Deepfake Detection in Images and Videos

Vishal Asnani, Sneha Mhaske

Introduction:

Deepfake (Deep Learning + Fake) is a technique that can superimpose face images of a target person to a video or image of a source person to create a manipulated media form to perform analogous actions as the source. Deepfake is not only limited to images but can be extended to videos as well. Advances in media generation techniques have made it very easy to generate forged images and videos that humans cannot distinguish from authentic ones. Their quality too has been improving with development of advanced models and generative adversarial networks (GANs). With social media today serving as a powerful source of information, fake news with manipulated multimedia can spread quickly and can have significant impact. Deepfake can affect the quality of public discourse and the safeguarding of human source. These techniques can be a source of misinformation, harassment and manipulation. Tackling the problem of deepfake is a very demanding and rapidly involving challenge. This is the main aim of this project of detecting fake images and deepfake videos.

There are many state-of-the art methods proposed in the field of anti deep fake videos. Guera and Delp proposed in [1] a temporal-aware pipeline to detect fake videos using a combination of CNNs and RNNs. Yang et al. in [2] estimated 3D head poses from face images to detect fake videos. Li et al. in [3] proposed another approach based on detecting face warping artifacts. Rossler et al. created a new database FaceForensics++[4]. They evaluated a CNN based approach [5] and also used handcrafted steganalysis features approach [6] for fake videos detection.

Nguyen et al. in [7] proposed a CNN based system to simultaneously detect fake videos using multi-task learning. Stehouwer et al. in [8] proposed a system as a combination of CNN and attention mechanisms to process and improve the feature maps of the classifier model. Agarwal et al. proposed a detection method in [9] based on head movements and facial expression. Sabir et al. [10] proposed a method based on using the temporal information present in the stream. Afchar et al. in [11] proposed a CNN based network to observe mesoscopic properties of the images.

In [12]. The authors have used the DFDC preview dataset to provide baseline results. The dataset contained around 5000 videos. They have provided a thorough description of the

dataset including its performance on three models: CNN mode, Xception model on face image and Xception model on full images.

Table 1: Recent State-of-the-art models on Deepfake videos detection

Study	Method	Best performance	Database used
Guera and Delp [1]	CNN+RNN	Acc=97.1%	Own
Yang et al. [2]	SVM	AUC=89.0% AUC=47.3% AUC=54.8%	UADFV FF++ Celeb-DF
Li et al. [3]	CNN	AUC=97.4% AUC=79.2% AUC=53.8%	UADFV FF++ Celeb-DF
Rossler et al. [6]	CNN	AUC=94.0%	FF++
Nguyen et al. [7]	Autoencoder	AUC=65.8% AUC=76.3%	UADFV FF++
Stehouwer et al. [8]	CNN+attention mechanism	AUC=99.4% EER=3.1%	DFFD
Agarwal and farid [9]	SVM	AUC=96.3%	Own
Dolhansky et al. [12]	CNN	Precision=93.0% Recall=8.4%	DFDC Preview
Sabir et al. [10]	CNN+RNN	AUC=96.9%	FF++
Afchar et al. [11]	CNN	Acc.=98.4% AUC=84.3% Acc.=90% Acc.=53.6%	Own UADFV FF++ Celeb-DF

Similar efforts have been made towards detection of fake images. In the traditional image forgery detection approaches, two types of forensics schemes are commonly used, active schemes and passive schemes. In the active schemes, an externally additive signal (i.e., watermark) is embedded in the source image without visual artifacts. The passive image forgery detectors use the statistical information on the source image that is high consistency between different images. As a result, the intrinsic statistical information can be used to detect the fake

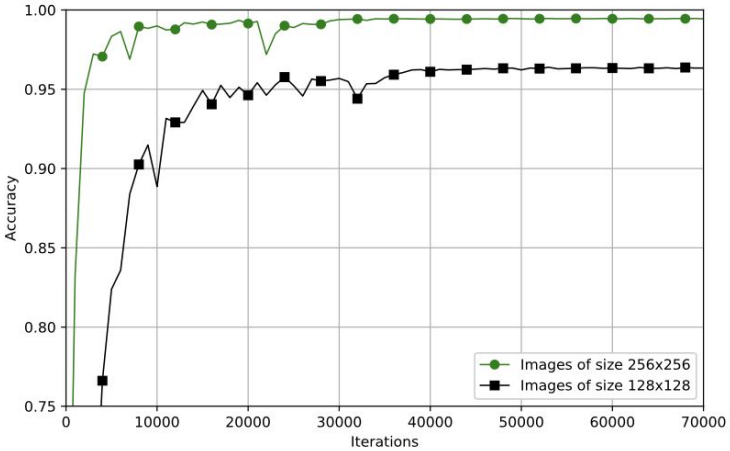
regions in the image. However, GANs synthesize images from just a low-dimensional random vector i.e. the fake images generated by the GANs are not modified from the original images and hence are difficult to detect.

Recently, the deep learning-based approach for fake image detection using supervised learning has been studied. In other words, fake image detection has been treated as a binary classification problem (i.e., fake or real image). In [14], a manipulated face detection algorithm was proposed based on a hybrid ensemble learning approach. In [15], the performance of the fake face image detection was improved by adopting the most advanced CNN–Xception network where the source images were translated to target images using CycleGAN. These methods, however, were used to detect only altered/translated images using the source image and not ones synthesized completely from random noise. In other works, the convolution neural network (CNN) network was used to develop the fake image detector [16,17]. They used the supervised learning approach to train the model on state-of-the-art GAN(PCGAN and BEGAN) generated images and hence, the classification performance depends on the training set and does not generalize well to other GANs.

