2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService) | 978-1-6654-3483-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/BigDataService52369.2021.00014

Deepfake Detection using GAN Discriminators

Sai Ashrith Aduwala, Manish Arigala, Shivan Desai, Heng Jerry Quan, Magdalini Eirinaki
Computer Engineering Department
San José State University (SJSU)
San José, CA, USA
Email: {saiashrith.aduwala, manish.arigala, shivan.desai, heng.j.quan, magdalini.eirinaki}@sjsu.edu

***Abstract*—Deepfake videos are videos where the features of a person are replaced with the features of another person. Videos can be manipulated using powerful Deep Learning techniques. This technology may be used maliciously as a means of misinformation, manipulation, and persuasion. There are currently not many solutions to identify products of Deepfake technology, although there is significant research being conducted to tackle this problem. One often researched deep learning technology is the Generative Adversarial Network (GAN). These networks are commonly used to generate Deepfake videos but not used for their detection. In this work, we explore solutions based on GAN discriminators as a means to detect Deepfake videos. Using MesoNet as a baseline, we train a GAN and extract the discriminator as a dedicated module to detect Deepfakes. We test several discriminator architectures using multiple datasets to explore how the efficacy of the discriminator varies with different setups and training methods. Finally, we propose a model to boost the efficacy of a group of GAN discriminators using ensemble methods. Our results show that GAN discriminators, even augmented by ensemble methods, do not perform well on videos from unknown sources.**

***Index Terms*—Deepfake Detection, Generative Adversarial Networks, GAN, Discriminator, Deepfake Videos, Deep Learning, Image Processing, Feature Recognition**

I. Introduction

With the advancement of Deep Learning technologies and with the introduction of Generative Adversarial Networks (GANs), it has become easier to generate fake videos that look real to the human eye. It is becoming increasingly difficult to detect these types of videos even with advanced computational algorithms. When released to the public, Deepfake videos could spread misinformation that mislead people and have detrimental effects on public discourse [1]. For example, creating videos of fake speeches of politicians could lead to misinformation and exacerbate political or religious tensions between countries or within a country. More importantly, people may lose trust and begin to question the authenticity of any kind of information present on the internet. Additionally, these technologies could be used to manipulate stock markets or even election results that could lead to chaos in society [1].

Nonetheless, Deepfake videos do have a few advantages, like filling up certain sections of movies that are left behind by lost actors [2] or creating fun applications like FaceApp [3] where users can visualize their faces in different forms of age or gender. Nevertheless, the negatives of this technology outweigh the positives. The advancement of this technology along with increasing availability of data has made it very easy to produce convincing fake videos. As a result, it is not only public figures who are targeted–common people can be targeted as well. Hence, it has become very important to address the problem of accurately identifying such Deepfake videos.

Currently, there are no solutions that perfectly identify whether a video has been manipulated or not. Some websites like deepware.ai [4] try to provide a similar service, but these websites may crash or fail when a video is uploaded that is either too long or when a generic video is uploaded.

We aim to build a framework with which to check the authenticity of a given video. Generative Adversarial Networks (GANs) are commonly used to generate Deepfake videos, but as far as we know, there are no detection solutions based on GANs. In this work, we evaluate the GAN discriminator's potential to be used as a Deepfake Detector. We explore the impact of data quality and training methods on the GAN discriminator and analyze the drawbacks and limitations of using the discriminator as a Deepfake Detector. Finally, we suggest some methods that can be utilized to improve the performance of these discriminators.

The rest of the paper is organized as follows: we review related work in the area of Deepfake detection in Section II, provide some background on the framework design in Section III, and details about the proposed architecture in Section IV. We then present our evaluation methodology using multiple input datasets and GAN-based workflows in Section V and conclude with our experimental findings and discussion on challenges, shortcomings of current state-of-the-art and possible improvements in Section VI.

II. Related Work

Deepfake video detection is a difficult problem to tackle, with many proposed solutions that have their own strengths and weaknesses. Some of these methods are summarized in a comprehensive study on the current state of Deepfake detection by Nguyen et al. [5]. The authors detail the many conventional machine learning methods that have been developed to combat Deepfaking, including a variety of recurrent, convolutional, and artificial neural networks trained on behaviors like eye blinking, facial texture, and other extracted features.

Currently, the efficacy of any one classifier depends highly on the kind of data used to train the model. Nguyen et al [5] briefly touched on Deepfake creation, noting that Deepfakes originated — and still specialize — in facial swapping using deep learning techniques. As these forgeries became more widespread, increasingly more data has been released as a basis for developing discriminators that could detect if a video has been compromised. According to Dolhansky et al [6], the first generation of these datasets includes DF-TIMIT [7], UADFV [8], and FaceForensics++ DF(FF++DF) [9] and the second generation contains datasets like the Google Deepfake Detection Dataset [10], Celeb-DF [11] and DFDC Preview Dataset [6]. Dolhansky et al. proposed a third generational dataset with videos that have more frames than previous generations. More importantly, the individuals shown in these videos have given explicit consent for these videos to be used to create Deepfakes [6]. The resulting dataset contains $119,154$ ten-second training videos, $4,000$ ten-second validation videos and $10,000$ ten-second test videos. Other datasets, like the one proposed by Rössler et al. [12], contain videos of different quality levels from low quality (LQ) to high quality (HQ), to simulate real world scenarios. Other popular methods of creating Deepfakes are with Auto-encoders or Generative Adversarial Networks (GANs) [13].

