Understanding the Deepfake Detectability of Various Affinity Groups Using a Dual-Stream Network

Rohit V Gopalakrishnan
Department of Computer Science
Columbia University
rvg2119@columbia.edu

Directed Research Report – Fall 2022
Advisor: Prof. John Kender

December 31st, 2022

Abstract

The rapid progress in the creation and circulation of deepfaked media has led to the need for countering the same. While existing implementations have constantly been improving in detecting deepfakes globally, little effort is being taken into considering the effects of deepfake detectability in different affinity groups differentiated by race, sex, etc. To deal with this, this paper discusses in length a robust deepfake detection system, that improves upon previous implementations that is trained on an unbiased global dataset, and a comparative analysis performed on deepfake videos of people differing in affinity groups such as race and sex. The system utilizes two streams for the detection. The first stream analyzes the semantic details of the given face in question, while the other analyzes the noise features using Spatial Rich Model (SRM) filters. By passing multiple frame inputs simultaneously, it also accounts for the temporal features of the video in question. Results of the experiment indicate differences in deepfake detectability of different affinity groups.

1. Introduction

Deep learning has been successfully applied to unravel various complex problems starting from big data analytics to computer vision and human level control. Although the benefits are high, it has also been employed to make software which will cause threats to privacy, democracy and national security. Deepfake (from 'deep learning' and 'fake') is a technique that, using deep learning, can superimpose facial images of a target to any media of a source person to create media of the target doing or saying things the source person does. Deepfake algorithms can create fake images and videos that humans cannot distinguish from authentic ones. These models are used to examine facial expressions and movements of a person and synthesize facial images of another person making analogous expressions and movements. Deepfake methods normally require a large amount of image and video data to train models to create photo-realistic images and videos. The proposal of technologies that can automatically detect and assess the integrity of digital visual media is therefore indispensable.

Figure 1.1 shows an example of how deepfakes work. The first image shows the donor person, who in the case of a deepfake attack will be the victim. The second image is the target person, where the donor person's face will be superimposed upon. The last image showcased the final deepfake result. As one can see, the third image, although is a synthetic fake image generated, looks very realistic to the naked eye. If the first two images weren't there, it would be hard to know that the third image is a fake.

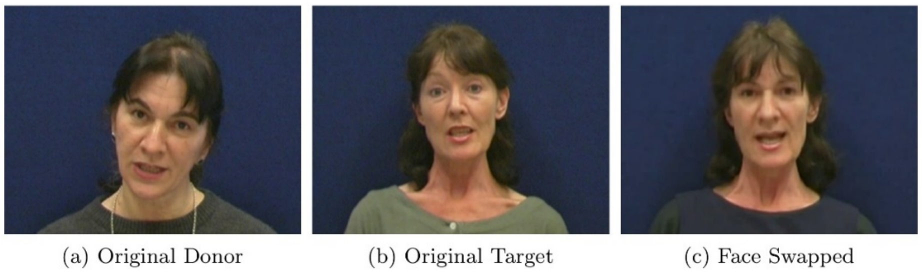


Figure 1.1 Example of a Deepfaked Image

Even though there are positive uses of deepfakes such as creating voices of those who have lost theirs or updating episodes of movies without reshooting them, the number of malicious uses of deepfakes largely dominates that of the positive ones, a prevalent use being tampering of evidence/faking evidence. Another example is compromising the credibility of the media the general public loses their trust in journalism and other forms of reliable sources of news, like create sexually compromising videos to blackmail people, or building fake-news campaigns to manipulate public opinion. In the long run, it may also reduce trust in journalism, including serious and reliable sources. Finding the truth in the digital domain therefore has become increasingly critical and there is a need in society to detect deepfakes.

There have been numerous methods proposed to detect Deepfakes. Over the years, we have seen improvements in the methodologies employed using deep learning. However, it is noticeable that there is minimum work done in studying and observing the trends of deepfake detection based on different affinity groups such as race, sex, etc. It is well known that in the field of facial detection there are some disparities when it comes to detecting people from different affinity groups. The same can be considered as the hypothesis for the deepfake detection problem.

The primary purpose of this project is to deal with the deepfake problem from a new angle. While work in deepfake have been purely technical, there haven't been any analytical angles based on inferences being undertaken. By performing an analysis and observing the effects of deepfake detection with various affinity groups, we can hope to obtain new information that can be further used to train better models to better deal with the observed results. Additionally, we can also identify errors in training that we have been employing previously to better train existing systems to yield better results.

The proposed system describes a new deep learning-based method that can effectively distinguish deepfaked media from the real ones, and using this system to perform a comparative analysis of deepfake detection taking actors from different affinity groups (based on race and sex), and find if there are any significant observations.

The link for the project files can be found here.

2. Methods

The proposed system is one that utilizes a dual stream network: one stream which takes in a regular facial image to extract the semantic features of the image and another stream which converts the facial image into a noise map by passing it through a series of SRM filters, and passes it through the deep learning model. The system takes in video inputs and hence there is a module to extract time separated frames from the video to account for the temporal features of the input. Figure 2.1 shows the overall system architecture of the proposed system (the pink color indicates that modifications were made from previous implementations).

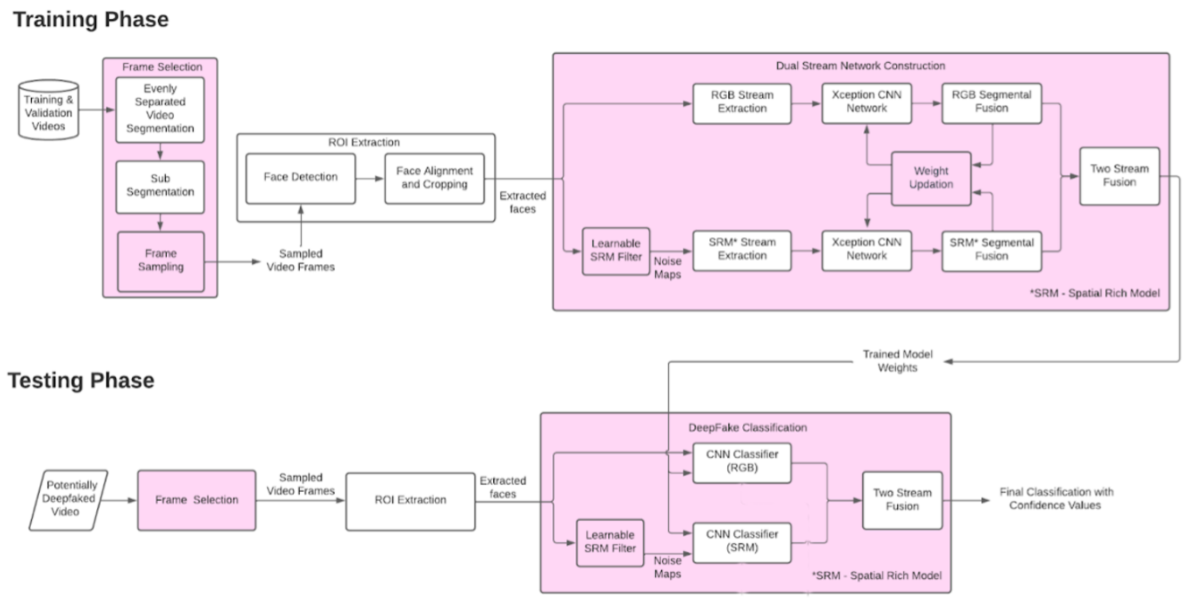


Figure 2.1 Overall Architecture Diagram

The proposed method has the following modules:

1. **Frame Selection**
2. **ROI Extraction**
3. **Dual Stream Network Construction**
4. **Deepfake Classification**

The proposed system includes the Frame Selection module to generate the frames from the input video by sampling with a certain number. The obtained frames are further pre-processed using libraries to generate images that have been aligned and cropped (ROI Extraction). The system has two streams which would be fused towards the end to generate a combined result to obtain the final results (Dual Stream Network Construction). The outputs from the first stage would be sent as input to the two streams in the system. Once the outputs are obtained from each stream, a weighted average would be taken to obtain the final confidence value along with the classification of the type of input (Deepfake Classifier).

2.1. Frame Selection

The Frame Selection module can also be considered as the preprocessing module of the proposed system. As the name suggests, this module is responsible for the selection of important frames in order to account for the spatial features of each frame and the temporal information between the frames for the overall video. Figure 2.2 shows the sequential flow of the module.

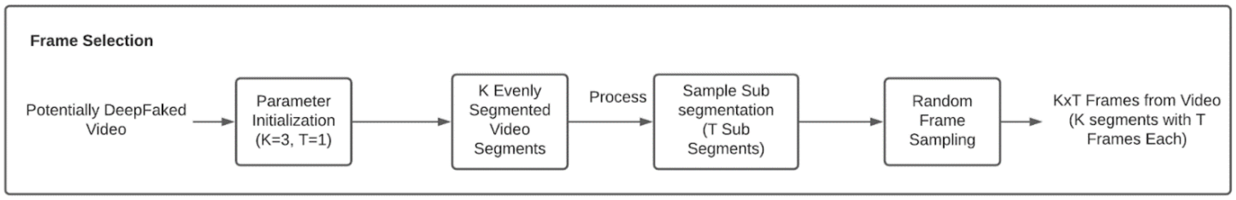

Frame Selection
Potentially DeepFaked Video
Parameter Initialization (K=3, T=1)
K Evenly Segmented Video Segments
Process
Sample Sub segmentation (T Sub Segments)
Random Frame Sampling
KxT Frames from Video (K segments with T Frames Each)

Figure 2.2 Frame Selection

To understand the flow of this module better, Pseudocode 1 explains the working of the module from the input to the output.

Pseudocode 1 Frame Selection

```
Start
{
Int K=3,T=1 ;      //Gpu Limitation
split the video evenly into K segments {$v_1$,$v_2$,$v_3$,.....$v_k$} ;
for (each $v_i$){
        Split into T sub-segments {$v_i$1,$v_i$2,$v_i$3,.....$v_i$T} ;
        Random sampling on  {$v_i$1,$v_i$2,$v_i$3,.....$v_i$T} ; (*Figure out sampling Algo*)
        }
 Dlib ( sub-segments) ;   (*Random Sampled*)
}
End
```

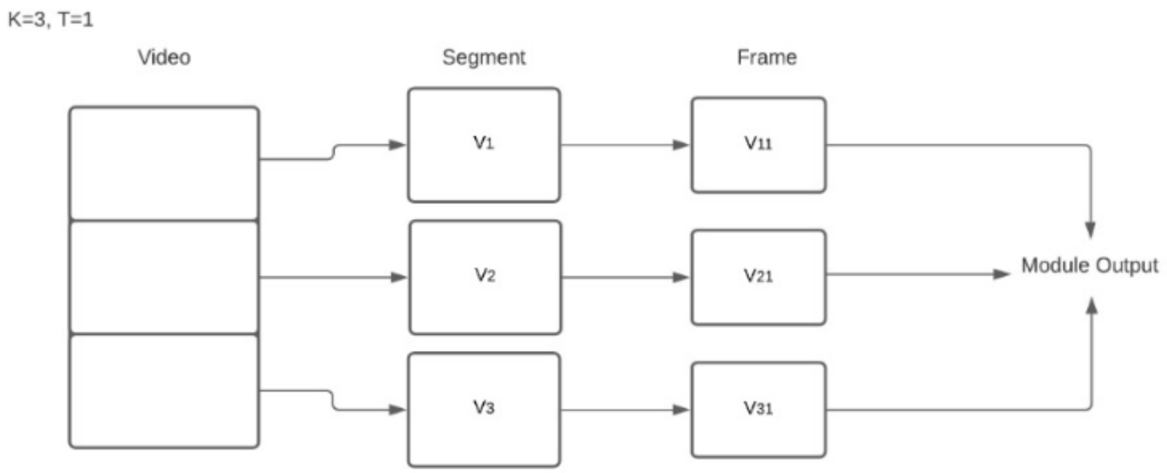


Figure 2.3 Sample Example of the Frame Selection Algorithm

The user initializes the number of segments the video must be split into (‘K’) and the number of subsegments that must be considered for frame sampling (‘T’). From each of these subsegments a frame is randomly sampled using a generator function, The K segments will be parallelly processed to account for the temporal data, while the T subsegments will be processed together and will not account for the time sensitive information. Figure 2.3 shows an example where K=3 and T=1.