Table 2: Results from [16] where CNN+Boosting was used-

Model	Classifier	Accuracy (%)	AUC
VGG16	Softmax	76	80.5
CGFace	Softmax	98	81
ADA-CGFace	**AdaBoost**	**97.3**	**89.4**
XGB-CGFace	XGB	92.6	84.2

Fig. 1 Results from [17] where CNN was used on PCGAN -



Problem Statement:

1. Detecting Fake images generated by GANs which synthesize realistic images from just a random noise vector. In GANs, we have a generator(G) and a discriminator(D) where both G and D play the following two-player minimax game with value function V (G, D):

$$min\ G\ max\ D\ V\ (D,\ G)\ =\ Ex\ _{\sim pdata}(x)\ [log\ D(x)]\ +\ Ez\ _{\sim pz}(z)\ [log(1-D(G(z)))]$$

   to the point where the generated images are indistinguishable from the training distribution and the discriminator predicts with probability P=0.5 i.e. random guessing. Fake images are generated using this adversarial network so as to generate images which are indistinguishable from the real ones fed as input. Several state-of-the-art GANs are available today which accomplish this task quite exceptionally. GANs are hard to train and converge so it becomes difficult for fake images to perfectly match the real data distribution. We can make use of this property and few semi-supervised techniques to get a better feature representation to identify fake from real. To this end, we make use of the following semi-supervised model to identify the fake ones from real. We first use a Siamese network which accepts pairwise inputs and learns the discriminative features(CFFs) of the real and fake images. We then attach a classifier on this learned model and train it using cross-entropy loss function to classify the images (1 for real and 0 for fake). In our project, we attempt to create a generalized model to identify images synthesized from several GANs so that even if one unseen GAN's images are fed to the model, it can still detect the fakes.

2. The problem of deepfake images can be extended to the problem of deepfake videos. There are several state-of-the-art techniques to tackle this problem of deepfake videos that have been proposed in recent years.

   In our project, we have tried to come up with a model capable of detecting fake videos. Due to the sequential nature of the videos, we introduce a CNN+RNN network while building our model. The sequential nature of videos gave us an idea to introduce RNN network into the model. The videos have to pre pre-processed first to extract the frames and detect face images which would then be passed into the model. The model was designed to do binary classification i.e. 1 for real and 0 for fake video.

   After training, experiments were done to see the performance of the model. Accuracy and log loss was calculated on the testing set to evaluate the performance of the model. We compared the performance of our model with all the participants of Deepfake detection challenge.

Dataset Description:

1. Deepfake Images:

We use the CelebA dataset which has around 2,00,000 images as our real image pool. For the fake image pool, since there is no existing fake image dataset available, we make use of several state-of-the-art GANs to generate fake face images.

- DCGAN (Deep Convolutional GAN)
- WGAN (Wasserstein GAN)
- WGAN-GP (Wasserstein GAN with Gradient Penalty)
- LSGAN (Least Squares GAN)
- PCGAN (Progressive GAN)

We train the above GANs available publicly to generate images for our fake image pool. Out of the five GANs, only the PCGAN has made available a pre-trained network which we used to generate the images. For the others, we have trained each model for more than a day and then generated the images. We have set the image size as 64x64 and have downsized the images generated by PCGAN to match with the other GANs.

We select 3,95,000 images(real+fake) randomly for training and 10,000 for validation. We have excluded one GAN completely from the training and used it only in the validation set. In both the sets, the number of real and fake images is equal. We also generate 10,00,000 random pairs of [(real,real),(real,fake),(fake,real) and (fake,fake)] images to be used as input for the siamese network for training.

2. Deepfake Videos:

The training dataset used for the project is provided for Deepfake detection challenge on Kaggle. AWS, Facebook, Microsoft, the Partnership on AI's Media Integrity Steering Committee, and academics have come together to build the Deepfake Detection Challenge (DFDC). The dataset contains 50 folders, each folder containing approximately 2500 videos with a metadata file giving information whether the video is fake or not. Total videos in the dataset were around 120,000.

The data is composed of .mp4 files, split into reduced sets of around 10GB individually. A metadata.json accompanies each set of .mp4 files, and contains filename, label (REAL/FAKE), original and split columns, listed below under Columns.

The model contains 50 folders of data out of which we will randomly select 45 folders as training data. Remaining 5 folders would be used as test data to find the final performance of the model.

Methodology:

1. Deepfake Images:

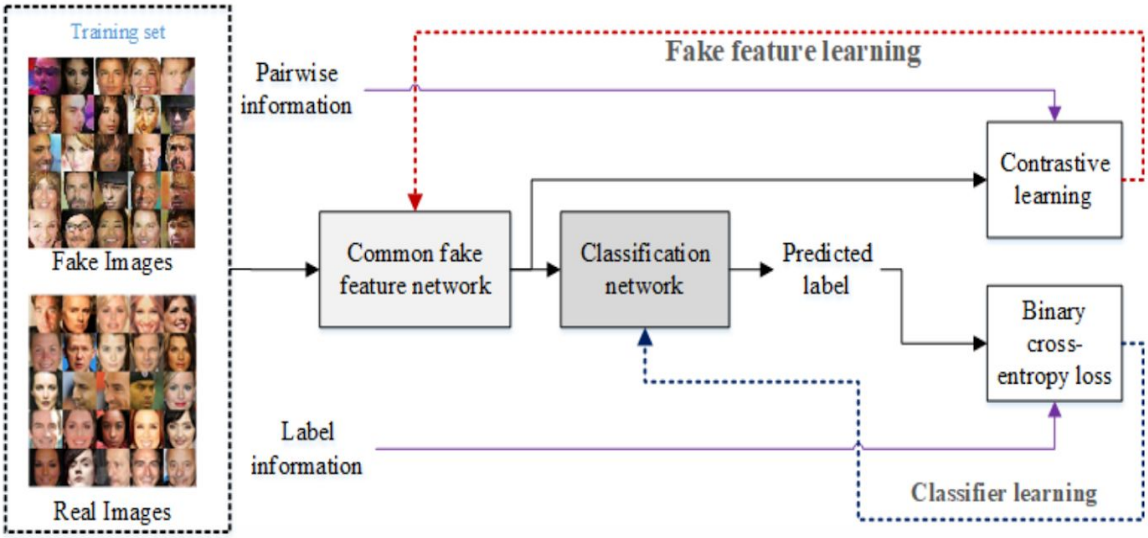


Fig. 2: Model for Fake Image Detection

In order to train the model not just on one GAN's images but to see how it fares when presented with images coming from several GANs, we use a two-step approach. In the first step, we use a siamese network to capture the discriminative Common Fake Features (CFFs). The fake and real images are paired and we use the pairwise information to construct the contrastive loss for the siamese network to learn the discriminative CFFs. Once the discriminative CFF is learned, the classification network captures the discriminative CFF to identify whether the image is real or fake and it makes use of the cross-entropy loss objective function to accomplish this task.