Once the dataset has been selected (or generated), there are a variety of methods that have been applied to identify Deepfake videos. Nguyen et al. point out that it is important to note the differences between Deepfake images and Deepfake videos. Unlike images, which are purely visual, videos also vary in time, adding possible temporal features to the analysis [5]. For both media, one approach is to treat the problem like the binary classification of a static image. Rössler et al. compared some methods implemented with this principle in mind to human subjects. They found that integration of domain-specific information improves the overall performance of the detector, allowing them to outperform the human benchmark [12]. Some indicators of tampering include lack of skin detail or difference in skin tone, blurring near edges, and face warping [13]. A common method of detecting these faults is through the use of convolutional neural networks (CNNs), sometimes with added features such as attached recurrent neural networks (RNNs) or extra layers.

Deepfake videos, in addition to visual elements, also contain a temporal element. This allows for a more thorough analysis of temporal features across multiple video frames. Nguyen et al. note that temporal coherence is not strictly enforced in many generated Deepfake videos, resulting in inconsistencies that may be exploited to identify potential fabrications. Some models use long short term memory (LSTM) coupled with CNNs to search for these temporal inconsistencies [5].

More recently, GAN-based methods are becoming increasingly popular as a method of detecting Deepfake videos. A GAN, or Generative Adversarial Network, consists of (at least) one generator and (at least) one discriminator. The generator takes in a noise signal and then aims to produce images or videos (content) that are as similar as possible to the set of real content. The discriminator takes in a dataset consisting of both real content and the fake content generated by the generator and aims to identify the fake content. Because of their opposing goals, the two networks are functionally adversarial. As one network improves its performance, the other receives better feedback with which to improve its own. Some variations expand on the types and number of networks involved, two examples being Deep Convolutional GANs (DCGANs) and Cycle-GAN [14]. In a DCGAN, both the generator and discriminator networks use convolutional neural networks (CNNs), each consisting of multiple layers. A Cycle-GAN stacks two DCGANs and cycles generated images between the two DCGANs. More complex GANs have also been made, like Nvidia's StyleGAN [15]. StyleGAN is highly complex, which allows it to generate very convincing Deepfake images, some examples of which are shown in Figure 1. At the present time, GANs are much more popular for Deepfake generation, with little to no sources detailing how a GAN discriminator does against Deepfakes not created by its generator. We evaluate the efficacy of a basic GAN discriminator and see if any possible improvements can be made.

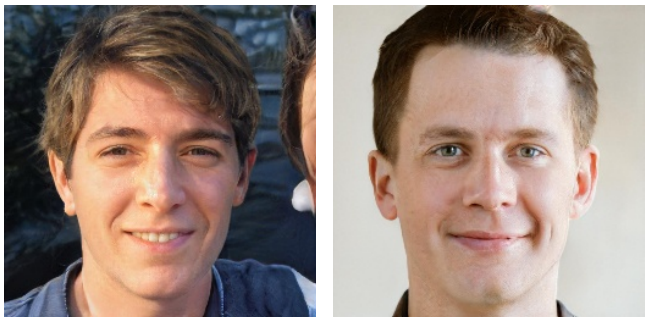

Fig. 1: Deepfake faces generated by StyleGAN.

Aside from traditional methods and GANs, some researchers have proposed less conventional ways of tackling the Deepfake detection problem. Researchers point out that Deepfake creation and detection evolve much like the two networks of a GAN. Both Deepfake generators and detectors coevolve, to the point where it becomes very difficult to identify forgeries. One example of this coevolution is blink rate analysis. During the incipiency of Deepfake technologies, generators were not trained with blinking taken into account, and detectors used this inconsistency to pick out fake videos. In response, Deepfake generators started to train with blinking data in the input, with sidelining blink rate act as a good indicator of forgery. Some have proposed trust models instead with blockchain technologies [16]. Others have tried to frame the question of Deepfake detection problem as a statistical hypothesis test [17]. Some of these unconventional paths may be worth exploring, with many solutions possibly yet undiscovered.

III. Methods / System Design

A. MesoNet Baseline Model

MesoNet [18] is one of the state-of-the-art facial forgery detection technology and is a combination of two independent architectures: Meso-4 and MesoInception-4. Meso-4 is a complex network with an initial four layers of successive convolutions and followed by a dense network with one hidden layer. The convolutional layers use ReLU activation functions to regularize the output by introducing non-linearities and normalization. The fully connected layers use Dropout for regularization. To test the performance of both the MesoNet architectures, we used the following weights to train and evaluate the architectures: Meso_DF [19] is the weight trained using the Meso-4 architecture on the DFDC dataset [6] and Meso_F2F [19] is the weight trained on the Face2Face dataset [20]; MesoInception_DF is the weight trained using the MesoInception-4 [19] architecture on the DFDC dataset [6] and MesoInception_F2F [19] is the weight trained on the Face2Face dataset [20]. The dataset used in testing is a subset of Deepfake Detection Dataset(DFDC) [6]. The main performance feature used in testing was the similarity score. After running multiple tests on the local machine, an average score of 92.3% was achieved for both the architectures which coincides with the author's average classification score of 90% [19].

