# Identification of Dominant Speaker in Short Videos

Harsh Sahu
2014ME10657

Vikas Deep
2014ME10692

Siddharth Agrawal
2014ME10688

## ABSTRACT

In this report, we discuss our approach, experiments and their results for solving the problem of identification of dominant speakers from short videos. We discuss our approach to data pre-processing, the model architecture used for the problem. We have also provided the reasons behind the choice of the experiments conducted and have included possible reasons to explain our observations.

## 1 Problem

The problem under consideration is the *Identification of Dominant Speaker from Short Videos.* The problem involves first preparation of training data from the given videos, removing noise, and then training a model for the multi-class classification problem. In the next few sections, we discuss each of the problems in detail.

## 2 Dataset

### *2.1 Training Data Using Face and Eye Detectors*

We extracted frames out of the videos by using by using Python openCV library. The frames from videos of different speakers were saved in separate directories so that we can assign naive class labels to our training data. For the video of each speaker we used openCV's haar cascade classifier. We used the cascade face detector to identify human faces in each of the frame from the video and then we cropped the region around the face to form our training set as shown in Fig 2.1.1 and Fig 2.1.2. We also used two types of cropping, in one of the types we didn't have any background noise in the image, we call this supercropped dataset. While in the other type, we include some background noise in the image. In the later sections, we compare the accuracies of the models obtained from training on these two datasets.

Since, there were a few frames in each of the video in which the audience was also shown or someone from the audience was shown. Therefore, while filtering we imposed a condition that we assign the label of a particular speaker to a frame only when the number of detected faces in the frame was exactly equal to one. This helped us to significantly reduce the number of images of the crowd in our training set.

This strategy seemed to work fine for the five classes other than the Sadhguru. However, due to the presence of a long beard on Sadhguru's face the cascade face detector was not able to identify the facial features and therefore didn't identify it as a human face. Therefore, for the videos of Sadguru, we made use of the eye detector to filter out relevant frames as shown in the fig 2.1.3. We gave the label of Sadguru to a particular frame only when the number of detected eyes in a frame was exactly equal to two.

Fig2.1.1 Over-cropped

Fig2.1.2 Less-cropped

Fig 2.1.3 Sadhguru identification using eye detector

### *2.2 Other Heuristics*

We measured the correlation between successive frames. We did this by making use of the numpy.corrcoef function. After this, we plotted this correlation factor value with frames as shown in fig 2.2.1. We also did the fast fourier transform on these correlation values to separate the noise from the speaker frames. The idea was that the speaker and the noise frames would have a very low correlation factor.

We made use of another heuristic approach to filter noise from the frames. For each of the speaker, we kept a reference image during filtering. So, for each frame we calculated its correlation with the reference image and the the previous frame. If the maximum of the two correlations was higher than a threshold than the frame got labelled as the particular speaker. But if we identified a particular frame as noise, we didn't use that to calculate the correlation of that with the next frame order. Basically we skipped that frame and the correlation of the next frame was calculate with the reference image and the last identified speaker frame. This way we got a training set for the seven classes, six for the speakers and one for the noise. We then prepared numpy arrays of this data for training our model.

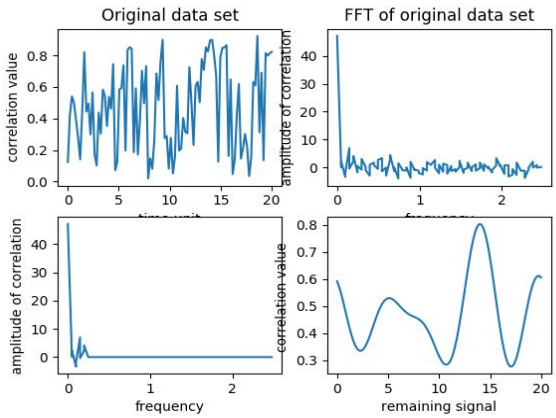


Fig 2.2.1 Correlation value and its FFT

## 3 Model Architecture

We made use of the Keras Deep Learning framework to implement our models. We used a pre-trained inception net network trained on Imagenet dataset and then added two fully connected layers to it, with the last one being a softmax layer.Our output was a seven dimensional vector, corresponding to the seven classes as mentioned previously. We trained our models by passing different datasets to the model. The model was trained using ADAM, whose implementation is provided as a function call in Keras. The loss function we made of use of was *categorical cross-entropy* and the activation used was reLU, with a dropout probability of 0.5.

