DeepFake Detection

1st Kaamran Khan
Roll Number : MT19064
Indraprastha Institute of Information Technology (IIIT-D)
Delhi, India
kaamran19064@iiitd.ac.in

2nd Vedant Desai
Roll Number : MT19074
Indraprastha Institute of Information Technology (IIIT-D)
Delhi, India
vedant19074@iiitd.ac.in

I. Problem Statement and Motivation

Deepfake techniques, which present realistic AI-generated videos of people doing and saying fictional things, have the potential to have a significant impact on how people determine the legitimacy of information presented online. These content generation and modification technologies may affect the quality of public discourse and the safeguarding of human rights—especially given that deepfakes may be used maliciously as a source of misinformation, manipulation, harassment, and persuasion. The generation of these malicious techniques will not only increase in magnitude but also advance technically. Thus, we aim to device innovative strategies that will suffice the need to counter such deepfake techniques.

II. Dataset Used

The Deepfake dataset was available on Kaggle. But the entire dataset was 470 GB of size. It was divided into 50 chunks of 10GB each. Using this entire data was computationally infeasible for us.So, we decided to use 4 chunks of it with total data around 40GB. After reading the metadata json for each chunk we discovered 871 real videos while rest were fake. This is because the overall data distribution had an 80:20 split ratio of fake to real. To train the models properly for generalized predictions we decided to have 50:50 fake to real ratio and we selected 871 fake videos. The total videos consisted 1742 videos in total. Now each video had 300 frames that can be read. This meant we had enough data to properly train the model.

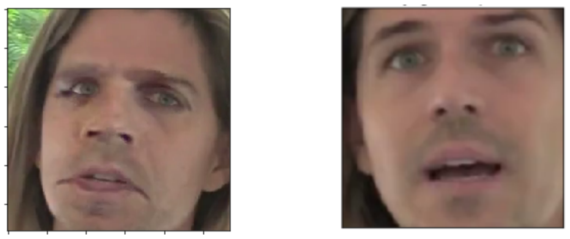


Fig. 1. Left: Frame from fake video, Right: Frame from real video

III. Preprocessing

Faces that we have extracted from frames were of different sizes so we resize each face to size 224*224*3. Like most other torch-vision models, the model we're using (ResNeXt50) requires that input images are normalized using mean and std. deviation. So, we normalized each face using mean and std. deviation. In our classification model of sound, the features have very high dimension that we have extracted from audio due to which we were getting memory crashed error so we resized them to size 1103*40. In MFCC feature, we were getting NAN value in some pixels so we have replaced NAN value with the mean of all the pixels oh that image. And also, in Spectrogram features of audio, we were getting some minus infinity pixel so we replaced those pixels with zero value.

IV. Training details

The 1742 videos in data was split into 70:20:10 ratio of train, validation, and test. Meaning train, validation, and test had 1220, 348, and 174 videos respectively. Now out of the 300 frames for each video we decided to pick 50 frames distributed uniformly in the video. This meant we had 61000, 17400, and 8700 frame images for training, validation, and testing. Since the data is 50:50 split and is a binary prediction problem, we used accuracy as the evaluation metric.

V. Methodology

A. Method 1: Classification by taking entire frame as image

The first method we tried was to divide the video into frames and then send the entire frame as image for training. So, 61000 frames as images were sent to training. Since this is an image classification problem, we decided to use the popular ResNext-50 architecture to train the model. We also had ResNext-50 (32*4d) architecture's pretrained weights, trained on ImageNet dataset with us. The ResNext-50 architecture consisted of cardinality as 32 where each block contained 4 layers. Each layer had the following configuration: [3, 4, 6, 3]. So, we applied 2 training models, one training using the pretrained weights and other without using the pretrained weights. While observing the training process we found out that though the training was happening, but it was happening very slowly as each epoch was taking a lot of time. Even after training both model for hours the maximum validation accuracy we got was 56.32%.
While observing carefully the real and fake videos, we noticed that fake videos were usually distinguishable with respect to the face features only. So, next we planned to extract the faces using face detection algorithm and then send these faces instead of entire frames for training.

