Bilkent University Spring 2019-2020

CS 564 - Computational Geometry

Term Project Final Report

25.05.2020

" Face Anomally Detection with Voronoi Regions as Features"

Irmak Türköz

21400487

1. Introduction and Background Information

1.1 Introduction

As the growth of identities available as open-source on media, it is now possible to create fake representations of human faces seamlessly to steal identities and deceive the viewers. DeepFakes - or fake videos produced to look real through the use of artificial intelligence - became more indistinguishable from real faces and they can posses a serious threat of spreading false information or fake news that can be used to threaten governments, create conflict in politics, and enable cybercrime. Thus, this topic raises significant concerns for the implication of society [6]. DeepFakes are created with taking input a video of a specific individual ('target') and outputting another video with the target's faces replaced with those of another individual ('source'). They are mostly generated with deep learning tools and these tools can automatically map facial expressions of the source to the target. With proper post-processing the two faces that are mixed to construct a realistic DeepFake that it is even hard to distinguish fake faces with human eyes [1]. In this project, the main objective is to find out if Voronoi diagrams can be used to detect anomaly of faces which would help to discriminate the DeepFakes.

1.2 Background Information

To detect DeepFakes, there have been several areas of research which involve using special artifacts such as face action units [8], eye blinking, [9], biological signals [10], facial features optical distortions [11][12]; there are also neural network methods which proposes CNN based models [13], CNN and RNN based models [14], two-stream neural networks [15], autoencoders [16] and; general image manipulation detection methods with analyzing frequency domain [17] or using a Long-Short Memory model [17]. Although there are many proposals with neural networks based on the face images, the Voronoi diagram based inputs to the machine learning algorithms have not been investigated. In this paper, I will try to use the Voronoi diagram analysis to detect the anomaly.

Voronoi diagram is defined as the partitioning of a plane with n points into convex polygons such that each polygon contains exactly one generating point and every point in a given polygon is closer to its generating point than to any other[2] and dual of these graphs give Delaunay triangulation mapping of the images. In computer science, Voronoi graphs are used in research topics such as face recognition, face segmentation, 3D face reconstruction from 2D images, etc. as they are extracting features and segmentation information of the given images [3]. Voronoi diagrams can be helpful to detect DeepFakes as they are capable of finding the ratio of the facial feature regions such as moth, eye, nose; which we can use to extract geometric features. However, the distances are not meaningful when we consider a FaceSwap algorithm. FaceSwap is a manipulation method to switch the face in the region with another face, without using any facial features of the target face. So the second source face has the exactly same features. Since we require a DeepFake detection algorithm that is capable of detecting any unseen faces, using only the geometrical distances would not give any information when we swap the faces, the proportions of the faces are consistent.

1.3 Dataset

The current system is evaluated with FaceForensics++ dataset consisting of 977 downloaded videos from youtube, 1000 original extracted sequences that contain an unobstructed face that can be easily tracked [4]. It contains 1000 original video sequences that have been manipulated with four automated face manipulation methods: Deepfakes, Face2Face, FaceSwap, and NeuralTextures. The dataset also merged with Google's DeepFakeDetection dataset [5] and JigSaw which contains over 3000 manipulated videos. It is a great resource to use for deep learning techniques. However, one drawback with this dataset used was all of the other dataset categories than the "DeepFakeDetection" had corrupted videos. I this project, the manipulated videos are only from the category "DeepFakeDetection". In the first phase, I have used only a small subset of the dataset, with 30 original videos and 30 manipulated videos to train; 10 videos each for validation and test sets.

2. Methodology

2.1 Preprocessing

This pre-processing step is required for both of the models described in 2.2 and 2.3, as a common step because it is used by the face detector. These models have different preprocessing steps after the common preprocessing, which involves reshaping and illumination normalization.