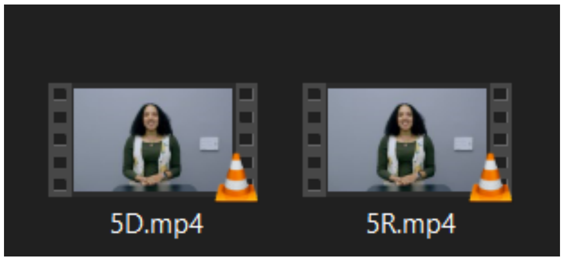


Figure 2.4 Sample Videos from Dataset

The dataset consists of videos of length varying from 10 seconds to 50 seconds (Figure 2.4 shows examples). Due to the variable length when the module splits the video, it must be ensured that it extracts valuable information regardless of the length. Hence, 8 is chosen as the number of segments K. In doing so, segments having lengths of approximately 1-6 seconds are obtained, which is the right amount of length to account for the temporal details.

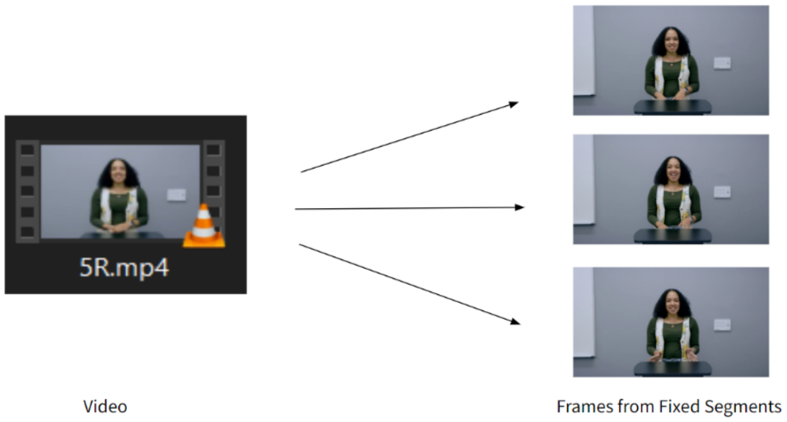
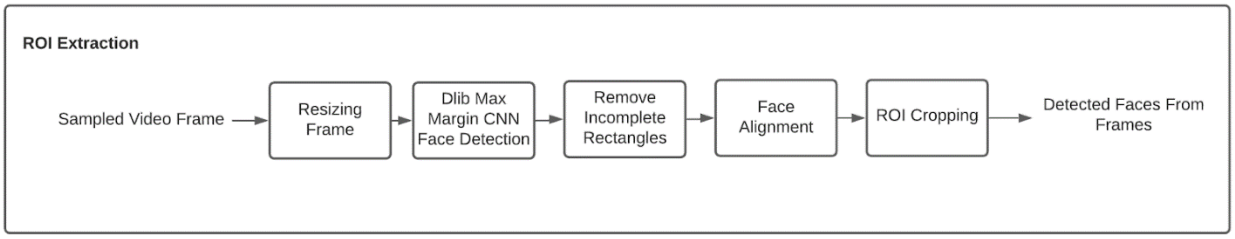


Figure 2.5 Segmentation and Frame Selection

For the sake of computational complexity, the number of subsegments is set to 1, and thus, the module only extracts 1 from each segment. Figure 2.5 shows a simplistic real time example of the working of the Frame Selection module.

2.2. ROI Extraction

Passing the entire frame as a network may provide a lot more unnecessary features for the network to handle. Hence to make the network more effective, this module identify and extract the Region of Interest. For a Deepfake application, the region of interest would be the face. Hence this module is aptly named the ROI Extraction module. Figure 2.6 shows the sequential flow of the module.


ROI Extraction
Sampled Video Frame
Resizing Frame
Dlib Max Margin CNN Face Detection
Remove Incomplete Rectangles
Face Alignment
ROI Cropping
Detected Faces From Frames
Figure 2.6 ROI Extraction

Pseudocode 2 Face Detection, Extraction & Alignment

```
Start
{
        Import Dlib;
        For (each i in V_i)
                dlib.mmod.Run(Sampled Frames);          //Run the MMOD CNN Detector in Dlib library
                For ( j in i)
                        If Faces.rectangles[j] is out of bound
                                Remove Faces.rectangles[j]
                        Faces.rectangles[j].create();   // Acquiring coordinates to detect ROI

        Align the faces identified using the facial features identified
        Output = image with coordinate of Faces.rectangles[]
}
End
```

To extract the ROI, the module utilizes face detection and image cropping methods such as image alignment and image extraction. For the Face Detection, Dlib library is used. To be more specific, the Max Margin Object Detection (MMOD) Face Detector is utilized for detecting the faces from the videos. The system then bounds the face with a viable rectangle based on the identified facial features. If the rectangles are not viable (i.e., partial or overlapping), it is discarded. There detections are then cropped and matched to the input size of the main networks. The working of this module can be better understood by understanding Pseudocode 2.

2.2.1 Face Detection

As mentioned above, the model uses the Dlib MMOD Face Detector in order to identify the facial landmarks of the given video. This face detector is used in particular as it yields better results with video frames compared to other face detectors. It also gives 68 different facial landmarks that can be used for the alignment of the facial image in the next step.

On detecting a face, using the facial features and the artifacts identified by the Face Detector. The model then places a rectangular box over the identified faces, to mark the boundaries of the extraction. However, the face is not extracted just yet, but only after it is passed to the next submodule. Figure 2.7 shows the rectangle being drawn on the identified face.

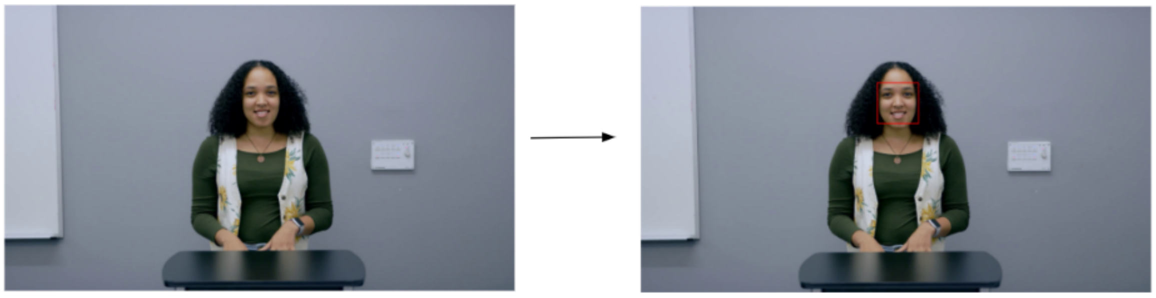


Figure 2.7 Rectangle to Show Detected Face

2.2.2 Face Extraction and Alignment

The model uses the facial artifacts that were found from the previous step and using those, the system aligns the face and crops it in a symmetric manner. In the proposed system, the line of the eye is made to be exactly horizontal (parallel to the top and bottom edge of the image).

All the extracted faces are saved onto the project directory along with all intermediate outputs (such as the frames, the rectangle drawn to identify the face, etc.). Figure 2.8 shows the final output of the ROI Extraction module.

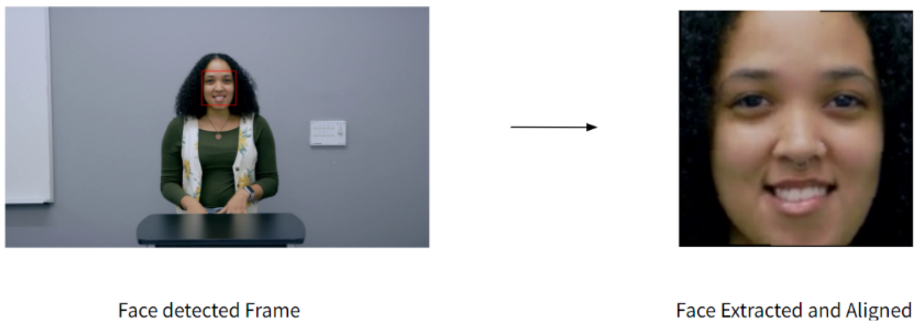


Figure 2.8 Face Extraction/Alignment

2.3. Dual Stream Network Construction

After ROI Extraction, the extracted faces are fed into the Dual Stream Network Construction module. The two streams are as follows:

1. **RGB Stream** – it directly uses the extracted faces as inputs which learns semantic features of the input image. The extracted facial images act as the input for this stream and there is no additional preprocessing required. Figure 2.9 shows the sequential flow of the RGB Stream network.

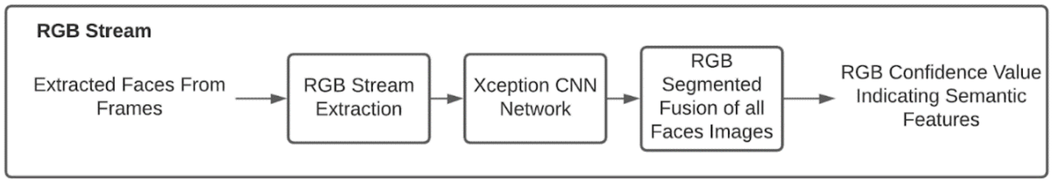

RGB Stream
Extracted Faces From Frames
RGB Stream Extraction
Xception CNN Network
RGB Segmented Fusion of all Faces Images
RGB Confidence Value Indicating Semantic Features

Figure 2.9 RGB Stream Construction

2. **SRM Stream** – SRM inputs are SRM noise maps learned through SRM filters applied on the extracted faces. Therefore, some SRM filters must be defined which would be best suited for this problem, and use that to acquire the noise map, which will be passed to the model. Figure 2.10 shows the sequential flow of the SRM Stream network.