B. Method 2: Classification by taking only face from each frame as image

1) Part 1: Face Detection: The main purpose of our project was not face detection but classification of real and fake videos. So, we decided to choose HaarCascade classifier from cv2 library for face detection. HaarCascade is not the most accurate but was chosen since it is a very fast detection algorithm. Since our number of frames were very large, we thought this would be the right choice. But after applying it on a sample of frames we found that the detection rate of HaarCasade is very poor for this data. And since we would train faces data, it would be infeasible to choose this algorithm for face detection. To select the ideal algorithm for face detection we decided to analyze multiple popular face detection algorithms according to our requirement. Our requirement was that we needed a fast algorithm but one which gave decent enough accuracy. We selected Dlib, MTCNN, and YOLO algorithm for analysis, keeping HaarCascade classifier as the baseline model. Dlib and MTCNN was used using their respective libraries while YOLO was used using its API with pretrained MobileNetV2 as the model. We performed the analysis on a sample of 300 frames. Each frame had a face so each algorithm had to detect the face and the total time taken in detection of 300 frames was noted. Fig 2 shows time taken and accuracy for each algorithm. As we can clearly derive that YOLO algorithm is the clear winner with the best accuracy while taking less time than Dlib and MTCNN.

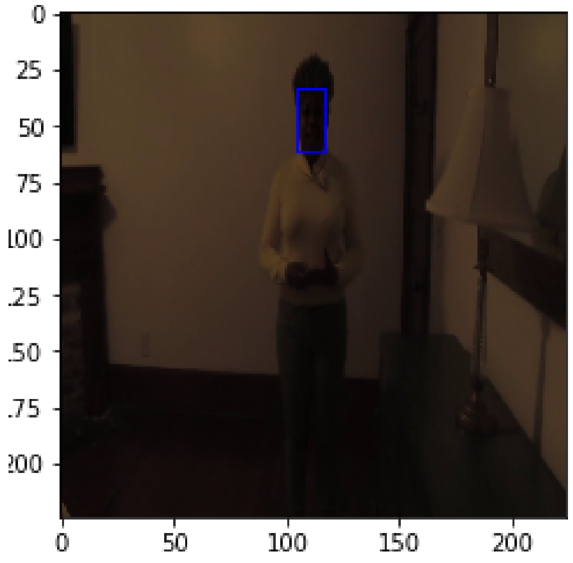


Fig. 2. Face Detection by YOLO

2) Part 2: Face Classification: Using the face detection algorithm YOLO, we extracted faces from 50 frames from each video and sent it for training. Thus, using 1220 videos there were 30500 real and fake face images amounting to total 61000 face images that were sent to training. For validation set we used 348 videos meaning 17400 face images. Next, we sent these face images to training on the ResNext-50

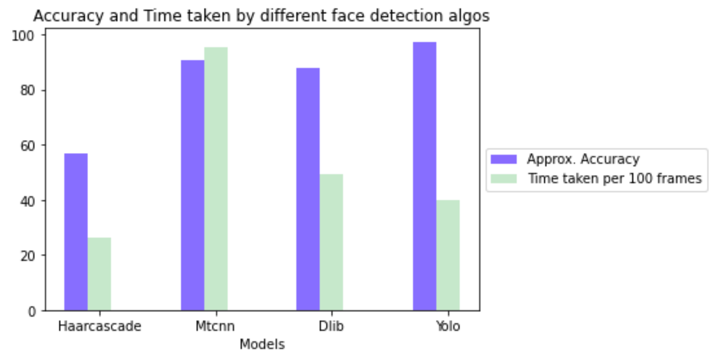
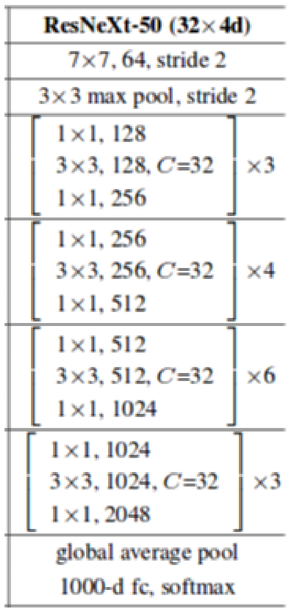


Fig. 3. Face Detection Algorithms analysis with respect to time and accuracy

architecture mentioned above. While training we observed that the data was being trained better than the training using entire frames. But this training was also very slow as each epoch took 13-14 minutes to train. So, despite seeing that the model would be trained properly with time, we had to abandon the training process since there was internet connectivity issues and also time constrains. Now, to train we used a toned-down