We generate 1 million pairs of real and fake images and label the [(real,real) and (fake,fake)] pairs as 1 and [(real,fake) and (fake,real)] pairs as 0. The siamese network then make use of the contrastive loss function:

$$L(W,\ (P,\ x1,\ x2))\ =\ 0.5 \times (yijEw^{\ 2})\ +\ (1-yij) \times max(0,\ (m-Ew))^{\ 2}$$

where

$$Ew(x1,x2)\ =\ ||fCFFN(x1)-fCFFN(x2)||^{\ 2}$$

The Energy function is the distance between the pairs and should ideally be minimum for (real-real) and (fake-fake) pairs and maximum for (real-fake) or (fake-real) pairs. Towards that end, the contrastive loss function adds a penalty if the energy function Ew is below the threshold m for (fake-real) and (real-fake) pairsor if the Ew is higher for (real-real) and (fake-fake) pairs.

By iteratively training the network using the contrastive loss, the CFFs of the collected GANs can be learned well. We use 0.5 as the margin/threshold *m.* The Siamese network uses Resnet as the backbone network. We then add a classifier network again using Resnet subnetwork on the learned model which uses cross-entropy loss function to train and detect fake images from real.

$$Lc(xi, pi) = -\sum_{1}^{N}(fCLS(fCFFN(xi))logpi)$$

Fig. 3: Details of the architecture of the models are presented in the below table →

Layers	D_1	D_2
1	Conv.layer, kernel=7*7, stride=4, channel=96	Conv. layer, kernel=3*3, channel = 2
2	Residual block *2, channel=96	Global average pooling
3	Residual block *2, channel=128	Fully connected layer, neurons=2 Softmax layer
4	Residual block *2, channel=256	
5	Fully connected layer, neurons=128 Softmax layer	

where D1 is the Siamese CFF network which is then followed by D2 which is the classifier network concatenated to the fourth layer of D1 to get an end-to-end model.

We use the first 2 epochs for training the CFF network to capture the discriminative features from the 1 million pairs that are input to it with the learning rate 1e-3. We then stop capturing CFFs by reducing contrastive loss after 2 epochs and add the classifier network to the learned model for the next 13 epochs with a reduced learning rate 1e-4. So the parameters in D1 will keep getting updated to improve the discriminative feature. Adam optimizer is used and we use a batch size of 32. The classifier gives binary output for predictions - 1 for Real and 0 for Fake.

2. Deepfake Videos:

There are five steps involved in the Deepfake video detection. These include:

1. Video preprocessing
2. Face detection
3. Building deep learning model architecture

All the steps will now be discussed in detail.

1. Video preprocessing:

   The videos in the training set first must be pre-processed to be able to pass it through the model. The duration of one video is of approximately 10 seconds with a frame rate of 30 frames per second which tells us that every video has around 300 frames. It was seen that the facial expression change can be observed in every second.Keeping in mind this observation and the computational resources, every 10^{th} frame of the video had been extracted. The main reason for extracting every 10^{th} frame is that we saw that the manipulation can be detected in the video by seeing the frame of the video every one second.

   Therefore, every video has now 30 frames extracted from it. These framesnow have to be further processed to detect faces.

2. Face Detection using MTCNN face detector:

   Recent years have shown that CNN is much better in face detection than all classic approaches. One of the most widely used models for face detection is the MTCNN detector [] which performs both face detection and face landmarking. The network is composed of three sub-networks: P-Net, R-Net and O-Net.

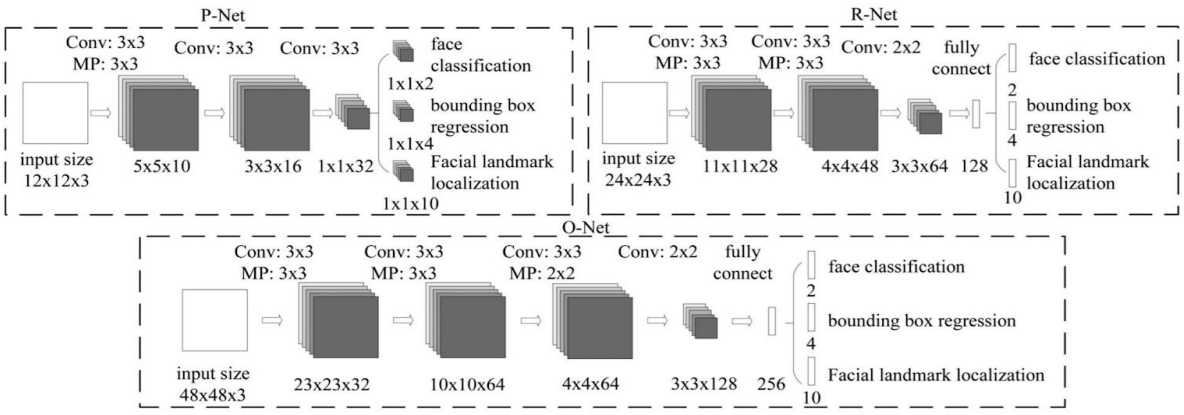


Fig. 4: Architectures of P-Net, R-Net, and O-Net

The first stage i.e. P-net does the coarse face detection producing proposal regions. R-Net refines the selected proposed regions and O-Net does the face landmarking. MTCNN is still a state-of-the-art face recognition system.

The MTCNN implementation in Pytorch[] has been used here. The MTCNN detector when given a frame as input will output a 160 x 160 size of cropped face image of frame. These frames are then stored into the memory to be later used in our Deepfake model.

3. Model Architecture:

The model we are using to tackle the problem of deepfake videos would be a CNN+RNN model. The pretrained Xception model is used for our CNN network. The Xception model is pre-trained on ImageNet data which consists of approximately 1.2 million Images for training and additional 50000 images for validation and 100,000 images for testing.

