Deepfake Detection

Atul Khobragade (B20CS027)

Abstract

Deepfakes use deep learning artificial intelligence to replace the likeness of one person with another in video and other digital media. This technology can be used to create large amounts of misinformation, damage reputations, and harm individuals. This makes deepfake detection more and more important. In thsi paper, We will be trying to create an end-to-end Machine Learning pipeline to recognize such fake videos out of a pool of videos by employing a variety of machine learning practices.

I. Introduction

IN this paper, we preprocess the videos by detecting a select number of frames from the videos. We have then detected faces from the frames using multiple face detection engines. The images have then been resized and transformed into trainable data.
This data has then been passed through different dimensionality reduction techniques and several classifiers have then been trained on this data. We provide the comparison between the different face detection techniques, the different classifiers and between the different dimensionality reduction techniques in the following text.

II. Face Detection Techniques

A. DLIB

DLIB is a toolkit for making real world machine learning and data analysis applications in C++. While the library is originally written in C++, it has good, easy to use Python bindings. It is used in both industry and academia in a wide range of domains including robotics, embedded devices, mobile phones, and large high performance computing environments.

B. MediaPipe

MediaPipe Face Detection is an ultrafast face detection solution that comes with 6 landmarks and multi-face support. It is based on BlazeFace, a lightweight and well-performing face detector tailored for mobile GPU inference.
The detector's super-realtime performance enables it to be applied to any live viewfinder experience that requires an accurate facial region of interest as an input for other task-specific models, such as 3D facial keypoint or geometry estimation, facial features or expression classification, and face region segmentation.

C. MTCNN

MTCNN or Multi-Task Cascaded Convolutional Neural Networks is a neural network which detects faces and facial landmarks on images. It was published in 2016 by Zhang et al. MTCNN output example. MTCNN is one of the most popular and most accurate face detection tools today.
It is an algorithm consisting of 3 stages, which detects the bounding boxes of faces in an image along with their 5 Point Face Landmarks.

III. Preprocessing

A. Loading Dataset

Dataset was uploaded to google drive and was included from there. Dataset is divided as follows:

1) *Train:* Fake, Real
2) *Val:* Fake, Real
3) *Test:* Fake, Real

B. Capturing Frames

Frame capturing was done by extracting and storing a fixed number(32) of evenly spaced frames from the respective video and onto the collab storage. For some videos, less than 32 frames were captured due to its short length. The paths to these frames were stored in correctly named variables.

C. Face Detection Engine

DLIB / MediaPipe / MTCNN face detection engine is applied here. All the stored frames are called and checked to see if they contain faces, if they do, the faces are isolated, extracted and resized to size (64,64,3). After which these images are stored on colab storage and their paths are written in respective variables.

D. Data Augmentation

Data we are provided is small in size. In order the classifiers to learn better, Data Augmentation is applied by adding noise to the images. Various noises were tried like Gaussian, Speckle and sp noise.
Noise useful here was observed to gaussian, which was meaningfully blurring the image, rest are not useful in our particular case. Hence the guassian noise is added in each image and each image is stored separately of the original image

E. Transforming Data

Custom function createFeatureDatabase is called on the training and testing part of the database. This function reads an array containing the path to the stored 'face' images, reads each image, converts the image into grayscale and appends the pixel values into an array. It also appends the label of each image and return 2 lists. One containing the trainable array containing pixel values for each image and the other which contains their corresponding labels.

IV. Dimensionality Reduction

A. Scaling

The training and the testing data is standardized using StandardScaler.

B. PCA

Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets. PCA is applied on the training and testing datasets and the the transformed data is stored in aptly named variables.

C. LDA

Linear discriminant analysis (LDA) is a dimensionality reduction technique used to reduce the number of features to a more manageable number by minimizing the intraclass distances and maximizing the distances between different classes. LDA is applied on training and testing datasets and the transformed data is stored in aptly named variables.