B. Technologies

A prototype of our framework was tested in Keras, and the model was later migrated to PyTorch. While Keras is more convenient for developing simple models, PyTorch gives us more control over the model's parameters and architecture.

IV. Project Architecture

A. GAN Overview

The GAN architecture has basically two components: the generator and the discriminator. The role of the generator is to generate images from random noise and the discriminator determines whether the image is generated or authentic. The goal of the generator is to fool the discriminator by increasing the classification error and the goal of the discriminator is to minimize classification error. These components play adversarial roles, hence this kind architecture is referred to as a Generative Adversarial Network (GAN). Both the Generator and the Discriminator are Convolutional Neural Networks (CNNs).

B. Generator

The generator is a complete Deep Convolutional Neural Network. It has a total of five layers joined sequentially. The input layer is a Transposed Convolutional Layer with 512 filters, stride 1 and 0 padding. This layer takes a random noise of 100 channels, 1x1 pixels each (i.e. noise of size $[batch\_size, 100, 1, 1]$ as input. The subsequent layers are also all Transposed Convolutional Layers. The second layer has 256 filters, the third has 128, the fourth has 64 and the final layer has 3 filters. All these layers have stride 2, padding 1 and a kernel size of 4. The Transposed Convolutional Layers are each followed by a Batch Normalization Layer and use ReLU activation function. The output layer uses Tanh activation and gives a generated fake image of size $[3, 64, 64]$ as output. This output is fed to the discriminator for detection and based on the response from the discriminator the generator is trained and vice versa. The architecture is depicted in Figure 2.

C. Discriminator

Similar to the generator, the discriminator for the GAN is a complete Deep Convolutional Neural Network (DCNN). It starts with a convolutional layer that also acts as the input layer. This has 8 filters and kernel size of 3, with a ReLU activation. This layer is supported with a max pooling layer. The second convolutional layer also has 8 filters and has a kernel size of 5. The third and the fourth convolutional layer both have 16 filters and kernel size 5. Each of these layers have a batch normalization and a max pooling layer attached to them and use leakyReLU activation. The convolutional layers are followed by a reshape layer and a fully connected layer. This fully connected layer has 256 nodes and uses Sigmoid activation. This network outputs the probability of the frame or an image being a Deepfake. Binary cross entropy loss is used as the loss function and Adam as the optimizer with a learning rate of 0.0002 and beta1 and beta2 having vales 0.5 and 0.999. The architecture is depicted in Figure 2.

D. Selecting a Discriminator Model

To select the most suitable model, five different discriminator architectures were built and tested. Table I shows the comparison of the tested models. All the tested models were Convolutional Neural Networks. The major difference between these architectures and resulting models is the number of convolutional layers and the parameter values of these individual layers. We have also experimented with different activation and loss functions. The model with the least final loss (Architecture #5) was selected as the base architecture and the final model was built upon it. We have also tested these models to include an LSTM, but later determined that it does not contribute to the improvement of the models efficiency.

V. Evaluation Methodology

A. Datasets

Our main dataset is the Deepfake Detection Challenge (DFDC) dataset [6]. The DFDC dataset contains a collection of clips taken from a library of videos. We used a subset of the DFDC dataset totaling 400 10-second videos. Approximately two-thirds of the videos are Deepfake videos, while the remaining third are real videos. In addition to this, we used two other datasets of real faces, the Celeb-A Dataset [21] containing $203,000$ real images of celebrities and the 70k real faces dataset [22], to supplement our GAN. Since the generator will be learning to create fake images, we need only to provide a dataset of real faces, which the generator will attempt to emulate. Lastly, we used the 140k Real and Fake Faces dataset [23], which is generated by Nvidia's StyleGAN [15]. This dataset contains equal parts real and fake faces, and is the superset of the 70k Real Faces Dataset. Of all the datasets mentioned, only the DFDC dataset requires preprocessing, discussed in detail in Section V-C.

B. Workflow

The project uses four datasets of which the 70k Real Faces dataset is a subset of the StyleGAN dataset. Of the four,