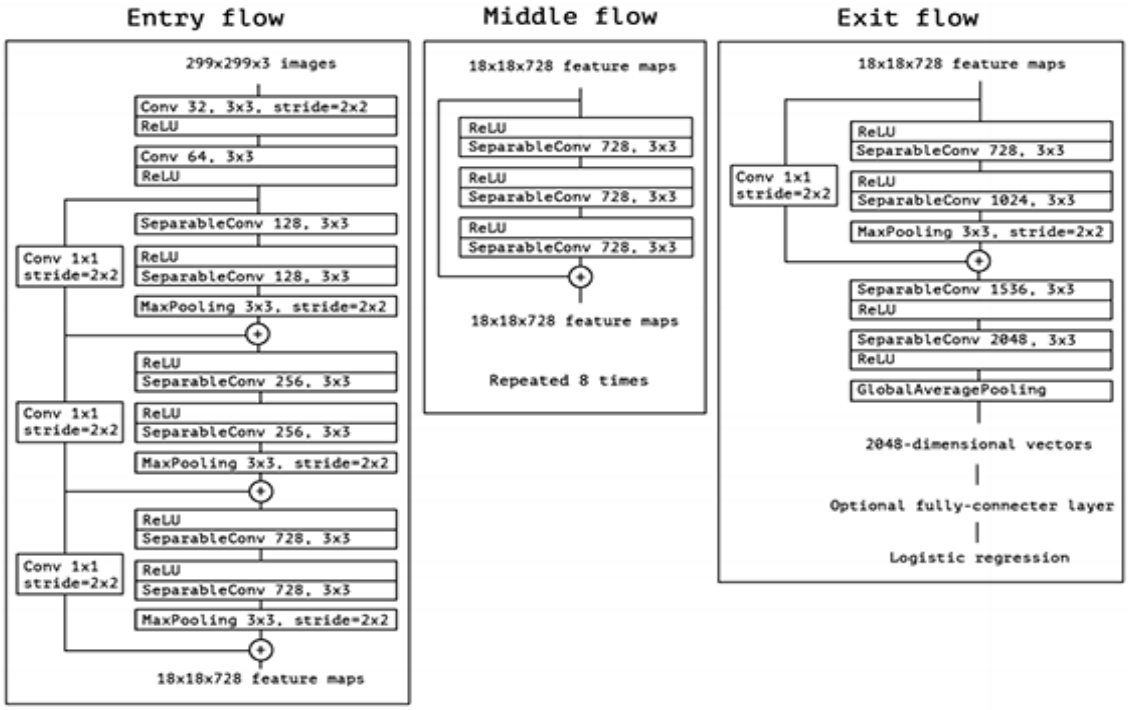


Fig. 5: Xception net architecture

Xception is an extension of the inception Architecture which replaces the standard Inception modules with depth wise Separable Convolutions. Separable convolution consists of depthwise convolution followed by pointwise convolution. Depthwise convolution is the channel-wise nxn spatial convolution. Pointwise convolution is a 1x1 convolution to change the dimensions.

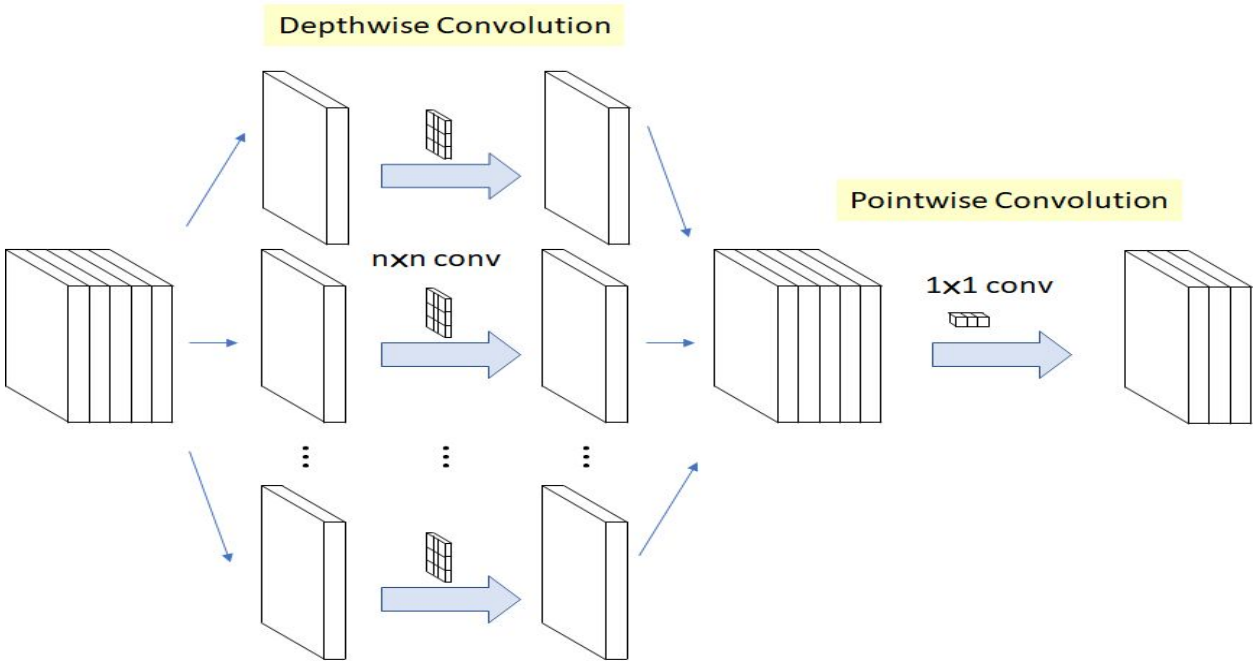


Fig. 6: Depthwise Separable Convolution

For xception model, a modified depthwise separable convolution layer is built. There are two differences between original and modified depthwise separable convolution layers. First, the modified depthwise separable convolution performs 1x1 convolution followed by channel-wise spatial convolution. Second, the introduction of nonlinearity after the first operation such as intermediate ReLU nonlinearity is removed.