V. Training

This section describes the training of 3 different models namely- ***SVC, MLP and KNN***.Each of the 3 models train on 3 different datasets namely- ***original, PCA transformed and LDA transformed***.

Labelling Criteria: Result for every image is predicted from the respective trained models and stored. The final results for a video is calculated on the basis of majority voting of Real and Fake tags generated by classifiers for the images belonging to that particular video. If the majority of the images for a video come out as Fake, the video is classified as Fake or vice-versa.

The same preprocessing and dimensionality reduction steps have been carried out for face detectors MTCNN and MediaPipe.Their results will also be compared below.

A. SVC

1) Accuracy:

Dataset	DLIB	MediaPipe	MTCNN
Normal	0.66	0.74	0.66
LDA Transformed	0.48	0.44	0.5
PCA transformed	0.66	0.72	0.66
Normal + Noise	-	0.72	-
PCA + Noise	-	0.72	-

2) F1 Score:

Dataset	DLIB	MediaPipe	MTCNN
Normal	0.691	0.787	0.721
LDA Transformed	0.567	0.517	0.667
PCA transformed	0.702	0.77	0.721
Normal + Noise	-	0.774	-
PCA + Noise	-	0.774	-

3) Log Loss:

Dataset	DLIB	MediaPipe	MTCNN
Normal	11.743	8.980	11.743
LDA Transformed	17.960	19.342	17.269
PCA transformed	11.743	9.671	11.743
Normal + Noise	-	9.671	-
PCA + Noise	-	9.671	-

B. MLP

1) Accuracy:

Dataset	DLIB	MediaPipe	MTCNN
Normal	0.5	0.60	0.48
LDA Transformed	0.48	0.48	0.52
PCA transformed	0.74	0.70	0.68
PCA + Noise	-	0.64	-

2) F1 Score:

Dataset	DLIB	MediaPipe	MTCNN
Normal	0.667	0.714	0.071
LDA Transformed	0.535	0.581	0.556
PCA transformed	0.772	0.762	0.724
PCA + Noise	-	0.709	-

3) Log Loss:

Dataset	DLIB	MediaPipe	MTCNN
Normal	17.270	13.815	17.960
LDA Transformed	17.960	17.960	16.578
PCA transformed	8.980	10.361	11.052
PCA + Noise	-	12.434	-

C. KNN

1) Accuracy:

Dataset	DLIB	MediaPipe	MTCNN
Normal	0.56	0.68	0.62
LDA Transformed	0.5	0.44	0.5
PCA transformed	0.62	0.66	0.62

2) F1 Score:

Dataset	DLIB	MediaPipe	MTCNN
Normal	0.593	0.724	0.689
LDA Transformed	0.561	0.562	0.545
PCA transformed	0.655	0.712	0.667

3) Log Loss:

Dataset	DLIB	MediaPipe	MTCNN
Normal	15.197	11.052	13.124
LDA Transformed	17.270	19.342	17.269
PCA transformed	13.125	11.743	13.124

VI. CNN

A. InceptionResNetV2

Inception-ResNet-v2 is a convolution neural architecture that builds on the Inception family of architectures but incorporates residual connections (replacing the filter concatenation stage of the Inception architecture).
The InceptionResNetV2 was applied on DLIB and MediaPipe Dataset. Hyper-parameter tuning was applied on it by changing the number of epochs and batch size. The model was trained with and without noisy data. On high number of epochs and batch size, it over fitted, while reducing them prevented overfitting .The results were the following on DLIB and MediaPipe : Results were a bit better on MediaPipe with accuracy score 0.5 and F1-score 0.625, while DLIB didn't give satisfactory results with accuracy scores below 0.3

B. EfficientNet

EfficientNet is a convolutional neural network architecture and scaling method that uniformly scales all dimensions of depth/width/resolution using a compound coefficient. The EfficientNet was applied on DLIB and MediaPipe Dataset. EfficientNet0 and EfficientNet4 were tested as the initial architecture and the model was validated through the Adam loss. Layers were modified by adding the Dropout .