First, I have read the videos frame by frame using OpenCV. For each of the frames, I have considered them as images to do the rest of the algorithm. As I have described in the progress report, I have implemented preprocessing to reduce the lighting ill effect. The original images have been fed into the pre-trained model to find facial landmarks, however, OpenFace might fail in the very low contrast setups. The illumination normalization is required to fix the detection consistency.

The preprocessing involves the following steps: (1) fix the image size to 400 x 700, convert image to lab space, (2) apply Contrast Limited Adaptive Histogram Equalization [18], (3) convert lab space back to RGB space. The preprocessing was required for the OpenFace detector, which is used to detect the facial area in the given frame. The preprocessed face detection model by dlib [19] is more capable of detection of the facial area after preprocessing the image as can be seen in Figure.1.

Figure 1: Lightning ill affectted image (Left) vs preprocessed image (Right)

After the extraction of the face regions, I have cropped the facial areas. These cropped images of the facial areas are used in both of the models, in section 2.2 face regions are inputs to detect the facial features, in section 2.3 the cropped face regions are inputs the detection model.

2.2 Voronoi Region Features Model

2.2.1 Preprocessing for Voronoi Model

After the illumination normalization which enhances the contrast, the facial regions are accurately extracted from the frames. However, the extraction of facial landmarks uses a predictor which required another preprocessing step. The cropped facial regions are also applied with gamma normalization, to get more accurate facial landmarks from the faces. Adjusting the gamma is done by building a lookup table mapping the pixel values [0, 255] to their adjusted gamma values. In some of the videos, the face sizes are too small, and they approach camera location by later frames. Therefore, because I require high-quality images of faces; I do not extract the faces from the distant objects from the camera.

2.2.2 Extracting Facial Landmarks

Using the same open-source library dlib[19] to detect the face regions, it has also a predictor that allows us to extract 68 landmarks from the facial regions. These landmarks are targeting most descriptive parts of the faces such as eye-lids, mouth, nose, eyebrows, etc.[20]. Dlib model was made using the classic Histogram of Oriented Gradients (HOG) feature combined with a linear classifier, an image pyramid, and a sliding window detection scheme [21]. The common preprocessing is very useful since it allows the predictor to extracts the features from already detected regions, instead of giving the whole image. This is more performance-friendly since the algorithm does not iterate through all of the image's sliding windows and it generates more accurate results because we are already limiting the face boundary box. After defining the predictor from this model, I get the output of the predictor with the following:

```
        landmarks = predictor(img, rect)
```

The "rect" here is the region found by detector, but detector might find more than one faces so I trace through each of the found faces to detect every face's landmark. Thus, my algorithm also works with multiple faces in a frame. In the progress report the pre-trained model is not working well with the profile angle faces, however, cropping the face regions as inputs to the Voronoi regions has improved the landmark extraction. If the face cannot be detected, the landmarks will not be generated, thus the noisy landmarks from the predictor will not be generated.

2.2.2 Creating Voronoi Diagrams

Once I have extracted 68 landmarks from faces, I have constructed a Voronoi diagram on these points. I have used scipy.spacial's Voronoi to generate Voronoi [7]. And Voronoi regions are generated done automatically, and it is performance friendly. In the progress stage, I did not use the boundary box of the face but the whole image, which caused to get the output of Voronoi vertices at infinity. However, because now the landmarks are extracted from only on the face boundary box, it automatically eliminates those vertices. The scipy.spacial Voronoi allows us to get several properties described in Figure 2.

Attributes

points	(ndarray of double, shape (npoints, ndim)) Coordinates of input points.
vertices	(ndarray of double, shape (nvertices, ndim)) Coordinates of the Voronoi vertices.
ridge_points	(ndarray of ints, shape `(nridges, 2)`) Indices of the points between which each Voronoi ridge lies.
ridge_vertices	(list of list of ints, shape `(nridges, *)`) Indices of the Voronoi vertices forming each Voronoi ridge.
regions	(list of list of ints, shape `(nregions, *)`) Indices of the Voronoi vertices forming each Voronoi region. -1 indicates vertex outside the Voronoi diagram.
point_region	(list of ints, shape (npoints)) Index of the Voronoi region for each input point. If qhull option "Qc" was not specified, the list will contain -1 for points that are not associated with a Voronoi region.

