future internet

Article

Deepfake-Image Anti-Forensics with Adversarial Examples Attacks

Li Fan [1], Wei Li [2] and Xiaohui Cui [1,*]

[1] Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430040, China; fl2019@whu.edu.cn
[2] School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi 214000, China; cs_weili@jiangnan.edu.cn
* Correspondence: xcui@whu.edu.cn

Abstract: Many deepfake-image forensic detectors have been proposed and improved due to the development of synthetic techniques. However, recent studies show that most of these detectors are not immune to adversarial example attacks. Therefore, understanding the impact of adversarial examples on their performance is an important step towards improving deepfake-image detectors. This study developed an anti-forensics case study of two popular general deepfake detectors based on their accuracy and generalization. Herein, we propose the Poisson noise DeepFool (PNDF), an improved iterative adversarial examples generation method. This method can simply and effectively attack forensics detectors by adding perturbations to images in different directions. Our attacks can reduce its AUC from 0.9999 to 0.0331, and the detection accuracy of deepfake images from 0.9997 to 0.0731. Compared with state-of-the-art studies, our work provides an important defense direction for future research on deepfake-image detectors, by focusing on the generalization performance of detectors and their resistance to adversarial example attacks.

Keywords: adversarial examples; deepfake; general detectors; Poisson noise

check for updates

Citation: Fan, L.; Li, W.; Cui, X. Deepfake-Image Anti-Forensics with Adversarial Examples Attacks. *Future Internet* **2021**, *13*, 288. https://doi.org/10.3390/fi13110288

Academic Editor: Izzat Alsmadi

Received: 12 October 2021
Accepted: 12 November 2021
Published: 17 November 2021

1. Introduction

As a result of the advent of image and video synthesis technology, it is easy to generate high-confidence fake images and videos. Usually, a "deepfake" refers to people or objects generated by artificial intelligence that look realistic but do not exist in the real world. The most ordinary patterns of a deepfake are the manipulation and generation of human face images. For example, there is the DeepFake method of face replacement in video [1] and the realistic generative adversarial network (GAN) artificial synthesis method [2]. Generally, deepfakes can be divided into four categories [3,4]: reenactment, replacement, editing and synthesis. Reenactment [5] utilizes the source identity X_s to drive the target identity X_t, making X_t behave the same as X_s. Reenactment technology allows the attacker to use the fake identity to control things they say or do, such as defamation, dissemination of misinformation. Replacement [6] refers to replacing the target identity X_t with the content of the source identity X_s, so that the target identity X_t becomes the source identity X_s. The attackers may replace the victim's face with the body of a pornographic actress to insult, slander, or blackmail the victim. Editing technology [7] adds, changes, or deletes the attributes of the target identity; for example, changing the target object's clothes, beard, age, weight, appearance, and even race and skin color. Synthesis [4] refers to a DeepFake character created without a target identity as the basis. The danger is that attackers can create and add characters online for fraud and other illegal activities.

Deepfakes are implemented mainly by using a convolution neural network structure (CNN), encoder-decoder network structure (ED), generative adversarial network (GAN) [8], or image-to-image translation network (CycleGAN) [9]. Most of these are based on CNNs to generate deepfake images and then synthesize fake videos. Therefore, the detection of deepfakes can start with the detection of fake images generated by CNNs. It is relatively simple to detect whether an image is real or fake. However, some methods may have poor generalization when testing on new datasets because their effects are closely related to the dataset used in image generation training. As a result, finding a general image-forensics detector is essential for the detection of deepfakes.

The current research on general detectors mainly aims to find the common features of the CNN network structure and the fingerprints of CNN-generated images [10,11]. Luca et al. [12] proposed a method to extract fingerprints from images generated by deepfakes. They used the Expectation-Maximization algorithm to extract and detect fingerprints that represent convolution traces left in the process of generating images by GAN. Some studies have also shown that the up-sampling and down-sampling operations of neural networks readily leave fingerprints, such as checkerboard artifacts produced by deconvolution layers [13]. Zhang et al. [14] proposed a method called AutoGAN, which can simulate the artifacts produced by the common pipeline shared by several popular GANs, and tested it on CycleGAN and StarGAN. They found that the up-sampling components included in common GANs may cause artifacts, which appear as a copy of the spectrum in the frequency domain. They innovatively used the image spectrum as the input of the classifier instead of the image pixels. This method reached the highest level at the time, even if the target GAN model was not seen during the training process. However, Wang et al. [15] found that not only GANs but other CNNs also observed periodic patterns in the spectrogram when generating images. They proposed a “general” detector for CNN-generated fake images. They demonstrated that with careful pre- and post-processing and data augmentation, a standard classifier trained on ProGAN, an unconditional CNN generator, can be generalized surprisingly well to unseen architectures, datasets, and training methods.

Correspondingly, there are many anti-forensics measures. For example, adversarial example generation methods have been used to modify fake images to attack deepfake detectors. Szegedy et al. [16] found that, in many cases, the deep neural network model will misjudge the input samples after adding a slight perturbation. Later, Gandhi et al. [16,17] used adversarial examples to enhance deepfake images to achieve the purpose of deceiving the VGG16 and Resnet18 detectors. Neekhara et al. [18] proposed using an adversarial example (iterative gradient sign method) to reversely modify the fake videos to bypass the detector. They tested it on the XceptionNet and MesoNet classifiers to evaluate the DeepFake detector for fake videos. Moreover, some researchers [19] directly used generative adversarial networks to generate adversarial examples from benign input images. This method can produce perturbations that can better align with the underlying edges and shapes in the input, making it look more natural.

Several related studies [20] show that neural networks are vulnerable to adversarial example attacks. Herein, we conduct a simple adversarial example attack against the “general” detector proposed by Wang et al. [15]. Our method can enhance the deception ability of the images generated by CNNs on the general detector. We use the general detector’s network structure to construct a new classifier, which is used in the Poisson noise DeepFool (PNDF). By adding perturbations in different directions, PNDF can easily generate adversarial examples. Finally, we input the adversarial examples to the detector and observe the classification accuracy. PNDF can also generate an approximate minimum perturbation, which can be used to evaluate the robustness of the classifier. To prove that our method is not limited to only one forensic classifier, we attack another general detector for GANs proposed by Zhang et al. [14].

