Fake Image Generation and Detection using GANs

Aishwarya Balajee
Electrical Engineering Department
Rochester Institute of Technology

Najmus Sahar
Electrical Engineering Department
Rochester Institute of Technology

***Abstract*—With the growth of Artificial Intelligence, specifically the generative AI models, deepfakes are becoming more common and sophisticated, making them almost indistinguishable from authentic data. In this project, we generate and identify fake facial images using the implementation of a convolutional Generative Adversarial Networks (DCGAN). This approach not only strengthens our capabilities to generate realistic images, but also helps us distinguish the real images from synthetically generated images.**

I. Introduction

With the widespread accessibility to the Internet and groundbreaking improvements made in the field of deep learning technology, it has become fairly easy to generate fake images and post them on social media for a large audience. While fake images are usually generated for entertainment, this AI technology is often misused for malicious purposes. It can be used to create Deepfakes to spread false rumors about public figures or create fake news. It becomes vital for internet users to be able to verify the authenticity of content that is available online to avoid widespread distrust in digital media content. Further, criminals can manipulate images to commit fraudulent activities and evade the law. As machine learning models improve at fake image generation, it becomes difficult for humans to detect if an image is real or fake. This situation brings in the need for better ways to spot fake images.

AI image generators generally use neural networks and are trained on large amounts of data comprising image datasets. During training, the machine learning algorithm learns different features of the images in the dataset. Based on the features it has learned, it will generate new images similar in style to the dataset used to train the model. The most commonly used type of AI model is the Generative Adversarial Networks, known as GANs. GANs comprise two components: the generator network and the discriminator network. The generator generates fake images based on a random input vector. The discriminator acts as a binary classifier and labels the input image as real or fake. Several variations of GANs have been developed and implemented for diverse applications and to improve the resolution of the image generated. However, the evolving complexity of GANs remains challenging for existing fake image detection technologies.

Further, the lack of availability of diverse and large datasets of GAN-generated images proves to be a challenge when it comes to training fake image detection models. Such datasets are vital for training models that can generalize well on diverse demographics and conditions. A bigger challenge is developing a robust detection model that generalizes well on diverse datasets. Scalability also becomes an issue, especially when it comes to the detection of large amounts of fake content that is generated online. It becomes necessary to develop a robust and scalable model while striking a perfect balance between minimizing false positives and false negatives.

This project will combine convolutional and conditional GANs for fake image detection. We will also create a dataset of generated images that can be publicly available for training other newly developed discriminatory models.

II. Related Work

The most common practice in machine learning is the creation of supervised learning models. While these methods generally produce accurate results post-training, they require datasets that comprise labeled/classified input samples. Hence, the training process requires extensive human intervention and effort.

To surpass the shortfalls mentioned above, the current research focus of the machine learning world needs to be supervised training algorithms. Generative Adversarial Networks fall under the unsupervised learning category, which efficiently generates data and discriminates. This algorithm was proposed by Ian Goodfellow et al. in their paper [3]. The authors explain the architecture of GANs as an unsupervised two-player minimax game. The framework consists of a generator and discriminator that are trained simultaneously. The generator takes random noise as its input and generates fake data fed to the discriminator network in combination with real data. The discriminator then has to output the probability of the data being real or fake. Hence, the generator aims to deceive the discriminator and the discriminator to identify real vs fake data correctly.

The GAN architecture proposed by Ian Goodfellow et al. consists of a simple multi-perceptron layer. However, many architecture variations were introduced and analyzed for diverse applications of GANs based on various input types. The paper by Daniel Jiwoong Im et al., [5] discusses the widely used derivatives of GANs, such as Deep Convolutional GAN (DCGAN), Wasserstein Deep Convolutional GAN (W-DCGAN), etc. and evaluates them based on various metrics such as Wasserstein Distance, Maximum Mean Discrepancy, etc. The authors were able to statistically compare the output performances of GANs using the evaluation metrics under different hyper-parameter settings and suggest possible future work on other commonly used models to understand the

best model that can be deployed for various data types and requirements.

