}$) by reducing the artifact patterns introduced by that technique, retouching the input fake image to make it more realistic. This task is significantly challenging since the content of fake images could be quite diverse and various GANs introduce different artifact textures into fake images. Because of the double complexity of the content and texture, it is rather difficult to employ DNNs to handle the image directly. To address the challenge, we formulate the DeepNotch as a general deep image filtering problem to perform implicit notch filtering, where the input image is processed by pixel-wise kernels estimated from an offline trained DNN:

$$\hat{\mathbf{I}} = \mathbf{K} \circ \mathbf{I} \quad \text{with} \quad \mathbf{K} = \text{DNN}(\mathbf{I}),$$

where DNN(·) denotes a UNet-like [58] deep neural network used to predict the pixel-wise kernels $\mathbf{K} \in \mathbb{R}^{H \times W \times K^2}$. $\circ$ denotes pixel-wise filtering. More specifically, the $p$-th pixel in the image $\mathbf{I}$ is processed by the corresponding $p$-th kernel in $\mathbf{K}$ denoted as $\mathbf{K}_p \in \mathbb{R}^{K \times K}$, where $K$ represents the kernel size. Then, we can offline train the DNN with fake-real image pairs and the $L_1$ loss function. Obviously, the aforementioned structure seems a feasible and intuitive solution for DeepNotch: First, the fake image is processed by only one single-layer filtering without any upsampling operations, avoiding the retarget risk. Second, the kernels are generated from a DNN, which takes full advantage of deep learning in perceiving the image content and predicting suitable kernels for each pixel.

However, naively training such a DNN for DeepNotch, unfortunately, often cannot reduce the artifacts. This wrong technical route is put in Fig. 1 as a warning. As shown in Fig. 1(a), the typical artifact patterns of a DeepFake image can be visualized in both spatial and frequency domains. When employing the directly trained deep image filtering, i.e., Eq. (1), to the image, the artifacts cannot be easily reduced, i.e., the trace indicated by red arrows in Fig. 1(b). The same conclusion is also reached on our large-scale evaluation (see Table I of the experimental section), where the naive deep image filtering fails to retouch the fake image and misjudge DeepFake detectors. The large-scale experimental results confirm that it is difficult to realize effective DeepNotch via single-layer filtering even if the DNN is equipped. Furthermore, in theory, as indicated by the analysis of the spectral bias of Deep Neural Networks [59], DNNs tend to be biased toward learning lower-frequency functions. This bias implies that comprehending high-frequency variations (e.g., checkerboard patterns) is a challenging task for DNNs. Consequently, it is not straightforward for DNNs to autonomously acquire the knowledge necessary to automatically learn notch filters capable of reducing artifacts in DeepFake images.

Such results force us to rethink the retouching solution. Intuitively, DNN-driven image filtering has demonstrated big advantages for image denoising. Meanwhile, noise can also be regarded as a perturbation in the spatial domain. Therefore, we come up with a bold idea, which first employs noise to destroy the artifacts, and then recovers the deliberate noisy image with DNN-driven image filtering. In other words, we find a way to implement notch filtering in the spatial domain implicitly, that is, first adding random noise to the image to break the periodic checkerboard-like noise pattern in the spatial domain, followed by per-pixel deep image filtering. We present how to realize this idea with the guidance of random noise (i.e., Sec. 2.3). Moreover, we further propose an advanced version by employing the popular adversarial attack to generate semantic-aware noise for guidance (i.e., Sec. 2.4).

C. Random-Noise-Guided Image Filtering

We first give a simple example to explain that adding random noise can effectively reduce the artifacts. As shown in Fig. 5, we use a pair of real-fake images from StarGAN as an example. In the first two columns, we show real-fake image pairs and their corresponding spectrums. We can find that DeepFake techniques can corrupt the spectrum of the real face image and introduce the bright blob patterns shown in the spectrum of the fake image. After adding zero-mean Gaussian noise with a standard deviation of five to the fake image, see the third column, surprisingly, the artifacts in the spectrum are reduced. To demonstrate the effect of noise across different fake images, we conduct extra experiments by adding noise to DeepFake images and calculating the average spectrum of these images (compared with visualizing one image). As shown in Fig. 6, we illustrate the spectrums of SNGAN [21], CycleGAN [60], SITD [61] and StarGAN [24],
the same network in Eq. (1) to predict the kernels, which is the same conclusion. Given a fake random-noise-guided image filtering method. The second column shows the result of adding Gaussian noise ($\sigma = 5, \mu = 0$) to the fake image. It does not have blobs in the spectrum. In addition, the deeply filtered image using deep image filtering on the noised image also exhibits no artifact patterns.

We choose these four GANs because the artifacts are obvious in their spectrums and thus we can display the effect of noise better. For each of them, we use dozens to tens of thousands (the same as in the experiment setting (Sec. V-A)). The two noises chosen by us are Gaussian noise and uniform noise. The mean of Gaussian noise is 0 while the standard deviation is 10. The lower bound and upper bound of uniform noise are -20 and 20, respectively. Take the last column as an example (the StarGAN column), we can find that the average spectrum is 10. The lower bound and upper bound of uniform noise are -20 and 20, respectively. Take the last column as an example (the StarGAN column), we can find that the average spectrum is 10. The lower bound and upper bound of uniform noise are -20 and 20, respectively. Take the last column as an example (the StarGAN column), we can find that the average spectrum is 10.

According to the above observation, we first propose the random-noise-guided image filtering method. Given a fake image $I$, we add random noise to it and process it with pixel-wise kernels, thus we can reformulate Eq. (1) as

$$\hat{I} = K \oplus (I + N_\sigma),$$

where $N_\sigma$ denotes the Gaussian noise map with standard deviation $\sigma$ and has the same size with $I$. We can employ the same network in Eq. (1) to predict the kernels, which is offline trained with the fake-real image pairs and $L_1$ loss.

As shown in Fig. 5, with the random-noise-guided image filtering, both the artifacts and deliberate noise are clearly reduced, i.e., the spectrum of the deeply filtered image is clean and similar to the real one. Nevertheless, as a kind of degradation, the random noise can potentially reduce the image quality. Ideally, it is highly desirable the noise addition is as minor as possible, at the same time, reducing fake artifacts effectively. As a result, the fake can not be detected by the advanced DeepFake detectors. This requires the noise addition method to be ‘smart’ or semantic-aware, i.e., adding the noise to the proper locations.

