**Phase 2 Weekly Report**

**Course Initial and name: Senior Project Design II ( 499B)**

**Project Name: Deepfake video detection**



Project Supervisor: Md. Ashrafuzzaman Khan (Azk)

Section: 07

Group Members:

Shayekh Mohiuddin Ahmed Navid-153124702

[Shamima Haque Priya-1620069042](https://github.com/fatema13/CSE499-04-FakeVideoDetection/tree/master/Shamima%20Haque%20Priya-1620069042)

[Umme Kulsum Ritu- 1511619042](https://github.com/fatema13/CSE499-04-FakeVideoDetection/tree/master/Umme%20Kulsum%20Ritu-%201511619042)

[Fatema Tuz Zohra- 1620191042](https://github.com/fatema13/CSE499-04-FakeVideoDetection/tree/master/Fatema%20Tuz%20Zohra-%201620191042)

Department of Electrical & Computer Engineering

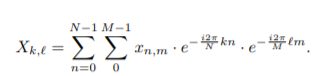
North South University

**To be continued after phase 1**

**Week 5 (16/02/2020-22/02/2020)**

After phase 1, there was a group study of different papers that was provided by the facebook deepfake detection challenge. The papers were studied and different methods were implemented. Two of the papers that were chosen for implementation were Mesonet and Deepfake face extraction using simple features. The code that was provided along with the ‘Deepfake face extraction using simple features’ paper was preprocessed and was used for model training. Different methods of machine learning were applied here. During the initial stage, SVM was used for training the model, followed by random forest. After the implementation of these methods, a simple neural network was used to train the model to check and observe a better output.

In **simple features**, two different methods are being implemented. Fourier transformation and Radial profile. Within this paper they address the issue, namely fake faces, for detecting certain artificial image material. They implement a modern machine-based learning approach to assess the essence of such diagrams. Their method is based on a classical frequency analysis of images which shows various high frequency behaviors. The study of the frequency domain accompanied by a basic supervised or unmonitored classification identifies these artifacts in their system. Remember that it does not require or need huge volumes of data in this particular pipeline, which is a very useful aspect for the data scrambling scenarios. However, they are launching the latest Faces-HQ sequence for the experimental assessment, used to complement the CelebA data collection and FaceForensics++. The Discrete Fourier Transform (DFT) is a statistical method used to decompose a binary signal into sinusoidal elements of varying frequencies from a spectrum of 0 (i.e. the constant frequency, which coincides with the medium picture value) to a maximum frequency, given its spatial resolution. For signals sampled on equidistant lines, it is the relatively equivalent of the continuous Fourier transform. This can be measured as for two-dimensional data format M / N in the following way:



The Radial Profile Plot displays the luminance at the position of all pixels. This corresponds to a calculation of the entire width of the target in the middle of the plot at maximum half (FWHM). Mira also displays the Gaussian + Constant suit, the context value well away from the Gaussian profile and the profile peak value. This approach analyzes a gray image and generates a profile plot with a normalized blended width around concentric circles, which depends on the distance between the image points. The center of the rectangle that borders on the current ROI is identified automatically as the stage. In a dialog box, the location of this point can be changed. The size of the pixel values around a circle is at a given distances from the point. This circle has the middle point and the distance as radius from the point. The built-in amplitude is determined by the number of pixels in the image frame, which gives normalized relative values.

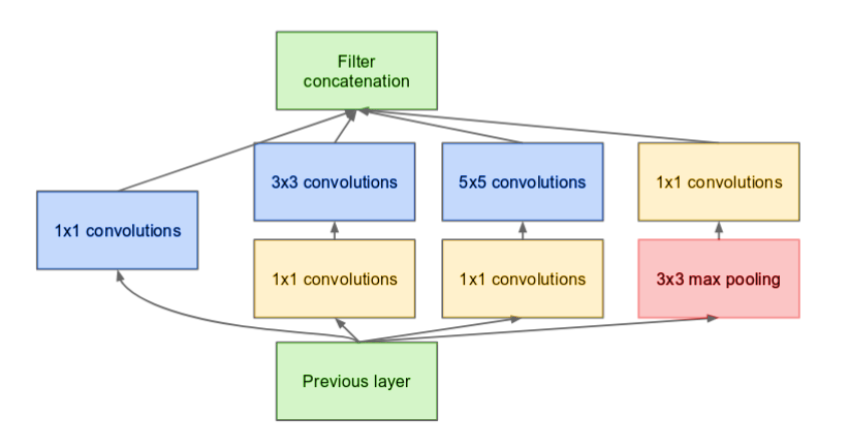
After the implementation of these methods, the classification layers are adjusted. The classifiers can be SVM, random forests or a neural network. For all of the classification layers, a small amount of datas was used. Considering there were some problems with the code during preprocessing or in the neural network architecture, the model gave a validation accuracy of 100% for all the different classifiers. However, when the model was run again using the Kaggle dataset, the model couldn’t identify or define any functions. Hence, it couldn’t predict whether the provided inputs were real or fake.