The survey paper by Pan et al., [9] analyzes and classifies the derivatives of GANs that were developed to overcome the deficit of the original GANs. In terms of architecture, GANs are classified into convolution-based GANs, condition-based GANs, and autoencoder-based GANs. The DCGAN is a convolution-based GAN, which is essentially the original GAN with the fully connected layers of generator and discriminator replaced by a deep convolutional neural network architecture. DCGANs can be specifically used for image generation tasks with exceptional performance. In condition-based GANs such as Conditional Generative Adversarial Networks (CGANs), InfoGAN, and Auxiliary Classifier GANs (ACGANs), additional information is given as input to the generator in addition to the random noise vector. This condition helps restrict the input and increase the efficiency of the training. Autoencoder-based GANs consist of an encoder and decoder in place of the generator. The encoder's input is converted into a hidden layer which can then be used to reconstruct the actual input by the decoder. This algorithm results in a type of unsupervised learning since the input labels would not be required during the training phase. In addition, the authors also list some useful metrics for evaluating the performance of GANs while also providing the pros and cons of these metrics so that an appropriate metric can be used to evaluate a specific GAN model.

Images manipulated by sophisticated image editing tools can be analyzed using forensic techniques. However, this does not apply to GAN-generated images. To combat identity theft resulting from such practices, Do et al. proposed a DCGAN to generate face data images, which were then used to train their model [2]. The generated face images were then processed to extract facial feature representations using VGG-Net. Finally, the deep neural network was fine-tuned to adjust the weights for the binary classification of images as real or fake.

GANs are especially valuable when there is a clear deficit of available data. An apt example would be the medical field. With respect to data collection in the medical field, patient consent is important, and a crucial deficit lies in the availability of rare or abnormal data. One such example is provided in the paper by Deepankar Nankani and Rashmi Dutta Baruah [8] in their attempt to investigate the use of deep convolutional GANs with the combination of conditional GANs to generate irregular electrocardiogram (ECG) signals. The authors created a model that generated the required types of abnormal beats, and these were evaluated based on the standard available medical databases.

While deep convolutional GANs have improved the quality of image generation, further improvements can be implemented based on necessity. The paper by Sukarna Barua et al., [1] proposed the implementation of fully connected layers and pooling layers in deep convolutional GANs can be used to increase the quality of image generation. The authors proved the efficiency of this model architecture by applying it to four benchmark datasets. Based on inception and FID scores, they showed that this model outdid the current benchmarks.

The expansion of Artificial intelligence (AI) has led us to a world where fake data can be very easily generated and spread around. We require efficient AI-generated databases to combat this and use discriminatory models. GANs can be a very useful tool to generate such databases. One example of generating databases is shown in the paper by Hanwen Yang et al., [11]. The authors of this paper use GAN to generate a dataset of finger veins to aid in the field of biometrics. The authors built a dataset that consisted of much clearer images compared to the pre-existing benchmarks. The paper also suggests possible future work to enhance the image quality of generated images.

Several variations in GANs have been developed so far, either to overcome the limitations of pre-existing versions of GAN or to fit the needs of a specific application. In [6], the authors conducted an exploratory study to test the generalizability of existing deep learning methods for fake face detection. They also created a database known as the Fake Face in the Wild (FFW) with a diverse set of fake images generated using different fake face image generation techniques. They then evaluated the performance of different algorithms, such as AlexNet, VGG19, and ResNet50, on the detection of fake images from the FFW dataset. The performance metrics indicate that in the presence of unknown data, deep-learning methods fail to detect fake faces with good accuracy. This brings in the need for a generalizable fake face detection model that is equipped to handle diverse collections of data.

With increased interest in researching and devising better GAN models, it has become possible to create hyper-realistic fake images more easily than ever before. Such GAN-generated images can have high resolution and are indistinguishable from real images. While image classifiers can be trained to identify fake images, training an image classifier on a large number of GAN-generated fake images is impossible as it is difficult to identify the specific GAN model used for fake image generation by the perpetrator. In the paper by Zhang et al., [12], the authors developed the AutoGAN, which has a comparable structure to the generator and discriminator used in GANs. The AutoGAN takes advantage of the up-sampling mechanism used in GAN models to create high-resolution images. By studying the frequency spectrum of such images rather than the pixels, the AutoGAN can detect the GAN-induced artifacts and thereby get trained to generalize and detect other GAN-generated fake images with similar artifacts. This paper makes it possible to train a fake image detector without the use of a readily available dataset of fake images. The AutoGAN uses real images to incorporate the artifacts of GAN-generated images into the real image during training. Further, this enables the model to detect fake images of different types.

