Frame by Frame Classification of Videos

Kapil Ahuja 2014MT60663, Ayush Gupta 2014CS50281, Ashish Hooda 2014EE10429

***Abstract*—In this assignment, we have tried to solve the task of classifying frames in a video based on the face present in the frame. There are seven categories; six correspond to speakers and a seventh category is for the noise images, and also as a none-of-these category. The idea was to generate a dataset from the provided videos, extract faces out of these frames, train a convolutional network to classify the categories and then use these saved models to get predictions for new face images. We experimented with VGG16, ResNet18 and AlexNet models, fine tuned them for our case and also evaluated t-SNE visualizations for visualizing how the networks classify the dataste images (see what the network learns). For this assignment, we used PyTorch and Keras to train and test models.**

I. Data curation

We took 5 videos of each person to generate the dataset. We wrote a script that takes a video, extracts faces in each frame using a classifer(discussed later), crops these faces and then saves them in directories based on the category. After these images are generated, it shuffles them and splits the dataset in a 70:30 train:test ratio. The shuffling is done to ensure images from all videos are taken in the split.
Since a video runs at typically 24fps, and in the kind of videos that we have, there isn't much activity over a period of 1 second. So the frames were extracted at a step of 1 in 20 frames. (20 because sometimes there were no faces and we needed more images).

A. OpenCV - Haar Cascade

The classifier we used initially was the default OpenCV Haar cascade classifier based on the Viola Jones detector. This classifier could detect faces in every video, except those of Sadguru Jaggi Vasudev. So we came up with a heuristic to create a training set for Sadguru's category. We now extracted eyes from the image, estimated the width of the face by the separation between eyes (with an offset of 50 pixels chosen by trial and error), and then estimating the height of the face by using the Golden Ratio(1.61) which holds for most human faces. This method now gave us Sadguru's images to train the network, but was not robust in that it wouldn't work for a new test image. So we came up with the following:
We trained a new Viola Jones detector to feed the OpenCV cascade classifier which could detect Sadguru's face as well as the other faces without any heuristic involved. To train this classifier, we took about 700 images, 100 from each category(non noise) as positive examples of faces, and 2000 negative examples from face.urtho.net which are grayscale images of trees, cars, houses and all other object examples which are not faces. Now, the classifier was able to detect Sadguru's face with ease and worked better than the above two approaches.
This method still suffers for some unseen images where Sadguru looks side ways.

B. Dlib - HOG based Extraction

We also tried face bounding box extraction using python's dlib library which works by detection of 68 key facial landmarks. It uses Histogram of Oriented Gradients (HOG) combined with a linear classifier. Both, haarcascade and dlib worked very decently on frames of all people except sadhguru. Many of the facial images extracted using OpenCV(haarcascade) were noisy and contained random areas in the frame or the beard. Dlib was able to extract Sadhguru's faces with much less noise than OpenCV but still the rate of Face extraction was very less. From around 40k frames containing Sadhguru, Dlib was able to extract just around 4k(10%) facial images. Both these methods worked decently well for other people though. Dlib's better rate of extraction can perhaps be due to it detecting facial landmark regions, which it could have detected well in upper half of Sadhguru's face which is not detected by beard.

II. Training the network

A. Using FaceNet pretrained Network

We used transfer learning on VGG-Face model, pretrained on face(2622 celebrities) dataset. We can safely assume that the features encoded in the model weights to discriminate the 2622 celebrities are enough to accurately describe any face. As model is trying to predict things similar to what it was trained on(faces), transfer learning alone should do the trick. We removed the last two FC layers and got hence get a feature vector of 7 x 7 x 512 for each face. We then learn our model on top of these features using 2 FC layers and finally a Softmax activation.

We had initially not kept a seventh class as a noise class and were planning on keeping a threshold of final soft-max probability for eliminating faces which did not belong to these six classes but we got varied values of probability for faces which didn't belong to these six people, hence we decided on training the model using the seventh noise class which contained faces from several other datasets and also some random non-face photos as we do get non-face images sometimes.