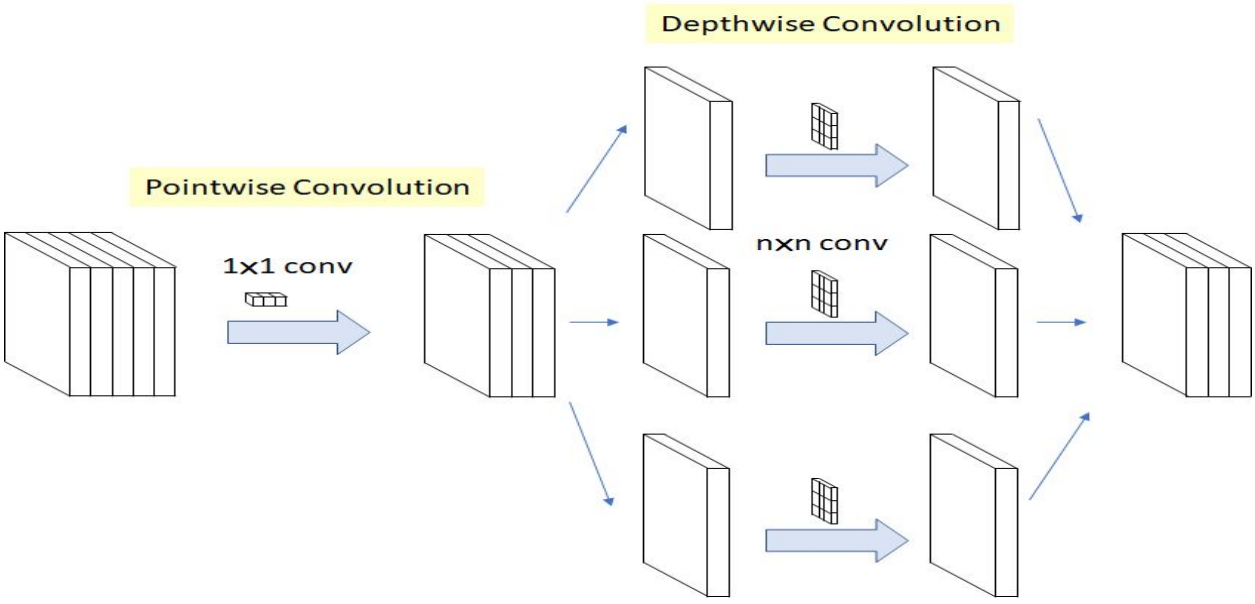


Fig 7: The Modified Depthwise Separable Convolution

The Xception model outputs a 2048 dimensional feature vector. This feature will be fed into a Recurrent neural network (RNN). RNNs are a type of artificial neural network designed to recognize patterns in sequences of data. We would be using a variant of general RNN networks known as Long Short-Term Memory units or LSTMs. LSTMs units provide a bit of memory by storing in a memory cell state so that it can remember the past. LSTMs consist of three gates namely, Input gate, Forget gate and Output gate. Below are the equation of all the three gates:

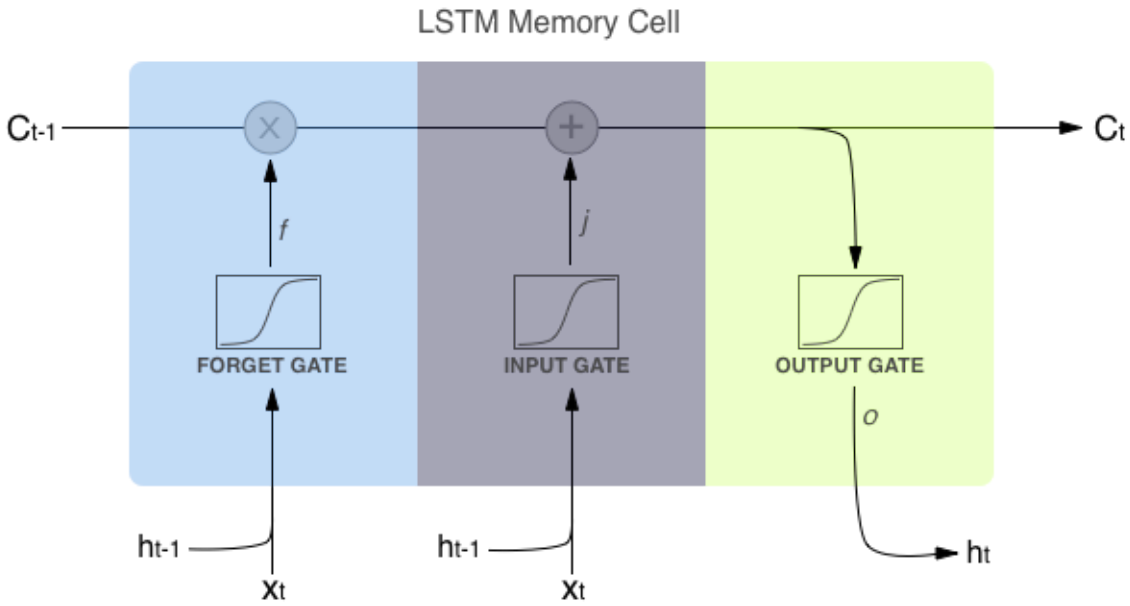


Fig. 8: Structure of the LSTM Memory cell

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i)$$

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f)$$

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o)$$

$i_t \rightarrow represents\ input\ gate.$
$f_t \rightarrow represents\ forget\ gate.$
$o_t \rightarrow represents\ output\ gate.$
$\sigma \rightarrow represents\ sigmoid\ function.$
$w_x \rightarrow weight\ for\ the\ respective\ gate(x)\ neurons.$
$h_{t-1} \rightarrow output\ of\ the\ previous\ lstm\ block(at\ timestamp\ t-1).$
$x_t \rightarrow input\ at\ current\ timestamp.$
$b_x \rightarrow biases\ for\ the\ respective\ gates(x).$

Fig. 9: LSTM gate equations

Now, till now, we explained about what neworks we are going to use in this model. Next, we will discuss the CNN and RNN network specifications.

We know that the MTCNN when given an input will output an image of size 160 x 160. These face images are then passed into the Xception model which outputs a 2048 dimension feature vector. This feature vector is then passed into the RNN network which consists of a LSTM layer followed by a fully connected layer, Relu and dropout layer. The LSTM layer has 3 hidden layers and 256 hidden nodes. 20% dropout is used for the dropout layer. Fully connected layer is used after LSTM layer and dropout layer. ReLU activation is used after a fully connected layer to introduce non linearity into the architecture.