In summary, the major contributions of this study are described as follows:

This is the first work proposing a novel approach, Poisson noise DeepFool (PNDF), which can promote the images to be perturbed in different directions by adding Poisson noise during the update.

By adding perturbations to the deepfake images using PNDF in reverse, we find that general detectors are not adversarially robust.

Through comprehensive experiments on eleven datasets, we demonstrate the proposed approach can effectively and simply attack both the accuracy and generalization of the detector.

The remainder of this paper is organized as follows. In Section 2, the existing works using adversarial examples to attack deepfake detectors are discussed. In Section 3, the attacked detectors and attack technology (PNDF) are introduced in detail. We show the specific experimental details and experimental results in Section 4. Section 5 serves as our discussion and conclusion.

2. Related Work

We introduce several classic adversarial examples generation methods and show several related works of using adversarial examples to attack the deepfake detectors in this section.

2.1. Adversarial Examples Generation Method

The adversarial example generation methods are divided into white-box attacks and black-box attacks. To better understand adversarial examples, we introduce a typical white-box attack method (FSFM) and a black-box attack method (Universal Adversarial Perturbations) in detail.

FSGM: Goodfellow et al. [21] proposed an effective calculation method to generate adversarial perturbations, termed the "Fast Gradient Sign Method" (FGSM). This method is gradient-based. In the white-box environment, the derivative of the model to the input is found first. Then the sign function is used to obtain its specific gradient direction and multiplied by a step size ε to obtain the perturbation. Finally, the perturbation is added to the original input to obtain the FGSM example. The FGSM attack is expressed as follows:

$$x' = x + \varepsilon \cdot sign(\nabla_x J(x, y)) \quad (1)$$

where x' is the adversarial example, x is the original image, and $\varepsilon \cdot sign(\nabla_x J(x, y))$ is the added perturbation. The added perturbation can increase the loss so that the model is classified incorrectly and the attack is successful.

Universal Adversarial Perturbations: Since FGSM, etc. can only calculate the perturbation of a single image, Moosavi-Dezfooli et al. [22] proposed a universal perturbation method that can be used for arbitrary images to fool neural networks with high probability. This universal perturbation can cause most images to be misclassified after adding the noise; that is, to perturb a new data point, a universal perturbation is only needed to be added to the image (i.e., there is no need to solve the optimization problem/gradient calculation). The objective function is as follows:

$$\begin{gathered} \hat{k}(x+v) \neq \hat{k}(x) \ for\ most\ \mathrm{x} \sim \mu \\ \parallel \mathrm{v} \parallel_p \leq \varepsilon \\ \underset{x}{p}\left(\hat{k}(x+v) \neq \hat{k}(x)\right) \geq 1-\delta \end{gathered} \quad (2)$$

where μ is the image distribution, v is the universal perturbation. P represents the p-norm, ε controls the size of the perturbation v, and δ is used to measure the expected interference rate of v on all samples.

2.2. Adversarial Examples Attacks on Deepfake Detectors

Gandhi et al. [16] used FGSM and C&W methods to generate perturbations under white-box attack and black-box attack settings, respectively. They used the "Few-Shot Face Translation GAN" to create 5000 fake images to test the VGG16 and Resnet18 classifiers. In the black-box case, the accuracy of both classifiers was decreased significantly, and in the white-box case, the accuracy of both classifiers was equal to or close to 0.

Because Gandhi et al. [17] only attacked image classifiers, and did not extend their work to videos, Neekhara et al. [18] proposed using adversarial examples (iterative gradient sign method) to evaluate the DeepFake (mainly face images) detector for fake videos. They proposed using the existing DeepFake generation method to reversely modify the fake video synthesis to bypass the detector, and tested it on the XceptionNet and MesoNet classifiers.

There are various techniques for generating fake images and videos, and various detectors. However, Neekhara et al. [18] only attacked DeepFake detectors, rather than all forensic detectors, which was a limitation. Carlini et al. [23] proposed five attack case studies for a new classifier for adversarial attacks aiming at the vulnerability of deepfake forensic classifiers. They attacked the two forensic classifiers under the premise of without network parameters (black-box) and accessible network parameters (white-box). Their white-box attacks reduced the area under the ROC curve (AUC) from 0.95 to below 0.1. Even when the classifier's parameters could not be directly accessed, the black-box attacks still reduced the AUC to below 0.22. However, they considered the attack effect as a whole and randomly selected 10,000 images for modification, and did not carefully study every generative model. The attack methods they chose were relatively high in intensity and were not accurate enough to assess the robustness of the detector. Furthermore, they attacked a general detector, but did not analyze it from the perspective of generalization.

3. Materials and Methods

3.1. Attacked Detector

In our work, the main attacked detector is the "general" detector proposed by Wang et al. [15] for CNN deepfake generation models. The main components of the detector are as follows:

1. ProGAN's real images and its generated fake images were used to train a binary classifier with a Resnet50 network structure. The size of the training set is 720,000 images. To extend a single dataset's training results to other datasets, the team used a data augmentation method. All images were first flipped left and right, and then Gaussian blur, JPEG compression, and Blur + JPEG were used to process the image. Data augmentation and data diversity improve the generalization ability of the model and the robustness of classification.

2. Fake images generated by ProGAN and the other ten CNN-based image generation models (StyleGAN, Big-GAN, CycleGAN, StarGAN, GauGAN, DeepFakes, cascaded refinement networks (CRN), implicit maximum likelihood estimation (IMLE), second-order attention super-resolution (SAN), seeing-in-the-dark (SITD)) were used to test classification accuracy and generalization. The real images and fake images generated by each generative model are shown in Figure 1.