**D. Adversarial-Noise-Guided Image Filtering**

Inspired by adversarial attacks, which perform semantic-aware and sparse noise perturbation, we further propose the adversarial-noise-guided image filtering, where a simple DeepFake detection method trained by us is employed as a subject model to generate the adversarial noise and an $L_1$ constraint is used to reduce the noise strength.

Based on Eq. (2), we retouch the fake image $I$ via the guidance of an adversarial noise map and obtain

$$\hat{I} = K \oplus (I + A \odot N_\sigma),$$

where ‘$\odot$’ denotes the element-wise multiplication and $A$ is a binary adversarial noise map with value ‘1’ for adding noise $N_\sigma$ to the fake image. It is highly desirable that $A$ is sparse while pinpointing key positions where noises can do best to perturb the artifact patterns. To this end, we calculate $A$ from the viewpoint of adversarial attack and employ a DeepFake detector (i.e., $D(\cdot)$) as a subject model. Please note that $D(\cdot)$ is a simple detector which unrelated to the detectors we aim to attack. Then, we have the following objective function:

$$\arg \max_A J(D(I + A), y) + ||A||_1,$$

where $J(\cdot)$ denotes the cross-entropy loss function, $y$ is the ground-truth label of $I$. Here, we have $y = 1$ since $I$ is a fake image. The second term encourages $A$ to be sparse to add less noise to the fake image.

In summary, algorithm 1 shows our method DeepNotch. There are four key steps. First, in line 2, we select a fake image $I$ from the fake image list $\mathcal{I}$. Second, in line 3, we generate an adversarial guided map $A$ produced by attacking a simple pre-trained DeepFake detector $D(\cdot)$ according to Eq. (4). Third, in line 4, with a prepared noise map $N_\sigma$ which is of uniform or Gaussian noise, we element-wise multiply $N_\sigma$ with the adversarial guided map $A$ to be $A \odot N_\sigma$ and add the noise to the fake image $I$. At last, in line 5, the specified kernels $K$ generated by deep neural network $DNN(\cdot)$ are efficiently used to embellish the noised image.

**V. EXPERIMENTS**

In the experiments, we design two different validation methods to demonstrate the effectiveness of our method. First, we test whether the reconstructed images can reduce the detection accuracy of various fake image detection methods.
Algorithm 1 DeepNotch

**Input:** Fake images $I^r$, DeepFake detector $D(\cdot)$, Ground truth label $y$ of fake images, Cross-entropy loss function $J(\cdot)$, Noise map $N_r$, Pixel-wise kernels $K$.

**Output:** Reconstruction image $\hat{I}$.

1. for $i = 1$ to $|I^r|$ do
2.  Sample an image $I \sim \{I^r\}$;
3.  Generate adversarial noise map $A$ via
4.    $\arg\max_A J(D(I + A), y) + \|A\|_1$;
5.  Produce $\hat{I}$ by adding noise to image $I$ via $\hat{I} = I + A \otimes N_r$
6.  Apply image filtering to achieve reconstruction image $\hat{I}$ from $\hat{I}$ via $\hat{I} = K \odot \hat{I}$;

Then, we further perform quantitative measurement of our reconstructed changing magnitude by using similarity metrics.

**A. Experimental Setup**

1) **Fake Detection Methods:** Existing fake detection methods largely fall into 3 categories. For each category, we choose one representative fake detection method. For fingerprint-based, image-based and spectrum-based methods, we select GANFingerprint [3], CNNDetector [4], and DCTA [2], respectively. [3] is the latest work using fingerprint. CNNDetector [4] detects a large number of GANs, which is suitable for testing the effectiveness of our method on different GANs. For spectrum-based methods, DCTA is popular and outstanding.

2) **Datasets:** We tried our best to cover diverse datasets and select 3 popular real image datasets (CelebA [62], LSUN [63], FFHQ [19]) used in previous work. CelebA and FFHQ are the most famous human face dataset while LSUN includes images of different rooms. We comprehensively tried a total of 16 GAN-based methods for fake image generation on the above real image datasets. Specifically, for GANFingerprint and DCTA, ProGAN [18], SNGAN [21], CramerGAN [64] and MMDGAN [22] are used as the fake image generators. For each of these four GAN-based image generation methods, we set the size of the testing dataset to be 10,000. For CNNDetector, we select ProGAN, StyleGAN [19], BigGAN [65], CycleGAN [60], StarGAN [24], GauGAN [66], CRN [28], IMLE [67], SITD [61], SAN [29], DeepFakes [68], StyleGAN2 [20], and Whichfaceisreal [69], a total of 13 GAN-based image generation methods. The size of the testing dataset of these GAN-based image generation methods ranges from hundreds to thousands. The elements in the datasets of CycleGAN, ProGAN, StyleGAN and StyleGAN2 have two or more categories (e.g., in StyleGAN2, four different categories (horse, car, cat, church) are contained). The datasets of other GANs have only one category. For example, Whichfaceisreal only has fake images of the human face.

3) **Evaluation Settings:** In our experiment, we use KPN [70] as the deep image filtering method. The training dataset of KPN has 10,000 pairs of images, each pair of which includes a real image chosen from CelebA and a fake image. The fake image is first produced by reconstructing the real image counterpart with STGAN [25], and then added with different types of noise. The two noises chosen by us are Gaussian noise and uniform noise. The mean of Gaussian noise is 0 while the standard deviation is 10. The lower bound and upper bound of uniform noise are $-20$ and $20$, respectively. To add deliberate noise, we first train a simple DeepFake detector with ResNet50. Then, we use PGD [71] to adversarially attack this detector, on the fake images as the inputs to obtain adversarial guided maps. The epsilon (maximum perturbation for each pixel) is 0.04. At last, depending on the adversarial guided map, we choose locations of the fake images to add noise. Please note that our method is a post-processing black-box image reconstruction method that doesn’t require any model information of the detectors. The DeepFake detector (based on Resnet50) used here is unrelated to the models of GANFingerprint, CNNDetector, and DCTA.

4) **Metrics:** Detection accuracy is one of our main evaluation metrics. We compare the detection accuracy of fake images and reconstructed images for each method. In addition, to be comprehensive, we further use cosine similarity (COSS), peak signal-to-noise ratio (PSNR), and structural similarity (SSIM) to measure the similarity between a fake image and its reconstructed image counterpart. COSS is a common similarity metric that measures the cosine of the angle. We transform the RGB images to vectors before calculating COSS. PSNR is a widely used measurement for the reconstruction quality of lossy compression. SSIM is one of the most popular and useful metrics for measuring the similarity between two images. A large value of COSS, PSNR and SSIM, indicates a better result. The value ranges of COSS and SSIM are both in $[0, 1]$.

