<u>ELL888-Assignment-2-Speaker Recognition</u>

Sai Teja Ramadev
2015EE10466
Sant Prasad Mishra
2017EEZ8208
Shantanu
2016EET2646

PROBLEM STATEMENT: Speaker recognition from YouTube videos.

MODEL TRAINING:

Four approaches were attempted to train the model (in the order of execution):
- Using Imagenet pre-trained models for image classification
- Using Keras Openface pre-trained model for facial recognition
- Using 7-layer CNN on cropped Images.
- Using spectographs with autoencoders

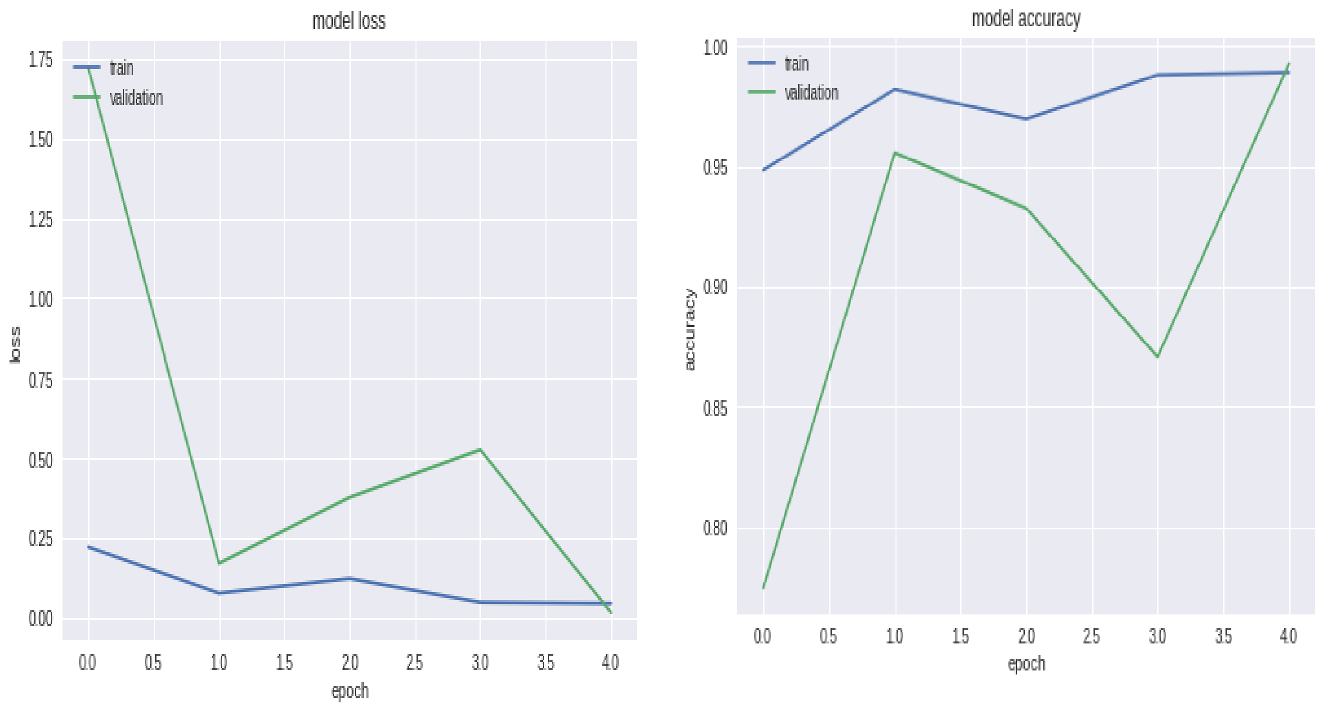
Only the fourth approach has given very good accuracies on downloaded YouTube videos not part of the problem statement. However the test data doesn't have voice and so the model cannot be tested on the given test data.