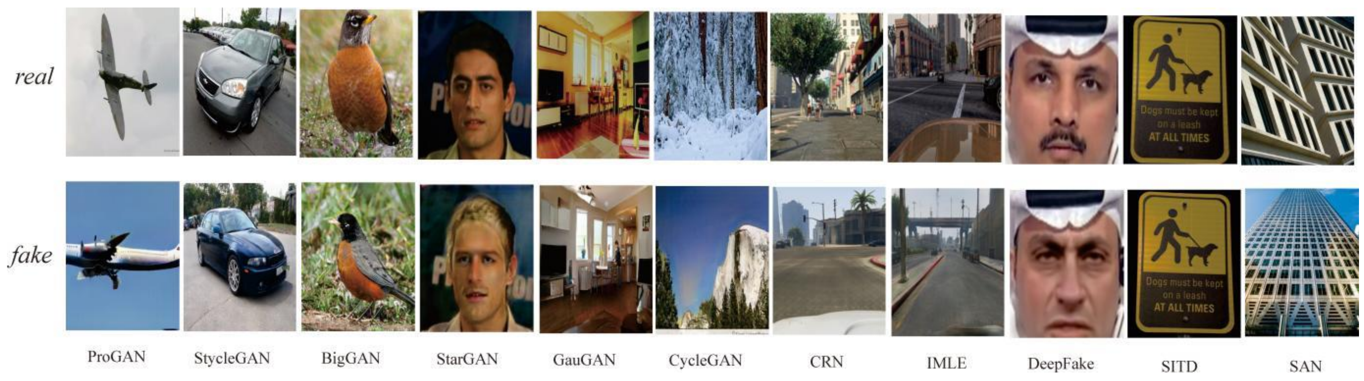

Figure 1. Examples of real and fake images generated by eleven CNN generation models (ProGAN, StyleGAN, Big-GAN, CycleGAN, StarGAN, GauGAN, DeepFakes, cascaded refinement networks (CRN), implicit maximum likelihood estimation (IMLE), second-order attention, seeing-in-the-dark (SITD)) super-resolution (SAN).

Additionally, the same method was used to attack the detector proposed by Zhang et al. [14]. The classifier for GANs takes a similar learning-based approach to that of Wang et al. [15], but the generalized model types are not as extensive as the latter. This detector is different from the previous typical method [24,25] of collecting a large number of GAN-generated images from one or more pretrained GAN models, and then training a binary classifier to design real and fake GAN image classifiers. Here, we emphasize that they noted the fingerprint information generated by the up-sampling module. They also used signal processing-related techniques so that their classifier input was a spectrum, rather than image pixels.

3.2. Adversarial Examples Attack

Due to deep neural networks' vulnerability to adversarial examples, adding perturbation to the input images can easily cause the classifier to misclassify. A small perturbation is added to an original image by adversarial example generation technology, which is enough to change its label. However, it is difficult for the naked eye to distinguish the new sample from the original image. We aimed to generate the minimum perturbations required to simply and effectively change the classification labels. Therefore, we used Poisson noise DeepFool (PNDF) to attack general detectors. This t is proposed for the directionality of DeepFool. Poisson noise is added in the updating process to disturb the samples in different directions. Due to its accurate estimation of the perturbations, PNDF also provides an index to evaluate the robustness of the classifier.

3.3. PNDF for Binary Classifier

Usually, for a given classifier, we define a minimum perturbation r sufficient to change the original classification label l as follows:

$$\Delta(x;l)\min_{r} \parallel r \parallel_2$$

$$\text{subject to } l(x+r) \neq l(x) \tag{3}$$

where x is an image and $l(x)$ is the estimated label; further, we call $\Delta(x;l)$ the robustness of l at point x. Moosavi-Dezfooli et al. [26] proposed the computation of a minimal norm adversarial perturbation for a given image in an iterative manner called DeepFool. They assumed the original image is located in the region defined by the decision boundary, and a small vector is added each iteration. Until the image label changes, these vectors are accumulated to obtain the final perturbation. Specifically, assume $l(x) = sign(f(x))$, where f is an arbitrary scalar value image classification function, $f: R^n \to R$. Firstly, we analyze the case where f is a linear classifier $f(x) = \{w^T x + b\}$, and then generalize it to a general binary classifier. DeepFool can generate perturbations for binary classifiers and multi-classifiers. Because a multi-class classifier can be regarded as a collection of binary classifiers, and the goal of this work is to attack the binary classifiers, here we only discuss the algorithm of binary classifiers.

When f is linear, it is easy to find the robustness of f at point x_0, $r_*(x)$ is equal to the distance from x to the boundary hyperplane $\mathcal{F} = \{x : w^T x + b = 0\}$ (Figure 2). At this time, the minimal perturbation that changes the classification result of the classifier is the orthogonal projection of x_0 onto the hyperplane $\mathcal{F}$. Assuming now that f is a general binary differentiable classifier, we adopt an iterative procedure to estimate the robustness $\Delta(x_0; f)$.

However, DeepFool only considers the projection from the sample point to the hyperplane and ignores the perturbations' directionality. Moreover, the minimum perturbation is the orthogonal projection from the point to the hyperplane, which is a small probability event. It is difficult to ensure that every perturbation is just the orthogonal projection. Poisson noise is very small and obeys the regular Poisson distribution. In addition, because of its signal dependence and good performance in improving the signal-to-noise ratio, it can ensure better image quality when increasing noise to obtain directional perturbations.

Therefore, considering the perturbations' directionality and generality, we add Poisson noise in the iterative update perturbation. When we update $r_*(x)$, we draw a circle with x_0 as the center and $r_*(x)$ as the radius. In this case, the hyperplane $\mathcal{F}$ is the tangent of the circle, and it forms a right triangle with the quarter circle, as shown in Figure 2. The coverage area of the smallest perturbation in the shadow part is defined as r_s. Obviously, because both x_0 and $\mathcal{F}$ are fixed, r_s is only related to $r_*(x)$.