ResNeXt-50 (32×4d)	
7×7, 64, stride 2	
3×3 max pool, stride 2	
[1×1, 128; 3×3, 128, C=32; 1×1, 256]	×3
[1×1, 256; 3×3, 256, C=32; 1×1, 512]	×4
[1×1, 512; 3×3, 512, C=32; 1×1, 1024]	×6
[1×1, 1024; 3×3, 1024, C=32; 1×1, 2048]	×3
global average pool 1000-d fc, softmax	

Fig. 4. ResNext-50 (32*4d) Architecture

CNN architecture but with enough complexity. This model trained much faster than ResNext-50. But without tuning it the model started to overfit the data. So, we applied a few tricks like dropout and batch norm and performed hyperparameter tuning. This successfully thwarted the overfitting problem as we achieved 75.86% train accuracy and best 69.25% validation accuracy. Since we had to classify an entire video and each validation video had 50 frames, we could apply a threshold on number of fake frames needed in 1 video to classify the video as fake. Like this we could devise a strategy to increase the accuracy by performing analysis on the threshold to each video. Also, to select the number of frames we should extract from the test video to classify it better, we decided to perform the above-mentioned analysis on different number of frames. So, from the validation set we picked 60 videos 30 real and 30 fake. For each video, we extracted different number of face frames for analysis. The numbers we choose were:

1) 1st frame of each video – 1 frame
2) 1st and last frame of each video – 2 frames
3) 5 uniformly distributed frames – 5 frames
4) 10 uniformly distributed frames – 10 frames
5) 20 uniformly distributed frames – 20 frames
6) 50 uniformly distributed frames – 50 frames

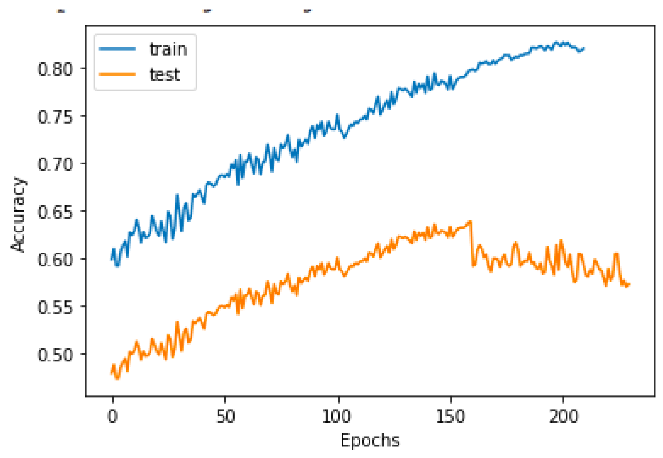


Fig. 5. Overfitted model accuracy curve

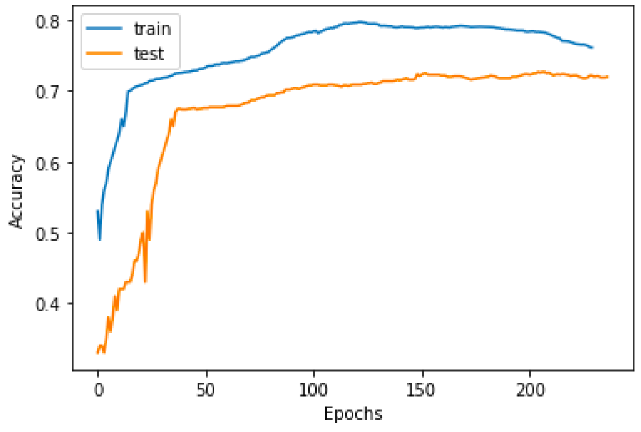


Fig. 6. Accuracy curve of our best model

So, each of the 60 videos were validated on the trained model and for each video a particular number of frames were predicted as fake frames. Using this number for each video we calculated the threshold where the validation set will give highest accuracy. In the figure 4 we have created multiple graphs depicting the threshold calculation for the different values of frames. So, below the threshold the entire video would be classified as real and above it the video would be classified as fake.