Approach 1:Pre-trained Networks (ResNet50, InceptionV3, VGG16)

Preprocessing:
1) Generating frames from youtube videos:
File: download&preprocess_data.py
We used Pytube module to automatically download all videos (360p .webm frormat) to separate folders for every speaker. Then VideoCapture method of opencv-python module is used to generate frames from each video and placed in separate folders for every speaker.

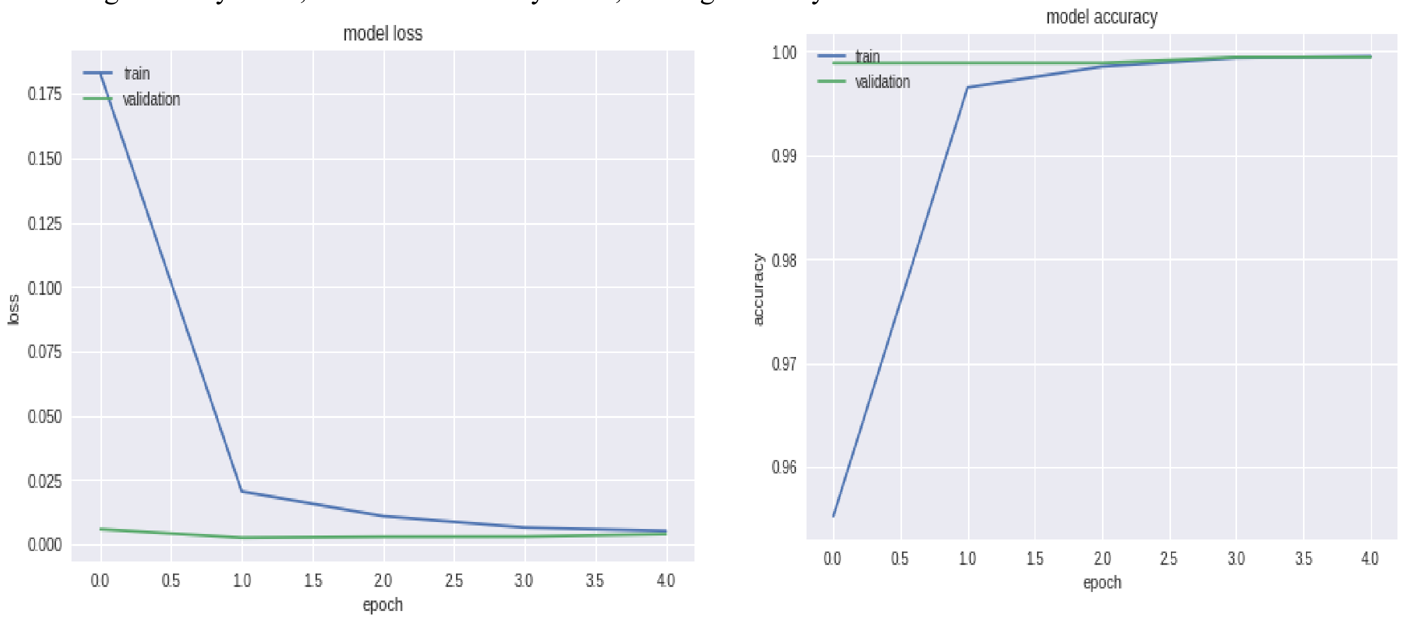
2) Getting training, validation and test data from frames:
File: final_preprocess.py
We took 4 videos of each speaker and extracted 750 frames from each video. So total number of images in **dataset is 6*4*750= 18000 frames**. We further divided dataset into training (80% of dataset), validation (10% of dataset) and test set (10% of dataset). We separated noise frames manually from this dataset and considered them as 7th class. We used flow_from_directory method in keras to load data in batches.The top softmax layer of each model is replaced with a Global Average Pooling layer, a fully connnected latyer and a softmax layer with 7 categories: 6 speakers and noise. Batch size is 32 images and adam is used as optimizer.