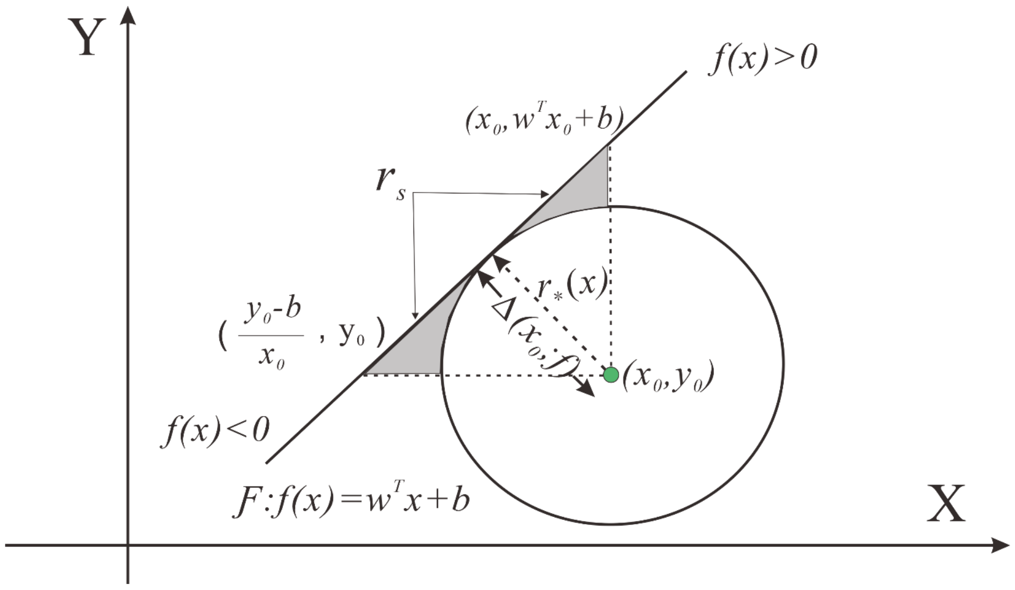

Figure 2. Calculation about shadow part r_s in PNDF for a binary classifier.

r_s is calculated as follows:

$$r_s = \frac{\left(x_0 - \frac{y_0-b}{w^T}\right)(w^T x_0 + b - y_0)}{2} - \frac{\pi r_*^2(x_0)}{4} \tag{4}$$

We add Poisson noise to properly increase $r_*(x)$ and decrease r_s to obtain the perturbation in different directions. The calculation formula of $r_*(x)$ is updated as follows:

$$r_*(x) = \mathcal{P}_{(x_{0,l(x_0)})}\left(\underset{r}{\operatorname{argmin}} \parallel r \parallel_2\right) \tag{5}$$

where $\mathcal{P}(\cdot)$ means to add Poisson noise at the sample point x_0 with label $l(x_0)$. The Poisson noise DeepFool (PNDF) algorithm for binary classifiers is summarized in Algorithm 1. The algorithm is simple in theory and easy to implement.

Algorithm 1 Poisson noise DeepFool (PNDF) for binary classifiers

```
Input: Image $x$, classifier $f$
Output: Perturbation $r$
  1: Initialize $x_0 \leftarrow x,\ i \leftarrow 0$
  2: While $sign(f(x_i)) = sign(f(x_0))$ do
  3: $r_i \leftarrow \mathcal{P}_{(x_0,l(x_0))}\left(-\frac{f(x_i)}{\|\Delta f(x_i)\|_2^2}\Delta f(x_i)\right)$,
  4: $x_{i+1} \leftarrow x_i + r_i$,
  5: $i \leftarrow i + 1$,
  6: end while
  7: return $r = \sum_i r_i$
```

In this work, perturbations are generated by PNDF to reverse modify the images to deceive the classifier. Moreover, we use the original DeepFool to reverse modify ProGAN's dataset, which is our baseline. In PNDF, the perturbation r is iteratively added to image x until the predicted label of the image $x + r$ changes from fake to real or from real to fake. We add perturbations to both fake images and to real images. However, the reverse modification of the real images requires greater perturbations, as found by Carlini et al. [23], and we also observe from the experimental results that modifying about 1% of the pixels can ensure 50% of fake images are classified as real, whereas misclassifying 50% of real images as fake requires modifying about 6% of the pixels (Figure 3). Therefore, we pay close attention to the change in the fake image detection accuracy (f-ac). Figure 4 is a schematic diagram of reverse modifying the image to deceive the detector. The network structure and parameters of "Classifier" are extracted from "Detector".

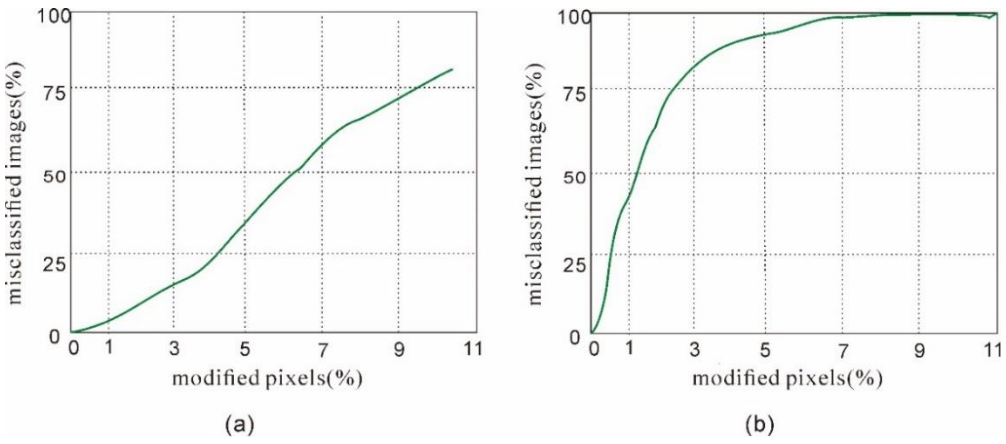

Figure 3. The required modified pixels to fool the classifier into identifying (**a**) real images as fake or (**b**) fake images as real.