Now we replicated the entire validation data into 6 types of different frames as given above. And for each type we

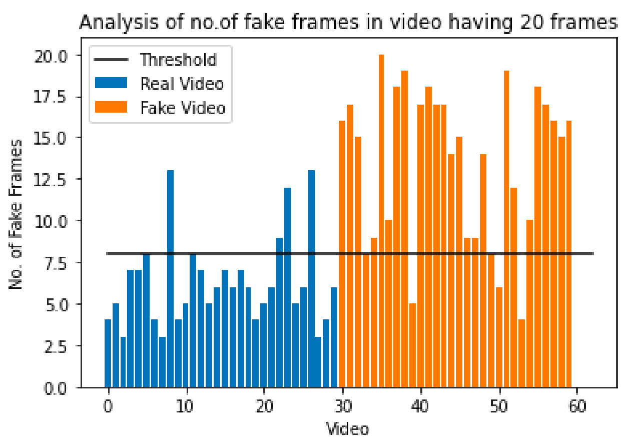


Fig. 7. Threshold setting analysis for 20 frames per video

validated them on the trained model. Now, we applied the already calculated threshold on them and like this classified each video into real and fake. The different accuracies for each frame type are given in figure 5 with keeping 20 frames fetching the highest validation accuracy of 81.03 which is a vast improvement with respect to classification on frames only.

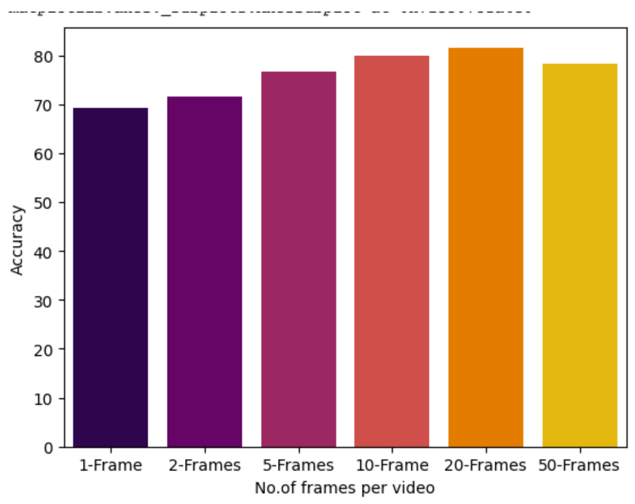


Fig. 8. Validation Accuracy with respect to number of frames per video

3) Method 3: Classification by taking features of Audio: The third method we tried was to extract the audio from video and then extract the features from those extracted audio and then passing those extracted features to our model for training. As we are giving features of sound to our models, which are nothing but images and we know that CNN works better than other models like RNN, LSTM, etc. for image classification because CNN extracts spatial information whereas, RNN extracts temporal features and also in the deep network of RNN there is also a problem of vanishing gradient whereas in CNN starting convolution layers gives us finer information and deep convolution layers gives us coarser information about the image. By combining coarser and finer information, we can classify the image more appropriately.

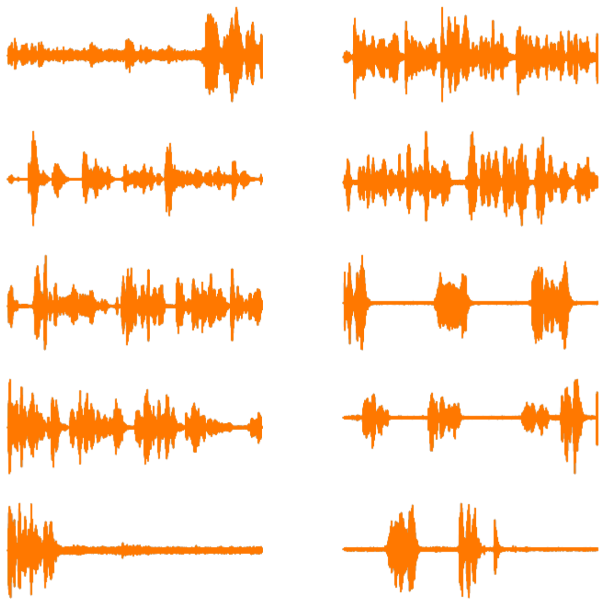


Fig. 9. Sound plot with real on the left and fake on the right

So we decided to use a simple CNN-based model for classification, simple because we have 1220 videos, which means only 1220 images of audio features for training. We applied three different features of sound to our training model. To build a baseline for our audio classification, we simply passed the raw data i.e., the simple waveform of audio as an image to our model, and we got the validation accuracy as 57.43%. In the second model, we passed the MFCC features of audio as an image to our model, and we got the validation accuracy as 77.83%. In the third model, we passed the Spectrogram features of audio as an image to our model, and we got validation accuracy as 83.04%.