Hyperparameters were tuned to overcome the overfitting and improve accuracy. However, the accuracy didn't increased significantly.

Results were same on both datasets with accuracy 0.5.

Convolution Neural Networks are generally better at image classification especially where it is trained on RGB Images. But they tend to work better with larger datasets, here in our work, datasets provided weere small in size. Due to the same accuracy CNNs tend to give less validation accuracy.
On training the CNNs with original and noisy data, results wear even worse, where due to noising the CNN is not able to learn the face features due to the noise addition. If the dataset had been large, results could have been better.

VII. Graphs

Cross - validation for different datasets and different classifiers was carried out and was plotted on a boxplot for comparison(for DLIB, graphs for other face detection engines are available in the .ipynb files submitted). (Graphs are available at the bottom of the document)

VIII. Conclusion

Thus, we conclude that Classifier Multi-layer Perceptron works better on DLIB Dataset while SVC works better on MediaPipe Dataset.

The **maximum accuracy** was achieved by clssifier **MLP** using the **DLIB** face detection while the data was **PCA** transformed.

Though we have a lot of features here, namely 64 x 64, since the number of classes here is 2 the number of Linear Discriminants can be a maximum value of 2-1 = 1.Thus, we only have 1 Linear Discriminant here, which is not sufficient for classifying the data. Thus we see low values for LDA as compared to PCA. Also, the noise wasn't applied on LDA because of as there was only 1 Linear Discriminate and applying noise on a single Linear Discriminant would deviate the data much, which can't be further solely relied . While the ensembling of CNNs could have been better according to the Paper , lack of Large datasets and Limited Hardware constrained implementing that.

Acknowledgment

The authors would like to thank...

- Dr. Richa Singh
- Harsha M.
- Rest TAs of the PRML Course

References

[1] https://arxiv.org/pdf/2004.07676v1.pdf
[2] https://scikit-learn.org/stable/
[3] https://keras.io/