ResNet50:
File: resnet.ipynb
We used keras to load pretrained ResNet50 model and fine-tuned this model on our dataset
Training accuracy: 98%, Validation accuracy: 99%, Testing accuracy: 99%



<u>Fig: Results for ResNet50</u>

InceptionV3:
File: inception.ipynb
We used keras to load pretrained IncepionV3 model and fine tuned this model on our dataset
Training accuracy: 99%, Validation accuracy: 99%, Testing accuracy: 100%



<u>Fig: Results for InceptionV3</u>

VGG16:
File: VGG.ipynb
We used keras to load pretrained VGG16 model and fine tuned this model on our dataset
Training accuracy: 99%, Validation accuracy: 99%, Testing accuracy: 99%

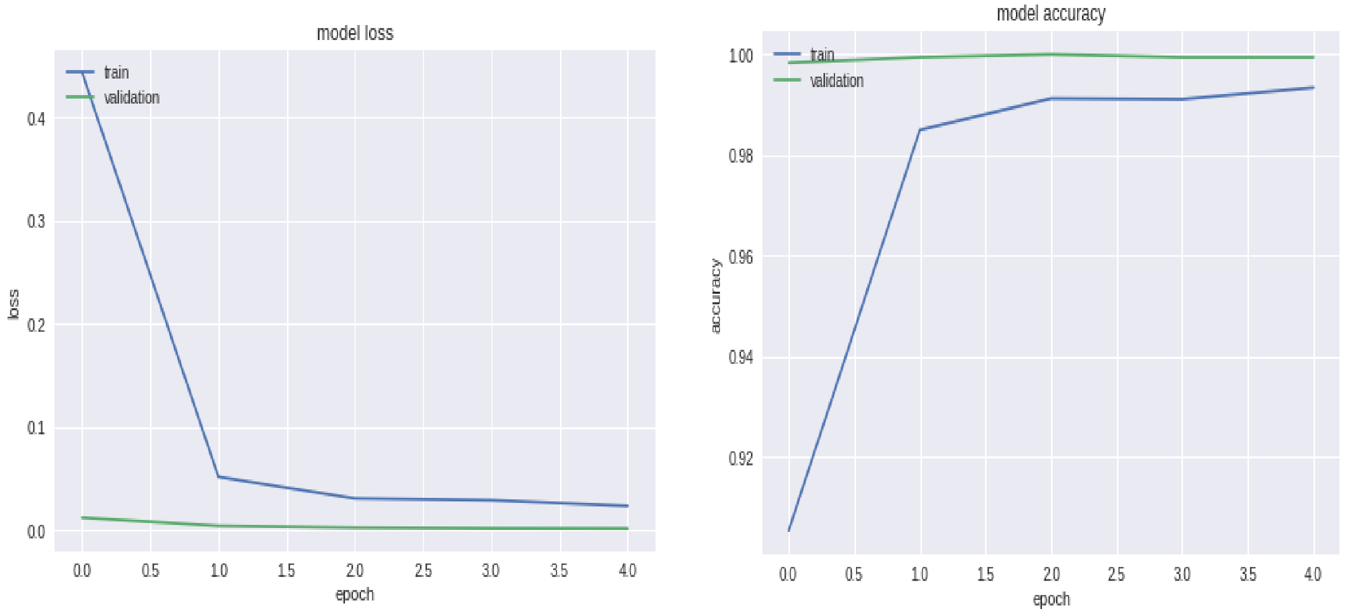


Fig: Results for VGG16

The accuracies on the test_vid_1.avi is 35% by ResNet50, 55% by VGG16 and 60% by InceptionV3.
The accuracies on the test_vid_2.avi is 36% by ResNet50, 56.4% by VGG16 and 62.5% by InceptionV3.

We didn't get very good results with ResNet50, InceptionV3 and VGG16 networks because they are overfiiting to backgrounds of images and other noises.