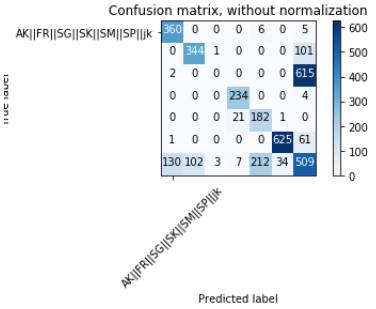
When model was being trained on just the six faces, with no noise class, then the transfer learning worked very well just 60 random faces (in total,from all the classes) from each class achieving above 99.5% accuracies on the validation data in just 4-5 epochs. This may also be attributed to validation faces being extracted from the same videos(may lead to same background and having similar face wear, eg.same specs or same turban).

If a frame contains more than one face, then its labelled as the person who face has highest probability of matching with any face of the frame.

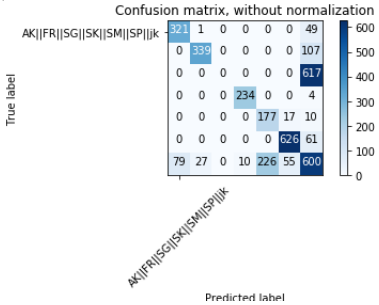
When model was being trained with a seventh noise class, validation accuracy increases as more dataset is added, but still doesn't need much data to reach high accuracy. It increases from 95.7% for 100 train faces to 99.6% for 400 train faces.The increase hear can be attributed to model getting to learn noise better as more and more examples are added.

On test video, we get accuracy of 63.3% using the dlib for extraction and using transfer learnt VGG16 face model. We use 400 faces in total, each class having about same number of samples. We see from the confusion matrix below that Sadhguru is not getting extracted at all. So, extraction is the main bottle neck in our classification. As frames having no face is labelled as 'jk', so all Sadhguru frames are labelled as 'jk'.

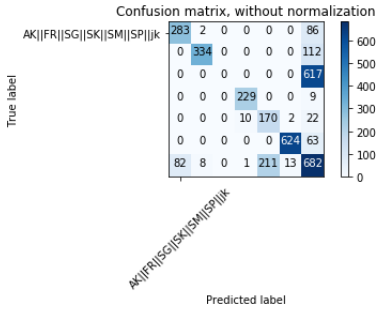
A nice thing to see here is that most true labels which are negative get labelled as Sandeep Maheshwari, its because most of the random people wear specs(atleast someone in audience), and it makes such faces closer to SM than to noise because Noise has only partial faces who don't wear specs.



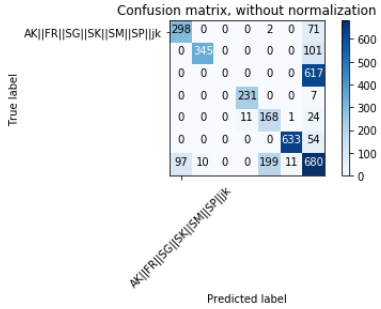
Increasing the bias in noise class, having 2000 noise faces and 200 faces each of our six people increases the recall of all classes, this is due to regularization effect that class noise has on the model. Earlier almost every person with specs was labelled as Sandeep Maheshwari and bald person as Sourabh Pant, but adding class noise decreases this effect and accuracy increases to 64.5%.



Increasing the bias in noise class even further,that is having 10000 noise faces and 200 faces each of our six people decreases the recall of all the persons, that is model gets over regularized and even faces that should be one of six persons gets labelled as Noise. This however still increases the accuracy to 65.2% due to this particular test data. In this case, lots of noise frames which were earlier labelled as SP get labelled as 'jk' due to increase in noise bias.



Testing on the Second Video data increases the accuracy to 66.15%. Number of faces extracted from frames increased from 3397 to 3781. As there was difference in pre-processing, Noise induced in video_1 made it difficult for dlib to identify facial landmarks. Almost all the values along the diagonal see an increase as facial extracted data is less noisy too.