An unsupervised learning approach is used by Hsu et al. in their paper [4], to develop a fake face image detector called the Common Fake Feature Network (CFFN). To address the shortcomings of the supervised learning method used for fake image detection, the authors used a pairwise learning strategy to learn fake features, thereby enabling the fake image

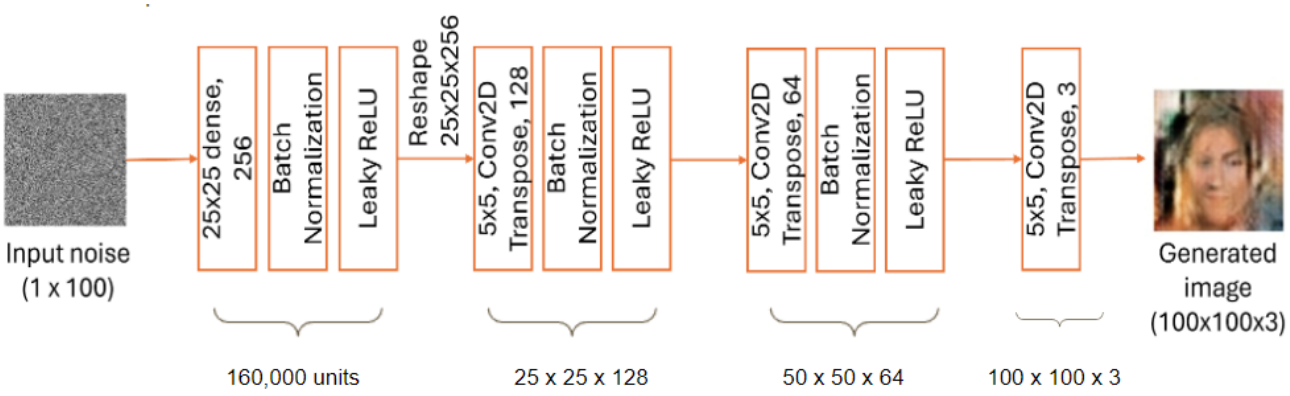

Fig. 1. Generator architecture

detector to generalize and successfully detect fake images that are generated by new types of GANs that were not initially included in the training set. The performance of CFFN was compared with other fake image detectors such as the BIGGAN, SA-GAN and SN-GAN. The results of the performance metrics, such as precision and recall, indicate that the CFFN far outperforms the existing general image detectors.

III. Proposed Methodology

A. Dataset

The dataset used in this project is the Large-scale CelebFaces Attributes (CelebA) Dataset [7]. This dataset comprises more than 200,000 diverse face images, including various poses and backgrounds. This dataset also contains annotations of faces in each image, which will improve the quality of future work done on this project.

B. Generator Architecture

The generator and discriminator architectures are chosen to be simple, so that the model can be trained on low computational power [10]. The model architecture of the generator is as shown in figure 1. The noise vector input to the architecture is an array of dimension 1x100. This vector is passed through the dense layer, which creates a latent space of 160,000 units, laying the foundation for image creation by the generator. This is further passed through the batch normalization layer, which essentially normalizes the previous layer's activation, maintaining the mean and standard deviation of the output close to zero and one, respectively, avoiding the complications in training that arise from the covariate shifts between subsequent layers, therefore reducing the network's computational requirements. We then pass the output through the leaky ReLU layer. We use the leaky ReLU instead of the standard ReLU to address the dying ReLU problem, i.e., the standard ReLU outputs zero for any input less than or equal to zero, leading to inactive training neurons. Leaky ReLU eliminates this issue by introducing a small slope for negative values, thereby allowing backpropagation through those neurons, aiding in more efficient training.

In order to increase the spatial dimensions of the input noise vector to an RGB image, we have to up-sample feature maps. Hence, we pass the latent space vector through multiple Conv2D transpose layers in phases. The first conv2D Transpose layer consists of 128 filters and is of kernel size 5x5 with a stride of 1; in this case, we keep the original width and height of the output while changing the depth of the output to 128, creating an output of dimensions 25x25x128. The batch normalization and leaky ReLU immediately follow this layer to stabilize our output for the next upsampling phase.