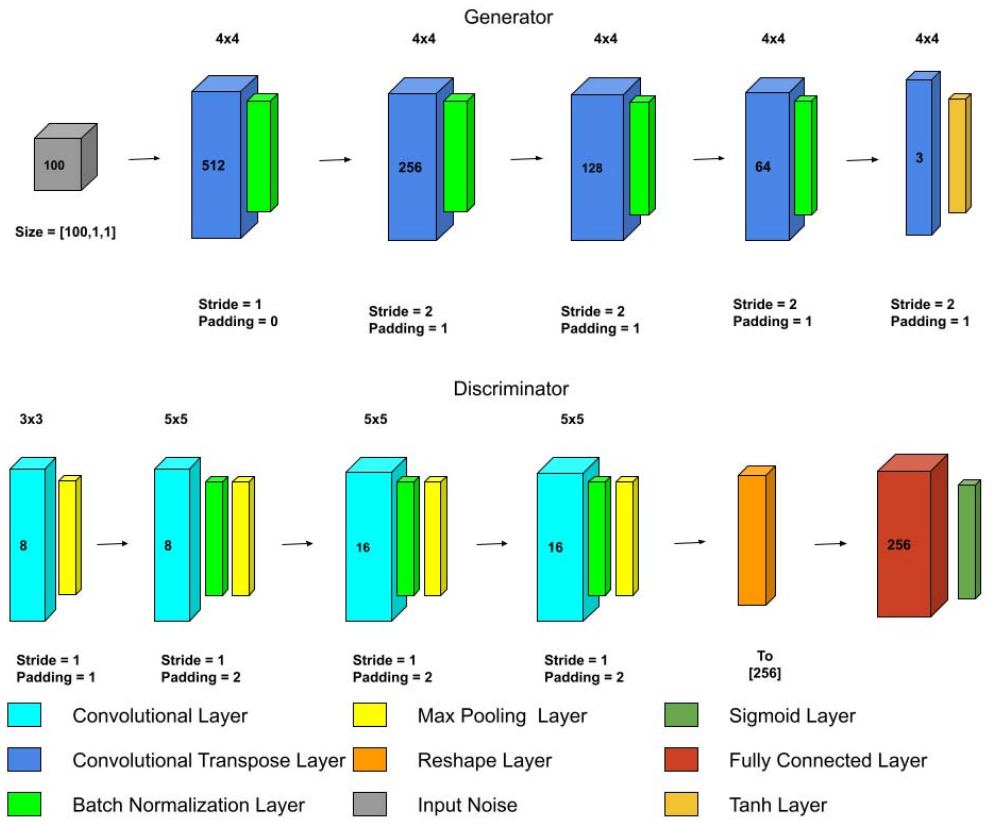

Fig. 2: Generator Architecture (top) and Discriminator Architecture (bottom)

Architecture	# of layers	Kernel Size	Activation Function	Loss Function	Final Loss
1	8	2,3,5	ReLU	Binary Cross Entropy	0.55
2	8	4	ReLU	Binary Cross Entropy	0.59
3	10	3,7	ReLU	Binary Cross Entropy	0.57
4	10	3,4	ReLU and LeakyReLU	Cross Entropy	0.72
5	7	3,5	ReLU and LeakyReLU	Cross Entropy	0.51

TABLE I: Comparison of the Tested Discriminator Models

the DFDC dataset consists of videos, whereas the other three datasets are images. To convert the videos into images, preprocessing was performed on the DFDC dataset. We trained three different models all of which were combined into an ensemble method. The first model was a pre-trained discriminator that was further trained using a GAN. The discriminator was first pre-trained on the DFDC dataset and then trained in a GAN using a combination of the Celeb-A and 70k Real Faces datasets. The second model was first trained using a GAN and then post-trained on the DFDC dataset. Similar to the previous model, the GAN training used a combination of the Celeb-A and 70k Real Faces datasets. The third model was a GAN classifier that was trained on the StyleGAN dataset. These three models were then used in an ensemble method. The results from the three were recorded and an SVM was trained on them to classify the video to be a Deepfake or not. The ensemble method is hosted on a cloud server. The users can upload their video, which is first preprocessed to extract the frames before these frames are sent to the ensemble method that returns the probability of the video being a Deepfake.The workflow diagram is shown in Figure 3.

C. Data Preprocessing

The DFDC dataset [6] contains both real and Deepfake videos cut to be 10 seconds long. The subset of data we used contains 400 such videos at 30 frames per second for a total of $12,000$ frames. First, each input video is separated into frames using OpenCV in Python. These frames had a resolution of 1920 x 1080 and contained a lot of extraneous information since our analysis was focused on faces. Using the default pre-trained Haar Cascade facial detection algorithm from OpenCV [24], we tried to reduce every frame to a number of 400 x 400 pixel areas, each containing a face. Unfortunately, the optimal parameters for each video varied significantly.

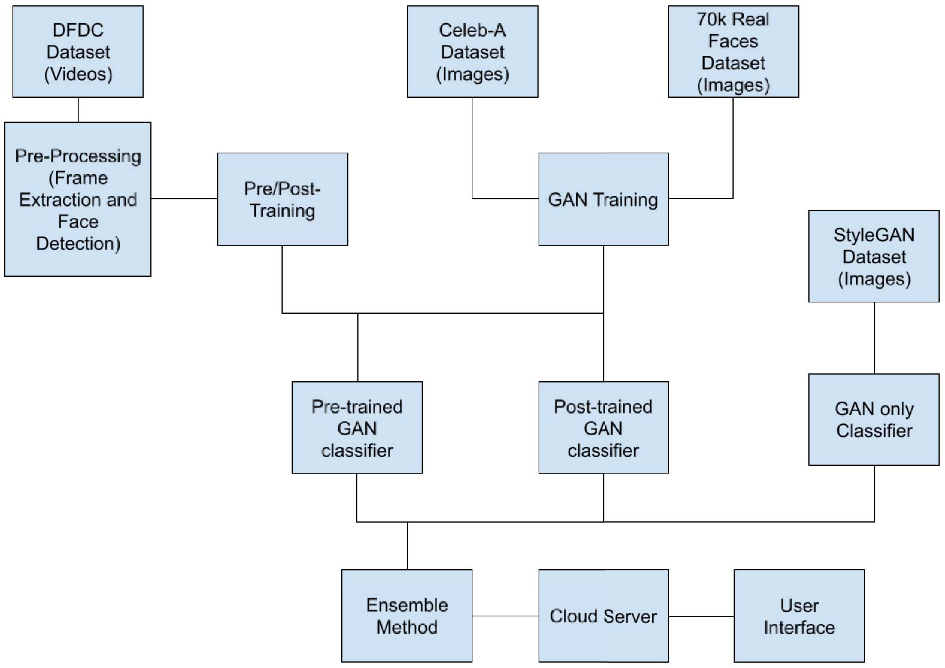
DFDC Dataset (Videos)
Celeb-A Dataset (Images)
70k Real Faces Dataset (Images)
Pre-Processing (Frame Extraction and Face Detection)
Pre/Post-Training
GAN Training
StyleGAN Dataset (Images)
Pre-trained GAN classifier
Post-trained GAN classifier
GAN only Classifier
Ensemble Method
Cloud Server
User Interface