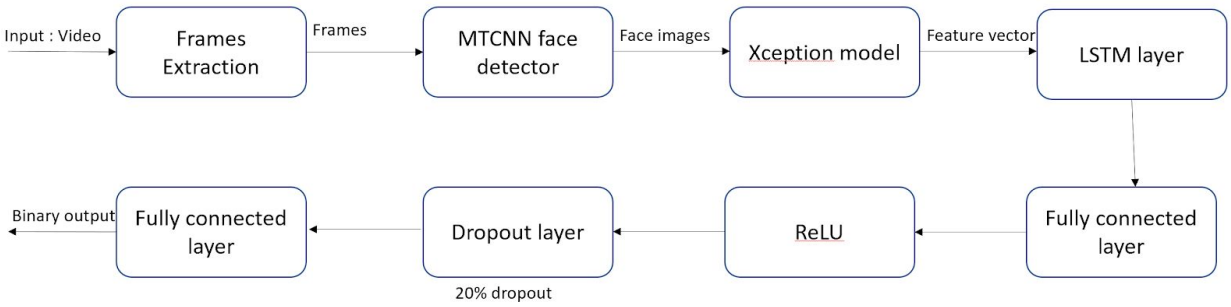


Fig. 10: Model architecture

Results:

1. Deepfake Images:

   Below are some of the sample images generated by each GAN as part of our fake image pool generation. PCGAN managed to generate the best realistic images as the model grows progressively from 4x4 resolution and goes upto 1024x1024 resolution by adding layers as training progresses - something that is fairly difficult to achieve as other GANs can only synthesize good quality images with low resolution.

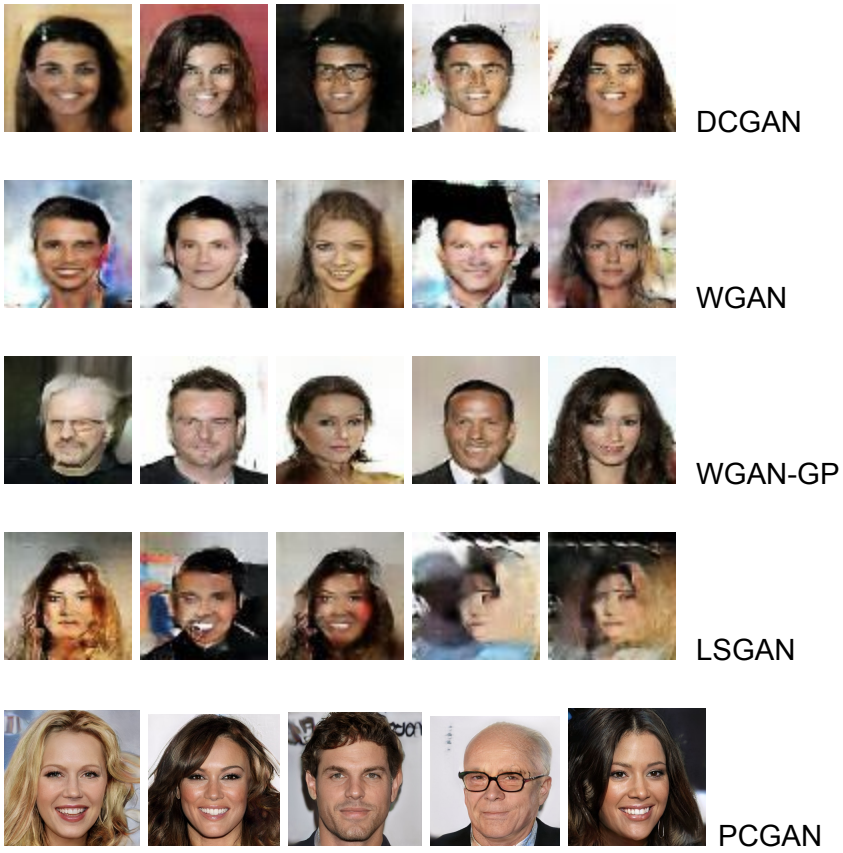
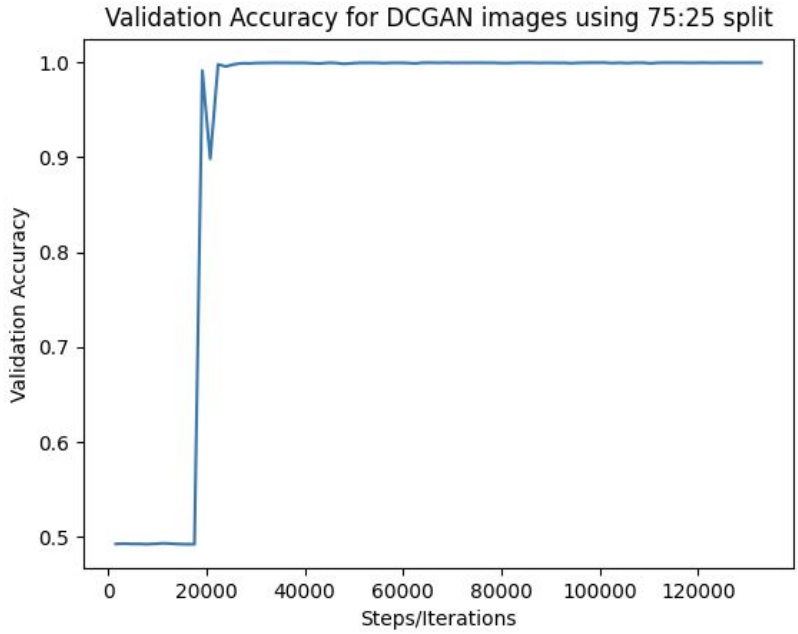


Fig. 11: Images generated by different GANs

We have first tested our model on a few individual GANs - DCGAN and WGAN-GP - to see how it fares and we achieve 0.99 validation accuracy with 0.99 precision and 0.99 recall for DCGAN and 0.99 accuracy with 0.99 precision and 0.98 recall for WGAN-GP.



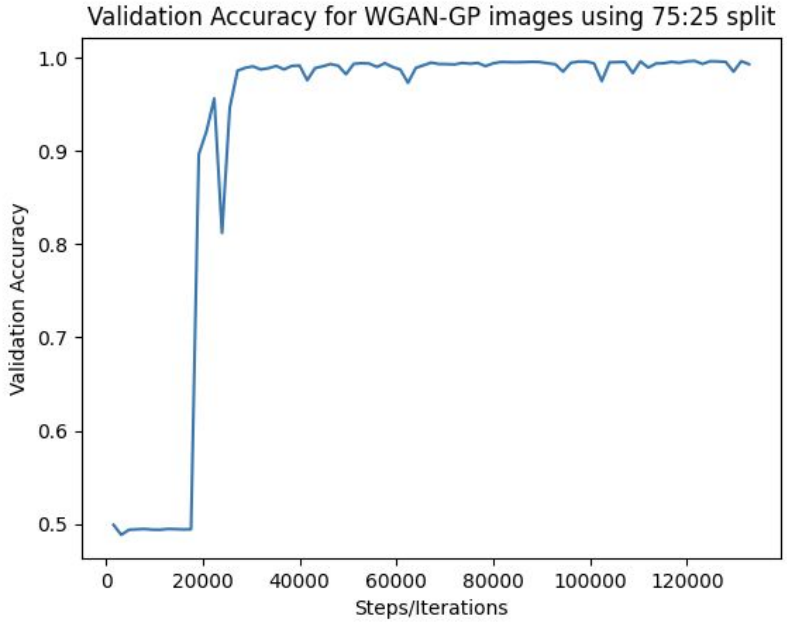


Fig. 12: Validation accuracy for DCGAN and WGAN-GP

However, to test whether our model generalizes well to images not seen during training, we use images from 4 of the GANs for training and exclude 1 GAN completely from training. We only use this GAN's images in the validation set.