The next phase of implementing Conv2D Transpose with

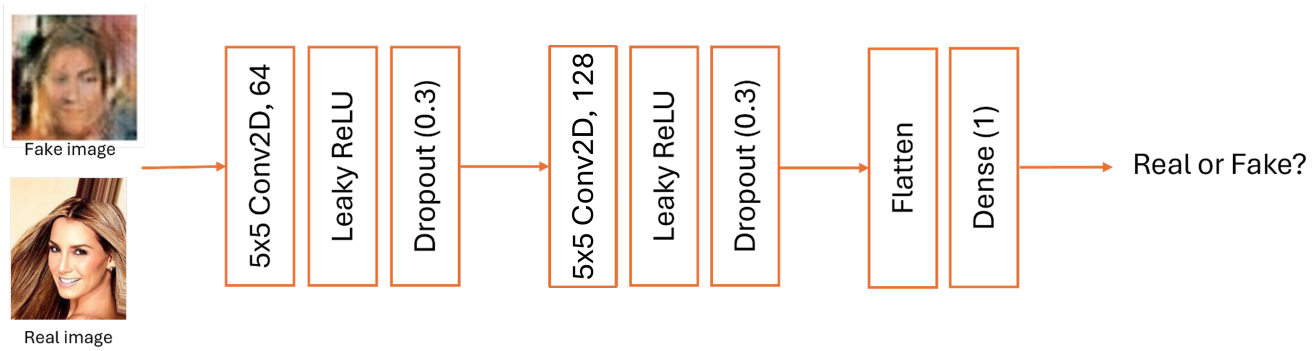

Fig. 2. Discriminator architecture

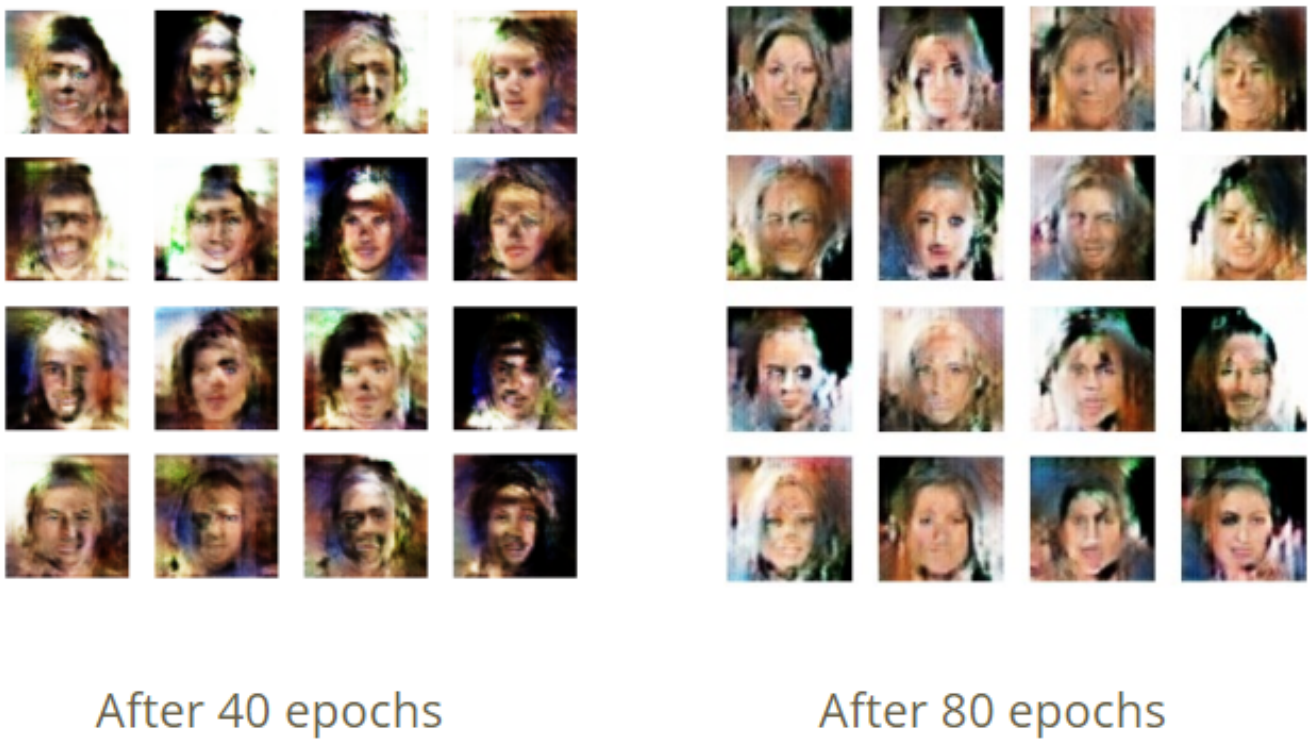

Fig. 3. GAN Generated Images

a stride of 2, a kernel size of 5x5, and consisting of 64 filters, is done to increase the width and height of the output while decreasing its depth, giving an output of dimensions 50x50x64. This is once again followed by batch normalization and leaky ReLU. For the final phase, we implement the Conv2D Transpose of 3 filters, 5x5 kernel size, and stride of 2, once again doubling the width and height of the output and reducing the depth. We finally arrive at the required RGB image of 3 channels and 100x100 width and height, respectively.

C. Discriminator Architecture

The model architecture of the discriminator is as shown in Figure 2. The role of the discriminator is to take in an input image and classify it as real or fake. The input image is passed through the Conv2D layer which performs a convolution operation of the input image, with 64 kernels of size 5x5 and a stride of 2. The input image is padded in such a way that the height and width of the output feature maps match that of the input The output is then passed through a leaky ReLU layer to introduce non-linearity. This is followed by a dropout layer of 30%. This dropout layer randomly sets 30% of input units to zero during training to overcome the problem of overfitting and improve the generalizability of the model. Further, the dropout layer also helps in balancing the performance of the discriminator so that it does not begin to outperform the generator.

In the next phase, we implement another Conv2D layer similar to the previous convolutional layer. The second Conv2D layer has 128 filters of size 5x5 and a stride of 2. Similar to the first convolutional layer, the second convolutional layer is also followed by a Leaky ReLU layer and a 30% dropout layer.

In the final phase, the output of the previous layer is sent to a flatten layer so that the multidimensional input is converted to a one-dimensional tensor. This one-dimensional tensor is then passed through a dense layer which will perform binary classification to produce a single value output. The dense layer will output 1 for real images and 0 for fake images. Overall, the discriminator acts as an image classifier.

D. Optimization and Loss function