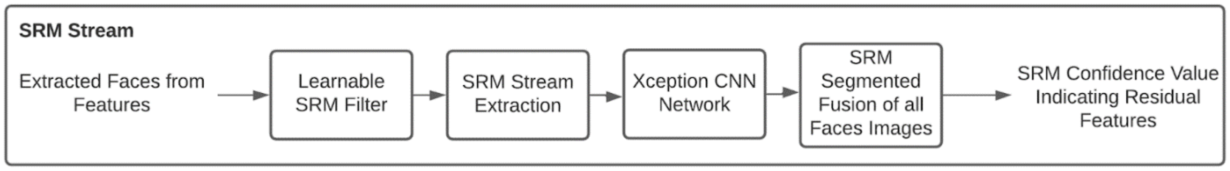

SRM Stream
Extracted Faces from Features
Learnable SRM Filter
SRM Stream Extraction
Xception CNN Network
SRM Segmented Fusion of all Faces Images
SRM Confidence Value Indicating Residual Features

Figure 2.10 SRM Stream Construction

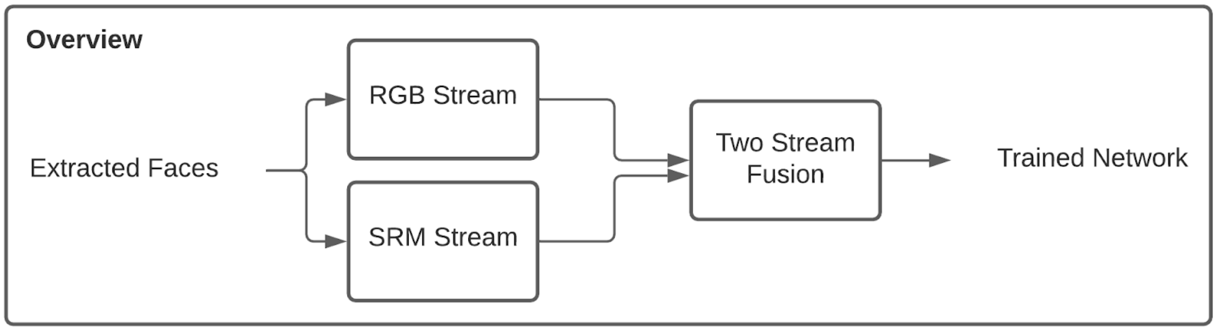


Figure 2.11 Dual Stream Network Construction (Overview)

The network for both is similar, however, the inputs and features extracted vary. Both streams are trained individually (due to GPU limitations) with both the training set and the validation set. Once both networks have been trained, the two streams are fused and when it is passed to the classifier both models play a role in the final classification.

2.3.1. RGB Stream

Once the ROI Extraction module extracts the faces from all segments of the video, it is all placed together in a single folder. This is then passed as the RGB Stream Network input. Figure 2.12 is an example of the RGB input.

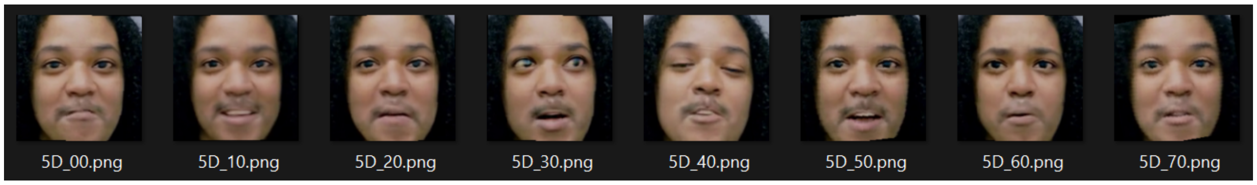


Figure 2.12 Input for RGB Stream

This stream is named the RGB stream simply to differentiate between the SRM stream. RGB is used to indicate that the pixels of this image correspond to the original RGB channel image without any further processing.

2.3.2. SRM Stream

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 2 & -4 & 2 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 & 2 & -2 & 2 & -1 \\ 2 & -6 & 8 & -6 & 2 \\ -2 & 8 & -12 & 8 & -2 \\ 2 & -6 & 8 & -6 & 2 \\ -1 & 2 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 2.13 SRM Filters to Obtain Noise Maps

The RGB Stream is duplicated and each image is passed through a series of filters known as learnable SRM filters. The model uses the SRM filters shown in Figure 2.13 as they have been proven to be effective in identifying the noise features that differentiate between real and face images. These were selected using Han et al. method as a reference.

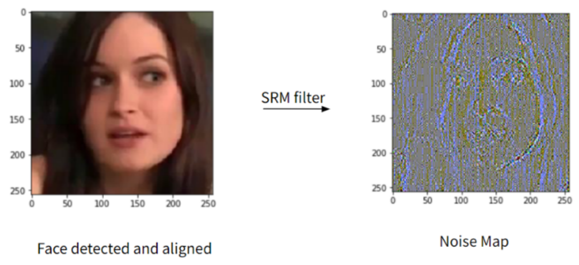


Figure 2.14 Applying the SRM Filters

As shown in Figure 2.14, the noise map of the facial image is obtained. This process is repeated for all the extracted faces present from the input video passed to the proposed system. The system saves the noise maps again in a folder similar to the RGB Stream. This is now the input of the SRM Stream Network, as shown in Figure 2.15.

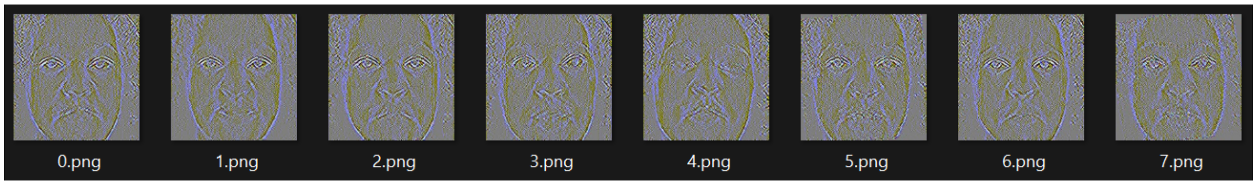


Figure 2.15 Input for SRM Stream

2.3.3. Model Construction

This step involves the creation of the model that is planned to use in the proposed system. XceptionNet pretrained on ImageNet is used as the backbone network for both the RGB model and the SRM model. Both stream networks are trained in a similar fashion, with the only difference being the input. Figure 2.16 shows the structure of XceptionNet.

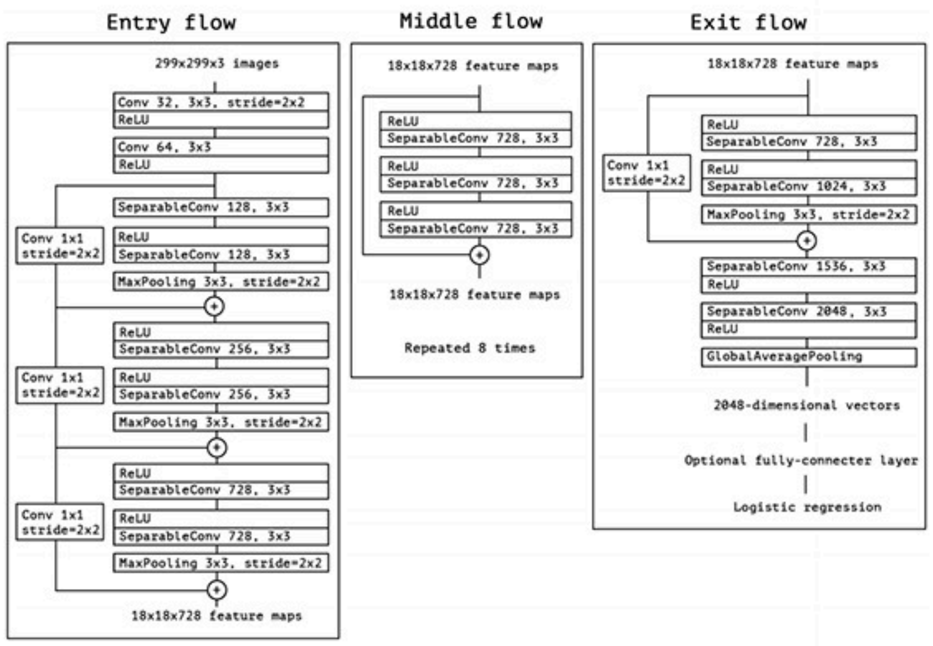


Figure 2.16 XceptionNet Architecture

XceptionNet consists of 14 convolutional blocks, which can be divided into three sections. The first 4 blocks are a part of the Entry flow. This is responsible for simplifying the image for the computer to better understand the objects present in the network. The next 8 blocks are a part of the Middle flow which is primarily responsible for extracting the features required for the problem at hand. The last 2 blocks are a part of the Exit flow, and their primary goal is to reduce the features extracted into simpler numerical values that can be interpreted more easily for classification/ human understanding.

For the proposed system, the Entry flow is frozen, as the model is not being trained to recognize the objects in the picture better. Training this section will be time consuming for less benefit. Hence, the focus is on training both the Middle flow and the Exit flow. The first 36 layers of XceptionNet (that is, the first 4 blocks) are frozen to achieve the same.

As XceptionNet was initially trained on ImageNet, the final output layer consists of 1000 nodes representing the 1000 classes of the ImageNet database. That layer must be removed and add a new output layer with simply 1 node, to accommodate the binary classification problem.

2.3.4. Model Training

Creating the dataset for the training of the modules must be known. For that the videos of the FaceForensics++ dataset is used, and applied with both the Preprocessing (Frame Selection, with 3 segments) and ROI Extraction to obtain facial images. The models are trained with these images. For the RGB network the images are fed as is, while for the SRM network the SRM filters are applied before passing it through the network.
The system then places the real and deepfake image data in separate folders under a folder called data. This important for flow from directory. The pixel values in the image data exist in the range between 0 and 255. large integer coefficients like this complicate gradient descent when using typical learning rates so the next step is to scale the data by a factor of one divided by 255 that way the pixel values fall into the range from zero to one. Then instantiate the image data generator that rescales the images and instantiate generator to feed images through the network.

The training is implemented on this model using the following parameters:

- Adam Optimizer (learning rate = 0.001)
- Batch Size = 32
- Epochs = 10
- Training Steps per Epoch = 160
- Validation Steps per Epoch = 40

The following values are set because of the dataset size. It divides into 200 batches of size 32 (160 batches belonging to training while 40 belonging to validation). Each batch will run in one step of an epoch and hence why the following values are set. 32 is set as the batch size as it makes training the model a lot faster and it stops at 10 epochs because it is observed that after that, there were indications of overfitting.

2.4. Deepfake Classification

After training the model, the network can be put to the test using the videos from the testing set. One video is tested at a time in the Deepfake Classification module. The CNN Classifier has the trained weights loaded and the extracted face images from the ROI Extraction module are passed as input to the Classifier. The entire sequential flow of the Deepfake Classification module is seen in Figure 2.17.

For the classification, each segmented frame values are taken and is passed individually through the Classifier to find out the model output of that particular facial image. This process is repeated for all the other segmented frames for both the streams in the system. For each stream, all the values are aggregated for a segmentally fused value. Equation 2.1 shows the segmental fusion for the RGB Stream, while Equation 2.2 shows the segmental fusion for the SRM Stream. Equation 2.3 shows the two-stream fusion in a simplistic manner.