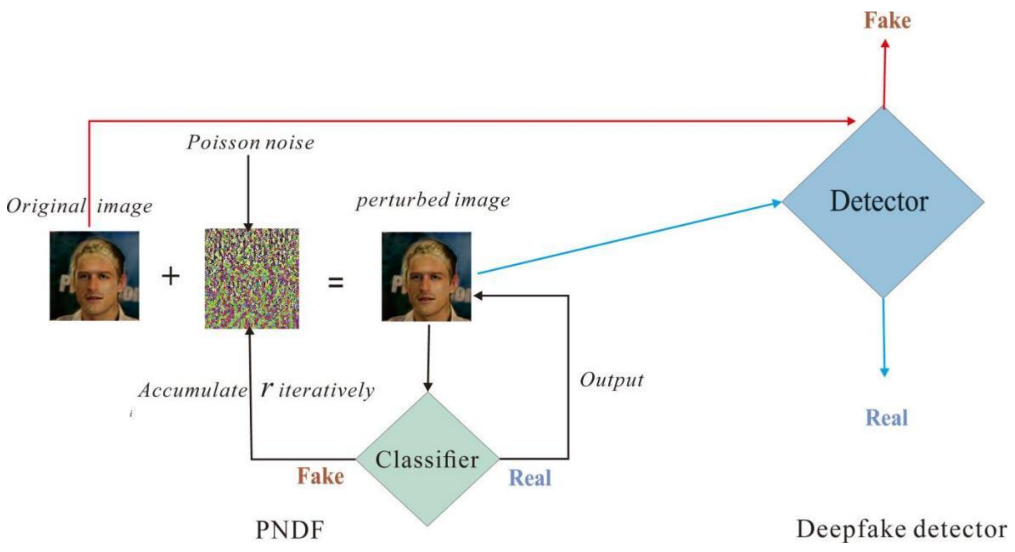

Figure 4. Schematic diagram of modifying the image reversely to deceive the detector. On the left is the process of using PNDF to iteratively add a perturbation r to the original image. Poisson noise is added when r is updated to obtain the perturbation in different directions. When the label obtained by the classifier changes, the perturbed image is outputted as the detector's input to make the detector's judgment result opposite to the original image.

4. Experiment and Discussion

4.1. Datasets and Models

We evaluated our proposed attack algorithm on the general detectors proposed by Wang et al. [15] and Zhang et al. [14]. The datasets used in our attack were eleven test sets which were divided into four categories: various GAN networks, Perceptual loss, Low-level vision, and DeepFake. The image source of each generative model and the number of real and fake images they contain are listed in Table 1.

Table 1. Generation Models Datasets.

Category	Models	Fake Images	Real Images	Image Source
Conditional GANs	CycleGAN	1321	1321	Style/object transfer
	StarGAN	1999	1999	Celebes
	GauGAN	5000	5000	COCO
Unconditional GANs	ProGAN	4000	4000	LSUN
	StyleGAN	5991	5991	LSUN
	Bagan	2000	2000	ImageNet
Low-level vision	SITD	180	180	Raw camera
	SAN	219	219	Standard SR benchmark
Perceptual loss	CRN	6382	6382	GTA
	IMLE	6382	6382	GTA
DeepFakes	FaceForensics++	2700	2700	Videos of faces

GANs We used the latest unconditional GANs trained on the LSUN or ImageNet dataset: ProGAN [27], StyleGAN [2], and BigGAN [28]. ProGAN and StyleGAN train different networks for each category. BigGAN has a single-block class conditional structure, which is trained on a very large batch scale and uses a self-attention layer. We also used three image-to-image translation method conditional GANs: GauGAN [29], CycleGAN [9], and StarGAN [30].

Low-level vision: Seeing-in-the-dark (SITD) [31] uses a high-resolution fully convolutional network. It approximates the long-exposure photography from short-exposure raw camera input in low-light conditions. We also tested a state-of-the-art super resolution model, the Second Order Attention Network (SAN) [32].

Perceptual loss: We tested no the confrontation training process which directly optimized the generation model of perceptual loss: cascaded refinement networks (CRN) [33] and implicit maximum likelihood estimation (IMLE) [34].

DeepFakes: Face replacement images are provided in FaceForensics++ [35]. The model uses an auto-encoder to generate a face and then uses a post-processing step of mixing the generated image with the real content Poisson image.

4.2. Attack Settings

Our attack was implemented with Python 3.6 based on Pytorch. GeForce GTX 1080Ti GPU and CUDA 9.2 were used for training acceleration. When adding perturbations to the images of each test set for reverse modification, the network structure of the detector and parameters must first be extracted. When using PNDF to add perturbation in different directions, the image's original label is first given by the classifier. According to the PNDF calculation perturbation formula, the perturbations are added to generate adversarial examples as the classifier inputs, and the processing is iterated until the classifier result changes.

To quantify the attack of the adversarial example on the detector, the f-ac, AUC (AUC is defined as the area under the ROC curve that compares the false positive rate to the true positive rate and must be less than 1; the ROC curve is the receiver operating characteristic curve, drawn with the true positive rate as ordinate), and AP before and after adding perturbation are recorded. The f-ac is the fake images' accuracy, which refers to the attacked detector's detection accuracy only for fake images. Average precision (AP), an important index of object recognition, is precision averaged across all values of recall between 0 and 1. Equations (6)–(9) are the formulas for calculating f-ac, recall, precision and average precision, where TP = True Positive, FP = False Positive, TN = True Negative, FN = False Negative, P refers to precision, and r refers to recall.

$$\mathrm{f-ac} = \frac{TN}{TN + FN} \tag{6}$$

$$\mathrm{recall} = \frac{TP}{TP + FN} \tag{7}$$

$$\mathrm{precision} = \frac{TP}{TP + FP} \tag{8}$$

$$\mathrm{AP} = \int_0^1 P(r)dr \tag{9}$$

AP and AUC were calculated after modifying real and fake images to evaluate the detector's robustness for detecting real and fake images. These indicators were calculated by the built-in functions of Pytorch.

4.3. *Attack Results*

To illustrate our contribution, we conducted many experiments, and here show the experiment results from the accuracy and generalization of the "general" detector. Moreover, to prove that our method is not limited to only one deepfake image detector, we also attack the detector proposed by Zhang et al. [14].