To optimize the training process for both the generator and discriminator, the Adam optimizer with a learning rate of 0.0001 is used. The Adam optimizer can adjust the learning rates based on the performance of the model during training. Using Adam for DCGANs can lead to more stable performance as the adaptive learning rate reduces the degradation of model performance due to suboptimal choices. This leads to the generation of more realistic images.

The loss function of the generator indicates the quality of the image generated and how well it can fool the discriminator. If the discriminator classifies the fake image as real, then the performance of the generator is good. The loss function of the discriminator indicates how accurately the discriminator

can classify real images as real and fake images as fake. Binary cross entropy loss is used for both the generator and discriminator loss functions since it is essentially a binary classification task. The binary cross entropy loss for DCGANs is given by equation 1:

$$\sum_{i=1}^{N}[y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))] \quad (1)$$

where N is the number of inputs, y_i is ground truth of the i^{th} input and $p(y_i)$ is the predicted probability of the i^{th} input.

The binary cross entropy loss measures how different the predicted probability of a label is from the actual label of the input image. In DCGANs, the binary cross entropy loss measures the difference between the output of the discriminator and the required output by the generator or discriminator.

E. DCGAN Architecture

The overall DCGAN architecture as shown in figure 4 consists of a generator and a discriminator network. The generator network takes in a random input noise and generates a fake image. The discriminator network is fed either a real image or a generated image and is tasked with classifying the input image as real or fake.

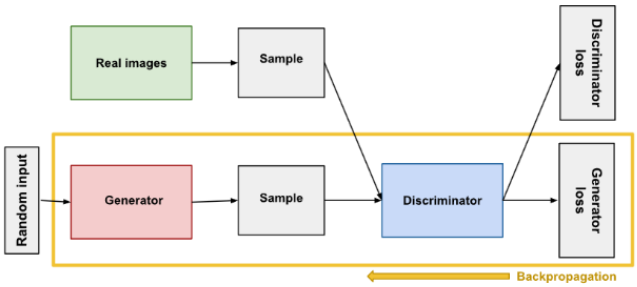

Fig. 4. DCGAN Architecture [3]

IV. Results and Discussion

The discriminator and generator were trained, while updating the losses in each case for backpropagation. Figure 5 shows the training loss achieved after 80 epochs. Initially, the generator loss is higher and the discriminator loss is lower since the generator is unable to generate convincing fake images to fool the discriminator. As a result, the discriminator can distinguish between real and fake images successfully. However, as the number of epochs increases, the generator loss decreases as it is well-trained to generate convincing fake images. This results in an increase in the discriminator loss. Further, there is a spike in discriminator loss midway through training, however, we were not able to ascertain the reason for this spike.