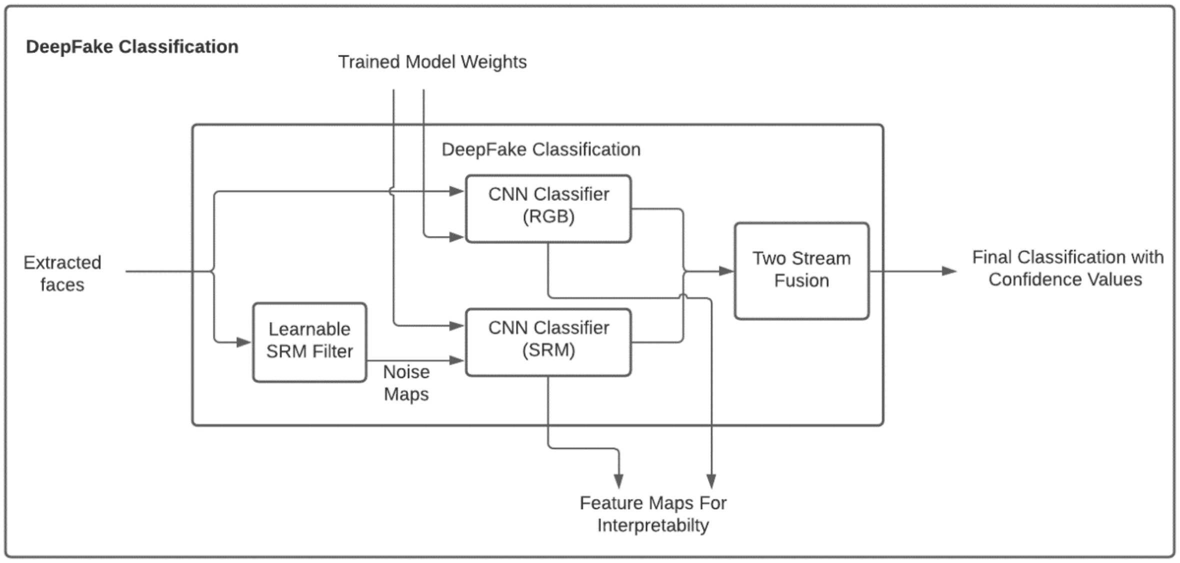


Figure 2.17 Deepfake Classification

$$F_R = F((I_R^1 \odot W_R), (I_R^2 \odot W_R), \dots, (I_R^K \odot W_R)) \qquad (2.1)$$
$$F_S = F((I_S^1 \odot W_S), (I_S^2 \odot W_S), \dots, (I_S^K \odot W_S) \qquad (2.2)$$
$$P = H(\sigma(F_R), \sigma(F_S)) \qquad (2.3)$$

FR – RGB Segmental Fusion
FS – SRM Segmental Fusion
P – Two Stream Fusion
$\odot$ – Convolution Function
σ – Sigmoid Function

Fusion between the two streams happens by the Weighted Sum method with the RGB stream getting a weightage of 1, and the SRM stream getting a weightage of 1.5. The model then classifies the output based on the final confidence value. Equations 2.4 and 2.5 show us the weighted sum of both streams for the problem statement that is deepfake detection. This added weightage is given to the SRM stream as the noise maps generated by the SRM filters capture more detail and noise interference when it comes to tampering with the image.

$$H(F_R, F_S) = (F_R \times w_R) + (F_S \times w_S) \tag{2.4}$$
$$w_R = 1, w_S = 1.5$$
$$H(F_R, F_S) = (F_R \times 1) + (F_S \times 1.5) = F_R + 1.5F_S \tag{2.5}$$

For the classification, the fully trained and tested models for both the RGB stream and the SRM stream are required.

2.4.1. RGB Stream

For the RGB stream, as the number of segments is set as 8, all 8 images from the segments are passed into the model, and acquire the confidence values for each picture passed through the model. The system then needs to display the model outputs for each of the 8 segments' facial image. It can do this using the pyplot function in python and simply print the image along with model confidences under it. Figure 2.18 shows an example of the individual confidence values of the RGB stream.

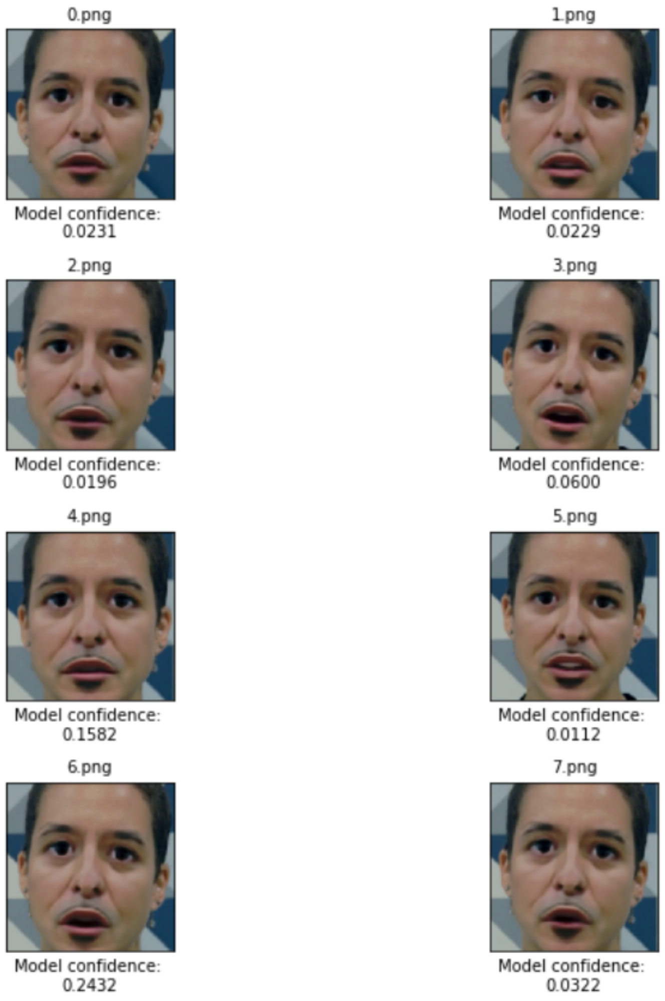


Figure 2.18 RGB Stream Confidence Values

2.4.2. SRM Stream

Similar to the RGB stream, the same is done for the SRM stream. As the number of segments is set as 8, all 8 images from the segments are passed into the model, and acquire the confidence values for each picture passed through the model. The system then needs to display the model outputs for each of the 8 segments' facial image. It can do this using the pyplot function in python and simply print the image along with model confidences under it. Figure 2.19 shows an example of the individual confidence values of the SRM stream.

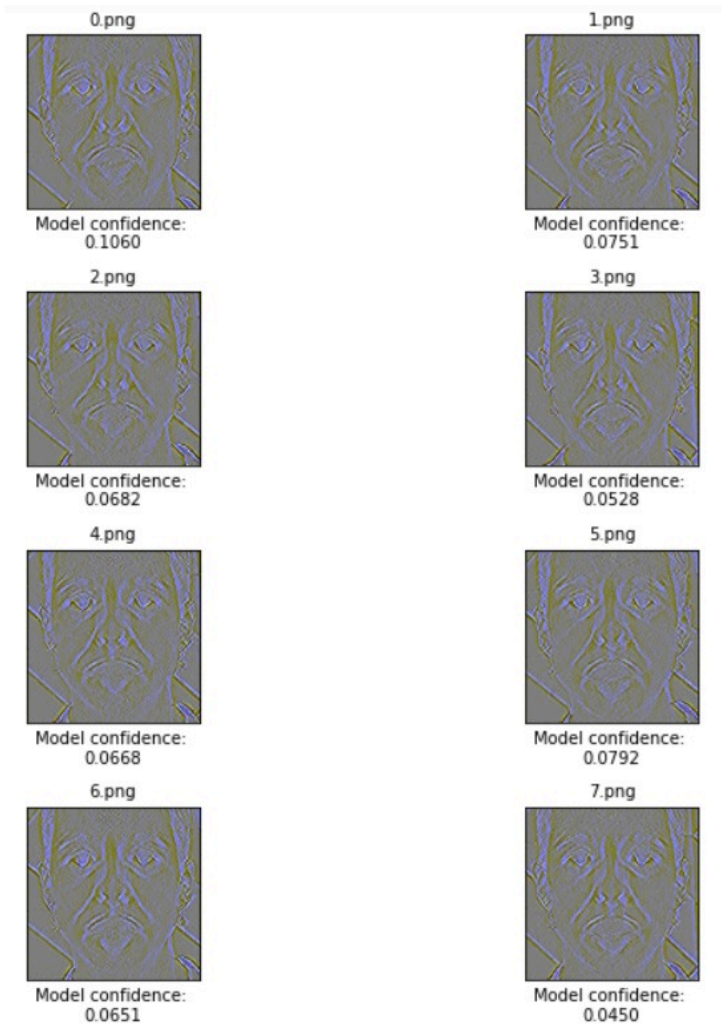


Figure 2.19 SRM Stream Confidence Values

2.4.3. Dual Stream Fusion

The model first needs to aggregate the individual streams' confidence value. This can be achieved by taking simple averaging. Once the system has the segmentally fused value of both the RGB and SRM stream, it now does the weighted sum to aggregate those two values to finally obtain the final confidence value for classifying the input video. Equation 2.5 is used to obtain the two-stream fusion value. If the final value is greater than 0.5, then it is classified as real, else it is a deepfaked video. Hence, the prediction on the video is made using this final confidence value.

3. Experiments

Now that we have gone into depth on the methodology of the proposed system, now it is time to describe the experiments that we will be conducting.

We first talk about the datasets used in this project. In summary, we use the FaceForensics++ dataset to train and test the proposed system, and use the DeeperForensics 1.0 dataset (derivative of FaceForensics++) to conduct our affinity group-based experiments.

We then discuss the performance of the proposed dual stream network, and compare it with previous methodologies to see how it performs in comparison.

Then we conduct two experiments based on affinity groups. In the first experiment, we take an equal sample of actors from different races and sexes who have been superimposed onto news videos (DeeperForensics 1.0), and compare the detectability of deepfakes among the various groups. For the second experiment, we dive deeper into the disparity between sexes and this time, we also account for the difference in source person. That is, we check for the sex of both the source and the target actors of the deepfaked video, and try to see if there are any disparities within the groups.

3.1. Dataset Description

3.1.1. FaceForensics++

For the implementation of the proposed system, the network models were trained and tested with the FaceForensics++ Dataset. FaceForensics++ is a forensics dataset consisting of 1000 original video sequences that have been manipulated with five automated face manipulation methods namely:

1. DeepFakes
2. Face2Face
3. FaceSwap
4. NeuralTextures
5. FaceShifter

In other words, each of the 1000 real videos have been manipulated by these 5 face manipulation methods. Giving the total of 5000 deepfaked videos (1000 of each type). Table 3.1 shows the number of videos under each type. Note that although there are 5 different manipulation techniques, they are all deepfaked media and hence the model should also classify them as deepfake as well.