Conclusively, 20,000 datas were used and the accuracy achieved after model training was 17%. The accuracy was considered to be reliable as during the preprocessing stage, it was stated that the model was not able to differentiate whether the provided input was fake or real.

**Week 6 (23/02/2020-29/02/2020)**

This week, methods of ‘**Mesonet’** paper were used for implementation, along with the extraction of the entire dataset from the ‘facebook deepfake detection challenge’. The paper demonstrates a strong detection rate for Deepfake of more than 98% and Face2Face of 95%. During the last decades, interactive pictures and videos have been popularized as smartphones and social networks. Following the enormous use of digital photography, for example, the use of modeling tools such as Photoshop has expanded techniques for image modification. Digital forensic photographs are devoted to the identification of counterfeit photographs in order to monitor their transmission. Compressed image objects are also studied to provide valuable insights into image processing. The length of the experiment ranges between two and three minutes with a fixed resolution of 854 X 480 pixels. A H.264 codec format, but with different degrees of compression, which allows them evaluated under actual circumstances. They investigated how additional facial fragmentations can be identified using the new architecture in addition to the Deepfake dataset. The FaceForensics data collection contains over one thousand faked Face2Face-approached videos and their initial dataset as the correct candidate. The data collection is now split into a preparation, evaluation and research package. A further benefit of the FaceForensics System is the extension of the use of the proposed architecture into another classifications function, as it allows fewer missing compressed videos to test the robustness of their models at various compression rates. A further benefit of the FaceForensics System is the extension of the use of the proposed architecture into another classifications function, as it allows fewer missing compressed videos to test the robustness of their models at various compression rates.

During the initial stages, only 20% of the dataset was used. However, as different methods from different papers were implemented, it was necessary to use the entire dataset to see the peak difference that is observed during different architectures along with different classifiers being implemented. After the entire dataset was extracted, methods of ‘Mesonet’ paper was implemented. However, the top of the keras layer didn’t really work as stated on the paper and it was becoming difficult to call the object in order to train the model. The implementations didn’t work as predicted. The model was able to predict whether input was real or fake. However, it wasn’t possible to train the model overall. Later on, a visit to Kaggle was committed to check the different methods of Mesonet that were implemented. One of the implementations was Mesonet using inception layers. The inception network is a dynamic (highly designed) network. It has used various tricks in terms of consistency and precision to improve efficiency. Its ongoing growth leads to several network models. Understanding updates will help us create personalized classificatory that are both easily and reliably configured. As stated before, the computational cost of deep neural networks is high. The developers reduce the number of input channels to make it cheaper by additional 1x 1 convolution between 3x 3 and 5x 5 convolution. While adding an extra operation can sound counterintuitive, 1x1 convolutions are much cheaper than 5x5 convolutions, while still helping to minimize the number of input channels.



Using this layer, a model was built. The code for the model was taken from Kaggle and then it was used in the model used for the deep fake detection challenge. The weight provided by the mesonet inception layer was taken and later the classification layer was taken for the built-in model. After this, the entire Mesonet layer was trained using a neural network architecture. But the training didn’t take place. This is because, it was taking a huge amount of execution time. 9 hours to be exact. It was very important to check the execution time in order to see the running time complexity. After this, steps from the Keras layers were reduced. Eventually, 20% of the datasets were used after this observation. And for each epoch it took 1 hour. There were 50 epochs. And the accuracy reached 88%. The most interesting part was that, as soon as the epochs started the accuracy started to increase. Therefore, the assumption is, even though the execution time, will increase at the end of the experiment, by increasing the dataset, it is possible that the accuracy will increase further than the current stage.

**Week 7 (01/03/2020-07/03/2020)**

Another experiment that was conducted was the implementation of the **Dense121** network. This network is also known as Dense121 network. Dense Convolutionary Network (DenseNet) that links each layer in a feed-forward manner. In the case of standard L-layer convolutionary networks, the network has direct L (L+1)/2 links-one between each layer and the layer that preceded it. The feature maps of all previous layers are used as inputs for each layer, and the feature maps themselves are used as inputs for all layers. The persuasive benefits of DenseNet are to ease the issue of fades, improve the replication of features, and promote reuse of features and the number of parameters considerably. The opposite-intuitive result of this dense pattern of networking is that less parameters are needed than conventional convolutionary networks, as redundant maps must not be re-learned. Traditional feed architecture can be interpreted as algorithms with a layer-by-layer configuration. Each layer reads and writes to the following layer from its preceding layer. DenseNet's suggested architecture specifically distinguishes between network added information and retained information. DenseNet layers are very small, for example, 12 function maps per layer. The "collective information' of the network is only supplied with a few character-maps and the other feature-maps are left unchanged.

20% of the full dataset was extracted and used for training the model, with the Dense121 network. The accuracy that was conceived was not satisfactory. It was below 80%. Also, the densenet architecture consists of many parameters. As a result, it took some time to converge. Also, it was observed that the densenet architecture was biased towards the minority class, i.e. the real class. However, there was no conclusive results about how the biasness occurred.