4.3.1. Attack Accuracy of the "General" Detector

Given that the detector is trained on the fake images generated by ProGAN and its corresponding real images, the detection of fake images generated by ProGAN should achieve a high accuracy. The comparison of the classifier's accuracy before and after perturbations are added by DeepFool and PNDF is listed in Table 2. We find that the attack intensity of PNDF with perturbation in different directions is stronger than that of DeepFool. Further, we measured the amount of added noise by calculating the percentage of modified pixels. The results show that the percentage of modified pixels of all images with perturbation is less than 10%. The detector's accuracy for original fake images generated by ProGAN is close to 1, and both AP and AUC reach 0.9999; that is, the performance of the detector is ideal when no perturbation is added. When using PNDF to add perturbation to the images, the f-ac is reduced from 0.9997 to 0.0731, and the AUC is reduced from 0.9999 to 0.0331. Because the recognition accuracy of the classifier is not equal to 100%, when the fake images are reverse modified, the fake images' labels after modification may be still fake; that is, the sum of the accuracy before and after modification is not equal to 1.

Table 2. Accuracy comparison of ProGAN.

	f-ac	**AP**	**AUC**
unperturbed	0.9997	0.9999	0.9999
Perturbed by DF	0.0923	0.3356	0.0430
Perturbed by PNDF	0.0731	0.3212	0.0331

4.3.2. Attack Generalization of the "General" Detector

The detector's generalization means that the detector trained on a certain training set can distinguish fake images, directly or under minor adjustment, regardless of the architecture or datasets. Wang et al. [15] demonstrated that with careful pre- and post-processing and data augmentation, a standard image classifier trained on only one specific CNN generator (ProGAN) can be generalized surprisingly well to unseen architectures, datasets, and training methods. Due to its superiority based on our test results, PNDF was used directly to add perturbations to the datasets of the ten other CNN generation models

(StyleGAN, Big-GAN, CycleGAN, StarGAN, GauGAN, DeepFakes, cascaded refinement networks (CRN), implicit maximum likelihood estimation (IMLE), second-order attention super-resolution (SAN), seeing-in-the-dark (SITD)) and the changes in the accuracy of the detector's fake image detection were recorded. In this process, the structure and parameters of the detector network are not changed, and we observe the extent to which its generalization ability is affected by the adversarial examples.

The generalization performance comparison before and after adding perturbations by PNDF is exhibited in Table 3. Before adding perturbations, most of the classification indicators (AUC and AP) of the ten generation models were above 0.9, and the f-ac of CRN and IMLE reach 0.99; that is, the generalization effect of the detector on the detection of original fake images is ideal. Our attack method is relatively effective in terms of the generalization performance, which is distinctly demonstrated by the comparison of indicators before and after addition with perturbations. The f-ac is reduced to below 0.1 (CRN's f-ac is equal to 0.0007) and the AP is around 0.3. The AUC's value is reduced from 0.95 to below 0.05. DeepFake's AUC even drops to 0 (bold in the table), which means that the classifier is completely invalid. Figure 5 clearly shows the changes in the three evaluation indicators before and after adding perturbations in the form of a histogram. The detection effect of the detector on the original fake images of SAN is very poor, and the f-ac is close to 0. Thus, the abnormal change in SAN after reverse modifying the fake images can be explained. Additionally, we find that AP and AUC are positively correlated, so their values can better reflect the changes in the accuracy of the detector.

Table 3. Generalization comparison of the detector before and after adding perturbation by PNDF.

Models	**f-ac**		**AP**		**AUC**	
	Unperturbed	**Perturbed**	**Unperturbed**	**Perturbed**	**Unperturbed**	**Perturbed**
CRN	0.9987	0.0007	0.9823	0.3135	0.9877	0.0394
IMLE	0.9976	0.0393	0.9840	0.3069	0.9884	0.0024
SITD	0.8666	0.0611	0.9723	0.3068	0.9756	0.0156
StarGAN	0.8129	0.0065	0.9400	0.3126	0.9780	0.0417
CycleGAN	0.7887	0.0832	0.9246	0.3346	0.9384	0.0799
StyleGAN	0.7426	0.1330	0.9959	0.3100	0.9958	0.0354
GauGAN	0.6480	0.0180	0.8948	0.3077	0.9206	0.0130
BigGAN	0.4690	0.0465	0.8450	0.3087	0.8819	0.0160
DeepFake	0.0685	0.0111	0.8902	0.3063	0.8844	0.0000
SAN	0.0182	0.5844	0.7046	0.3400	0.7185	0.1049

4.3.3. Attack of Zhang et al.

Zhang et al. [14] used a Resnet34 pre-trained by ImageNet as the basic network, and regarded the GAN detection task as a simple binary classification problem. They trained different classifiers for comparison:

Img: Trained with real images and fake images generated by CycleGAN.

A-Img: Trained with real image and fake image generated by AutoGAN.

Spec: The training data was the same as that of **Img**, but the classifier was trained with the spectrum input.

A-Spec: The training data was the same as that of **A-Img**, but the classifier was trained with the spectrum input.

The classifier trained based on fake images generated by the proposed AutoGAN (**A-Spec**) can achieve the best generalization effect. It achieved accuracy of 0.972 for the detection of CycleGAN. Therefore, to test our method on this detector, we applied the similar attack to this classifier (AutoGAN) for CycleGAN. We found that our method can also reduce its accuracy to almost 0 (0.052).