Table 3.1 Dataset Description

Manipulation Technique	Classification	No. of Videos
No manipulation	Real	1000
DeepFakes	Deepfake	1000
Face2Face	Deepfake	1000
FaceSwap	Deepfake	1000
NeuralTextures	Deepfake	1000
FaceShifter	Deepfake	1000

The data has been sourced from 977 YouTube videos and all videos contain a trackable mostly frontal face without occlusions which enables automated tampering methods to generate realistic forgeries. Figure 3.1 shows a few sample snapshots from the FaceForensics++ Dataset. The dataset can be accessed from the below link:

https://www.kaggle.com/sorokin/faceforensics

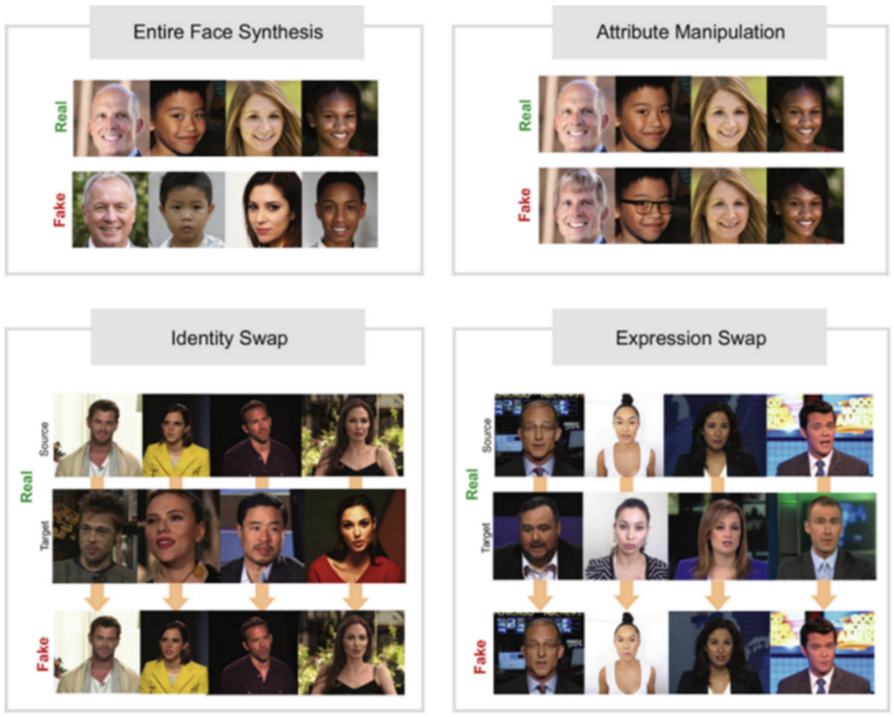


Figure 3.1 Sample Snapshots from the FaceForensics++ Dataset

3.1.2. DeeperForensics 1.0

DeeperForensics 1.0 is a derivative of FaceForensics++, where the owners of the dataset got 100 actors of varying ethnicities and affinity groups to record under different emotions, lighting and circumstances. This dataset consists of over 60000 videos, 10 times larger than any comparable dataset. This dataset accounts for the diversity that is typically lacking in other datasets. That is, for each of the 1000 videos, there are an additional 60 deepfakes created based on different actors chosen at random.

Given the fact that we know the actors who were superimposed onto the source video, we can then use the actor information such as their race and sex, to divide the dataset into videos of deepfake based on different groups of people. This dataset has both sexes (Male and Female) and has actors from four broad racial categories (Caucasian, Asian, Brown, Black). We use this to our advantage for the affinity group based comparative analysis.

The actor information is anonymous, so manual tagging for the skin tone was necessary. However, the sex information was provided by the dataset. Once we mapped all actors to the skin tone, we then form groups and we can use this for the analysis. Figure 3.2 shows how the dataset was created using the actors hired. The dataset can be accessed from the below link:

https://github.com/EndlessSora/DeeperForensics-1.0

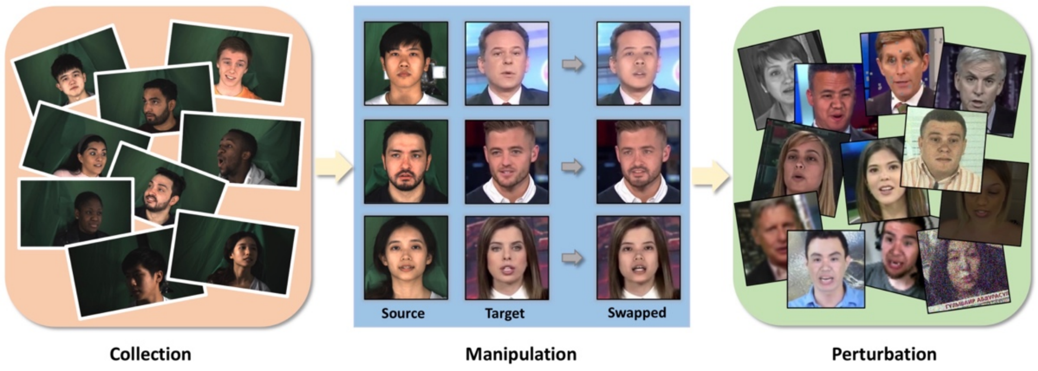


Figure 3.2 DeeperForensics 1.0 Dataset

3.2. Performance Evaluation of Network

3.2.1. Evaluation Metrics

The output obtained from the proposed system can be categorized into four categories (assume that Deepfake video is 'negative' and Real video is 'positive'):

1. Real predicted as Real (TP)
2. Deepfake predicted as Deepfake (TN)
3. Real predicted as Deepfake (FN)
4. Deepfake predicted as Real (FP)

Using this as reference, the Classifier model has been evaluated using the following performance metrics:

Accuracy: the proportion of true results (both true positives and true negatives) among the total number of cases examined (Equation 3.1).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.1}$$

Precision: (also called positive predictive value) - the fraction of relevant instances among the retrieved instances (Equation 3.2).

$$Precision = \frac{TP}{TP+FP} \tag{3.2}$$

Recall: (also known as sensitivity) - the fraction of the total amount of relevant instances that were actually retrieved (Equation 3.3).

$$Recall = \frac{TP}{TP+FN} \tag{3.3}$$

F1-score: - the weighted harmonic means of the test's precision and recall (Equation 3.4).

$$F_1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} = \frac{2TP}{2TP+FP+FN} \qquad (3.4)$$

The four aforementioned evaluation metrics are a standard for any binary classification problem.

3.2.2. RGB Network

After training the RGB model, the training and validation losses are identified for each epoch that the model trained for, and it is plotted in terms of a graph. As you can see from Figure 3.3 the losses decrease every epoch and tapers towards the end. If the model was trained for more epochs, there would not be any significant differences in the reduction of loss.

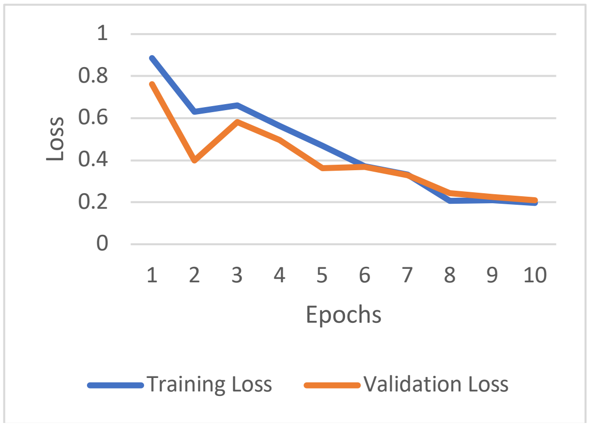
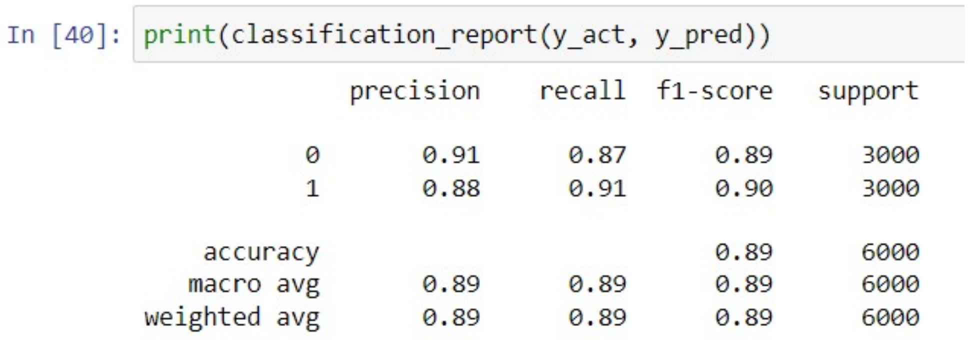


Figure 3.3 Losses vs. Epochs for RGB Model

With the use of the sklearn classification report, the performance measures mentioned above can be calculated. Figure 3.4 shows a snapshot of the classification report generated for the RGB model.



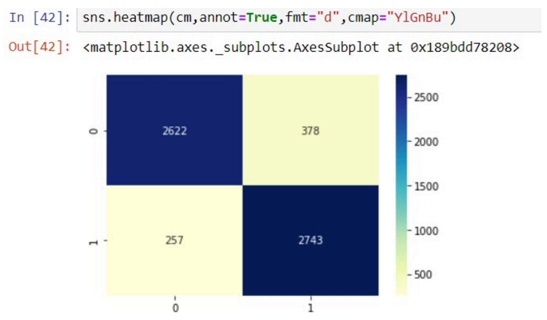
```
In [40]: print(classification_report(y_act, y_pred))
              precision    recall  f1-score   support

           0       0.91      0.87      0.89      3000
           1       0.88      0.91      0.90      3000

    accuracy                           0.89      6000
   macro avg       0.89      0.89      0.89      6000
weighted avg       0.89      0.89      0.89      6000
```

Figure 3.4 Classification Report for RGB Model

Additionally, a confusion matrix can be plot for the same as well. To visually understand it better, it can be plot in the form of a heatmap. By doing so, simply visualizing the diagonals of the matrix, it confirms that the model is working as it was intended. Figure 3.5 shows the confusion matrix heatmap plotted for the RGB model. The correctly predicted boxes are darker, while the incorrectly predicted boxes are lighter in shade.



```
In [42]: sns.heatmap(cm,annot=True,fmt="d",cmap="YlGnBu")
Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x189bdd78208>
```

Figure 3.5 Confusion Matrix for RGB Model

The accuracy is also calculated manually to confirm the robustness of the model. Using Equation 3.1 the RGB model gives an accuracy of 89.42% which a good accuracy percentage.

3.2.3. SRM Network

After training the SRM model, the training and validation losses are identified for each epoch that the model trained for, and it is plotted in terms of a graph. As you can see from Figure 3.6 the losses decrease every epoch and tapers towards the end. If the model was trained for more epochs, there would not be any significant differences in the reduction of loss.