Figure 2:scipy.spatial.Voronoi , attributes.[22]

In the progress report, I have used the vor.ridge_vertices which imports the indices of Voronoi vertices that construct the edges of Voronoi maps, and give these indices to the vor.vertices to visualize the Voronoi regions. However, this only allowed to extract the edges, not the faces of the Voronoi regions. In the final step, I have used vor.ridge_vertices that extract the indicies of the Voronoi ridges to give these indices to vor.regions to extract the polygons that represent the faces of Voronoi maps.

I have also implemented another algorithm, to crop the images with this given polygons. Algorithm explained step-by-step: (1) Crop the bounding rectangle of the polygon, (2) make a mask out of the rectangle, (3) do a bitwise operation with respect to the given polygon on the mask, (4) revert the pixels so that the background is black, (5) apply the mask on the image, (6) eliminate the Voronoi region if it has (a) black pixels / image size ratio is low, (b) if the size of the Voronoi region is small (less than 20x20), (7) save the resulting Voronoi regions to not generate them in every frame while doing training because it is time-consuming. These steps allow us to generate approximately 50-60 Voronoi regions for each of the faces in a video. However, the size

of the regions varies from 20x20 to 80 x 100. Therefore, to be not biased with the detector algorithm, padding is applied to all of the features so that they become the maximum size of 80 x 100. The feature extraction takes time since the dataset consists of videos and there are many frames (approximately 500-600) and for each of the frames there are many features. An interesting remark while doing the feature extraction, the algorithm that constructed to generate Voronoi region features extract more data points(more regions) for original images than the DeepFakes. Therefore, the elimination done by cropping the polygons out of the faces, eliminates more Voronoi regions for manipulated faces. Some of the extracted features are given in Figure.3.

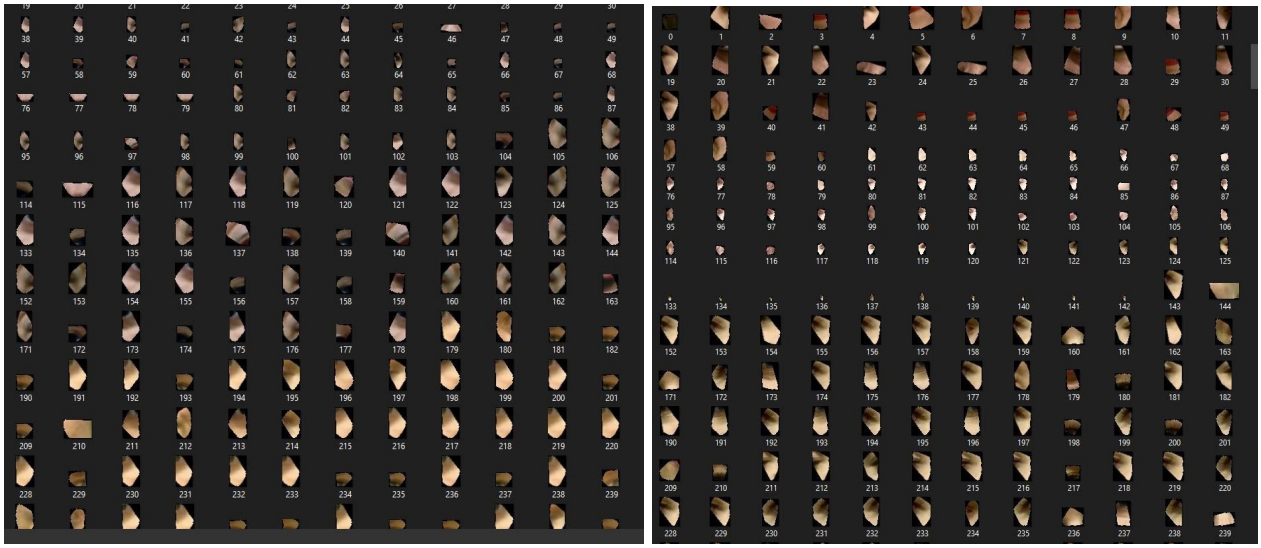

Figure 3: Extracted voronoi regions for manipulated(left) and original(right) videos.