## 4 Experiments

### *4.1 Cropping Methods*

We trained our model on the two different datasets: super cropped and normally cropped. We did this experiment as we wanted to assess the effect of introducing noise in the training data on the model accuracy. We has suspected that the supercrop model will likely overfit and do worse on the new frames which had different background noise than the training data. The results of our experiment is shown in the fig 4.1.1. We observed that the model performed better for the less cropped dataset with the highest accuracy of 52.5%. The highest accuracy observed for the more cropped dataset model was 46% approximately.

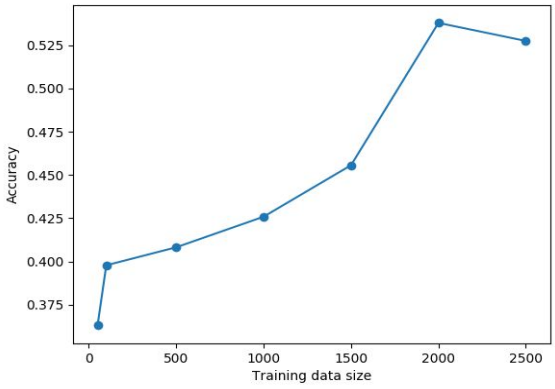


Fig 4.1.1 Accuracy plot for the model trained using less cropped images of Test Dataset 1

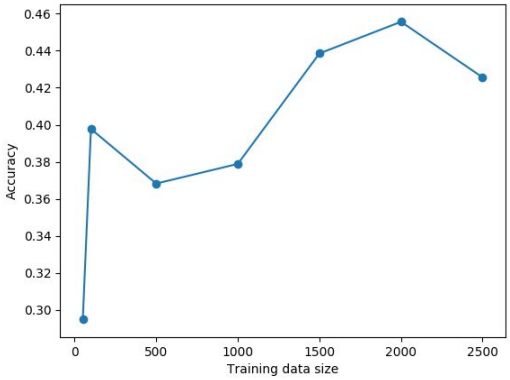


Fig 4.1.2 Accuracy plot for the model trained using more cropped images of Test Data Set 1.

## *4.2 Training Data and Accuracy*

We trained our model on different sizes of training data and observed the test accuracy. We trained our model on input data of sizes 100, 500, 1000, 1500, 2000 and 2500. For the more less cropped dataset model, we observed a steady increase in the test accuracy upto dataset size = 2000. However, for 2500 points in the dataset we observed a decrease in the test accuracy as depicted in Fig 4.1.1. For the model trained using the more cropped images, we observed a different trend of accuracy as depicted in Fig 4.1.2. There was an initial increase followed by a decrease and then another increase and decrease eventually.

## *4.3 Deep Learning Visualisers*

### 4.3.1 Gradient Ascent

Based on our learnt weights, we tried to construct the image which will maximise the probability of getting a particular class given the trained model. The generated images for the classes Atul Khatri is shown in the fig 4.3.1. However we were not able to extract any useful information from this image.

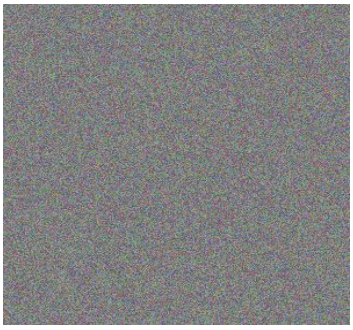

Fig 4.3.1 Generated image from gradient ascent

### 4.3.2 Visualisation of Activation Values

To get a sense of the features that are being learnt in the hidden layers of our model we plotted the activations of the layers as shown in Fig 4.3.2 .

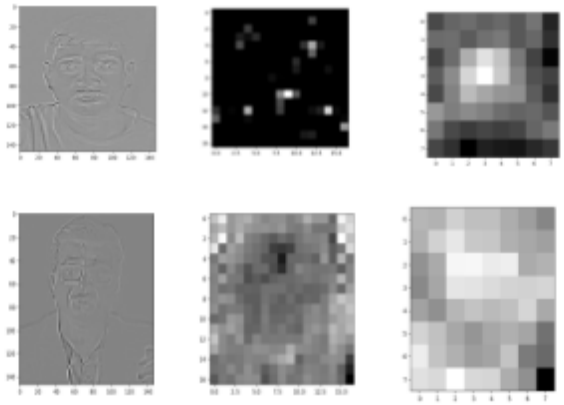

Fig 4.3.2 Layerwise activation value for Shailendra and Atul

## *4.4 Model Performance on Different Datasets*

For the test data set 2 we trained our model on input data of sizes 500, 1000, 1500, 2000 and 2500 of less cropped training data as CNN model trained on less cropped data was giving better accuracy. We plotted the accuracy on test data 2 wrto training data size as shown in figure 4.4.1