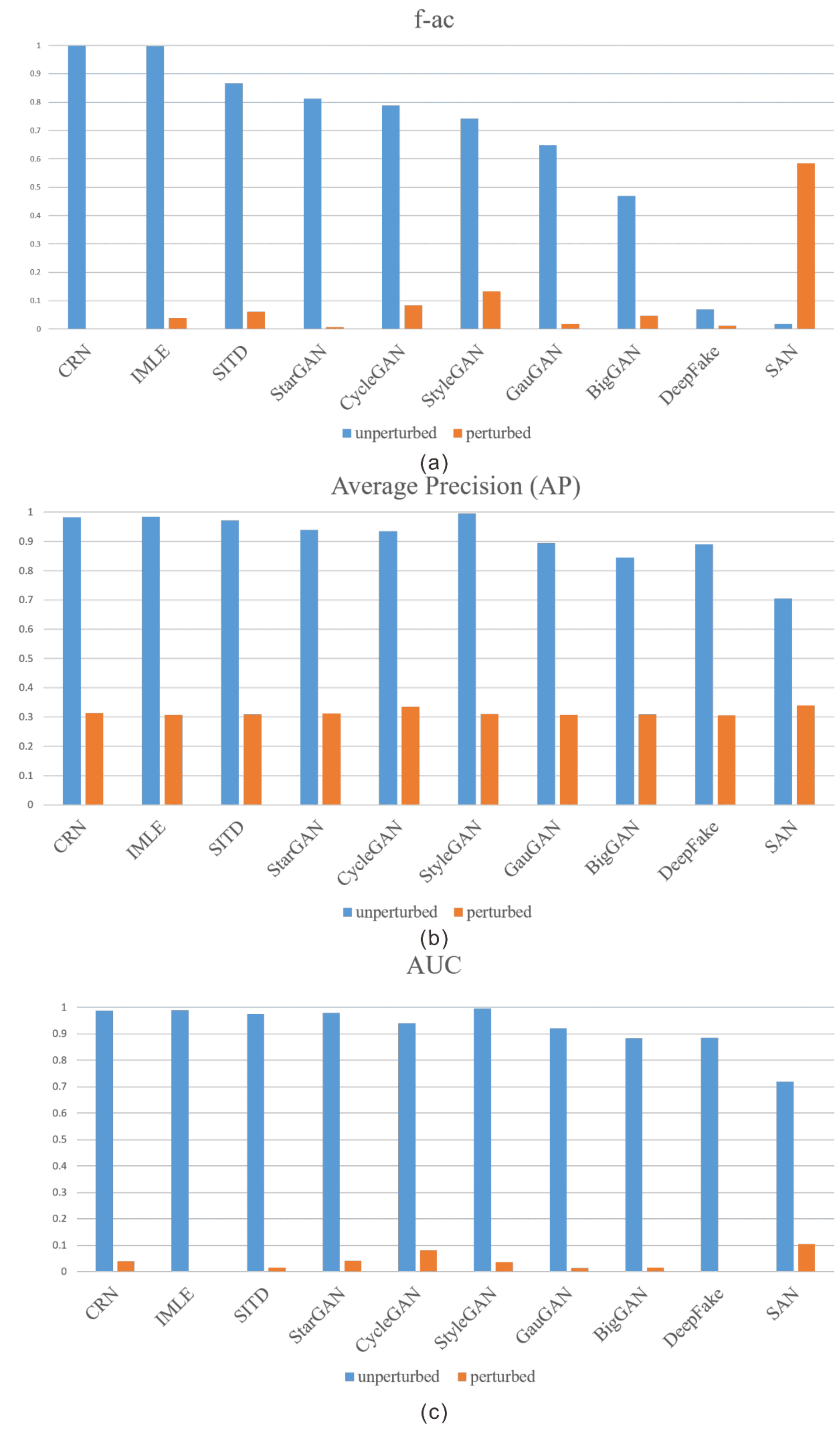

Figure 5. Comparison of histograms of f-ac (**a**), AP (**b**), and AUC (**c**) for the other ten generator models before and after adding perturbation. The blue histogram represents the indicator value before adding perturbations, and the orange histogram represents the indicator value after adding perturbations.

5. Conclusions

As a result of the rapid development of malicious deepfake technology, the detection of deepfake images has become a major security issue. Although many detectors have been proposed and improved, some related research shows that they are still vulnerable to adversarial example attacks. In this paper, we propose a method using Poisson noise DeepFool to reverse modify the image to deceive detectors, which can also evaluate the robustness of the detectors. To verify our proposed method's effectiveness, we attacked two popular general detectors proposed by Wang et al. [15] and Zhang et al. [14]. For the former, we assessed the accuracy and generalization. After reverse modifying the images generated by ProGAN, the AUC was reduced from 0.9999 to 0.0331. This shows that our attack has a significant impact on the accuracy of the classifier. When we modified the datasets of ten other generation models, the best experimental result was the reduction in the AUC to 0 (DeepFake). For the remaining CNN generation models [36], the classification AUC dropped below 0.05, indicating the classifier failed. For the detector trained by Zhang et al. [14] on AutoGAN, the detection accuracy of CycleGAN decreased from 0.972 to almost 0.

This work proves mainly that general detectors can be deceived by a simple adversarial example generation method with small perturbations. The experimental results also show that deepfake-image anti-forensics using adversarial example attacks is effective. Although our attack method is successful, we used a white-box attack method, which requires access to the network structure and parameters. Therefore, one of our future works will aim to achieve the attack effect of this article using black-box attack methods, such as the One-Pixel attack. Moreover, we will continue to study attacks on adversarial detectors or mitigation technologies to help improve the performance of general detectors.

Author Contributions: Conceptualization, L.F. and X.C.; Methodology, L.F. and W.L.; Software, L.F.; Writing—original draft, L.F.; Writing—review & editing, W.L. and X.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Key Research and Development Program of China (NO.2018YFC1604000).

Data Availability Statement: The source code for our attack is available on https://github.com/fanliaveline/DeepFool-attack-CNN-generated-detector (accessed on 12 October 2021).