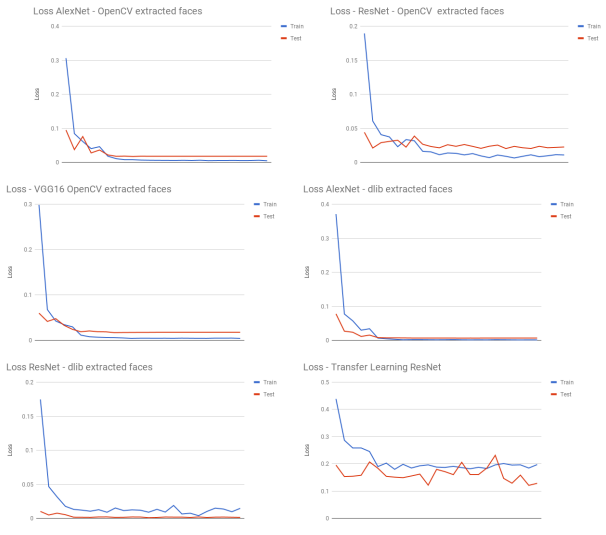


As we clearly see that extraction is the bottle neck due to inability of dlib to extract Sadhguru's faces, We can improve upon this drawback by using our own trained OpenCV cascade file for sadhguru on frames where dlib detects no faces. This will help many of 'jk' labelled faces of now to get labelled as 'SG' and will highly boost test video accuracy.

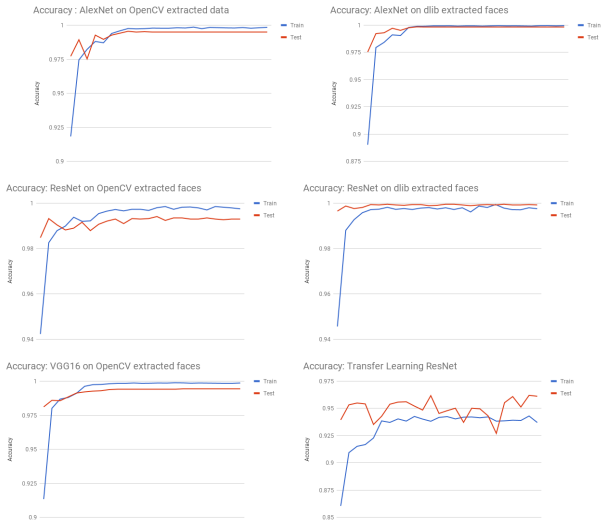
B. Using ImageNet pretrained Network

We trained several networks with each extraction method that we used. The ResNet, AlexNet and VGG16 pretrained networks were used for the initialization. A fully connected layer was added to each model with inputs as the original model's output, and 7 class outputs. CrossEntropy was used as the loss function and SGD with momentum 0.9 was used as the optimizer. Also, the learning rate was initialized at 0.001 and successively decayed by a factor of 0.1 every 5 epochs. Since the networks are pre-trained and we have a relatively small dataset, we only trained them for 25 epochs. We also tried transfer learning by training only the final layer.

The loss-epoch plots for each network are plotted below. Each of them reach over 99% accuracies. However, test accuracies differ significantly. This is probably because the networks are trained on images from videos where the speaker's facial reactions, lighting background are constant for a significant period of time, which causes trouble when new light conditions come up.



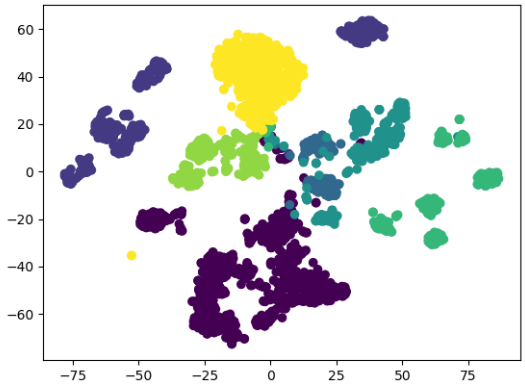
The first three plots are the training loss plots when models are trained over OpenCV extracted faces. The modified cascade classifier was used to train them. It performs better than the default OpenCV classifier. The next two plots represent losses for networks trained over data extracted by dlib, which uses Local Binary Patterns to detect faces, and gives better bounding boxes. As the graphs show, losses are lower when dlib faces are used for training.
The final graph shows the losses for the transfer learning case, when we train only the final fc layer. This has much higher values compared to the previous methods. The data for this was extracted using the OpenCV cascade classifier. The accuracy graphs on the test data we got (from the same videos) are plotted below:

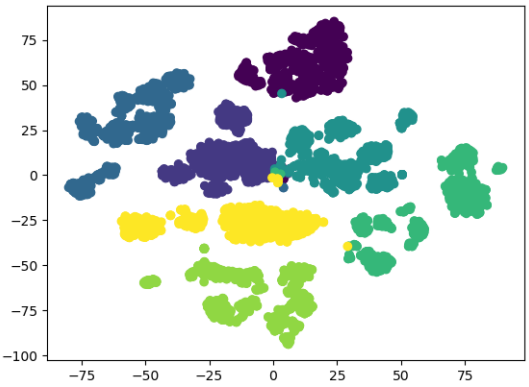


Similar to the case of loss plots, the accuracies on extracted faces are higher for dlib extracted faces, and lower for the modified OpenCV cascade classifier. The ”transfer learnt” ResNet model has much lower accuracies than the fine tuned ones.

III. t-SNE Visualizations

The t - distributed Stochastic Neighbor Embedding (t-SNE) method allows us to visualize high dimensional vectors, in a lower dimension and it keeps the separation between the vectors as much as possible. So if our network was learning correctly, the activations just before the final layer for every test image should form separate clusters. The better the clusters are made, the better is the model. The following are the t-SNE visualizations of the second last layer for ResNet in the case of i) OpenCV extracted faces, and ii) dlib extracted faces.





The above two plots show more evidence that the same model learns and classifies better in the case when it is trained on dlib extracted data than when it is trained on OpenCV extracted data. The 7 categories have clear demarcations.

IV. Alternate Methods

A. Audio based classification model

Each audio was split into several audios of 5 seconds duration. These audio have been sampled at 48000 frame per second. MFCC features are extracted from the samples and a GMM model is learned with 6 components. We trained a two layer neural network with 50 hidden nodes in each layer. We trained on MFCC features obtained from 220 audio each of 5 second duration. The model achieved training accuracy of 100% and on a our test data-set containing 220 disjoint

audio each of 5 s duration, it got 80% accuracy. On further fine tuning the hyper-parameters of network and increasing the depth of network or using CNN architecture, the test accuracy may increase. We could have used a heuristic of audio and image analysis, if test data contained audio encoded Video.

B. YOLO based person detection

Using Lightnet (python interface to Darknet), we extracted persons from frames with very good accuracy including Sadhguru. We then intended to train a pretrained network to classify people using these body images but the bounding box covered a lot of background, and it is very unlikely to work well on test data as it learns background as well as people's clothes because there is not much variation in dresses across 5-6 videos. YOLO based face detection has been shown to detect face bounding box really, even with half face hidden and at different orientations. We weren't able to find face weights but we expect this to surely work better than dlib or OpenCV as has also been shown by many people (see example).

V. SUMMARY

We achieved best accuracy of 65.2% on video_1 and of 66.15% on video_2. Noise induced in video_2 affects dlib face extraction and also induces noise in extracted facial data. The main reason for less accuracy however is not face classification but rather face extraction. This can in part be resolved by using a yolo-based face detection. Faces extracted with OpenCV is noisy but dlib extracted faces are better as can also be seen by loss graphs while training.

Transfer learning works very well in predicting things similar to what model is trained on, and hence face-net trained data worked very well with only a few images of each class. The problem here remains of adding a noise class and its seen that model gets better as more faces are added in the noise class which helps better in predicting noisy faces(not belonging to our six people).

VI. REFERENCES

1) Keras FaceNet VGG Blog
2) Keras FaceNet VGG Code Reference
3) Transfer Learning in Keras Code Reference
4) Confusion Matrix Code Reference
5) Faces for Noise Class - 1
6) Faces for Noise Class - 2
7) PyTorch Transfer Learning Tutorial
8) Training OpenCV Haar Classifier
9) CS231n: Visualizing CNNs
10) t-SNE visualization by Laurens van der Maaten