**B. Experiment I: Evading GANFingerprint**

GANFingerprint can judge which GAN-based image generation method is used to produce the fake image. For a double-check and validation of our settings, we reproduce their experiments on fake image detection. For each GAN-based fake image generation method (i.e., ProGAN, SNGAN, CramerGAN, MMDGAN), we produce 10,000 fake images from 10,000 randomly chosen CelebA real images. Furthermore, we replace their fake images with our reconstructed images to test the detection accuracy of their method.

1) **Compare With the Baseline:** Table I shows the detection accuracy of different GANs. We use ProGAN (Pro) in the second column as an example for elaboration. In particular, Pro has five sub-items (columns): CelebA, ProGAN (Pro), SNGAN (SN), CramerGAN (Cramer), MMDGAN (MMD). These sub-items represent the possibilities of the ProGAN images to be classified as one of them. In the first column are the types of the source of input images.

The baseline is the state-of-the-art method [72]. They use the shallow reconstruction method to polish fake images. In the seven rows above the gray line, Fake represents the fake images. BL-PCA and BL-KSVD represent the PCA-based and KSVD-based reconstruction of the baseline, respectively. D(rn)-gau, D(rn)-uni, D(an)-gau, D(an)-uni mean the reconstructed images generated from our method. In particular, ‘rn’ and ‘an’ mean adding random noise and adversarial noise.
TABLE I
DETECTION ACCURACY BEFORE AND AFTER RECONSTRUCTION OF GAN-SYNTHESIZED IMAGES IN GANFINGERPRINT. THE ACCURACY IN THE COLUMNS WITH YELLOW BACKGROUND COLOR IS WHAT WE NEED TO DECREASE. BECAUSE WE USE OUR METHOD TO RECONSTRUCT THE GAN IMAGES OF THAT COLUMN HEAD

| Accuracy (%) | CelebA | Pro | ProGAN (Pro) | SN | Cramer | MMD | CelebA | Pro | SNGAN (SN) | SN | Cramer | MMD |
|--------------|--------|-----|-------------|----|--------|-----|--------|-----|----------|----|--------|-----|
| Fake         | 0.03   | 0.01| 0.03        | 0.01| 0.02   | 0.03| 0.08   | 0.01| 0.05     | 0.05| 0.02   | 0.04|
| BL-PCA       | 88.90  | 68.87| 6.99        | -92.92| 0.01| 0.02| 0.03   | 0.01| 0.02     | 0.03| 0.02   | 0.04|
| BL-KSVD      | 72.50  | 21.47| 78.10       | 21.81| 0.01| 0.09| 0.20   | 0.17| 0.10     | 0.08| 0.08   | 0.09|
| StatAttack   | 79.33  | 92.90| 4.79        | -95.12| 0.16| 0.13| 0.21   | 0.18| 1.49     | -1.47| 0.03   | 0.04|
| DN(m)-gau    | 93.39  | (93.36)| 4.77       | -95.14| 0.01| 0.14| 0.21   | 0.18| 1.49     | -1.47| 0.03   | 0.04|
| DN(m)-uni    | 91.79  | (91.76)| 6.57       | -93.34| 0.11| 0.10| 0.20   | 0.17| 0.22     | 0.20| 0.02   | 0.03|
| DN(nan)-gau  | 96.00  | (95.97)| 2.37       | -97.54| 0.24| 0.23| 0.19   | 0.16| 1.20     | -1.18| 0.02   | 0.03|
| DN(nan)-uni  | 95.76  | (95.73)| 2.89       | -97.02| 0.16| 0.15| 0.12   | 0.09| 1.07     | -1.05| 0.02   | 0.03|
| Fake-gau-5   | 92.75  | (92.72)| 4.50       | -95.41| 0.17| 0.16| 0.12   | 0.09| 2.36     | 2.36| 0.02   | 0.04|
| Fake-gau-10  | 99.47  | (99.44)| 0.02       | -01.97| 0.05| 0.04| 0.06   | 0.05| 0.24     | 0.24| 0.02   | 0.04|
| Fake-uni-5   | 55.10  | (55.07)| 38.42      | -63.49| 0.17| 0.16| 0.32   | 0.49| 7.79     | -7.77| 0.04   | 0.05|
| Fake-uni-10  | 95.96  | (95.93)| 2.20       | -97.71| 0.03| 0.05| 0.08   | 0.05| 1.70     | -1.68| 0.04   | 0.06|
| Fake-uni-15  | 99.24  | (99.21)| 0.33       | -99.58| 0.06| 0.05| 0.05   | 0.02| 0.32     | 0.32| 0.05   | 0.07|
| Fake-uni-20  | 99.66  | (99.63)| 0.07       | -99.84| 0.04| 0.03| 0.04   | 0.01| 0.19     | 0.19| 0.03   | 0.05|
| Filt(nn)     | (0.03)| (0.03)| 0.91       | (0.01)| 0.03| 0.02| 0.02   | 0.02| 0.00     | 0.00| 0.00   | 0.00|

respectively. ‘uni’ and ‘gau’ mean adding uniform noise and Gaussian noise respectively.

In the Fake row, the input images are 10,000 ProGAN fake images. We can find that 99.91% of the fake images have been considered as being produced by Pro. The percentages of the ProGAN fake images that are misclassified as CelebA, SN, Cramer and MMD are 0.03%, 0.01%, 0.03% and 0.02%.

In the table, we highlight the difference in detection accuracy between reconstructed images and fake images (e.g., by color and number). As shown in the DN(an)-uni row, the inputs are the reconstructed images by adding and denoising adversarial-noise-guided uniform noise on the counterpart fake images. Most of the 10,000 reconstructed images are considered as CelebA type. The accuracy raises from 0.03% to 95.76%. We use blue color and (+95.73) to highlight the difference. Similarly, the ratio of images classified into Pro decreases from 99.91% to 2.89%. We use red color and (−97.02) to show the difference. Most of the fake images generated by ProGAN are misclassified to be real images after using our method. Compared with BL-PCA and BL-KSVD, DN(m)-gau and DN(m)-uni degrade the detector more. Our method also surpasses the BL significantly in the other three parts (i.e., SN, Cramer, MMD).