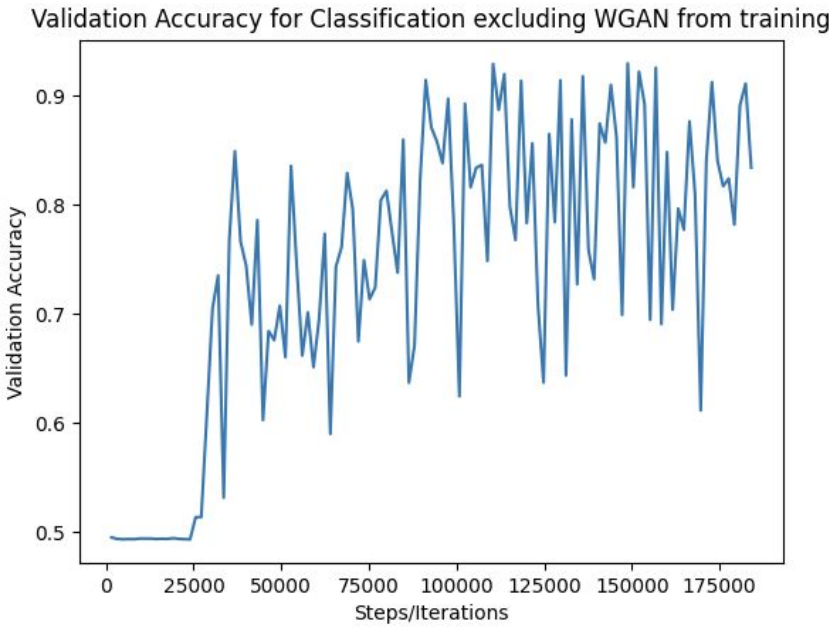


Fig. 13: Validation accuracy for WGAN after training

Excluding WGAN from training gives the above result for validation accuracy. The highest accuracy reached is 92%. But it oscillates and we will have to work on the hyper-parameters to stabilize it. One very likely reason why it oscillates might be that the size of validation data is very small - 10,000 whereas training uses 3,95,000 images - so just around 2.5% of training data. We can try increasing the size of the validation set and test again. Another thing we can try is tuning the learning rate and batch size to see if it helps.

We also plot contrastive loss to see if it varies a lot after stopping the contrastive loss optimization after the 2nd epoch and it stays fairly low and constant throughout the training.

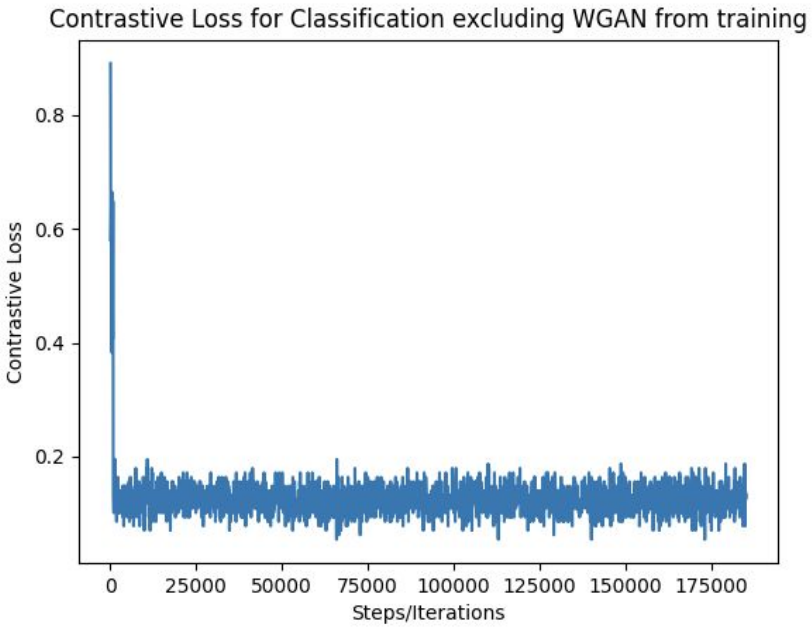


Fig. 14: Contrastive loss excluding WGAN from training

One area where our model failed is when excluding PCGAN images from training. It generates poor results as PCGAN generates the best realistic images out of all the GANs and is more complex than the other GANs with the way it generates images by progressively adding layers and increasing the resolution. The training accuracy was 100% but validation accuracy achieved was 50% - something that could be achieved by random guessing. Our model failed to pick up the fake features of PCGAN since the training set images were nowhere near close to the quality of PCGAN's images.

2. Deepfake videos:

Deepfake Detection Challenge (DFDC) dataset is used for the training of the model described before. The training was done on 46 folders and the remaining 4 folders were used for the testing of the model. Initially, the frames from the videos in all folders are extracted. 30 frames from every video were extracted. The faces from these frames are then extracted using the MTCNN face detector.

The model was then trained using the 46 folders which have around 100,000 videos. All frames of a video were passed into the model at once. Thus, we can say that 30 was the batch size of our model. The trained model was then tested for four folders from the dataset which contained around 10,000 videos.

For the first experiment, accuracy was calculated for the predicted value on the testing set after training the model. The accuracy for all four folders in the testing set is given below.

Folder	Accuracy(%)
Folder 45 (2347 videos)	78.97
Folder 46 (2203 videos)	78.92
Folder 47 (2407 videos)	78.78
Folder 48 (2464 videos)	81.49

The average accuracy was found to be 81.49%.

