National College of Ireland

MSc Research Project

MSc Data Analytics

Tej Rup Sai Munagala

Student ID: x19196628

School of Computing

National College of Ireland

Supervisor: Cristina Muntean

I. Why CNN model was investigated in this research?

After an extensive literature review, the most compelling results are found to be from various CNN models like MoesInception-4 CNN model [1], Inception V3 model [2], and Sequential CNN RNN model [3]. These models resulted with higher accuracy of 99% and took lesser epochs to run the model. These CNN models also include LSTM features, Adam and tanh optimizers and Max pooling layers which helps in reducing the run time of the model and improves the prediction accuracy.

However, other models like Y-shaped autoencoder decoder have shown an accuracy of about 92% but the model needed to run for 50 epochs which is for longer period of time for model compilation [4] and few other deep learning models shown lesser accuracy in deepfake detection. Therefore, different layered approach of CNN model is chosen for this deepfake detection research work.

II. Describe a real-life scenario that would benefit of your research outcome?

This research outcome may result numerous possible merits in various fields like forensics detection and multi-media authentication. For instance, by using these deepfake detection methodologies in forensics detection can protect many world-famous[1] leaders by watermarking their deepfake content over internet. Recently, Facebook has updated its privacy policies to stop the circulation of deepfake content that helped US 2020 election by not misleading the public by deepfake news[2].

Thus, this research work results can help fellow researchers and deepfake detection research community for improving the deepfake detection tools and helps them this subject to a deeper extent.

III. In section 3.3 (Data Preparation) is mentioned that was applied pre-processing techniques to the data, but no details are provided. Can you detail which pre-processing techniques were applied? Also in relation to the ROI detection mentioned, it should be provided details about this implementation and results?

The data preprocessing typically include two major steps:

***Step-1: Conversion of videos into frames**:*

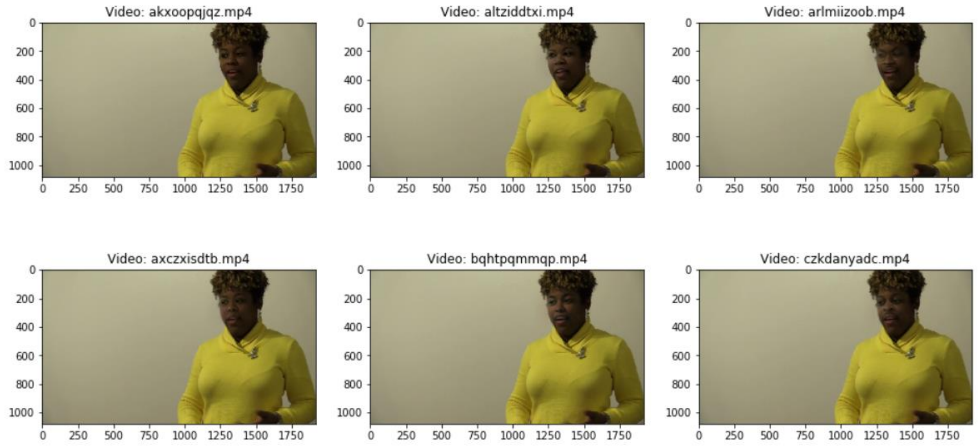
[1] MIT technology review: www.technologyreview.com/2019/06/21/134815/a-new-deepfake-detection-tool-should-keep-world-leaders-safefor-now/

[2] Theguardian: www.theguardian.com/technology/2020/jan/07/facebook-bans-deepfake-videos-in-run-up-to-us-election

At this stage of the data preprocessing, each individual training video is converted into various frames as specified below:

```
In [24]: same_original_fake_train_sample_video = list(meta_train_df.loc[meta_train_df.original=='meawmsgiti.mp4'].index)
         display_image_from_video_list(same_original_fake_train_sample_video)
```

```
<Figure size 432x288 with 0 Axes>
```



Step-2: Detection of facial features from all the videos:

Facial features are detected by using cascading tools like haarcascade and blazeface packages. In order to identify these facial features, user defined classes like ObjectDetector(), extract_image_objects(), and detect() are used to capture the facial features particularly with help of blazeface packages. These object classes helped in identifying the following facial features like eyes, nose, frontal face, and smile from all the videos. The illustration of the user-defined classes as follows:

1) **ObjectDetector():** This class is created to cascade the facial features from the images.