Fig. 3: Workflow Diagram

To get around this, we used the structural similarity index measure (SSIM) [25] to pick out similar faces between frames, eliminating much of the noise resulting from the Haar Cascade detector. Additionally, we sampled each video at half-second intervals to reduce the number of frames that needed to be processed. Much of this process has been done manually (e.g. cleaning the final dataset), and this will be automated for the final production. Nevertheless, there are some performance drawbacks. One issue is that the script requires a user to manually select faces they want to track on the first frame. If the first frame has no faces, the preprocessing script will have some difficulty processing the video. This isn't an issue on the DFDC dataset, but it will affect future deployments of our model, thus we plan to address it in future iterations of the framework. The final dataset consists of about 3000 labeled frames with a $2:1$ ratio between Deepfakes and real frames, a sample of which is shown in Figure 4. This dataset is small because it is only used for the discriminator's pre-training. When the generator is added, the network will start to generate its own Deepfakes and will only require real faces, which we source from other parts of the DFDC dataset or from alternate datasets like the Celeb-A [21] and 70k [22] datasets.

D. Network Parameters and Hyperparameters

Given our current architecture was mostly inspired by Mesonet [18], many of our parameters were inherited from their model. In other areas, like those in the convolutional layers, parameters like the kernel size, stride, and padding were calculated to have a certain effect on the dimensions of the input. Typically, they are calculated to either preserve the current dimensions or to scale them down by some factor.

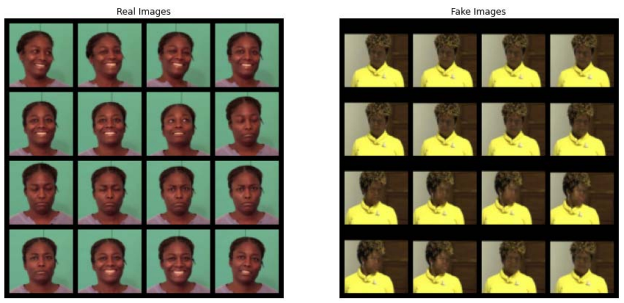

Fig. 4: Preprocessed Faces from real (left) and fake (right) videos.

Every time the input dimensions are scaled down, the number of channels is scaled up.

We had to adjust some other hyperparameters to fine tune the network. Examples include the learning rate and learning rate scheduler. We initially started with too high of a learning rate, which prevented the model from improving. Our GAN still requires some learning rate tuning as the discriminator and generator influence each other.

E. Training and Testing

Training consists of mainly two parts: pre-training the discriminator and alternately training the discriminator and generator. To pre-train the discriminator, we split up the dataset with a $4:1$ ratio used for training and testing respectively. For the validation set, the ratio of Deepfakes to real faces is $2:1$. We trained our discriminator for 50 epochs, resulting in a validation accuracy of 78%. At this point we tried to train the discriminator with the generator together in the GAN, but it gave poor results, so more pre-training may be applied before the discriminator is jointly used with the generator again.

To train the GAN as a whole, the discriminator only requires a data set consisting of real faces and a validation set to see how the discriminator's accuracy changes. The training process then consists of the following steps:

1) Generate an image by feeding random noise into the generator
2) Use the generator's output as the discriminator's input
3) Compute the generator's loss using the discriminator's output and the "real" label
4) Backpropagate through the generator and adjust parameters

Steps 1, 2, and 4 are fairly straightforward. For step 3, we calculate the loss using the discriminator's output and a set of "real" labels because we want the generator to produce images that the discriminator thinks are real. To this end, the generator loss would be the difference between the discriminator's output on a generated image and a

The discriminator training consists of processing one batch of real face data and one batch of generated data with their respective labels. The loss for each batch is then computed and averaged. The optimizer then back propagates through the discriminator.

We are training the GAN for 10 epochs at a time with a learning rate of 0.001 for the generator and 0.00001 for the discriminator. Additionally, we have also tried to add in a learning rate scheduler to decrease the discriminator learning rate as training progresses.

F. Ensemble Method

To decide on an ensemble method, we tried different aggregation methods and chose one method for the classification of the entire input video. The weights from the Pre-trained GAN classifier, Post-trained GAN classifier, and GAN only classifier (as outlined in Section V-B and depicted in Figure 3) were collected and stored in a list. We then created some discriminator objects and loaded in the weights. The DFDC data is passed through these three discriminators, and the results are collected in a Pandas Dataframe. Next, we compared Logistic Regression and Support Vector Machines (SVM) models and tested the performance of each model on metrics like accuracy, precision, and recall. SVM performed much better than the Logistic Regression model on every metric, and hence we chose SVM as the classifier for our architecture.