2.2.3 Detection Model Architecture and Evaluation

My first approach was to apply VGG16, as it works best with the image type of inputs. In the first try, I have split the data into test, train, and validation before extracting the features. So, the person appears to be in the training set only appears in the training set and does not appear on validation and test sets.The specifications of the VGG16 network is following:

- Loss Function : Binary Cross Entropy
- Number of epochs : 20
- Optimizer : Adam optimizer
- Output layer activation : Softmax, Dense with one class (binary classification)
- Hidden layer activation: Relu

- Learning rate, decay rate : 0.1, $5e^{-3}$
- Batch size : 16
- Input shape : 80, 100

The accuracy I got was pretty low, and my system was not considered to be learning anything as can be seen in Figure.4. I have also considered if the epoch is not enough, so I have trained with 100 epochs and here are the results of 100th epoch:

Epoch 00100: LearningRateScheduler setting learning rate to 0.009999999776482582.

loss: 0.6101 - **accuracy**: 0.**7005** - val_loss: 0.8446 - **val_accuracy**: 0.**5662**

This shows that although the model learned some of the validation data, the test data had only 0.56 accuracy, so the model, overall did not learn anything at all (see Figure.4).

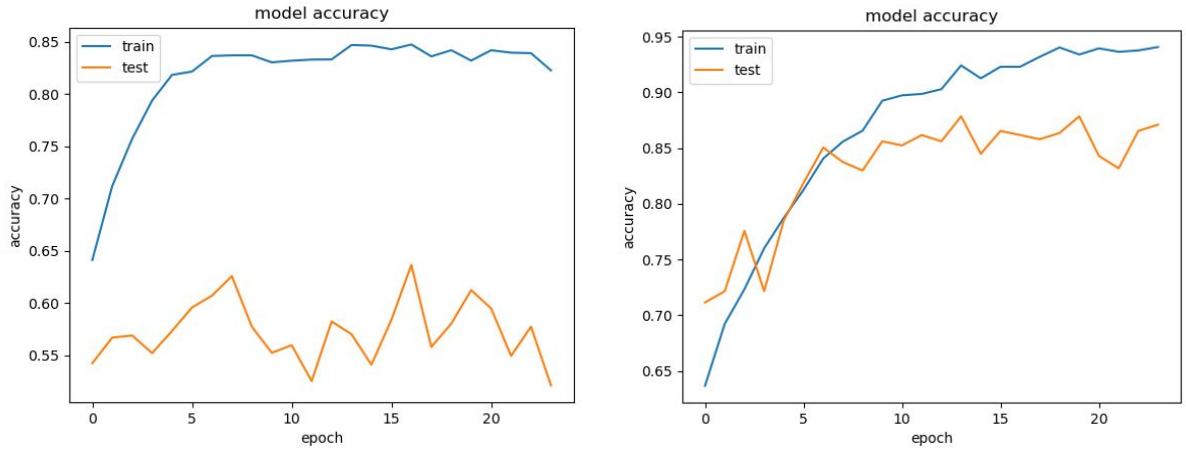

Figure 4: left: bad model, a person only appears in training set and does not appear in test and validation; right, better model, a person appears in both training and the validation sets.