In [27]:
```
class ObjectDetector():
    def __init__(self,object_cascade_path):
        self.objectCascade=cv2.CascadeClassifier(object_cascade_path)

    def detect(self, image, scale_factor=1.3,
               min_neighbors=5,
               min_size=(20,20)):
        rects=self.objectCascade.detectMultiScale(image,
                                                  scaleFactor=scale_factor,
                                                  minNeighbors=min_neighbors,
                                                  minSize=min_size)
        return rects
```

2) **Variables Declaration:** fd, ed, pd, sd variables are assigned to capture the facial features like smile, eye, smile, and front face are created by using haarcascade packages.

In [28]:
```
frontal_cascade_path= os.path.join(FACE_DETECTION_FOLDER,'haarcascade_frontalface_default.xml')
eye_cascade_path= os.path.join(FACE_DETECTION_FOLDER,'haarcascade_eye.xml')
profile_cascade_path= os.path.join(FACE_DETECTION_FOLDER,'haarcascade_profileface.xml')
smile_cascade_path= os.path.join(FACE_DETECTION_FOLDER,'haarcascade_smile.xml')

fd=ObjectDetector(frontal_cascade_path)

ed=ObjectDetector(eye_cascade_path)

pd=ObjectDetector(profile_cascade_path)

sd=ObjectDetector(smile_cascade_path)
```

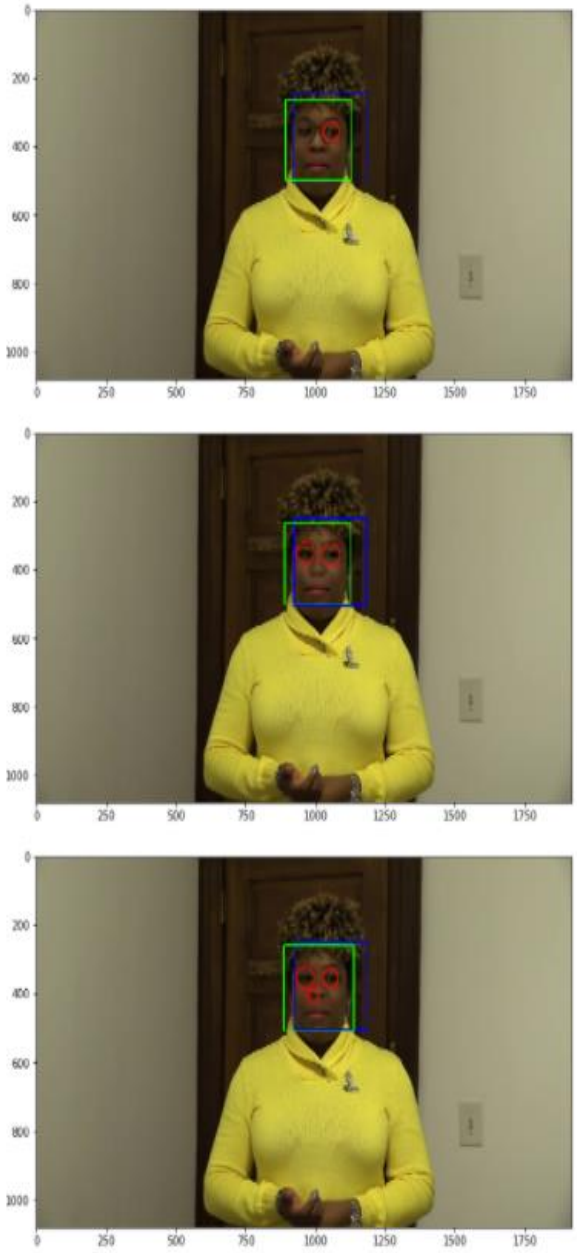
3) **Extract_image_object():** This class is defined to extract facial features from the training videos.