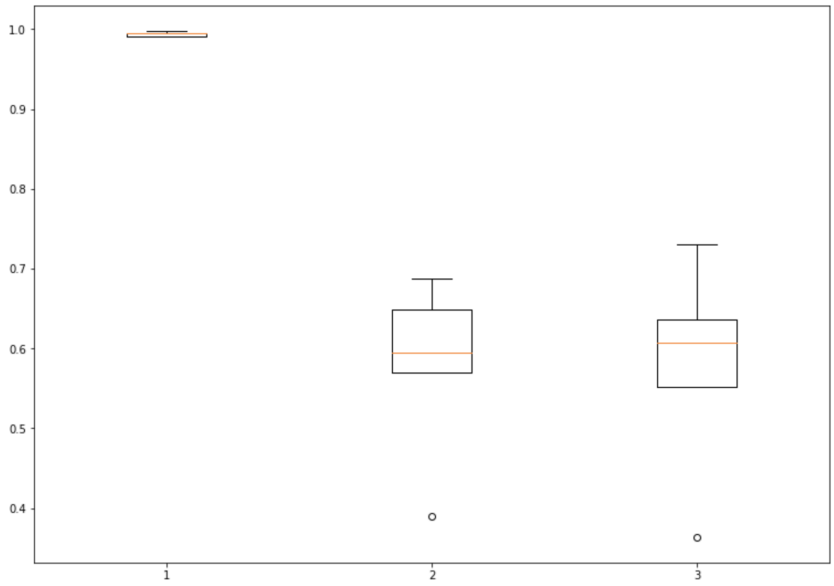

Fig. 1. SVC: 1-LDA, 2-PCA, 3-Normal

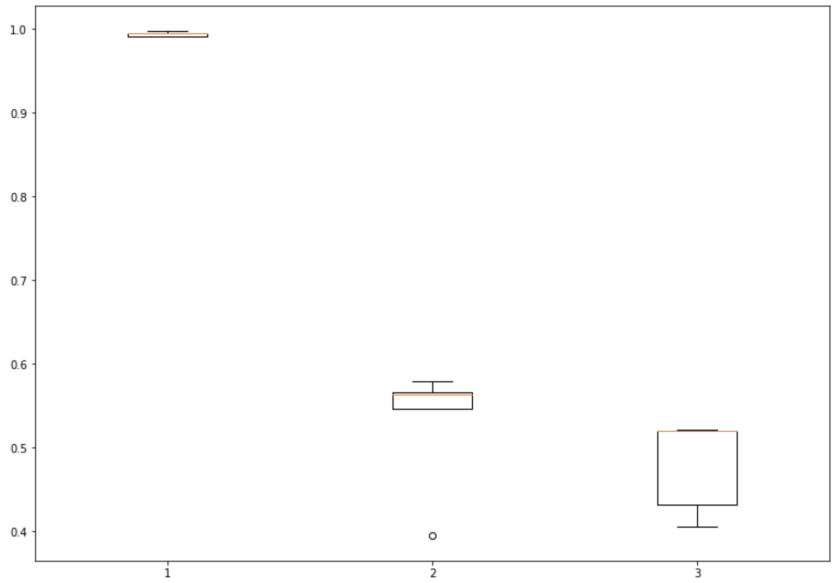

Fig. 2. MLP: 1-LDA, 2-PCA, 3-Normal

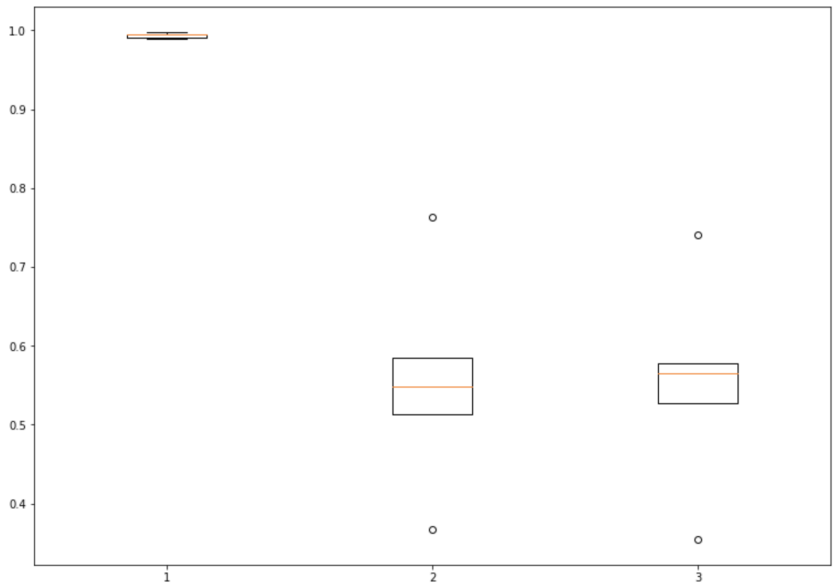

Fig. 3. KNN: 1-LDA, 2-PCA, 3-Normal