Another extension of the project, is the plug-in for the web browser. The plug-in consisted of a few parts. One of them was to take an image, using the plug-in and save it in a designated location. The next step was to take the saved image, and pass it to tensorflow.js model. However, the tensorflow.js model was very complicated which required some fixation. Overall, there was no tensorflow.js model. Instead, there was keras.h5 model. This model was later then converted to a .json model. In order to change the model, at first, tensorflow.js was provided as an input, followed by tensorflow.tf. Then, the model was loaded. The model was loaded by disabling the eager execution, which caused several problems during the initial stages. Secondly, after the model was loaded, in a variable called “model”, then **tensorflow.js** was imported. From the tensorflow.js library, which is a python library (and not a JavaScript library), the model was executed with JavaScript. However, as it is a python library, it needed some changes. In google collaborator, the easiest way to use JavaScript is to use in-line command.

Finally, tensorflow.js was imported using the python kernel. After importing tensorflow.js, a converter was used from the library, and the converted has a function which helped to save the built-in model as the model which was provided as an input. The path was set for the model to be saved. Once the model was saved and executed, several outputs were found which turned out to be tensorfow.js models and along with a model of model. Json. The latter model was used to predict and differentiate whether the input image was real or fake.

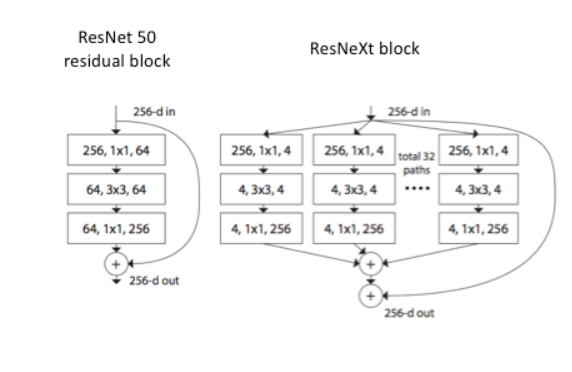
**Week 8 (08/03/2020-14/03/2020)**

It was important to check the problems that were present inside the TensorFlow library. The main problem that was occurring was the fact that during the implementation of Mesonet, the tensor flow was accidentally downgraded. For which, the old tensorflow cpu was installed and tensor flow gpu was not. For which, the model took more time for each epoch. After this, the tensor flow cpu was uninstalled and the tensorflow gpu was installed. The version of the library was 1.13 for tensorflow gpu. GPU is known to be the core of artificial intelligence, for Deep Learning. This chip processor works for comprehensive graphical and mathematical calculations is a single chip processor, which frees CPU cycles for other work. In fundamental education, the host system is running on the CPU and is operating on GPU as CUDA system.

CPU performs specific functions such as processing of 3D images, vector equations, etc.

GPU can have a question of bandwidth limitation while Processor may perform streamlined and complex tasks for long periods. I.e. it may be difficult to transfer vast volumes of data to the GPU. GPUs have been designed for bandwidth. CPUs have been designed for latency (memory access time).

The datas from face forensics, c40 level videos for the real class, was then extracted to images and was later merged with the existing Facebook DeepFakes detection challenge dataset (only the real class). This is because there was a shortage of the adequate amount of data for the real class. Right now, there is approximately 2,50,000 data for the real class from 1,50,000 which is a substantial improvement. After the fame extraction, **Resnext** architecture was used for model training. The ResNeXt architecture is a deep network enhancement that replaces the uniform residual block by a technique used in the original models for a "split-transformation-merge," that is, linking paths within a node. Basically, the input block is projected into a set to low-sizes (channel) of which they add a few convolutionary filters separately before convergence of the effects, rather than executing convolutions over the entire input region. The ResNeXt architecture simply recreates the ResNet models to replace the ResNeXt block.



The architecture was not compatible with keras for which the architecture was later replaced with methods of **EfficientNet** paper**.** Within this document, the method of the ConvNets scaling will be reconsidered and discussed. They investigate the key problem in particular: is there a theory method of expanding ConvNets which can increase precision and efficiency?

The observational analysis reveals that all network width / depth / resolution measurements are important to match, and interestingly such balancing can be accomplished by simply increasing every parameter by a constant ratio. They suggest a basic but powerful scaling approach based on this observation. They note empirically that various aspects of scaling are not distinct. Intuitively, they can expand network size for higher-resolution images, so that wider fields will help to collect identical functionalities that have more pixels in wider images. In order to capture more finely-grained images and more pixels in high quality, we can also expand network width when the quality is low. These intuitions suggest that various scaling dimensions can be organized and balanced instead of traditional scaling.

The accuracy achieved from this architecture is very disappointing, which is 67% to be very precise, for the first 10 epochs. However, there is a chance of the accuracy to improve as the epochs were taking only 35 minutes. In conclusion, EfficientNet didn’t provide the expected accuracy.