In [33]:
```
def extract_image_objects(video_file, video_set_folder=TRAIN_SAMPLE_FOLDER):
    video_path = os.path.join(DATA_FOLDER, video_set_folder,video_file)
    capture_image = cv2.VideoCapture(video_path)
    ret, frame = capture_image.read()
    #frame = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
    detect(image=frame,
           scale_factor=1.3,
           min_neighbors=5,
           min_size=(50, 50))
```

4) **Haarcascade Implementation:** All the user defined classes are incorporated to extract facial features frames**.**

```
In [39]: same_original_fake_train_sample_video = list(meta_train_df.loc[meta_train_df.original=='kgbkktcjxf.mp4'].index)
         for video_file in same_original_fake_train_sample_video[1:4]:
             print(video_file)
             extract_image_objects(video_file)
```

```
byqzyxifza.mp4
cwrtyzndpx.mp4
cwwandrkus.mp4
```



5) **Region of Interest (ROI) methodology**: ROI is implemented to identify the specific facial region of the frames. ROI () user defined class is defined indicated in figure 1 and its implemented on the extracted frontal cascaded images of the dataset and this class helps in extracting facial features accurately as shown in the figure 2.

```
In [41]: def ROI(img):

             face_img = img.copy()

             face_rects = face_cascade.detectMultiScale(face_img,scaleFactor=1.3, minNeighbors=5)

             for (x,y,w,h) in face_rects:
                 roi = face_img[y:y+256,x:x+256]

             try:
                 return roi
             except:
                 return []
```

Figure 1 - ROI class

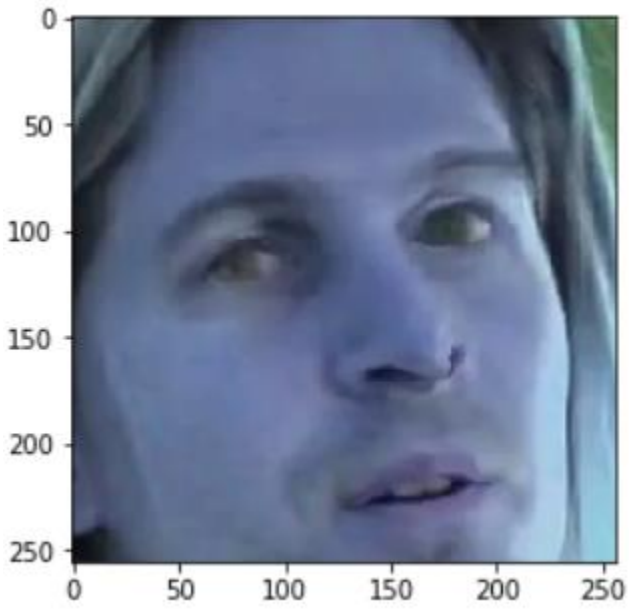


Figure 2 - Output of ROI methodology

The ROI methodology is applied on all videos of the dataset and the extracted frames are split into training and test sets for the validation.

IV. In section 3.3.2 is also mentioned that a frame of dimension 299*299 was used. Can you specify why this size was used?

The default input frame size of the Inception-V3 CNN model is 299*299. Thus, the default frame size is used for running the model [5].

V. In section 6 is mention that "The hyperparameters of the CNN model are tuned for better prediction accuracy." Can you specify which hyperparameters were used in this tuning process and the results obtained?

As part of the research, the hyper parameter tunning is implemented on the CNN model in three different ways as specified below:

Model 1:

Initially, a sequential CNN model was executed with a total of 3,745,890 parameters as shown in the figure 3 and the model resulted with an accuracy of 80 per cent that can be inferred from the figure 4. During this stage, a total of 5 layers of successive convolutional neural networks are used and, at each layer, Maxpooling2D and relu activation functions are used for processing the model.

Model 2:

The hyperparameters of the sequential CNN model are tuned further for better prediction accuracy. During the reimplementation of CNN model, the number of layers of neural network are reduced to 4 layers and at each layer, batch normalization is introduced along with Maxpooling2D and relu activation function. The batch normalization is used to regularize the output frames and it also helps in reducing the number of trainable parameters. During this stage, there are about 20,905 trainable parameters as shown in figure 5 and this parameterized model has shown an accuracy about 81 per cent as displayed in figure 6 and there is only one per cent increase noticed in the prediction accuracy. However, the time taken to train the model is reduced with an improved accuracy.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 254, 254, 32)      896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 252, 252, 32)      9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 126, 126, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 124, 124, 64)      18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 122, 122, 64)      36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 61, 61, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 59, 59, 128)       73856
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 57, 57, 128)       147584
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 55, 55, 128)       147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 27, 27, 128)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 25, 25, 256)       295168
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 23, 23, 256)       590080
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 21, 21, 256)       590080
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 10, 10, 256)       0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 8, 8, 256)         590080
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 6, 6, 256)         590080
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 4, 4, 256)         590080
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 2, 2, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 64)                65600
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 130
=================================================================
Total params: 3,745,890
Trainable params: 3,745,890
Non-trainable params: 0
_________________________________________________________________
```

Figure 3 - CNN model

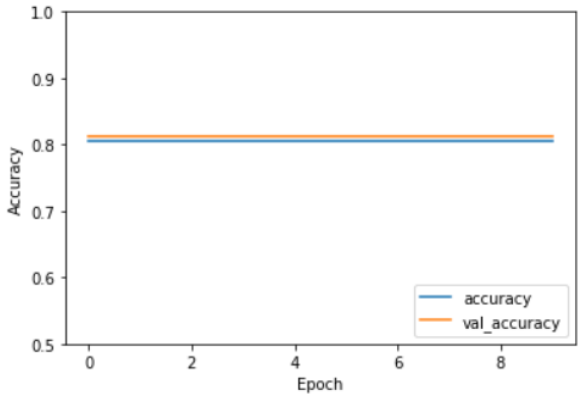


Figure 4 - Accuracy result of CNN model

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_13 (Conv2D)           (None, 254, 254, 8)       224
_________________________________________________________________
batch_normalization (BatchNo (None, 254, 254, 8)       32
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 127, 127, 8)       0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 123, 123, 8)       1608
_________________________________________________________________
batch_normalization_1 (Batch (None, 123, 123, 8)       32
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 61, 61, 8)         0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 57, 57, 16)        3216
_________________________________________________________________
batch_normalization_2 (Batch (None, 57, 57, 16)        64
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 28, 28, 16)        0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 24, 24, 16)        6416
_________________________________________________________________
batch_normalization_3 (Batch (None, 24, 24, 16)        64
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 6, 6, 16)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_2 (Dense)              (None, 16)                9232
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 17
=================================================================
Total params: 20,905
Trainable params: 20,809
Non-trainable params: 96
```

Figure 5 - Hyper parameterized CNN model

```
model_history[1][['loss','auc','accuracy','val_loss','val_auc','val_accuracy']]
```

	loss	auc	accuracy	val_loss	val_auc	val_accuracy
0	0.693147	0.5	0.80625	0.693147	0.5	0.8125
1	0.693147	0.5	0.80625	0.693147	0.5	0.8125
2	0.693147	0.5	0.80625	0.693147	0.5	0.8125
3	0.693147	0.5	0.80625	0.693147	0.5	0.8125

Figure 6 - Accuracy results of hyper parameterized model

Model 3: A transfer learning technique is implemented by using both CNN and RNN models. Here, inception V3 CNN model is the baseline model for initializing the weights for all the frames that are inputted and RNN model is used for adding the LSTM (long short-term memory) layers to neural architecture. The RNN model also loads activation functions like softmax and tanh, dropout parameters as shown in figure 7. Upon running the transfer learning model for 522 epochs, the model resulted with an accuracy outcome of about 82 per cent for a total of 2,426,626 parameters. The model prediction on the test dataset has shown on figure 9 & 10, the model is able to distinguish the fake video in test videos with an accuracy of 80 per cent as shown in figure 8.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 256)               2360320
_________________________________________________________________
dense (Dense)                (None, 256)               65792
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 514
=================================================================
Total params: 2,426,626
Trainable params: 2,426,626
Non-trainable params: 0
```

Figure 7 - Transfer learning model