Therefore, I considered a different approach. Since most of the DeepFakes are constructed from a familiar face, a celebrity; because of the available data online and open source. Therefore, DeepFake detection algorithm can be considered as a recognition task. Consider dataset have the same persons' original data and the DeepFakes of the same person, model can recognize the facial features with Voronoi regions created from section 2.2.2. For instance, if there would be 10 videos of real Obama speaking, and 10 manipulated videos of Obama, detection model could have found out a video is fake or not by comparing the actual Voronoi regions of the Obama. Moreover, if there are many other celebrities alongside Obama, the same idea holds; without giving the class information, because the model already extracted the parts of the faces. To test this idea out, I have

used a new collection of the data, where the same face appears in test and train set, however in different circumstances. In this subset, there are identities belong to many people, but not one; however the main difference is the face also appears in the test, train, and train set. So this detection model started to recognize the original face features of the people in the dataset and decided if a part of the face is real or fake by doing this. Even though I have not to give any information about the face id that the Voronoi region belongs to, it learns from the most similar data point. The results were pretty impressive compare to the first model, as I have got the accuracy of 0.95 on validation data and 0.85 on test data (see Figure .4). This method would also allow us to predict an unseen methodology of DeepFake of a person that has the original Voronoi features in the dataset. Further research with many more data could be constructed to see if it works. Moreover, the current model used is the only VGG16; however a much more complex model such as XCeption can be used to detect even more complex features of the face parts. Because of the time constraints of this project and the high time cost of deep neural networks, other models have not experimented.

2.3 Face Regions Model

2.3.1 Preprocessing for Face Regions

Preprocessing of the face regions was more simple and faster than the Voronoi regions, the size normalization with reshaping; and the frame rate was applied. If we would consider every frame in the dataset, the system would generate too many face regions. So, instead I have introduced the frame rate to the system. Moreover, since we might have a deformation on the background caused by face reenactment, I have not applied masks to the face regions and took the face regions as rectangles. There are approximately 30 images of face regions extracted from the model, and the model was built with 4980 images for training, 2015 for validation, and 1611 images for test belonging to 2 classes, manipulated and original.

2.3.2 Detection Model Architecture and Evaluation

The model I have used to compare the voronoi regions with face regions, is the same, VGG16. The specifications of the network is following:

- Loss Function : Categorical Cross Entropy
- Number of epochs : 24

- Optimizer : SGD optimizer
- Output layer activation : Softmax, Dense with two class (Categorical classification)
- Hidden layer activation: Relu
- Learning rate, decay rate : 0.1, $5e^{-3}$
- Batch size : 40
- Input shape : 150, 150

In this model, the first significant fact was the slowness. Because I have eliminated unnecessary information in the previous model, it could train in just 10 minutes. However, when we have the whole images, and the dataset is much bigger because we do not eliminate the feature regions in the Voronoi regions, moreover, this time directly giving the faces to the algorithm. The accuracy is very low, even lower than the previous approach when we only give the parts of the images without giving the same person in the train/validation. I have also tried to train the system with binary class entropy, however, the results have not got any better (see Figure 5.).