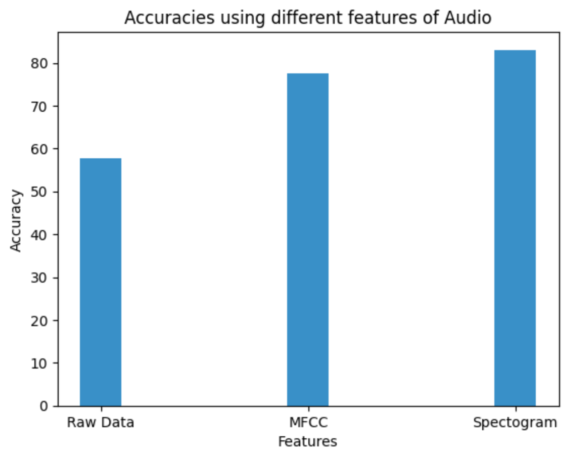
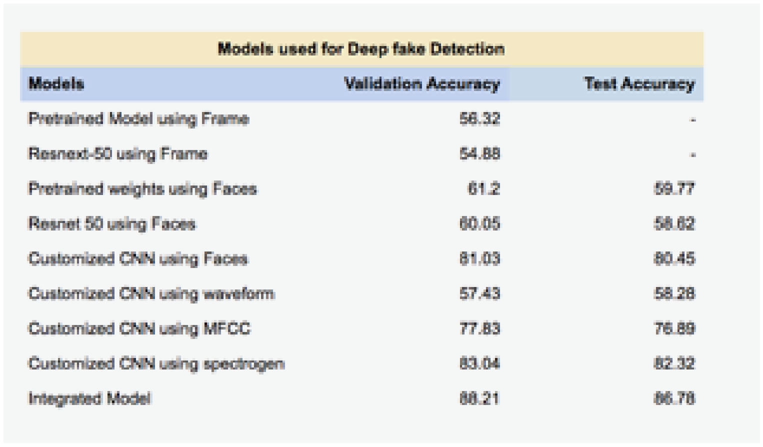


Fig. 10. Validation Accuracy using 3 different audio features

4) Method 4: Integration: To increase the accuracy further, we tried to integrate the results of classification using audio and face classification. We took the best models and hyper-parameters from both, 20 frames with 8 threshold giving 81.03% accuracy from face classification and using spectrogram features giving 83.04% accuracy from classification using audio. Next, on the same validation data we performed a very crucial analysis. In this, for a particular video, our combined model will predict fake of both the above two predict fake and real likewise. Now, in the case where audio gives fake and face gives real, we found that changing the threshold to above 4 in which fake will be predicted if more than 4 frames are fake otherwise real will be predicted. In the case where audio predicts real and face predicts fake, the threshold was kept at 13 where more than 13 frames would predict fake while less than 13 would predict real. So, using this combined model setting we achieved accuracy as high as 88.21%.

VI. Challenges Faced

We faced quite a few non-technical challenges. The obvious one is that since the data consist of videos, the size of data is very large. So, we had to use a subset of the data. Even that data was quite large as we had to upload it into drive since we used google colab to run our models. We had to upload the entire data in parts and then combine it later for training purposes.



Models used for Deep fake Detection		
Models	Validation Accuracy	Test Accuracy
Pretrained Model using Frame	56.32	-
Resnext-50 using Frame	54.88	-
Pretrained weights using Faces	61.2	59.77
Resnet 50 using Faces	60.05	58.62
Customized CNN using Faces	81.03	80.45
Customized CNN using waveform	57.43	58.28
Customized CNN using MFCC	77.83	76.89
Customized CNN using spectrogen	83.04	82.32
Integrated Model	88.21	86.78

Fig. 11. Results of all the models on validation and test set

References

[1] T. Nguyen, C. Nguyen, D. Nguyen, D. Nguyen and S. Nahavandi, "Deep Learning for Deepfakes Creation and Detection", Sep 2019.
[2] H. Nguyen, F. Fang, J. Yamagishi and I. Echizen, "Multi-task learning for detecting and segmenting manipulated facial images and videos", 2019
[3] Y. Li and S. Lyu, "Exposing DeepFake Videos By Detecting Face Warping Artifacts", 2018.