2) Ablation Study: To verify the effectiveness of key technology: adding noise, deep image filtering and adversarial-guided map.

- Adding noise. We also take ProGAN as an example. The conclusion is the same on the other three GANs. The experimental results are below the gray line. In the first column, Fake-gau-10 and Fake-uni-20 represent only adding Gaussian noise (gau) or uniform noise (uni) to the fake images. The accuracy of Fake-gau-10 and Fake-uni-20 images being classified as CelebA type raises from 0.03% to 99.47% and 99.66% respectively. This phenomenon shows the effectiveness of noise in reducing artifact patterns. On the other hand, if we only use deep image filtering without adding noise to the fake image, the result is shown in the row of Filt(nn), which means the reconstructed images with no noise (nn) added. We can find that the classification accuracy is basically unchanged. Furthermore, we evaluate the effect of different levels of noise on misleading the detector. The mean of Gaussian noise is 0 while the standard deviation is 5 and 10 (Fake-gau-5 and Fake-gau-10). The lower bounds and upper bounds of uniform noise are ±5 (Fake-uni-5), ±10 (Fake-uni-10), ±15 (Fake-uni-15), ±20 (Fake-uni-20), respectively. We can find that, with the increase in noise level, the noised fake images are more likely to evade GANFingerprint. At the same time, as shown in Fig. 7, the more noise added, the fewer artifacts in the spectrum. This evidence supports that reducing artifacts can help to evade the DeepFake detector. We also conduct similar experiments in Table V and achieve the same conclusion.

- Deep image filtering. We add guided filter [73] and BM3D [74] to compare the effect of other classical denoising methods with deep image filtering. Among the four GANs, to evade GANFingerprint, the accuracy of fake images being classified as real images with the guided filter or BM3D is 19.86% and 1.35% lower than the accuracy of the original fake images. At the same time, as shown in Fig. 7, the more noise added, the fewer artifacts in the spectrum. This evidence supports that reducing artifacts can help to evade the DeepFake detector. We also conduct similar experiments in Table V and achieve the same conclusion.
using the KPN filter with kernel size three. From the experiment, we can find that our method surpasses the guided filter by far and the performance of BM3D is similar to deep filtering. However, the time of generating 40,000 images upon experiment takes BM3D 24 hours while our method only needs half an hour.

- **Adversarial guided map.** Here we show the effect of the adversarial guided map by comparing DN(rn)-uni and DN(an)-uni. Since the guided noise only uses partial regions of the fake image to add noise, we increased the intensity of the noise to match the total intensity of DN(rn)-uni. In Table I, DN(an)-uni uses 80% of the area to add uniform noise, thus the lower and upper bounds of uniform noise are 1.25 times (-25/25) of that in DN(rn)-uni (-20/20). We highlight the best performance of the reconstruction method of all four GANs in bold font.

We also retrain the GANFingerprint model with reconstructed images to test whether the retrained model can detect reconstructed images. As shown in Table III, the model is retrained with reconstructed images generated from DN(rn)-uni. The model can successfully classify reconstructed images as fake, which means the images reconstructed by our method can help to improve the DeepFake detector to a new version.

C. Experiment II: Evading DCTA

DCTA has the same testing dataset as GANFingerprint. We follow the exact evaluation setting in its original paper, where the spectrums of the images are used as inputs. For each category of CelebA, ProGAN, SNGAN, CramerGAN and MMDGAN, we use 9,600 fake images as the input. The testing dataset has a total of 48,000 images. As shown in Table IV, DCTA successfully detects fake images with high accuracy. However, on reconstructed images, it decreases significantly. DN(rn)-gau, DN(rn)-uni, DN(an)-gau, DN(an)-uni all successfully drop the classification accuracy of DCTA and do better than baselines. DCTA only obtains 22.59% accuracy on DN(rn)-gau reconstructed images, at a dramatic drop of 66.4%. For DN(rn)-uni reconstruction, it shows slightly better (21.73%) than DN(rn)-gau reconstruction. Furthermore, DN(an)-uni and DN(an)-gau, the methods which exploit adversarial-noise-guided image filtering, achieve better performance than their corresponding random-noise-guided counterpart respectively. DN(an)-uni does the best, which is slightly better than DN(an)-gau. The results demonstrate the effectiveness of reconstructed images in misleading the DCTA.
TABLE IV
DETECTION ACCURACY BEFORE AND AFTER RECONSTRUCTION OF GAN-SYNTHESIZED IMAGES IN DCTA

| Accuracy(%) | ProGAN | Fake | SNGAN | CelebA | Fake |
|-------------|--------|------|-------|--------|------|
| Fake        | 0.43   | 99.57 | 0.22  | 99.78  |
| BL-PCA      | 74.50  | (74.07) | 25.50  | (74.07) | 56.85  | (56.63) | 43.15  | (56.63) |
| BL-KSVD     | 23.40  | (23.13) | 76.60  | (23.13) | 71.60  | (70.92) | 28.40  | (70.92) |
| StatAttack  | 76.18  | (75.75) | 23.82  | (75.75) | 76.06  | (75.85) | 23.94  | (75.84) |
| DN(rn)-gau  | 82.07  | (81.64) | 17.93  | (61.84) | 56.30  | (56.06) | 45.70  | (56.08) |
| DN(rn)-uni  | 83.84  | (83.41) | 16.16  | (83.41) | 70.11  | (69.89) | 29.89  | (69.89) |
| DN(an)-gau  | 87.59  | (87.16) | 12.41  | (87.16) | 60.43  | (60.21) | 39.57  | (60.21) |
| DN(an)-uni  | 88.55  | (88.12) | 11.45  | (88.12) | 73.62  | (73.40) | 26.38  | (73.40) |

D. Experiment III: Evading CNNDetector

We further perform a large-scale evaluation on CNNDetector with a total of 13 GAN-based image generation methods. The testing dataset of CNNDetector contains diverse types of images (e.g., animals, human faces, road). For each GAN category, the size of the testing dataset ranges from hundreds to thousands.

As shown in the first column of Table V, the two models used by CNNDetector are prob0.1 and prob0.5. We can find that both prob0.1 and prob0.5 have achieved high accuracy on a large proportion of GANs. Here we take the case prob0.1 as an example to introduce the table. In the second row, Fake is the testing dataset that contains fake images. The experimental result on fake images is the same as that in CNNDetector. DN(rn)-gau, DN(rn)-uni, DN(an)-gau, DN(an)-uni in the 5-8 rows are the results of our method. The accuracy data with red decrements are where our method succeeds. Compared with the detection accuracy of fake images, DN(rn)-gau and DN(rn)-uni reconstructed images both drop the accuracy of CNNDetector. Furthermore, DN(an)-gau and DN(an)-uni drop the accuracy of CNNDetector model more and have comparable performance with baselines.