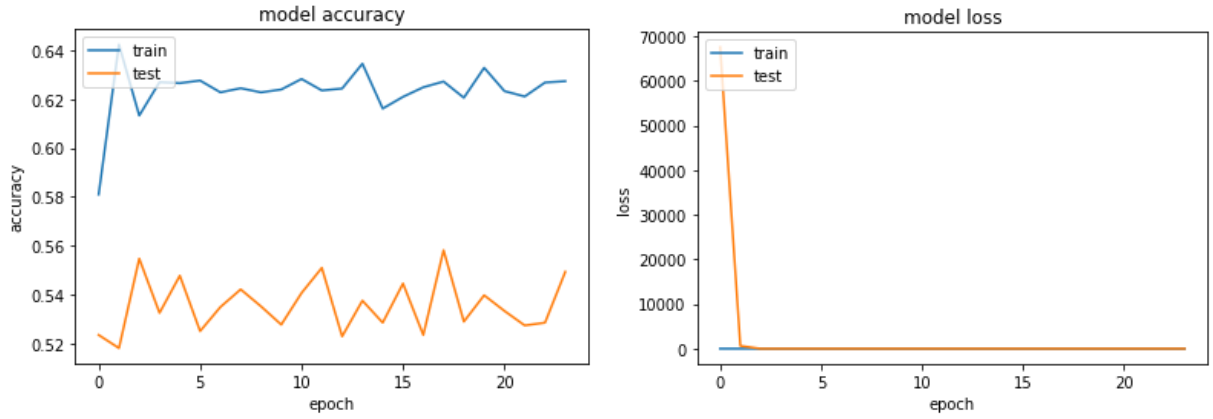

Figure 5: Usign Face Regions as input to VGG16 model

I have also experimented with different models for this step, such as ResNet50 and Xception. ResNet50 was slightly better with 0.7014 accarcy on validation data, however 0.65 accuracy on test data; on the 30th epoch. Xception, on the other hand did worse than the VGG16, with 0.5788 accuracy on validation data, and 0.5221 accuracy on test data on the 8th epoch. These models were not able to directly do the detection through the face regions. The most logical way to explain this, the DeepFakes are not detectable with the unnecessary information caused by the whole face region. Directly giving the regions does not work because the models are more compatible with

less feature information instead of 150x150x3 features generated from the whole image. This also explains the slowness of the models, because we have many features; we are trying to generalize too much knowledge without eliminating unnecessary information.

3. Conclusion

In conclusion, the task of DeepFake detection has gained more interest in the availability of open-source software and a variety of commercial applications provide an opportunity to generate fake media of a target face in several ways. There have been many approaches to detect the manipulated media, however none of the researches have yet come up with a solution involving Voronoi region features. In this paper, the Voronoi feature extraction technique with the help of OpenFace was explained in detail and a VGG16 based model is trained to experiment. To evaluate these features, another VGG16 model with using face region has been developed. Comparing two of the methodologies tried in this project, a DeepFake detection method always requires a preprocessing, followed by feature extraction. Without the extraction of feature information of images, the whole face region is not meaningful for a neural network to generalize the data. Geometrical features of the data without the texture information can be misleading to the neural networks because some of the DeepFake detection techniques are swapping the source face with the target face. Thus, the geometrical features should be used with texture features. A drawback of Voronoi features is also explained, the availability of the same person in the training and validation sets are required because of the Voronoi regions in the detector algorithm act as recognition task of the facial parts. On the other hand, this original feature information of the face can be used to detect unseen DeepFake detection system in the training data. A further expedition of voronoi regions can be used with larger data, in a less time-constraiened ecosystem and with a better performance server. Although the extraction of voronoi regions is optimized for time, extraction of many features from many frames of the video lasts for long. Another feature extraction method can be also used on the voronoi features, such as Local Binary Patterns, to improve the accuracy of the system, as a future work.

References

[1]Y. Li and S. Lyu, "Exposing DeepFake Videos By Detecting Face Warping Artifacts", *Openaccess.thecvf.com*, 2020. [Online]. Available: http://openaccess.thecvf.com/content_CVPRW_2019/papers/Media%20Forensics/Li_Exposing_DeepFake_Videos_By_Detecting_Face_Warping_Artifacts_CVPRW_2019_paper.pdf. [Accessed: 30- Mar- 2020].

[2]W. E, "Voronoi Diagram -- from Wolfram MathWorld", *Mathworld.wolfram.com*, 2020. [Online]. Available: http://mathworld.wolfram.com/VoronoiDiagram.html. [Accessed: 30- Mar- 2020].

[3]A. Cheddada, D. Mohamadb and A. Manaf, "Exploiting Voronoi diagram properties in face segmentation and feature extraction", *Fei.edu.br*, 2008. [Online]. Available: https://fei.edu.br/~psergio/CompGeom/Voronoi-Faces.pdf. [Accessed: 30- Mar- 2020].