For the second experiment, we thought of comparing or model's performance with all other participants in the challenge. The metric given by the challenge was log loss probability given below:

$$\text{LogLoss} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

where

- n is the number of videos being predicted
- $\hat{y}_i$ is the predicted probability of the video being FAKE
- y_i is 1 if the video is FAKE, 0 if REAL
- $log()$ is the natural (base e) logarithm

The processed videos from the test set is passed into the model whose output is passed to a softmax layer which outputs a probability of both classes i.e. FAKE and REAL. The probability of a video being FAKE is then used to calculate the score using the Log loss metric as mentioned above. A smaller log loss is preferred for the model. The use of the logarithm provides extreme punishments for prediction being both confident and wrong.

The best score achieved in the challenge was 0.1917. Our score on the testing set was 0.5505. Comparing our performance with all 2300 participants in the competition, we stand around rank 1100.

Conclusion and Future Scope:

In conclusion, we have tried to detect fake images from some of the several state-of-the-art GANs and even though our model fails in certain areas, we believe that we can still achieve good results by working on the model and trying to tune it. We already achieve almost 100% accuracy by training on the individual GANs and our model generalizes well to a certain extent across multiple GANs as well.

We tried to explore the deepfake videos domain as well. We tried to come with a CNN+RNN model giving almost 80 % accuracy on the testing set with a log loss score of 0.5505.

There are a lot of publicly available datasets available for deep fake video detection. Many people have proposed the state-of-the-art performance of their models on these datasets. But because of the recent introduction of the DFDC challenge's dataset, there are no results of the current state-of-the-art techniques for this dataset. We tried to come up with some baseline results and compared our model with the other models proposed. The dataset provided by competition is very large compared to the datasets currently available publicly. So the performance on this dataset can be used to generalize the model.

Even though both of our models are based more on deep learning, the underlying machine learning techniques still apply. The optimization of an objective loss function and tuning of the hyper-parameters of the model are based on core machine learning principles.

For the Fake Image Detection, we have used Resnet but other models like Densenet and Xception, etc can be used. The training set still needs to be representative of various GANs' images in order to get good results for a generalized model. GANs that are vastly different from the ones used in training might not give expected results.

There is a lot of work still to be done to predict whether a video is fake or real. Many state-of-the-art algorithms are proposed but they are all limited to only some particular dataset. This is because of the limited datasets that are publicly available. We tried to come up with a model which can still be finetuned to give better performance.

The CNN model chosen here is the Xception Model. Other models like VGG, Resnet, Inception, etc can be used to have a comparison between different models. A custom CNN network might also be useful and help in improving the score. Introducing new and custom CNN networks can help increase the performance of the model.

The LSTM layer used in the RNN network is the standard LSTM layer. Manipulations in the gate equations can be done to use modified LSTM layers which might help to improve the score.

Contribution by each Member:

Since meeting up to work on the project together was difficult, we decided to divide the two modules amongst ourselves. Vishal has primarily worked on Video Detection module and Sneha has worked on the Image Detection part.

References:

1. D. Guera and E. Delp, “Deepfake Video Detection Using Recurrent Neural Networks,” in Proc. International Conference on Advanced Video and Signal Based Surveillance, 2018.
2. X Yang, Y. Li, and S. Lyu, “Exposing Deep Fakes Using Inconsistent Head Poses,” in Proc. International Conference on Acoustics, Speech and Signal Processing, 2019.
3. Y. Li and S. Lyu, “Exposing DeepFake Videos By Detecting Face Warping Artifacts,” in Proc. Conference on Computer Vision and Pattern Recognition Workshops, 2019.
4. A. R¨ossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, “FaceForensics++: Learning to Detect Manipulated Facial Images,” in Proc. International Conference on Computer Vision, 2019.
5. B. Bayar and M. Stamm, “A Deep Learning Approach to Universal Image Manipulation Detection Using a New Convolutional Layer,” in Proc. ACM Workshop on Information Hiding and Multimedia Security, 2016.
6. D. Cozzolino, G. Poggi, and L. Verdoliva, “Recasting Residual-Based Local Descriptors as Convolutional Neural Networks: an Application to Image Forgery Detection,” in Proc. ACM Workshop on Information Hiding and Multimedia Security, 2017.
7. H. Nguyen, F. Fang, J. Yamagishi, and I. Echizen, “Multi-task Learning For Detecting and Segmenting Manipulated Facial Images and Videos,” arXiv preprint arXiv:1906.06876, 2019.
8. J. Stehouwer, H. Dang, F. Liu, X. Liu, and A. Jain, “On the Detection of Digital Face Manipulation,” arXiv preprint arXiv:1910.01717, 2019.
9. S. Agarwal and H. Farid, “Protecting World Leaders Against Deep Fakes,” in Proc. Conference on Computer Vision and Pattern Recognition Workshops, 2019.
10. E. Sabir, J. Cheng, A. Jaiswal, W. AbdAlmageed, I. Masi, and P. Natarajan, “Recurrent Convolutional Strategies for Face Manipulation Detection in Videos,” in Proc. Conference on Computer Vision and Pattern Recognition Workshops, 2019.
11. D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, “MesoNet: a Compact Facial Video Forgery Detection Network,” in Proc. International Workshop on Information Forensics and Security, 2018.
12. B. Dolhansky, R. Howes, B. Pflaum, N. Baram, and C. Ferrer, “The Deepfake Detection Challenge (DFDC) Preview Dataset,” arXiv preprint arXiv:1910.08854, 2019.
13. Chih-Chung Hsu, Yi-Xiu Zhuang, and Chia-Yen Lee, “Deep Fake Image Detection Based on Pairwise Learning”.

14. Dang, L.M.; Hassan, S.I.; Im, S.; Moon, H. Face image manipulation detection based on a convolutional neural network. *Expert Syst. Appl.* **2019**, *129*, 156–168.
15. Marra, F.; Gragnaniello, D.; Cozzolino, D.; Verdoliva, L. Detection of GAN-Generated Fake Images over Social Networks. In Proceedings of the IEEE Conference on Multimedia Information Processing and Retrieval, Miami, FL, USA, 10–12 April 2018, pp; 384–389.
16. Dang, L.; Hassan, S.; Im, S.; Lee, J.; Lee, S.; Moon, H. Deep learning based computer generated face identification using convolutional neural network. *Appl. Sci.* **2018**, *8*, 2610.
17. Huaxiao Mo, B.C.; Luo, W. Fake Faces Identification via Convolutional Neural Network. In Proceedings of the ACM.