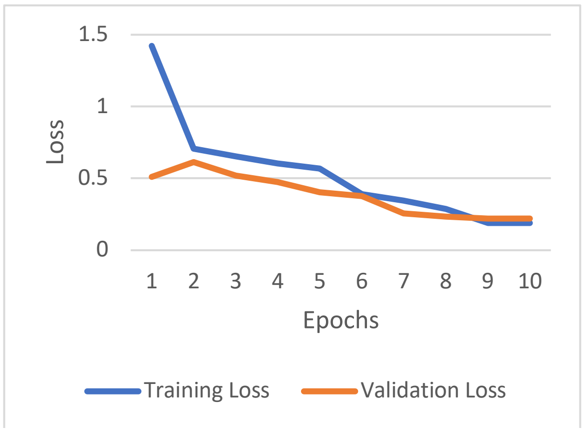
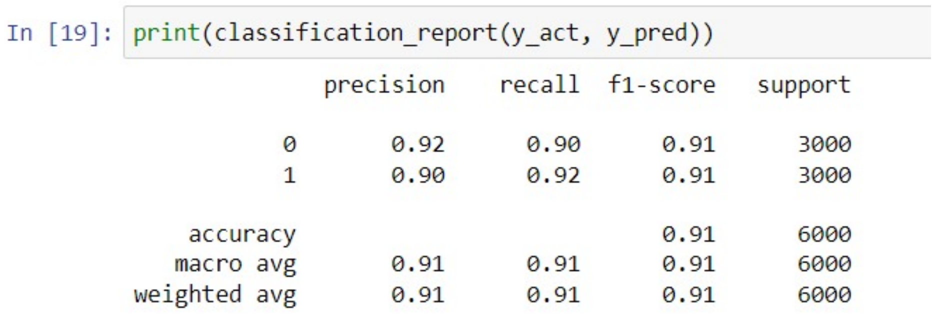


Figure 3.6 Losses vs. Epochs for SRM Model

With the use of the sklearn classification report, the performance measures mentioned above can be calculated. Figure 3.7 shows a snapshot of the classification report generated for the SRM model.



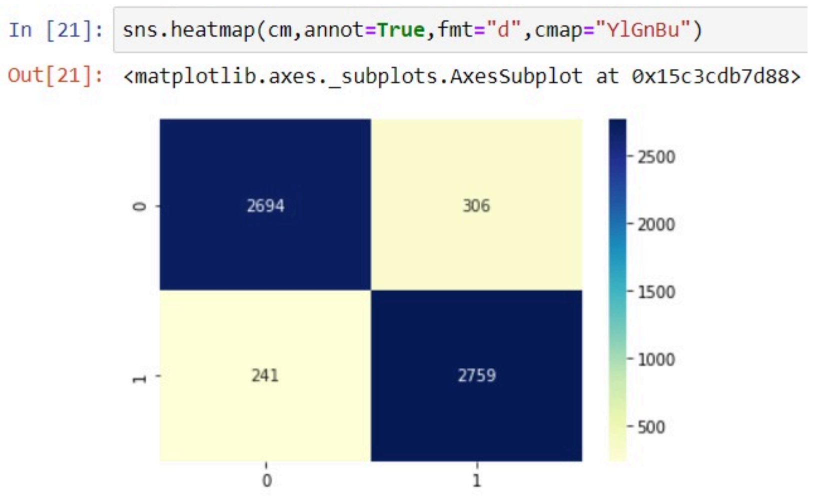
```
In [19]: print(classification_report(y_act, y_pred))
              precision    recall  f1-score   support

           0       0.92      0.90      0.91      3000
           1       0.90      0.92      0.91      3000

    accuracy                           0.91      6000
   macro avg       0.91      0.91      0.91      6000
weighted avg       0.91      0.91      0.91      6000
```

Figure 3.7 Classification Report for SRM Model

Additionally, a confusion matrix can be plot for the same as well. To visually understand it better, it can be plot in the form of a heatmap. By doing so, simply visualizing the diagonals of the matrix, it confirms that the model is working as it was intended. Figure 3.8 shows the confusion matrix heatmap plotted for the SRM model. The correctly predicted boxes are darker, while the incorrectly predicted boxes are lighter in shade.



```
In [21]: sns.heatmap(cm,annot=True,fmt="d",cmap="YlGnBu")
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x15c3cdb7d88>
```

Figure 3.8 Confusion Matrix for SRM Model

The accuracy is also calculated manually to confirm the robustness of the model. Using Equation 3.1 the SRM model gives an accuracy of 90.88% which a better accuracy percentage.

3.2.4. Analysis

Table 3.2 consists of all the necessary performance metrics that the model has tabulated. From this it can be seen that the model has very good accuracy and can be used in real world applications. One area of improvement must be the Deepfake Recall, which indicates that some Deepfaked videos are being identified as real. Although this count is low, it is important to consider in the future.

Table 3.2 Evaluation Metrics Obtained from Proposed Architecture

Implementation	**Accuracy**	**Precision (Real)**	**Precision (Deepfake)**	**Recall (Real)**	**Recall (Deepfake)**	**F1-Score (Real)**	**F1-Score (Deepfake)**
RGB Stream	89.42%	87.89%	91.07%	91.43%	84.40%	89.63%	89.20%
SRM Stream	90.88%	90.02%	91.79%	91.97%	89.80%	90.98%	90.78%
RGB+SRM*	91.07%	90.24%	91.93%	92.10%	90.03%	91.16%	90.97%

*The scores were aggregated by weighted sum (1:1.5, RGB: SRM)

From the given table, it can be seen that while the individual Stream Networks are quite robust on their own, the fusion of the two streams provides far better robustness and added flexibility. It is for this reason that the dual stream methodology was employed in this implementation. As it is easier to visualize data in the form of charts, Figure 3.9 shows the performance metrics of both the streams individually (blue and orange) and together (grey).

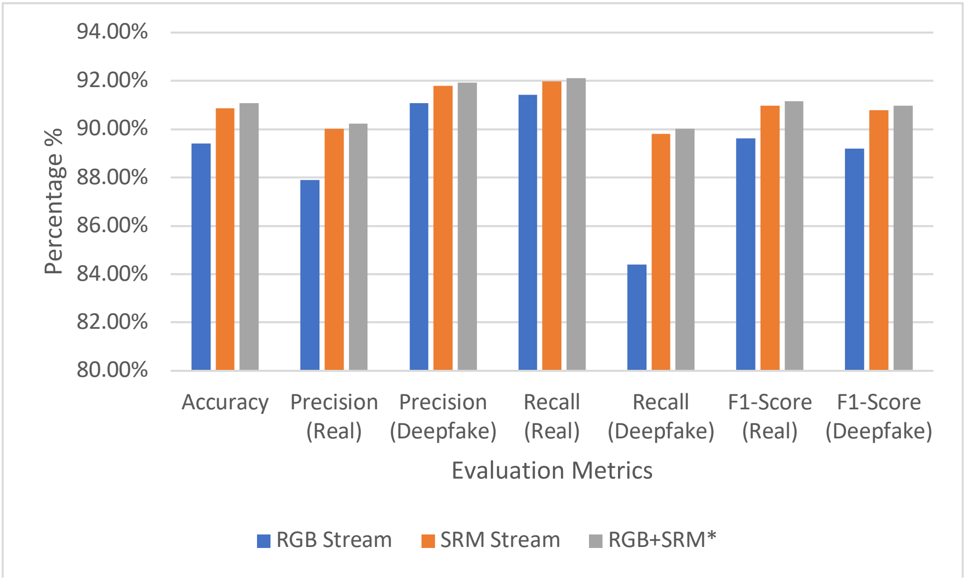


Figure 3.9 Evaluation Metrics from Proposed Architecture

Evidently, the grey bar is the tallest with respect to all performance measures. It is also noticeable that the SRM stream on its own, performs better than the RGB stream on its own, which further supports the reasoning for adding more weightage to the SRM stream outputs.

In order to understand how well a system works, it must be compared to other similar applications and methodologies to view how the proposed system is an improvement. When compared to the previous implementations and solutions to the Deepfake detection problem, the proposed methodology has shown considerable improvements in terms of accuracy, and general interpretability of the CNN model. As shown in Table 3.3, the proposed system is compared you with nine existing implementations.

Table 3.3 Comparison of Proposed System with Existing Methods

Methodology	Frames Extracted	Accuracy (%)
Bayar and Stamm	1	70.01
MesoNet	1	75.65
I3D	8	81.42
XceptionNet	1	85.49
TSM	24 (i.e., 3x8)	86.07
CapsuleNet	8	88.04
TwoStream-LightCNN	8+8	87.86
TwoStream-Res101	8+8	88.21
TwoStream-Xception (previous)	8+8	90.36
Proposed System	8+8	91.07

The table also showcases the number of frames extracted for the implementation of the methodology. It is evident that choosing 8 frames to be extracted was the right call, and is backed by successful previous implementations. In order to visualize the improvement in the method, Figure 3.10 depicts the information in Table 3.3 in the form of a bar graph. As you can see below, out implementation (towards the end in orange) has the highest accuracy of 91.07%. Although the improvement is not that much higher than the previous method, as the trend of the bar graph may show, an increase by even 1% is significant.

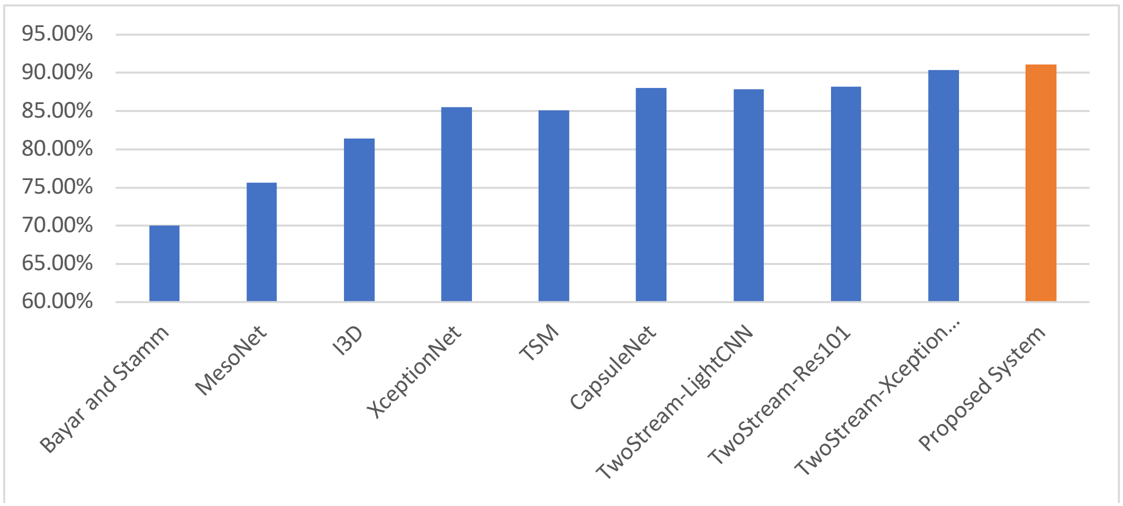


Figure 3.10 Comparative Analysis of Accuracy

Additionally, the model can be successfully deployed to detect deepfakes created by more techniques. While previous implementations have only shown success in detecting morphed faces created by DeepFakes, Face2Face and FaceSwap, the proposed methodology additionally differentiates well for Neural Textures and FaceShifter as well. This was shown when seeing the test cases pass through the model.