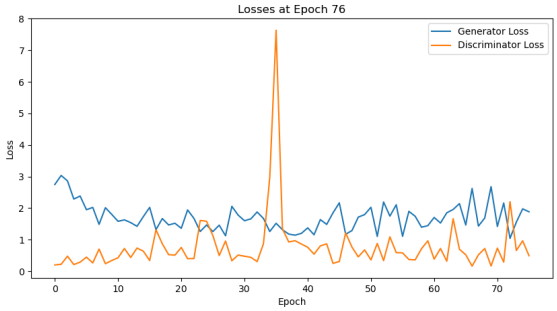

Fig. 5. GAN Training Loss

We were able to generate fake images as shown in figure 3. The image on the left shows the generated images after 40 epochs while the one on the right shows those after 80 epochs. As visible in the figure, while there is an improvement in the image quality upon visual inspection, more epochs of training are needed to generate better facial images.

The major constraint faced during the implementation of this project is the low computational power available to us. With more resources and time, this project can be expanded to achieve better results.

V. Conclusion and Future Work

With this project, we were able to achieve relatively low loss with fewer epochs and were able to generate fake images of faces. However, with low computational resources available at our disposal, the images created were not up to the standard. These results can be further improved by allowing the training process more epochs while using a more complex generator and discriminator network. For the purpose of this project, we only used 60,000 images for training. with more computational resources, a wider database can be utilized to further improve the results and get better fake images.

References

[1] Sukarna Barua, Sarah Monazam Erfani, and James Bailey. “FCC-GAN: A fully connected and convolutional net architecture for GANs”. In: *arXiv preprint arXiv:1905.02417* (2019).

[2] Nhu-Tai Do, In-Seop Na, and Soo-Hyung Kim. “Forensics face detection from GANs using convolutional neural network”. In: *ISITC* 2018 (2018), pp. 376–379.

[3] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems* 27 (2014).

[4] Chih Chung Hsu, Yi Xiu Zhuang, and Chia Yen Lee. “Deep Fake Image Detection Based on Pairwise Learning”. In: *Applied Sciences 2020, Vol. 10, Page 370* (2020).

[5] Daniel Jiwoong Im et al. “Quantitatively evaluating GANs with divergences proposed for training”. In: *arXiv preprint arXiv:1803.01045* (2018).

[6] Ali Khodabakhsh et al. “Fake Face Detection Methods: Can They Be Generalized?” In: *2018 International Conference of the Biometrics Special Interest Group (BIOSIG)*. 2018, pp. 1–6.

[7] Ziwei Liu et al. *CelebA Dataset — mmlab.ie.cuhk.edu.hk*. https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html. [Accessed 29-02-2024]. 2021.

[8] Deepankar Nankani and Rashmi Dutta Baruah. “Investigating deep convolution conditional GANs for electrocardiogram generation”. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–8.

[9] Zhaoqing Pan et al. “Recent Progress on Generative Adversarial Networks (GANs): A Survey”. In: *IEEE Access* 7 (2019), pp. 36322–36333.

[10] Tensorflow. *Docs/site/en/tutorials/generative/dcgan.ipynb at master · tensorflow/docs*. URL: https://github.com/tensorflow/docs/blob/master/site/en/tutorials/generative/dcgan.ipynb.

[11] Hanwen Yang, Peiyu Fang, and Zhiang Hao. “A gan-based method for generating finger vein dataset”. In: *Proceedings of the 2020 3rd International Conference on Algorithms, Computing and Artificial Intelligence*. 2020, pp. 1–6.

[12] Xu Zhang, Svebor Karaman, and Shih-Fu Chang. “Detecting and Simulating Artifacts in GAN Fake Images”. In: *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*. 2019, pp. 1–6.