```
history.history['accuracy']
```

```
[0.8256705]
```

Figure 8 - Sequential model accuracy

Figures 9 and 10 show the predicted values of sequential model for each individual video present in the test dataset.

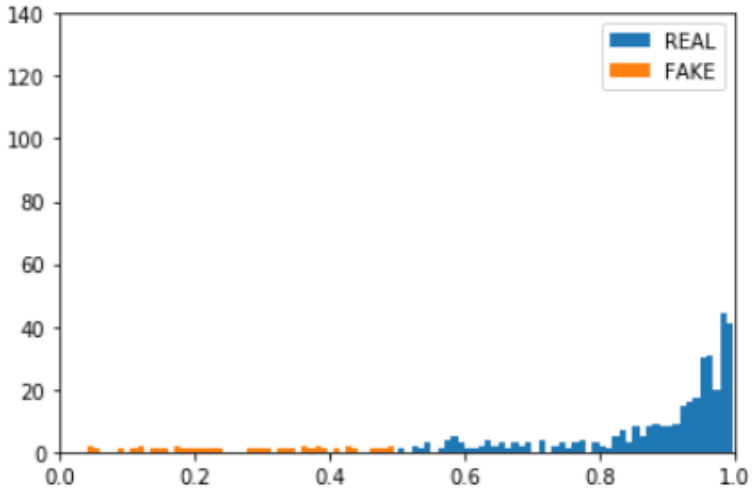


Figure 9 - Real and Fake video distribution in test dataset

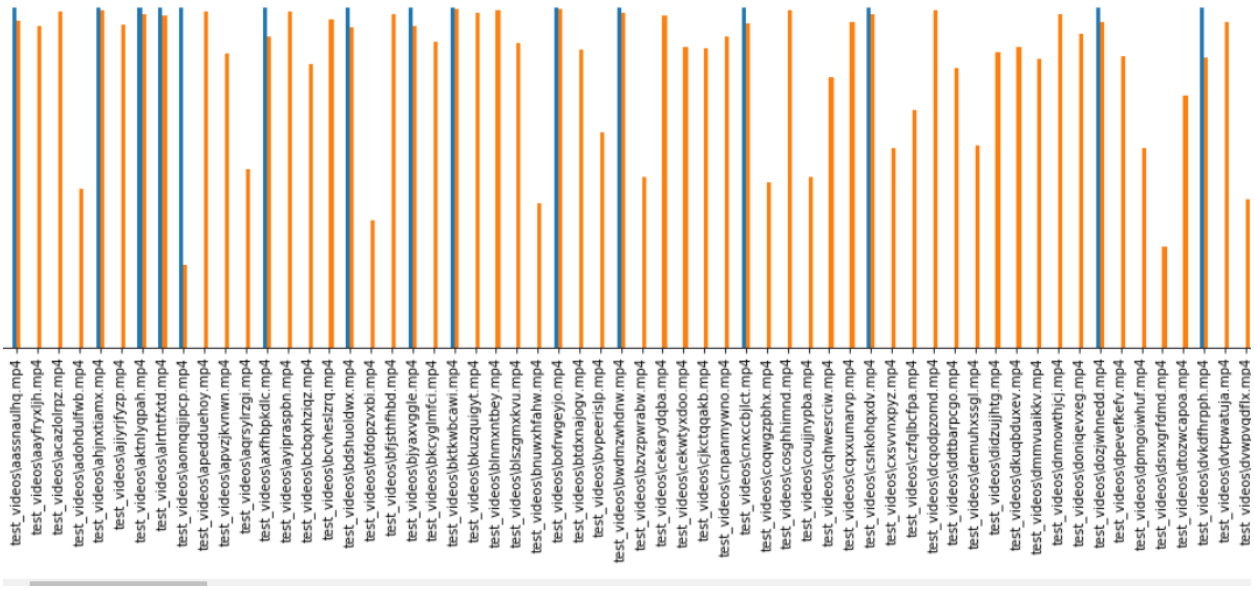


Figure 10 - Comparison of individual test video ground truth value with predicted value

References:

[1] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, “MesoNet: a Compact Facial Video Forgery Detection Network,” 2018 IEEE International Workshop on Information Forensics and Security (WIFS), Dec. 2018, doi: 10.1109/wifs.2018.8630761.

[2] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” www.cv-foundation.org, 2016. https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Szegedy_Rethinking_the_Inception_CVPR_2016_paper.html.

[3] D. Guera and E. J. Delp, “Deepfake Video Detection Using Recurrent Neural Networks,” 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Nov. 2018, doi: 10.1109/avss.2018.8639163.

[4] H. H. Nguyen, F. Fang, J. Yamagishi, and I. Echizen, “Multi-task Learning For Detecting and Segmenting Manipulated Facial Images and Videos,” arxiv.org, Jun. 2019, [Online]. Available: https://arxiv.org/abs/1906.06876.

[5] Q. Guan *et al.*, “Deep convolutional neural network VGG-16 model for differential diagnosing of papillary thyroid carcinomas in cytological images: a pilot study,” *Journal of Cancer*, vol. 10, no. 20, pp. 4876–4882, 2019, doi: 10.7150/jca.28769.