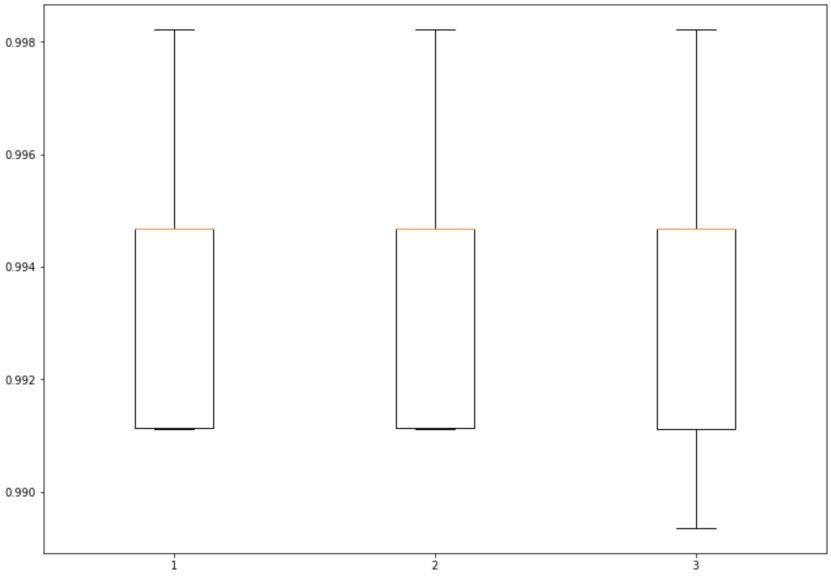

Fig. 4. LDA: 1-SVC, 2-MLP, 3-KNN

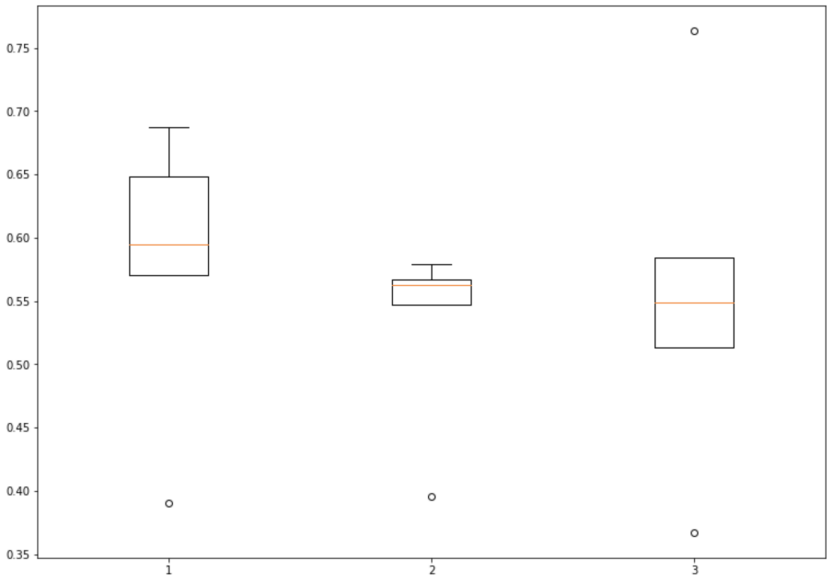

Fig. 5. PCA: 1-SVC, 2-MLP, 3-KNN

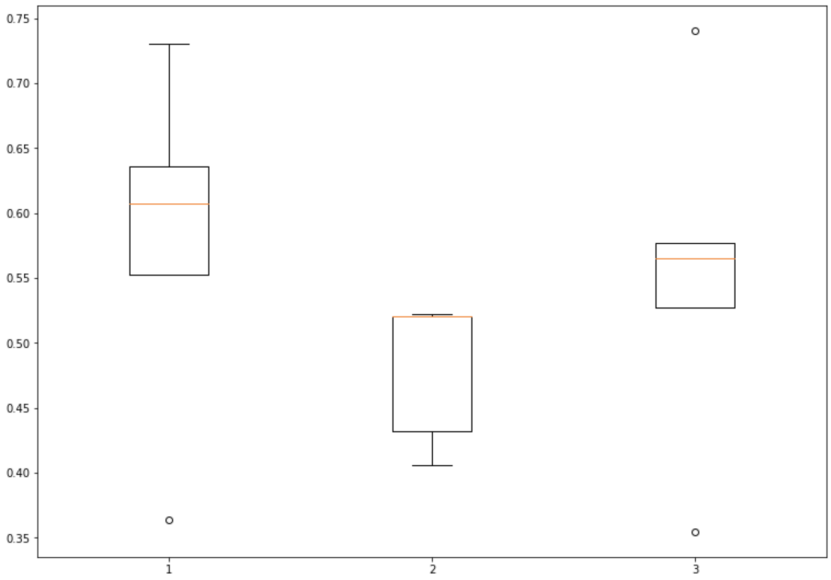

Fig. 6. Normal: 1-SVC, 2-MLP, 3-KNN