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. DeepFakes Faceswap Github Repository. Available online: https://github.com/DeepFakes/faceswap. (accessed on 12 October 2021).
2. Karras, T.; Laine, S.; Aila, T. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4401–4410.
3. Mirsky, Y.; Lee, W. The Creation and Detection of Deepfakes: A Survey. *arXiv*. 2020. Available online: https://arxiv.org/abs/2004.11138 (accessed on 12 October 2021).
4. Tolosana, R.; Vera-Rodriguez, R.; Fierrez, J.; Morales, A.; Ortega-Garcia, J. Deepfakes and beyond: A Survey of face manipulation and fake detection. *Inf. Fusion.* **2020**, *64*, 131–148. [CrossRef]
5. Tran, L.Q.; Yin, X.; Liu, X. Representation Learning by Rotating Your Faces. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *41*, 3007–3021. [CrossRef] [PubMed]
6. Schwartz, O. You Thought Fake News Was Bad? The Guardian. Available online: https://www.theguardian.com/technology/2018/nov/12/deep-fakes-fake-news-truth (accessed on 12 October 2021).
7. Samuel, S. A Guy Made a Deepfake App to Turn Photos of Women into Nudes. It didnfit Go Well. 2019. Available online: https://www.vox.com/2019/6/27/18761639/ai-deepfake-deepnude-app-nude-women-porn (accessed on 12 October 2021).
8. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. *Adv. Neural Inf. Proces. Syst.* **2014**, *27*, 2672–2680.
9. Zhu, J.; Park, T.; Isola, P.; Efros, A.A. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2242–2251. [CrossRef]
10. Nataraj, L.; Mohammed, T.M.; Manjunath, B.S.; Chandrasekaran, S.; Flenner, A.; Bappy, J.H.; Roy-Chowdhury, A. Detecting GAN generated Fake Images using Co-occurrence Matrices. *Electron. Imaging* **2019**, *2019*, 532-1–532-7. [CrossRef]

11. Cozzolino, D.; Thies, J.; Rssler, A.; Riess, C. Forensic Transfer: Weakly-Supervised Domain Adaptation for Forgery Detection. 2018. Available online: https://arxiv.org/abs/1812.02510 (accessed on 12 October 2021).
12. Guarnera, L.; Giudice, O.; Battiato, S. Fighting Deepfake by Exposing the Convolutional Traces on Images. *IEEE Access* **2020**, *8*, 165085–165098. [CrossRef]
13. Odena, A.; Dumoulin, V.; Olah, C. Deconvolution and Checkerboard Artifacts. *Distill* **2016**, *1*, e3. [CrossRef]
14. Zhang, X.; Karaman, S.; Chang, S.-F. Detecting and Simulating Artifacts in GAN Fake Images. In Proceedings of the 2019 IEEE International Workshop on Information Forensics and Security (WIFS), Delft, The Netherlands, 9–12 December 2019; IEEE: Piscataway, NJ, USA, 2019.
15. Wang, S.-Y.; Wang, O.; Zhang, R.; Owens, A.; Efros, A.A. CNN-Generated Images Are Surprisingly Easy to Spot . . . for Now. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 8692–8701.
16. Szegedy, C.; Zaremba, W.; Sutskever, I. Intriguing properties of neural networks. In Proceedings of the International Conference on Learning Representations, Banff, AB, Canada, 14–16 April 2014.
17. Gandhi, A.; Jain, S. Adversarial Perturbations Fool Deepfake Detectors. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 June 2020; IEEE: Piscataway, NJ, USA, 2020.
18. Hussain, S.; Neekhara, P.; Jere, M.; Koushanfar, F.; McAuley, J. Adversarial Deepfakes: Evaluating Vulnerability of Deepfake Detectors to Adversarial Examples. In Proceedings of the 2021 IEEE Winter Conference on Applications of Computer Vision (WACV), Waikoloa, HI, USA, 3–8 January 2021; pp. 3347–3356.
19. Zhang, W. Generating Adversarial Examples in One Shot with Image-to-Image Translation GAN. *IEEE Access* **2019**, *7*, 151103–151119. [CrossRef]
20. Akhtar, N.; Mian, A. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. *IEEE Access* **2018**, *6*, 14410–14430. [CrossRef]
21. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
22. Moosavi-Dezfooli, S.; Fawzi, A.; Fawzi, O. Universal adversarial perturbations. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
23. Carlini, N.; Farid, H. Evading Deepfake-Image Detectors with White- and Black-Box Attacks. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Seattle, WA, USA, 14–19 June 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 2804–2813.
24. Marra, F.; Gragnaniello, D.; Cozzolino, D.; Verdoliva, L. Detection of GAN-Generated Fake Images Over Social Networks. In Proceedings of the 2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), Miami, FL, USA, 10–12 April 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 384–389.
25. Yu, N.; Davis, L.; Fritz, M. Attributing Fake Images to GANs: Learning and Analyzing GAN Fingerprints. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 7555–7565.
26. Moosavi-Dezfooli, S.-M.; Fawzi, A.; Frossard, P. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; IEEE: New York, NY, USA, 2016; pp. 2574–2582.
27. Karras, T.; Aila, T.; Laine, S. Progressive growing of gans for improved quality, stability, and variation. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
28. Brock, A.; Donahue, J.; Simonyan, K. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In Proceedings of the International Conference on Learning Representations, Vancouver, Canada, 30 April–3 May 2018.
29. Park, T.; Liu, M.-Y.; Wang, T.-C.; Zhu, J.-Y. Semantic Image Synthesis with Spatially-Adaptive Normalization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 2332–2341.
30. Choi, Y.; Choi, M.; Kim, M. Stargan: Unified generative adversarial networks for multi-domain image-to-image transla-tion. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 19–21 June 2018; pp. 8789–8797.
31. Chen, C.; Chen, Q.; Xu, J.; Koltun, V. Learning to See in the Dark. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition; Institute of Electrical and Electronics Engineers (IEEE), Salt Lake City, UT, USA, 18–23 June 2018; pp. 3291–3300.
32. Dai, T.; Cai, J.; Zhang, Y.; Xia, S.-T.; Zhang, L. Second-Order Attention Network for Single Image Super-Resolution. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 11057–11066.
33. Chen, Q.; Koltun, V. Photographic Image Synthesis with Cascaded Refinement Networks. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1520–1529.
34. Li, K.; Zhang, T.; Malik, J. Diverse Image Synthesis from Semantic Layouts via Conditional IMLE. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 4220–4229.

35. Rossler, A.; Cozzolino, D.; Verdoliva, L.; Riess, C.; Thies, J.; Nießner, M. Faceforensics++: Learning to detect manipulated facial images. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019.
36. Li, W.; Ding, W.; Sadasivam, R.; Cui, X.; Chen, P. His-GAN: A histogram-based GAN model to improve data generation quality. *Neural Netw.* **2019**, *119*, 31–45. [CrossRef] [PubMed]