For the final phase, we used ensemble methods to create an overall label given the predictions from three different versions of the discriminator. The performance results of the discriminator when tested on the DFDC dataset are as follows. The discriminator had an overall accuracy of 0.763. However it had precision of 0.811, recall of 0.919 and F1 score of 0.546 when detecting Deepfake videos, and precision of 0.356, recall of 0.123 and F1 score of 0.183 for detecting real videos. On observation, the discriminator has a much better precision, recall and F1 scores for detecting Deepfakes than for detecting real videos. To address this, we split up each video into smaller batches. Previously, we would treat each video as one batch. Instead, we group every 4 frames (2 seconds of video) into one batch and aggregate the results over multiple batches for each video. This lowered the accuracy, precision, recall, and F1 scores of Deepfake detection to 0.73 (prev 0.763), 0.83 (prev 0.811), 0.75 (prev 0.919), and 0.78 (prev 0.546) respectively. Despite the small loss, we were able to improve the performance on real videos considerably, with final precision, recall, and F1 scores of 0.58 (prev 0.356), 0.70 (prev 0.123), and 0.63 (prev 0.183) respectively.

VI. Discussion

A. Results

For our project, we tested a variety of discriminators. All the discriminators have the same architecture, but are trained on different data sets. For each discriminator, we tested its accuracy using a test set from the dataset that the particular discriminator was trained on, as well as a test set from the DFDC dataset. The pre-trained and post-trained models have seen a subset of the DFDC data (about $3,000$ images) during pre-training and post-training, while the remaining models have not seen DFDC data at all.

The following is a description of each discriminator.

- *Celeb-A only*: Trained only on Celeb-A dataset
- *70k only*: Trained only on 70k dataset
- *Mixed*: Trained on both Celeb-A and 70k datasets, with epochs alternating between the two
- *Pre-trained*: Discriminator trained 100 epochs on DFDC before being trained on both Celeb-A and 70k datasets
- *Post-trained*: Discriminator trained on both Celeb-A and 70k datasets, then post-trained for 50 epochs on DFDC dataset
- *StyleGAN*: Trained only on StyleGAN Dataset

Results for each discriminator are summarized in Tables II-V
.

	Accuracy on Dataset			
Discriminator	Celeb-A	70k Real Faces	StyleGAN	DFDC
Celeb-A Only	0.689	0.594	-	0.413
70k Only	0.474	0.445	-	0.538
Mixed Celeb-A/70k	0.447	0.487	-	0.578
Pre-trained	0.488	0.503	-	0.442
Post-trained	0.298	0.355	-	0.697
StyleGAN	-	-	0.920	0.662

TABLE II: Accuracy of different discriminators

	Precision on Dataset			
Discriminator	Celeb-A	70k Real Faces	StyleGAN	DFDC
Celeb-A Only	0.702	0.600	-	0.610
70k Only	0.468	0.437	-	0.659
Mixed Celeb-A/70k	0.462	0.490	-	0.656
Pre-trained	0.491	0.503	-	0.754
Post-trained	0.269	0.310	-	0.806
StyleGAN	-	-	0.903	0.664

TABLE III: Precision of different discriminators

The first thing we notice is that compared to the *Celeb-A Only* and *70k Only* discriminators, the *Mixed Celeb-A/70k* did

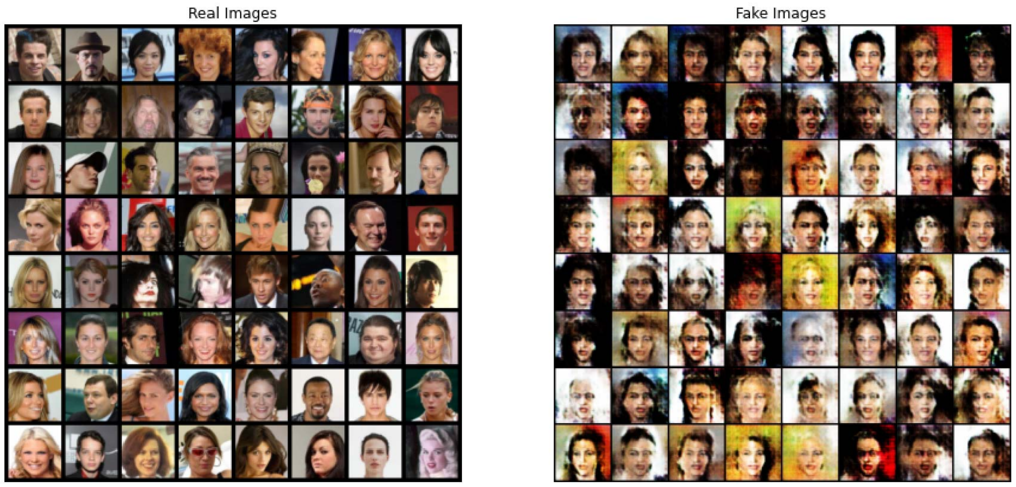

Fig. 5: Celeb-A Face Dataset Results: Real faces (left) and generated images (right).