E. Discussion

1) Similarity b/w Fake and Reconstructed Image: Table VI summarizes the similarity between fake images and reconstructed images of GANs in GANFingerprint and DCTA. For each of DN(rn)-gau, DN(rn)-uni and DN(an)-uni in the 5-8 rows are the results of our method. The accuracy data with red decrements are where our method succeeds. Compared with the detection accuracy of fake images, DN(rn)-gau and DN(rn)-uni reconstructed images both drop the accuracy of CNNDetector. Furthermore, DN(an)-gau and DN(an)-uni drop the accuracy of CNNDetector model more and have comparable performance with baselines.

2) Discussion on Deliberate Noise: Fig. 8 shows the performance between deliberate noise and random noise of the same noise setting (i.e., uniform noise with 20 upper bound and -20 lower bound). The horizontal axis of all four subfigures represents the percentage of the region ratio of the image, on which the noise is added. The vertical axis is the possibility of reconstructed images being classified as CelebA types by method GANFingerprint. The blue and orange lines represent the performance of deliberate noise and random noise, respectively. We can observe that in all the four GANs and among the different percentages of regions, the accuracy of deliberate noise is stably better than that of random noise. This indicates that our localization method is stable and effective.

3) Discussion on Adversarial Guided Map: As shown in the left subfigure of Fig. 9, the fake image in the first column is generated by ProGAN [18] with CelebA [62]. The images on the right are the adversarial guided maps. The labels R, G, B in the first row represent the red, green, and blue channels of the adversarial guided map. The adversarial guided maps under the labels are corresponding to one of the three channels of the fake image on the left (e.g., the red-channel adversarial guided map is used to confirm where to add noise to the red channel of the fake image). In each adversarial guided map, the pixel in it is either 0 or 1. The yellow pixel means 1 while the purple pixel means 0. The area of yellow pixels is where we will add noise to. The numbers in the last column mean other eight GANs have multiple categories. The images of CycleGAN, StyleGAN, StyleGAN2, and ProGAN are sorted into categories, they use different folders to store different categories of images. The other GANs (BigGAN, GauGAN, SAN, SITD) combine images of different categories into one folder. We use '-' to represent this category. We can see that the reconstructed images in CNNDetector are also very similar to their fake image counterparts. Fig. 13 shows the reconstructed image of DeepNotch on a cat and a human face, with very high reconstruction quality.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
TABLE V

| Detection Accuracy Before and After Reconstruction of GAN-Synthesized Images in CNNDetector |
|---------------------------------------------------------------|
| Accuracy (%) | ProGAN | StyleGAN | BigGAN | CycGAN | StarGAN | GanGAN | CBR | IMNET | SEDT | DeepPatches | StyleGAN | Wholesampled |
|---------------|--------|----------|--------|--------|--------|--------|------|-------|------|-------------|----------|--------------|
| Face          | 99.9   | 98.4     | 87.5   | 68.5   | 68.5   | 68.5   | 68.5 | 68.5   | 68.5 | 68.5        | 88.4     | 86.4         |
| BL-PCA        | 62.9   | (57.4)   | 3.80   | 70.3   | 12.3   | 34.5   | 35.8  | 43.0   | 36.0  | 35.9       | 40.0     | (30.6)       |
| BL-RVQD       | 94.9   | (56.0)   | 35.7   | (4.5)  | 30.0   | 18.8   | 49.7  | (10.1) | 48.9  | 38.7       | 51.0     | (13.8)       |
| StarAttack    | 0.00%  | 0.00%    | 0.00%  | 0.00%  | 0.00%  | 0.00%  | 0.00% | 0.00%  | 0.00% | 0.00%      | 0.00%    | 0.00%        |
| DIN(gau)      | 96.8   | (3.2)    | 28.6   | (4.6)  | 25.0   | 31.8   | 41.8  | (3.7)  | 43.2  | (4.5)     | 36.3     | (27.9)       |
| DIN-uni       | 95.3   | (4.0)    | 18.9   | (3.9)  | 21.7   | 23.3   | 40.3  | (7.8)  | 46.4  | (4.4)     | 36.3     | (13.9)       |
| DIN(uav)      | 66.8   | (33.1)   | 12.5   | (41.7) | 5.01   | (41.7) | 18.4  | (40.0) | 14.3  | (73.4)    | 53.4     | (39.2)       |
| DIN(uav)      | 68.6   | (33.1)   | 12.5   | (41.7) | 5.01   | (41.7) | 18.4  | (40.0) | 14.3  | (73.4)    | 53.4     | (39.2)       |
| Fake-5        | 91.5   | (8.40)   | 19.7   | (54.5) | 11.1   | (53.5) | 50.0  | (48.0) | 28.2  | (55.9)    | 8.66     | (44.6)       |
| Fake-10       | 12.1   | (67.50)  | 11.3   | (72.5) | 22.5   | (45.5) | 34.9  | (75.0) | 15.8  | (61.3)    | 6.86     | (61.3)       |
| Fake-50       | 99.9   | (98.90)  | 57.4   | (18.4) | 24.0   | (18.0) | 64.4  | (16.0) | 77.4  | (20.0)    | 51.5     | (15.9)       |
| Fake-50       | 99.9   | (98.90)  | 57.4   | (18.4) | 24.0   | (18.0) | 64.4  | (16.0) | 77.4  | (20.0)    | 51.5     | (15.9)       |
| Fake-100      | 84.5   | (58.00)  | 12.8   | (41.7) | 7.00   | (45.9) | 20.0  | (45.9) | 6.84  | (45.8)    | 6.84     | (45.8)       |
| Fake-15       | 44.6   | (53.50)  | 2.40   | (71.8) | 2.45   | (45.9) | 6.73  | (72.7) | 1.65  | (45.5)    | 0.08     | (99.9)       |
| Fake-20       | 20.9   | (79.10)  | 0.57   | (73.7) | 1.40   | (45.0) | 4.23  | (76.8) | 0.15  | (45.8)    | 0.00     | (99.9)       |