3.3. Comparative Analysis of Different Affinity Groups

3.3.1. Experiment

For the first experiment, as mentioned earlier, we take equal deepfake videos from various affinity groups and pass them through the Deepfake Detection system, and see how many are correctly identified as deepfake. Here, we consider only the affinity group of the source actor irrespective of the target actor. We are able to do this due to the knowledge of each individual actor superimposed onto a video from the FaceForensics++ dataset (known as the DeeperForensics 1.0 dataset). We don't have the same information for the target actor, as the source videos are from FaceForensics++ dataset, which does not contain such information and is harder to tag manually.

We consider two different attributes when distributing the actors between the affinity groups:

- **Race/Skin Tone:**
    - Caucasian
    - Asian
    - Brown
    - Black
- **Sex/Gender:**
    - Male
    - Female

While the actors were tagged with their respective sex, the same cannot be said for the race. Hence, manual tagging was performed for all 100 actors.

Using the two categories, we can form eight affinity groups (simply a cross product of the elements of each attribute). We randomly sample 500 videos for each category from the DeeperForensics dataset for a total of 4000 deepfaked videos. These videos were then passed through the network and classified as either Deepfake or Real. For a correctly classified video, the classification must be Deepfake.

We then note down the results of the prediction in the form of percentage correctly identified as deepfake. As the number of videos of each group is the same, we will have comparable results.

3.3.2. Results

All 4000 videos have been passed through the network and the following results were obtained based on the eight different affinity groups as shown in Table 3.4:

Table 3.4 Prediction Accuracy of the Different Affinity Groups

Race / Sex	Caucasian	Asian	Brown	Black	Total
Male	92.2	90.2	95.2	94	92.9
Female	91	85.2	88.6	91.4	89.05
Total	91.6	87.7	91.9	92.7	90.975

*All values are in %

To better understand these results, let us take a glance at Figure 3.11. It is the graphical representation of the prediction accuracy of each of the eight affinity groups. As shown, the group that had the highest prediction rate were 'Brown Male' with 95.2%, while the group with the lowest prediction rate were 'Asian Female' with 85.2%. That is a 10% gap in prediction accuracy among the two groups.

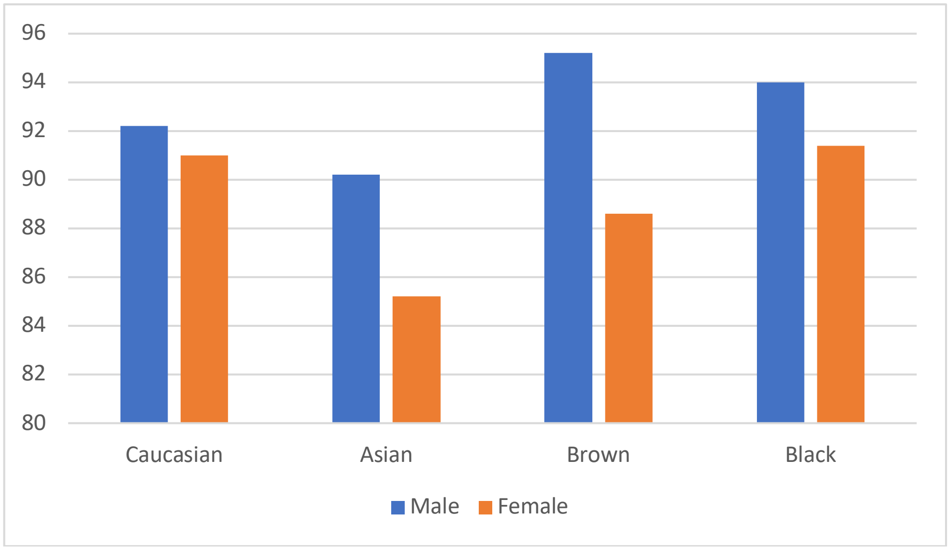


Figure 3.11 Prediction Accuracy of the Different Affinity Groups

Taking just race as the differentiating attribute, we obtain the graph as shown in Figure 3.12. The values were obtained by considering both Male and Female groups as one single group for each race. As we can observe, the group that had the highest prediction rate were ‘Black’ with 92.7%, while the group with the lowest prediction rate were ‘Asian’ with 87.7%. It is also worth noting that the other two groups have values above 91% (i.e., closer to the highest value than the lowest value). In other words, the lowest racial group is significantly lower than the other groups.

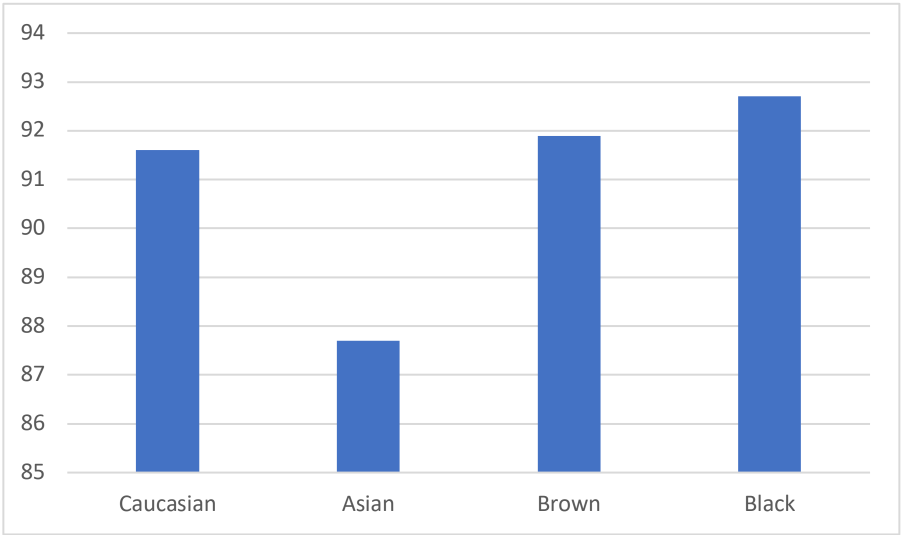


Figure 3.12 Prediction Accuracy of the Different Races

Similar to above, we can take just sex as the differentiating attribute. Figure 3.13 shows the graph obtained by considering all males and females in the video list as two separate groups. As we can see the ‘Male’ group has a higher prediction accuracy of 92.9% than that of the ‘Female’ group which has a prediction accuracy of 89.05%. While this disparity isn’t as large as one would prefer, it is still a significant difference worth noting.

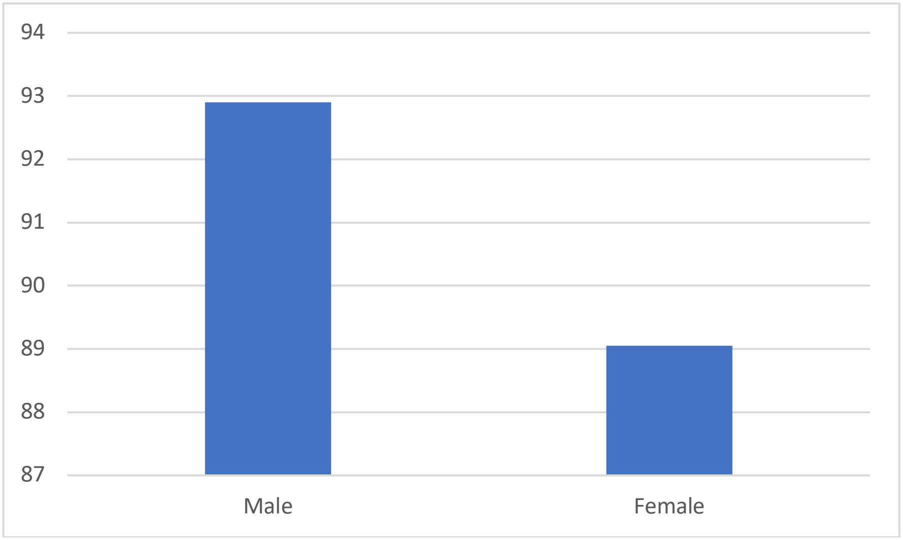


Figure 3.13 Prediction Accuracy of the Different Sexes

3.3.3. Inferences

To make meaningful inferences, we must first understand what the results are indicating, and come up with valid reasons that would account for the result obtained.

The very first observation that can be made is that, for each race on its own, the Male sex has a higher prediction rate than that of the Female sex. In fact, with the exception of the 'Asian Male' group, all other Male groups have higher prediction rates than any Female groups. Even taking sex as the differentiating attribute, it is obvious that the Male group has a higher prediction rate than that of the Female group. When a group has a better prediction rate, it is understood that it is easier to identify a deepfake video of videos using people from that affinity group. Hence, we can infer that on average, deepfake videos of Males are typically easier to identify as fake than those of Females.

Now let us consider the difference in race. In almost all situations, we can observe that 'Asian' affinity observes a lower prediction rate. Whether it is on a general level, or even considering just Males and Females individually, we can observe that the prediction rate of the Asian race is significantly lower than the other races. We can infer that on average, deepfake videos of Asians are typically harder to identify as fake than those of other races.

Other observable results which are significant are as follows:

- When considering all Male groups, the Brown affinity group had the highest prediction rate, followed by the Black race and Caucasian race.
- When considering all the Female groups, the Black affinity group had the highest prediction rate, followed by the Caucasian race and Brown race.
- When considering all groups with race being the differentiating attribute, the Black affinity group had the highest prediction rate, followed by the Caucasian race and Brown race.

A valid conclusion that can be made from the following observations and results is that facial attributes play a major role in determining whether a video is deepfake or not, and when there is more differences in facial attributes, it is easier to detect deepfakes.

This conclusion can be proven by seeing that Male deepfake videos are easier to identify as fake than Female deepfake videos. Typically for Males, the differentiating facial features are usually, facial hair, facial structure, skin texture, etc., while for Females, the differentiating facial features are usually their makeup and face shape. This is because, typically Females have makeup when appearing in reporting news, and skin texture doesn't play a role, but rather the style of makeup. They also lack facial hair such as mustaches and beards, which vary a lot with Males, but since it is nonexistent with Females, there isn't much of a differentiating factor. Hence, we observe better prediction rates for Men.

A similar conclusion can be drawn when considering race. Typically speaking, in comparison, there are significantly lesser Asian people with varying facial features. Women typically wear similar make up and have similar facial structure, while men usually don't don any facial hair, making it one less differentiating feature to help detect the deepfake. Meanwhile, among other races, especially Brown and Black people, there is a lot more difference in facial structure and facial hair. Hence it is safe to assume that such differentiating features play a huge role in detecting deepfakes.

Hence, from this experiment we learn that the distinctive facial features of the actors determine how easily the videos can be identified as deepfake.

3.4. Comparative Analysis of Sex-Based Deepfake Mappings

3.3.1. Experiment

For the second experiment, as mentioned earlier, we take equal deepfake videos from the two different sexes of actors, and then similarly pass them through the Deepfake Detection system, and see how many are correctly identified as deepfake. Unlike the last experiment, we take into consideration both actors in a Deepfaked video: the source actor and the target actor. While it was difficult to manually tag the race of the target actors in the FaceForensics++ dataset, it is much simpler to tag the sex/gender of the reporter. Hence, by doing that, we now have information of both the source and target actor, with respect to their sex. The actors of the DeeperForensics 1.0 dataset were already tagged with the corresponding sexes.