	Recall on Dataset			
Discriminator	Celeb-A	70k Real Faces	StyleGAN	DFDC
Celeb-A Only	0.657	0.562	-	0.324
70k Only	0.382	0.383	-	0.631
Mixed Celeb-A/70k	0.643	0.642	-	0.766
Pre-trained	0.641	0.638	-	0.237
Post-trained	0.236	0.236	-	0.713
StyleGAN	-	-	0.941	0.990

TABLE IV: Recall of different discriminators

	F1 Score on Dataset			
Discriminator	Celeb-A	70k Real Faces	StyleGAN	DFDC
Celeb-A Only	0.391	0.312	-	0.205
70k Only	0.193	0.184	-	0.363
Mixed Celeb-A/70k	0.282	0.295	-	0.415
Pre-trained	0.295	0.300	-	0.179
Post-trained	0.084	0.095	-	0.456
StyleGAN	-	-	0.598	0.495

TABLE V: F1 Score of different discriminators

a bit better than both on the final DFDC dataset. Thus, when we added pre- and post-training to the model, we chose to add them to the mixed model. The pre-training model actually performed worse on the DFDC dataset than the mixed model. After post-training though, the model (unsurprisingly) does much better than both the pre-trained and mixed models.

Additionally, we noted that some of the discriminators had a hard time increasing their accuracy on the data. For each discriminator, we considered both the accuracy and loss when deciding whether or not to stop training. Training too much can lead to overfitting. For most of our models though, the loss stops changing and the accuracy no longer improves after a certain point. One exception is the discriminator trained on the StyleGAN dataset. Despite having the same architecture, the *StyleGAN*-trained discriminator achieved high levels of accuracy on both its training set and testing set. We think that the complexity of the original StyleGAN generator allows it to generate images of much higher quality, which has a great effect on the accuracy of a discriminator trained on the dataset. Other generators can generate images, but the quality will be much lower than that of StyleGAN. As an example, compare the results from the *Celeb-A* generator (Figure 5, right) to those of the *StyleGAN* generator (Figure 1).

On the contrary, we notice that even though the *StyleGAN*-trained discriminator achieves 92% accuracy on StyleGAN's test set, it did not perform significantly better on the DFDC dataset than the other discriminators. This, combined with the poor results on DFDC from other datasets, leads us to conclude that the dataset on which a discriminator is trained is highly important, and even if a discriminator does well with the dataset on which it was trained, it is difficult to generalize such a discriminator to data that comes from a different source.

One way we tried to get around this is to try performing post-training on StyleGAN. We trained the *StyleGAN*-trained discriminator for 50 epochs on the DFDC dataset. We had hoped that we can find a balance between accuracy on the StyleGAN dataset and the DFDC dataset, however, this is far from what we observed. The results are shown in Figure 6.

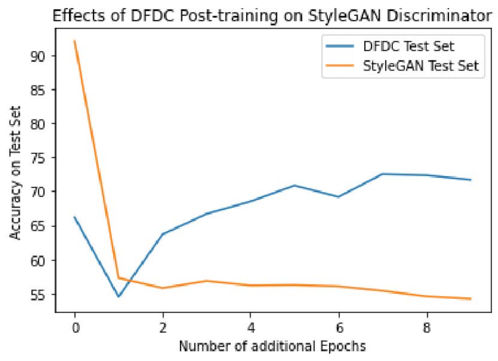

Fig. 6: Effect of post-training on StyleGAN

Despite starting with 92% accuracy on the StyleGAN dataset and 66% accuracy on DFDC, the discriminator scores much lower on both datasets after a single epoch of training. Although it does increase again after the first epoch, it doesn't do spectacularly, leading us to conclude that it would be difficult to generalize a discriminator by training it on separate datasets.

B. Conclusions

In conclusion, while GANs are a good method of training a generator, a GAN discriminator is not robust enough to be used as a general-purpose Deepfake detector. We found no evidence to suggest that a GAN discriminator performs any better than a non-GAN detector of the same architecture.

The results indicate that the discriminator accuracy on known datasets is high while accuracy on never-before seen datasets is low. This shows that GAN discriminators work best on images created by the generator it was trained with. In a real world setting, there is no guarantee that the source of the Deepfake material will be known. Thus, the GAN discriminator cannot be easily generalized to handle a wide variety of Deepfake videos, making it a poor choice for comprehensive use.

C. Implications and Future Improvements

This project has only scratched the surface of a vast body of literature, and given more time, there are a few things that we would have liked to improve on.

Part of our pipeline was limited by data preprocessing due to the limits of facial detection we had available. Since our project's focus was not on facial detection, we used the pre-trained Haar Cascade Facial Detector from the OpenCV package [24]. This only affects the model in minor ways, but it has a bigger impact on the front-end. Currently, the types of acceptable videos are limited by the quality of their initial frames and how well Haar Cascade performs on all the frames. In the future, we would like to explore alternative and more reliable facial detection algorithms.

As noted before, trying to generalize a StyleGAN discriminator does not work well when trained on separate datasets. It would be interesting to consider the effects of shuffling two datasets together. Interspersing images from different datasets may help prevent overfitting and make for a more generalized discriminator.

The datasets themselves are also interesting. It is clear that each discriminator is affected by the peculiarities of its dataset, be it the size, ratio of real to fake images, or the image quality. It would be interesting to see how each of these factors affect the training of a GAN discriminator.