Table VI

| Similarity Between Fake Image & Generated Image of GANs in GANFingerprint & DCTA |
|---------------------------------------------------------------|
| ProGAN | SNGAN | CramerGAN | MMGAN |
|-------|-------|----------|-------|
| DIN(r)-gau | COSS | 0.998 | 0.998 | 0.998 |
|        | PSNR  | 30.26  | 30.21  | 30.18  |
|        | SSIM  | 0.933  | 0.932  | 0.933  |
| DIN(r)-uni | COSS | 0.998 | 0.998 | 0.998 |
|        | PSNR  | 29.94  | 29.86  | 29.85  |
|        | SSIM  | 0.928  | 0.924  | 0.927  |
| DIN(a)-gau | COSS | 0.998 | 0.998 | 0.998 |
|        | PSNR  | 30.00  | 29.97  | 29.93  |
|        | SSIM  | 0.925  | 0.924  | 0.924  |
| DIN(a)-uni | COSS | 29.73  | 29.68  | 29.61  |
|        | PSNR  | 29.95  | 29.87  | 29.87  |
|        | SSIM  | 0.924  | 0.924  | 0.924  |

4) Discussion on Failure Samples:

We investigate the images that can still detected as DeepFakes despite undergoing the DeepNotch and call them “failure samples”. In contrast, the adversarial guided maps are semantic-aware.
images that are detected as Real undergoing the DeepNotch are called “success samples”. We find that the spectrum of their fake images show obvious artifacts (i.e., blobs), which is easy for us to observe. The SNGAN is detected by GANFingerprint and the StarGAN is detected by CNNDetector. Let’s take the first row as an example. The fake SNGAN images exhibit obvious artifacts in the spectrum. It’s important to note that the spectrum is computed by averaging the spectra of fake SNGAN images. When we apply DeepNotch and uniform noise to filter the SNGAN images, we gather failure samples and success samples. Subsequently, we calculate the average spectrum for both sets, denoting them as “Uniform fail” and “Uniform success” in Figure 11. Remarkably, we find that there are hardly any artifacts in either the failure or success samples. Similarly, when we filter the SNGAN images using DeepNotch and Gaussian noise, we observe almost no artifacts in both sets of samples. We can also find that in the second row (StarGAN), there are almost no artifacts in the spectrums of fake images and success samples. This finding implies that, in addition to the artifacts discussed in our paper, there may exist other indicators capable of distinguishing fake from real images. However, since DeepNotch effectively reduces artifacts and allows evasion of existing DeepFake detectors like GANFingerprint, DCTA, and CNNDetector, we believe that these alternative indicators have limited significance in current DeepFake detectors. It would be intriguing to identify these additional indicators, and we intend to investigate them in our future research.

5) Comparison With Adversarial Attack Methods: We further investigate the SOTA detector evasion methods and find that currently the most popular methods are based on adversarial attacks. Although the adversarial attack is another distinct method compared with image reconstruction methods, in order to show the comparison comprehensively, we add extra experiments. To our best knowledge, the SOTA published methods are TR-Net [75] and StatAttack [76]. TR-Net does not open-source the code. Compared with that, StatAttack open-source codes on GitHub. Thus we conduct experiments of StatAttack with the same attack setting in [76] and the same experiment setting in our paper on three different DeepFake detectors. Since DeepNotch is a target-detector-independent evasion method, we apply the StatAttack method in the black-box setting. As shown in Table I, when evading GANFingerprint, we can find that DeepNotch shows better performance than StatAttack. As shown in Table IV, when evading DCTA, we can find that StatAttack is a bit better than DeepNotch. As shown in Table V, when evading CNNDetector, we can find that StatAttack is better than DeepNotch on most of the DeepFakes. Furthermore, we compare the image quality of images generated by DeepNotch and StatAttack. As shown in Table VI and Table VII, we can find that the images generated by DeepNotch are more similar to the corresponding original images than those generated by StatAttack. As shown in Fig. 12, we take the DeepFake dataset of GANFingerprint and DCTA as an example. We can find that the images generated by StatAttack have obvious corruption, which makes the images seem uncommon. To sum up, compared with SOTA DeepFake evasion methods (i.e., FakePolisher) that are based on image reconstruction, DeepNotch shows better evasion performance. Compared with the SOTA DeepFake evasion method (i.e., StatAttack) of different types we can find that DeepNotch shows better image quality while a bit poor evasion performance.

6) Limitation: As shown in Table III, people could retrain the DeepFake detection model to defend our method, which is the limitation of our method. However, this means that our method can help to improve the DeepFake detector to a new version. Furthermore, improved detectors will encourage people to retrain their detection models.
method effectively reduces the artifact patterns introduced for improving the fidelity of GAN-based fake images. Our field.

to classify real and non-real images. This shows the function i.e

effectiveness. Hence, the detectors to find a better decision line (i.e., hard examples), the detector to be above the red line, which means the reconstructed images have room for improvement. The target of our method is to be actually real. Similarly, the images below the red line will be considered fake, although they may not be actually fake. From this point of view, we can find that the detectors still have room for improvement. The target of our method is to expose this defect with hard examples and help to improve the detectors with these hard examples. As shown in the left of Fig. 10, their classification ability has an intersection with part of the real distribution and fake distribution. If we use a red line to divide the ability of the detector into two parts, then we can find that the images above the red line will be considered as real, although they may not be actually real. Similarly, the images below the red line will be considered fake, although they may not be actually fake. From this point of view, we can find that the detectors still have room for improvement. The target of our method is to expose this defect with hard examples and help to improve the detectors with these hard examples. As shown in the left of Fig. 10, our method can shift the fake images (i.e., blue points) to be above the red line, which means the reconstructed images are considered as real by detectors. As shown in Table III, with our reconstructed images (i.e., hard examples), the detector can be retrained to successfully classify reconstructed images and real images. This means our reconstructed images help the detectors to find a better decision line (i.e., green line) to classify real and non-real images. This shows the function of our method to promote the improvement of the DeepFake field.

VI. CONCLUSION

In this paper, we propose the DeepNotch, a pipeline that performs implicit spatial-domain notch filtering by taking a hybrid approach of deep image filtering and noise addition for improving the fidelity of GAN-based fake images. Our method effectively reduces the artifact patterns introduced by existing fake image generation methods, in both spatial and frequency domains. By reducing fake artifacts, our further reconstructed image retains photo-realistic and high-fidelity, which can bypass state-of-the-art DeepFake detection methods. Our large-scale evaluation demonstrates that more general DeepFake detectors beyond leveraging fake artifacts should be further investigated. In future work, we aim to learn from other video-related work [77], [78], [79], [80], [81], [82] for constructing DeepFake evasion with more temporal information.