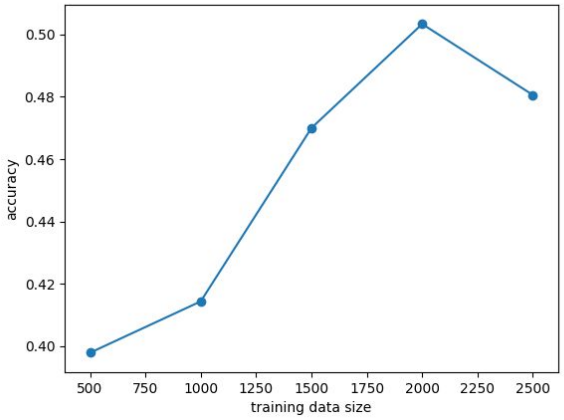


Fig4.4.1 Accuracy plot for the model trained using less cropped images of Test Data Set 2.

We compared our models' accuracies graphs on the two different datasets given to us. We observed that both graphs are following the same pattern. Maximum accuracy

obtained on 2nd test data is 50 % and the maximum accuracy obtained on both test data are nearly same.

To visualise the both test data, we made use of the tSNE algorithm. For the two sets of test data, we first reduced the dimension of the data to 50 by doing a PCA(Principal Component Analysis). After this dimension reduction, we further reduced the dimension of the data by using the tSNE algorithm. The results obtained for the two datasets, the one given earlier on the one given out just the day before are shown in the Fig4.4.2 and Fig 4.4.3 respectively.

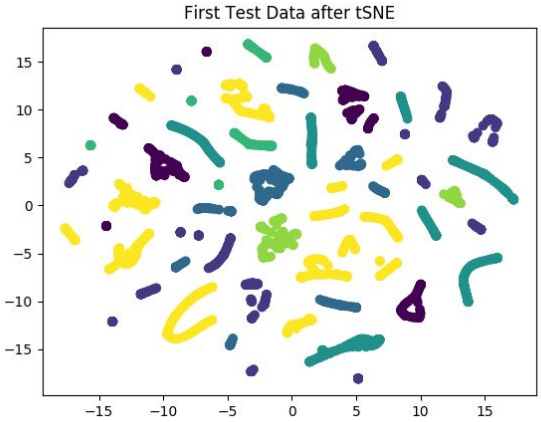


Fig 4.4.2 Clusters obtained after doing tSNE on Dataset 1.

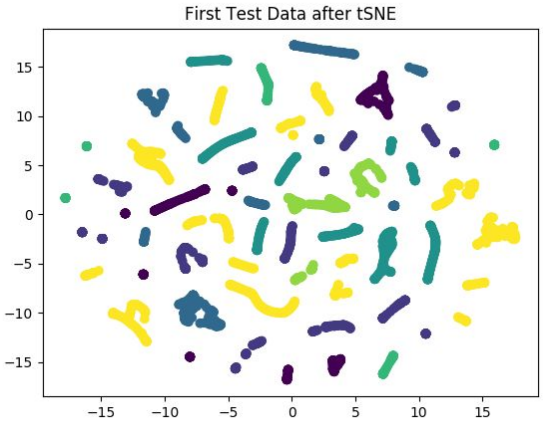


Fig 4.4.3 Clusters obtained after doing tSNE on Dataset 2.

From the plots obtained for tSNE in the above figures, we observed clusters corresponding to each of the seven classes in our data. There were multiple clusters for the same type of speaker in the data. Also, clusters for the class of noise were sparser than the other classes. The possible reason for observing different clusters for the same speaker, is that since different frames of a particular speaker has been taken from videos that have very different background noise. Therefore, though the images share some commonality, there are quite different from each other. The class of noise is sparser because very diverse frames have been classified with the same label in this class.

## 5 Discussion

We observe that our model performed better when we trained it using the less cropped images than the more cropped images. The reason for this can be that the model trained on the more cropped images suffered from overfitting. We also observe a decrement in the test accuracy after we increase the training data size beyond 2000. This also points to the importance of having an ambient size of training data to prevent overfitting. From the visualisation plots of the activations of the different layers of our model, we see that as we go deeper in the network, the layers try to learn more specific features of a class, while the initial layers learn more general features.This can be observed in the figure from the previous section, where we were able to identify the outer edges of the face of the speakers, from the images plotted corresponding to the activation of initial layers.