Another factor is the architecture. Our architecture can be considered fairly simple compared to something like Nvidia's StyleGAN. The complexity of a model clearly has some bearing on the quality of images produced by the generator, which would affect the efficacy of the discriminator. StyleGAN, which had the most complexity, produced the best quality images and trained a high quality discriminator. Future research is needed to fine-tune a complex model that may generalize better to many datasets.

Lastly, ensemble methods seem to be a good way to incorporate many specialized discriminators. We concluded that an individual discriminator tends to be too niche to be useful in a real world setting, where the source of a Deepfake video (and the method used to create it) are uncertain. However, an ensemble method could make use of many such specialized discriminators to cover a wider variety of Deepfake sources.

References

[1] M. Masood, M. Nawaz, K. M. Malik, A. Javed, and A. Irtaza, "Deepfakes generation and detection: State-of-the-art, open challenges, countermeasures, and way forward," *CoRR*, vol. abs/2103.00484, 2021. [Online]. Available: https://arxiv.org/abs/2103.00484

[2] A. Weisman, "Fast & furious 7 is using new technology to replace paul walker with his brothers," Apr 2014. [Online]. Available: https://www.businessinsider.com/how-fast-and-furious-7-replaced-paul-walker-with-brothers-2014-4

[3] "Ai face editor." [Online]. Available: https://www.faceapp.com/

[4] "Deepware.ai." [Online]. Available: https://deepware.ai/

[5] T. T. Nguyen, C. M. Nguyen, D. T. Nguyen, D. T. Nguyen, and S. Nahavandi, "Deep learning for deepfakes creation and detection," *CoRR*, vol. abs/1909.11573, 2019. [Online]. Available: http://arxiv.org/abs/1909.11573

[6] B. Dolhansky, J. Bitton, B. Pflaum, J. Lu, R. Howes, M. Wang, and C. Canton-Ferrer, "The deepfake detection challenge dataset," *CoRR*, vol. abs/2006.07397, 2020. [Online]. Available: https://arxiv.org/abs/2006.07397

[7] P. Korshunov and S. Marcel, "Deepfakes: a new threat to face recognition? assessment and detection," *CoRR*, vol. abs/1812.08685, 2018. [Online]. Available: http://arxiv.org/abs/1812.08685

[8] X. Yang, Y. Li, and S. Lyu, "Exposing deep fakes using inconsistent head poses," *CoRR*, vol. abs/1811.00661, 2018. [Online]. Available: http://arxiv.org/abs/1811.00661

[9] J. Shao, H. Shi, Z. Fang, G. Yin, S. Chen, N. Ning, and Y. Liu, "Siyu-c/robustforensics." [Online]. Available: https://github.com/Siyu-C/RobustForensics

[10] N. Dufour and A. Gully, "Contributing data to deepfake detection research," Sep 2019. [Online]. Available: https://ai.googleblog.com/2019/09/contributing-data-to-deepfake-detection.html

[11] Y. Li, X. Yang, P. Sun, H. Qi, and S. Lyu, "Celeb-df: A large-scale challenging dataset for deepfake forensics," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. IEEE, 2020, pp. 3204–3213. [Online]. Available: https://doi.org/10.1109/CVPR42600.2020.00327

[12] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "Faceforensics: A large-scale video dataset for forgery detection in human faces," *CoRR*, vol. abs/1803.09179, 2018. [Online]. Available: http://arxiv.org/abs/1803.09179

[13] A. A. Maksutov, V. O. Morozov, A. A. Lavrenov, and A. S. Smirnov, "Methods of deepfake detection based on machine learning," in *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 2020, pp. 408–411.

[14] T. Shen, R. Liu, J. Bai, and Z. Li, ""deep fakes" using generative adversarial networks (gan)." [Online]. Available: http://noiselab.ucsd.edu/ECE228_2018/Reports/Report16.pdf

[15] NVlabs, "Nvlabs/stylegan." [Online]. Available: https://github.com/NVlabs/stylegan

[16] H. R. Hasan and K. Salah, "Combating deepfake videos using blockchain and smart contracts," *IEEE Access*, vol. 7, pp. 41 596–41 606, 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2905689

[17] S. Agarwal and L. R. Varshney, "Limits of deepfake detection: A robust estimation viewpoint," *CoRR*, vol. abs/1905.03493, 2019. [Online]. Available: http://arxiv.org/abs/1905.03493

[18] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, "Mesonet: a compact facial video forgery detection network," *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, 2018.

[19] DariusAf, "Dariusaf/mesonet." [Online]. Available: https://github.com/DariusAf/MesoNet

[20] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2face: Real-time face capture and reenactment of RGB videos," *CoRR*, vol. abs/2007.14808, 2020. [Online]. Available: https://arxiv.org/abs/2007.14808

[21] [Online]. Available: http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

[22] Xhulu, "70k real faces." [Online]. Available: https://www.kaggle.com/c/deepfake-detection-challenge/discussion/122786

[23] Xhlulu, "140k real and fake faces," Feb 2020. [Online]. Available: https://www.kaggle.com/xhlulu/140k-real-and-fake-faces

[24] Opencv, "opencv/opencv." [Online]. Available: https://github.com/opencv/opencv/tree/master/data/haarcascades

[25] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.