Using the information of both the source and the target actor, we can form four different categories (simply the cross product of the sex information of the two actors). We randomly sample 250 videos of each category from the DeeperForensics dataset for a total of 1000 deepfaked videos. We do this by sampling 50 random real videos of the from the FaceForensics++ dataset for each sex of the target actor and then we sample 5 actors for each video from each sex of the source actor. These videos were then passed through the network and classified as either Deepfake or Real. For a correctly classified video, the classification must be Deepfake.

We then note down the results of the prediction in the form of percentage correctly identified as deepfake. As the number of videos of each group is the same, we will have comparable results.

3.3.2. Results

All 1000 videos have been passed through the network and the following results were obtained based on the eight different affinity groups as shown in Table 3.5:

Table 3.5 Prediction Accuracy of the Different Affinity Groups

Source Actor Sex / Target Actor Sex	Male	Female	Total
Male	88.8	96.4	92.6
Female	93.2	86	89.6
Total	91	91.2	91.1

*All values are in %

To better understand these results, let us take a glance at Figure 3.14. It is the graphical representation of the prediction accuracy of each of the categories. As shown, the group that had the highest prediction rate were 'Male (source) superimposed on Female (target)' with 96.4%, while the group with the lowest prediction rate were 'Female (source) superimposed on Female (target)' with 86%. That is a 10.4% gap in prediction accuracy among the two groups.

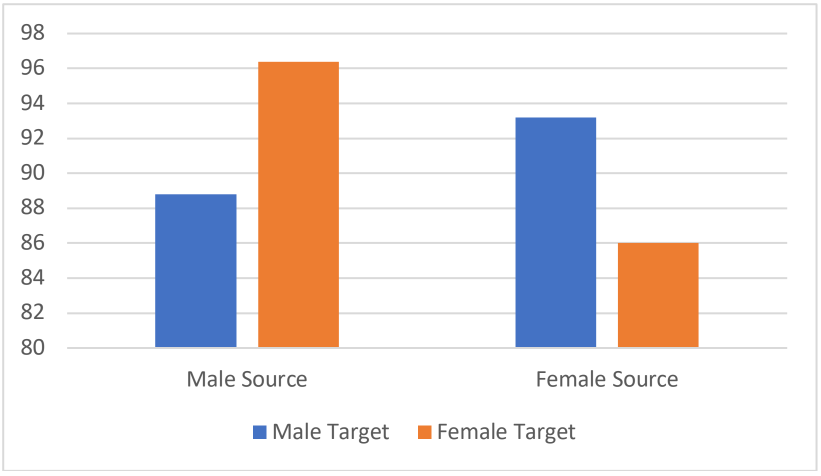


Figure 3.14 Prediction Accuracy of the Sex-Based Deepfake Mappings

In order to better visualize these particular results, Figure 3.15 shows a heatmap representation of the values shown in Table 3.5. As we can see, the prediction accuracy for categories with mismatching sex between the source actor and the target actor (i.e., 'Male (source) superimposed on Female (target)' and 'Female (source) superimposed on Male (target)') were significantly higher than that of matching sex between the source actor and target actor (i.e., 'Male (source) superimposed on Male (target)' and 'Female (source) superimposed on Female (target)').

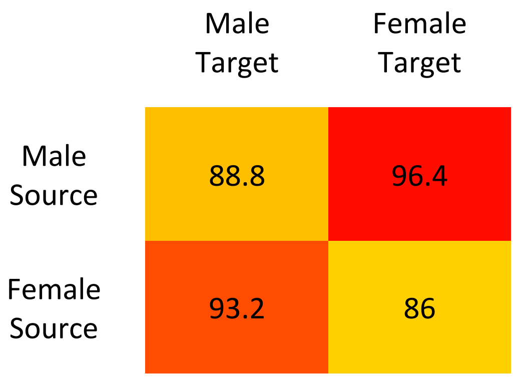


Figure 3.15 Heatmap of the Sex-Based Deepfake Mappings

3.3.3. Inferences

Similar to the first experiment, to make meaningful inferences, we must first understand what the results are indicating, and come up with valid reasons that would account for the result obtained.

The most obvious observation that can be made, is as mentioned before, when the source actor and target actor have mismatching sexes. That is when a Female was superimposed and a Male and vice versa. This means, that when the sexes were mismatched, it is easier to detect that the video was deepfake, when compared to when the sexes were matched.

It should also be noted, that in both the matching and mismatching scenarios, when a Male actor (source) was used to superimpose onto any person, the prediction rate was higher than when a

Female actor was used. This was also observed in the previous experiment when Males group have a higher prediction rate than that of Females.

Comparing again to the first experiment, we can draw this conclusion from the fact that distinctive facial features play a major role in determining whether a video is deepfake or not. When superimposing a male onto a female, there will obviously be a mismatch in the basic facial structure. This is the same vice versa as well. It is especially easier to detect deepfakes when a male actor has been superimposed onto a female actor. We have previously already established that males in general have more differentiating facial features than females. Hence, mapping a male onto a female could result in facial hair on a face that is shaped similar to a woman, making it more likely to be predicted as fake. Meanwhile, when the sex is matched, the facial structure tends to be similar and is harder to detect as fake. However, male videos are easier to detect as fake than females do to the existence of more differentiating facial features, for example, facial hair, prominent facial structure, etc.

Another interesting result to note, is that there is almost no difference when comparing any actor superimposed onto Male targets and any actor superimposed onto Female targets. This result indicates that the target person does not matter when it comes to detecting deepfake videos. This makes sense, as deepfake videos, don't retain or retain very minute original details of the target actor. Deepfakes are the superimposition of one's face onto another completely hiding the previous face, with the source face.

Hence the observations we can learn from this experiment are as follows:

- When the source and target actors are not from similar affinity groups, it is easier to detect as deepfake.
- When an affinity group has more distinctive facial features, it is easier to identify as deepfake.
- The features or information of the target actor has little to do with determining whether the video is deepfake or not.

4. Conclusion

In conclusion we were able to conduct experiments in the field of deepfakes that provided useful results that can be used to improve on detection methods in future works. We were able to create and train a system that is on par, if not better than the state-of-the-art system currently available. We utilized two network streams to account for both the semantic and finer details of deepfakes. We established the significant role of the source actor for a deepfake video and the lack of significance of the target actor. We also were able to establish how differentiating facial features of various affinity groups contributes to better detection of deepfake videos.

Future work on this project can be to put to use the information gained from this experiment, to compile new datasets that will account for the discrepancies and lower prediction rates that were observed with the various affinity groups. We can also experiment with new training methods that account for these discrepancies as well. Additionally, we can expand upon these experiments to be more specific to individual countries based on their cultural, political and social differences as well. This can be done provided that there are new datasets available with enough information to be more specific.

References

[1] Maksutov, V. O. Morozov, A. A. Lavrenov and A. S. Smirnov, (2020), "Methods of Deepfake Detection Based on Machine Learning," 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2020, pp. 408-411, doi: 10.1109/EIConRus49466.2020.9039057.

[2] Bayar and M. C. Stamm, (2016), "A deep learning approach to universal image manipulation detection using a new convolutional layer," in Proc. 4th ACM Workshop Inf. Hiding Multimedia Security, 2016, pp. 5–10.

[3] Han, X. Han, H. Zhang, J. Li and X. Cao, (2021), "Fighting Fake News: Two Stream Network for Deepfake Detection via Learnable SRM," in IEEE Transactions on Biometrics, Behavior, and Identity Science, vol. 3, no. 3, pp. 320-331, July 2021, doi: 10.1109/TBIOM.2021.3065735.

[4] Malolan, A. Parekh and F. Kazi, (2020), "Explainable Deep-Fake Detection Using Visual Interpretability Methods," 2020 3rd International Conference on Information and Computer Technologies (ICICT), 2020, pp. 289-293, doi: 10.1109/ICICT50521.2020.00051.

[5] Z. Yang, J. Ma, S. Wang and A. W. -C. Liew, (2021), "Preventing DeepFake Attacks on Speaker Authentication by Dynamic Lip Movement Analysis," in IEEE Transactions on Information Forensics and Security, vol. 16, pp. 1841-1854, 2021, doi: 10.1109/TIFS.2020.3045937.

[6] Cozzolino, G. Poggi, and L. Verdoliva, (2017), "Recasting residual-based local descriptors as convolutional neural networks: An application to image forgery detection," in Proc. 5th ACM Workshop Inf. Hiding Multimedia Security, 2017, pp. 159–164.

[7] H. A. Khalil and S. A. Maged, (2021), "Deepfakes Creation and Detection Using Deep Learning," 2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), 2021, pp. 1-4, doi: 10.1109/MIUCC52538.2021.9447642.

[8] H. H. Nguyen, J. Yamagishi, and I. Echizen, (2019), "Capsule-forensics: Using capsule networks to detect forged images and videos," in Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP), Brighton, U.K., 2019, pp. 2307–2311.

[9] J. Hu, X. Liao, W. Wang and Z. Qin, (2021), "Detecting Compressed Deepfake Videos in Social Networks Using Frame-Temporality Two-Stream Convolutional Network," in IEEE Transactions on Circuits and Systems for Video Technology, doi: 10.1109/TCSVT.2021.3074259.

[10] J. Yang, A. Li, S. Xiao, W. Lu and X. Gao, (2021) "MTD-Net: Learning to Detect Deepfakes Images by Multi-Scale Texture Difference," in IEEE Transactions on Information Forensics and Security, vol. 16, pp. 4234-4245, 2021, doi: 10.1109/TIFS.2021.3102487.

[11] M. C. El Rai, H. Al Ahmad, O. Gouda, D. Jamal, M. A. Talib and Q. Nasir, (2020), "Fighting Deepfake by Residual Noise Using Convolutional Neural Networks," 2020 3rd International Conference on Signal Processing and Information Security (ICSPIS), 2020, pp. 1-4, doi: 10.1109/ICSPIS51252.2020.9340138.

[12] M. Weerawardana and T. Fernando, (2021), "Deepfakes Detection Methods: A Literature Survey," 2021 10th International Conference on Information and Automation for Sustainability (ICIAfS), 2021, pp. 76-81, doi: 10.1109/ICIAfS52090.2021.9606067.

[13] Y. Li et al., (2021), "DeepFake-o-meter: An Open Platform for DeepFake Detection," 2021 IEEE Security and Privacy Workshops (SPW), 2021, pp. 277-281, doi: 10.1109/SPW53761.2021.00047.

[14] Y. Nirkin, L. Wolf, Y. Keller and T. Hassner, (2021), "DeepFake Detection Based on Discrepancies Between Faces and their Context," in IEEE Transactions on Pattern Analysis and Machine Intelligence, doi: 10.1109/TPAMI.2021.3093446.

[15] Z. Joseph and C. Nyirenda, (2021), "Deepfake Detection using a Two-Stream Capsule Network," 2021 IST-Africa Conference (IST-Africa), 2021, pp. 1-8.