[4]A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies and M. Nießner, "FaceForensics++: Learning to Detect Manipulated Facial Images", 2019. [Online]. Available: https://arxiv.org/pdf/1901.08971.pdf. [Accessed: 30- Mar- 2020].

[5]"ondyari/FaceForensics", *GitHub*, 2020. [Online]. Available: https://github.com/ondyari/FaceForensics/tree/master/dataset. [Accessed: 30- Mar- 2020].

[6] A. R ¨ossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, andM. Nießner, "Faceforensics++: Learning to detect manipulated facialimages," 01 2019.

[7]"scipy.spatial.Voronoi — SciPy v0.18.1 Reference Guide", *Docs.scipy.org*, 2016. [Online]. Available: https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.spatial.Voronoi.html. [Accessed: 30- Mar- 2020].

[8]S. Agarwal and H. Farid, "Protecting World Leaders Against Deep Fakes", *Openaccess.thecvf.com*, 2020. [Online]. Available: http://openaccess.thecvf.com/content_CVPRW_2019/papers/Media%20Forensics/Agarwal_Protecting_World_Leaders_Against_Deep_Fakes_CVPRW_2019_paper.pdf. [Accessed: 30- Mar- 2020].

[9]Y. Li, M. Chang and S. Lyu, "In Ictu Oculi: Exposing AI Generated Fake Face Videos by Detecting Eye Blinking", Computer Science Department, University at Albany, SUNY, 2018.

[10]U. Ciftci, ˙. Demir and L. Yin, "FakeCatcher: Detection of Synthetic Portrait Videos using Biological Signals", England, 2019.

[11]F. Matern, C. Riess and M. Stamminger, "Exploiting Visual Artifacts to Expose Deepfakes and Face Manipulations", Friedrich-Alexander University Erlangen-Nuremberg, 2020.

[12]I. Amerini, L. Galteri, R. Caldelli and A. Bimbo, "Deepfake Video Detection through Optical Flow based CNN", Italy.

[13]N. Do, I. Na, H. Yang, G. Lee and S. Kim, "Forensics Face Detection From GANs Using Convolutional Neural Network", in *ISITC 2018*, chool of Electronics and Computer Engineering, Chonnam National University, 2018.

[14]D. Guera and E. Delp, "Deepfake Video Detection Using Recurrent Neural Networks", Video and Image Processing Laboratory (VIPER), Purdue University, 2020.

[15]P. Zhou, X. Han, V. Morariu and L. Davis, "Two-Stream Neural Networks for Tampered Face Detection", *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2020. [Accessed 30 March 2020].

[16]D. Cozzolino, J. Thies, A. Rossler, C. Riess, M. Nießner and L. Verdoliva, "ForensicTransfer: Weakly-supervised Domain Adaptation for Forgery Detection", Germany, 2020.

[17]R. Durall, M. Keuper, F. Pfreundt and J. Keuper, "Unmasking DeepFakes with simple Features", Germany, 2020.

[18]"OpenCV: Histograms", *Docs.opencv.org*, 2020. [Online]. Available: https://docs.opencv.org/master/d6/dc7/group__imgproc__hist.html. [Accessed: 30- Mar- 2020].

[19]D. King, "Classes — dlib documentation", *Dlib.net*, 2013. [Online]. Available: http://dlib.net/python/index.html. [Accessed: 30- Mar- 2020]

[20]C. Sagonas, E. Antonakos, G, Tzimiropoulos, S. Zafeiriou, M. Pantic. “300 faces In-the-wild challenge: Database and results.”Image and Vision Computing (IMAVIS), Special Issue on Facial Landmark Localisation "In-The-Wild". 2016.

[21]*Dlib.net*, 2020. [Online]. Available: http://dlib.net/face_landmark_detection.py.html. [Accessed: 30- Mar- 2020].