ACKNOWLEDGMENT

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore, and Cyber Security Agency of Singapore.

REFERENCES

[1] X. Zhang, S. Karaman, and S.-F. Chang, “Detecting and simulating artifacts in GAN fake images,” in Proc. IEEE Int. Workshop Inf. Forensics Secur. (WIFS), Dec. 2019, pp. 1–6.
[2] J. Frank, T. Eisenhofer, L. Schönherr, A. Fischer, D. Kolossa, and T. Holz, “Leveraging frequency analysis for deep fake image recognition,” in Proc. Int. Conf. Mach. Learn., 2020, pp. 3247–3258.
[3] N. Yu, L. Davis, and M. Fritz, “Attributing fake images to GANs: Learning and analyzing GAN fingerprints,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 7555–7565.
[4] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, “CNN-generated images are surprisingly easy to spot... for now,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 8692–8701.
[5] Y. Ni, D. Meng, C. Yu, C. Quan, D. Ren, and Y. Zhao, “CORE: Consistent representation learning for face forgery detection,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), Jun. 2022, pp. 12–21.
[6] C. Dong, A. Kumar, and E. Liu, “Think twice before detecting GAN-generated fake images from their spectral domain imprints,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2022, pp. 7855–7864.
[7] Y. Jeong, D. Kim, Y. Ro, and J. Choi, “FrePGAN: Robust deepfake detection using frequency-level perturbations,” in Proc. AAAI Conf. Artif. Intell., 2022, vol. 36, no. 1, pp. 1060–1068.
[8] J. Li, H. Xie, J. Li, Z. Wang, and Y. Zhang, “Frequency-aware discriminative feature learning supervised by single-center loss for face forgery detection,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 6454–6463.
[9] H. Liu et al., “Spatial-phase shallow learning: Rethinking face forgery detection in frequency domain,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 772–781.
[10] H. Zhao, T. Wei, W. Zhou, W. Zhang, D. Chen, and N. Yu, “Multi-attentional deepfake detection,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 2185–2194.
[11] T. Dzanic, K. Shah, and F. Witherden, “Fourier spectrum discrepancies in deep network generated images,” in Proc. Adv. Neural Inf. Process. Syst., vol. 33, 2020, pp. 3022–3032.
[12] Y. Qian, G. Yin, L. Sheng, Z. Chen, and J. Shao, “Thinking in frequency: Face forgery detection by mining frequency-aware clues,” in Proc. Eur. Conf. Comput. Vis. Springer, 2020, pp. 86–103.
[13] I. Goodfellow et al., “Generative adversarial nets,” in Proc. Adv. Neural Inf. Process. Syst., 2014, pp. 2672–2680.
[14] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” 2015, arXiv:1511.06434.
[15] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein GAN,” 2017, arXiv:1701.07875.
[16] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of Wasserstein GANs,” in Proc. Adv. Neural Inf. Process. Syst., vol. 30, 2017, pp. 5767–5777.
[17] G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez, “Invertible conditional GANs for image editing,” 2016, arXiv:1611.06355.
[18] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of GANs for improved quality, stability, and variation,” 2017, arXiv:1710.10196.

[19] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 4396–4405.

[20] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and improving the image quality of StyleGAN,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 8107–8116.

[21] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, “Spectral normalization for generative adversarial networks,” 2018, arXiv:1802.05957.

[22] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos, “MMD GAN: Towards deeper understanding of moment matching network,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 2203–2213.

[23] Z. He, W. Zuo, M. Kan, S. Shan, and X. Chen, “AttGAN: Facial attribute editing by only changing what you want,” IEEE Trans. Image Process., vol. 28, no. 11, pp. 5464–5478, Nov. 2019.

[24] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, “StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 8789–8797.

[25] M. Liu et al., “STGAN: A unified selective transfer network for arbitrary image attribute editing,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 3668–3677.

[26] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, “StarGAN v2: Diverse image synthesis for multiple domains,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 8185–8194.

[27] Y. Gao et al., “High-fidelity and arbitrary face editing,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 16110–16119.

[28] B. Shi, X. Bai, and C. Yao, “An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 11, pp. 2298–2304, Nov. 2017.

[29] T. Dai, J. Cai, Y. Zhang, S.-T. Xia, and L. Zhang, “Second-order attention network for single image super-resolution,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 11057–11066.

[30] R. Tolosana, R. Vera-Rodriguez, J. Fierrez, A. Morales, and J. Ortega-García, “DeepFakes and beyond: A survey of face manipulation and fake detection,” 2020, arXiv:2001.00179.

[31] F. Juefei-Xu, R. Wang, Y. Huang, Q. Guo, L. Ma, and Y. Liu, “Countering malicious deepfakes: Survey, battleground, and horizon,” 2021, arXiv:2103.00218.

[32] R. Wang et al., “FakeSpotter: A simple yet robust baseline for spotting AI-synthesized fake faces,” in Proc. 29th Int. Joint Conf. Artif. Intell., Jul. 2020, pp. 3431–3435.

[33] H. Qi et al., “DeepRhythm: Exposing deepfakes with attentional visual heartbeat rhythms,” in Proc. ACM Int. Conf. Multimedia (ACM MM), 2020, pp. 4318–4327.

[34] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, “MesoNet: A compact facial video forgery detection network,” in Proc. IEEE Int. Workshop Inf. Forensics Secur. (WIFS), Dec. 2018, pp. 1–7.

[35] H. H. Nguyen, J. Yamagishi, and I. Echizen, “Capsule-forensics: Using capsule networks to detect forged images and videos,” in Proc. Int. Conf. Inf. Acoust., Speech Signal Process. (ICASSP), May 2019, pp. 2307–2311.

[36] H. H. Nguyen, F. Fang, J. Yamagishi, and I. Echizen, “Multi-task learning for detecting and segmenting manipulated facial images and videos,” in Proc. IEEE 10th Int. Conf. Biometrics Theory, Appl. Syst. (BTAS), Sep. 2019, pp. 1–8.

[37] R. Wang et al., “DeepSonar: Towards effective and robust detection of AI-synthesized fake voices,” in Proc. ACM Int. Conf. Multimedia (ACM MM), 2020, pp. 1207–1216.

[38] S. Fernandes et al., “Predicting heart rate variations of deepfake videos using neural ODE,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW), Oct. 2019, pp. 1721–1729.

[39] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, “Neural ordinary differential equations,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 6571–6583.

[40] H. Qi et al., “DeepRhythm: Exposing deepfakes with attentional visual heartbeat rhythms,” in Proc. 28th ACM Int. Conf. Multimedia, Oct. 2020, pp. 4318–4327.
[65] A. Brock, J. Donahue, and K. Simonyan, “Large scale GAN training for high fidelity natural image synthesis,” 2018, arXiv:1809.11096.

[66] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, “GauGAN: Semantic image synthesis with spatially adaptive normalization,” in Proc. ACM SIGGRAPH Real-Time Live, 2019, p. 1.

[67] K. Li and J. Malik, “Implicit maximum likelihood estimation,” 2018, arXiv:1809.09080.

[68] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Niessner, “FaceForensics++: Learning to detect manipulated facial images,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 1–11.

[69] B. Mildenhall, J. T. Barron, J. Chen, D. Sharlet, R. Ng, and R. Carroll, “Burst denoising with kernel prediction networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 2502–2510.

[70] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” 2017, arXiv:1706.06083.

[71] Y. Huang et al., “FakePolisher: Making deepfakes more detection-evasive by shallow reconstruction,” in Proc. 28th ACM Int. Conf. Multimedia, Oct. 2020, pp. 1217–1226.

[72] K. He, J. Sun, and X. Tang, “Guided image filtering,” in Proc. Eur. Conf. Comput. Vis. Crete, Greece: Springer, 2010, pp. 1–14.

[73] K. Dubov, A. Foi, V. Katkovnik, and K. Egiazarian, “Image denoising with block-matching and 3D filtering,” Proc. SPIE, vol. 6064, pp. 60641-1–60641-12, Feb. 2006.

[74] C. Liu, H. Chen, T. Zhu, J. Zhang, and W. Zhou, “Making deepfakes more spurious: Evading deep face forgery detection via trace removal attack,” IEEE Trans. Dependable Secure Comput., early access, Feb. 2, 2023, doi: 10.1109/TDSC.2023.3241604.

[75] Y. Hou, Q. Guo, Y. Huang, X. Xie, L. Ma, and J. Zhao, “Evading deepfake detectors via adversarial statistical consistency,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2023, pp. 12271–12280.

[76] C. Chen, M. Ye, M. Qi, J. Wu, Y. Liu, and J. Jiang, “Saliency and granularity: Discovering temporal coherence for video-based person re-identification,” IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 9, pp. 6100–6112, Sep. 2022.

[77] M. Liu, S. Jin, C. Yao, C. Lin, and Y. Zhao, “Temporal consistency learning of inter-frames for video super-resolution,” IEEE Trans. Circuits Syst. Video Technol., vol. 33, no. 4, pp. 1507–1520, Apr. 2023.

[78] X. Shu et al., “Large-scale spatio-temporal person re-identification: Algorithms and benchmark,” IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 7, pp. 4390–4403, Jul. 2022.

[79] J. Deng, L. Wang, S. Pu, and C. Zhuo, “Spatio-temporal deformable convolution for compressed video quality enhancement,” in Proc. AAAI Conf. Artif. Intell., 2020, vol. 34, no. 7, pp. 10696–10703.

[80] B. Chen, L. Zhu, G. Li, F. Lu, H. Fan, and S. Wang, “Learning generalized spatial–temporal deep feature representation for no-reference video quality assessment,” IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 4, pp. 1903–1916, Apr. 2022.

[81] M. Afonso, F. Zhang, and D. R. Bull, “Video compression based on spatio-temporal resolution adaptation,” IEEE Trans. Circuits Syst. Video Technol., vol. 29, no. 1, pp. 275–280, Jan. 2019.

Felix Juefei-Xu (Member, IEEE) received the B.S. degree in electronic engineering from Shanghai Jiao Tong University (SJTU), Shanghai, China, and the M.S. degree in electrical and computer engineering and in machine learning and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University (CMU), Pittsburgh, PA, USA. Currently, he is a Research Scientist with GenAI at Meta, New York City, where he works on robust perception and efficient learning problems in the domain of generative AI. He is also affiliated with New York University as an Adjunct Professor. Previously, he was a Research Scientist with the Alibaba Group, Sunnyvale, CA, USA. He was a recipient of multiple best/distinguished paper awards, including IJCB 2011, BTAS 2015 and 2016, ASE 2018, and ACCV 2018.

Qing Guo (Member, IEEE) received the Ph.D. degree in computer application technology from the School of Computer Science and Technology, Tianjin University, China. He is currently a Senior Research Scientist and Principal Investigator (PI) with the Center for Frontier AI Research (CFAR), A*STAR, Singapore. He is also an Adjunct Assistant Professor with the National University of Singapore (NUS), and a Senior PC Member of AAAI. Before that, he was a Wallenberg-NTU Presidential Post-doctoral Fellow with the Nanyang Technological University, Singapore. His research interests include computer vision, AI security, and image processing.

Yang Liu (Senior Member, IEEE) received the Bachelor of Computing degree (Hons.) from the National University of Singapore (NUS) in 2005 and the joint Ph.D. degree from NUS and MIT in 2010. He started his post-doctoral work in NUS and MIT. In 2012, he joined Nanyang Technological University (NTU), where he is currently a Full Professor and the Director of the Cybersecurity Laboratory. He specializes in software engineering, cybersecurity, and artificial intelligence. His research has bridged the gap between the theory and practical usage of program analysis, data analysis, and AI to evaluate the design and implementation of software for high assurance and security. By now, he has more than 400 publications in top tier conferences and journals. He has received a number of prestigious awards including MSRA Fellowship, TRF Fellowship, Nanyang Assistant Professor, Tan Chin Tuan Fellowship, Nanyang Research Award 2019, ACM Distinguished Speaker, NRF Investigatorship, and 15 best paper awards and one most influence system award in top software engineering conferences like ASE, FSE, and ICSE.

Yihao Huang received the B.S. and Ph.D. degrees from the Software Engineering Institute, East China Normal University, China, in 2017 and 2022, respectively. He is currently a Research Fellow with Nanyang Technological University. His research interests include computer vision and AI security.

Geguang Pu received the B.S. degree in mathematics from Wuhan University in 2000 and the Ph.D. degree in mathematics from Peking University in 2005. He is currently a Professor with the Software Engineering Institute, East China Normal University. His research interests include software engineering and system verification (including ICSE, FSE, ASE, and CAV).