Since our models are overfitting to background, we decided to learn models on cropped faces from image. We cropped faces from image with the help of openCV's deep learning detector. OpenCV's deep learning face detector is based on the Single Shot Detector (SSD) framework with a ResNet base network. These face detector failed to detect faces in many images.

Approach 2:Facial Recognition with Pretrained FaceNet(Keras-OpenFace)

File: facenet.ipynb, faceprepro.py
FaceNet encodes an 96X96X3 image into vector of 128 numbers(top layer features) by passing it through network. By comparing two such vectors, one can then determine if two pictures are of the same person.
Train data: 160 cropped images of each speaker
Test data: faces cropped from frames of test_vid_1.avi(Algorithm detected faces in 2162 images)
faces cropped from frames of test_vid_2.avi(Algorithm detected faces in 2217 images)
From 160 images of each speakers, we computed 160 encodings for each speaker.
For image in test data, we took average of distances from all 160 encodings for each speaker. We classified image to the class which it has least average distance.
The accuracy on test_vid_1.avi is 47.6%
The accuracy on test_vid_2.avi is 47.1%
The accuracies are not improved.

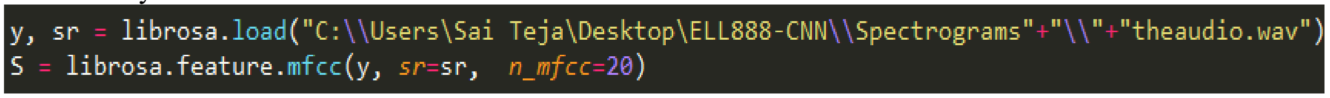
The problem with this model is that we could not do frame level classification since openCV fails to detect faces from every frame from video.

Approach 3: 7-layer CNN on cropped Images

File: CNN_faces.ipynb
Instead of using pretrained FaceNet, we trained 7-layer CNN on the same cropped data.
Train data: 160 cropped images of each speaker
Test data: faces cropped from frames of test_vid_1.avi(Algorithm detected faces in 2162 images)
faces cropped from frames of test_vid_2.avi(Algorithm detected faces in 2217 images)
The accuracy on test_vid_1.avi is 57%%
The accuracy on test_vid_2.avi is 58%

Approach 4: Voice Recognition (Using spectrograms with autoencoders)

Files: extract_spectrograms.py & autoencoder.ipynb
The reason we want to tackle this problem through audio is because visual recognition is overfitting to backgrounds and other noises. We extracted audio from video. Using Librosa module we extracted Mel-Frequency cepstrum Coefficients(MFCC) of every 5 seconds of audio. We also ignored first 20 seconds and last 20 seconds of audio since this will be mostly noise. We plotted 64X64 size spectrograms and trained 5-layer CNN on it. We used keras to construct CNN.



```
y, sr = librosa.load("C:\\Users\Sai Teja\Desktop\ELL888-CNN\\Spectrograms"+"\\"+"theaudio.wav")
S = librosa.feature.mfcc(y, sr=sr,  n_mfcc=20)
```

Training Set: 355 spectrograms of each speaker. Total Size = 2130 spectrograms
Validation Set: 39 spectrograms of each speaker. Total Size = 234 spectrograms
Test data: https://drive.google.com/open?id=1_Y9YfscBZr0prqsVOEhw3TnZwCkIakbT
This model gave better results than pre-trained models on images but we were not satisfied with results. The main problem of this model was that data set has noise like audience laughing, audience asking questions, background music etc. Our model was overfitting to this noise. So we used autoencoder to reduce noise. With the help of autoencoder we reduced the size of spectrogram from 64X64 to 16X16. Then we trained 5-layer CNN on this low dimensional spectrogram. The results improved significantly compared to previous models. We have downloaded couple of videos from youtube and tested this model on these videos. Our model has given around 85-90% accuracy.

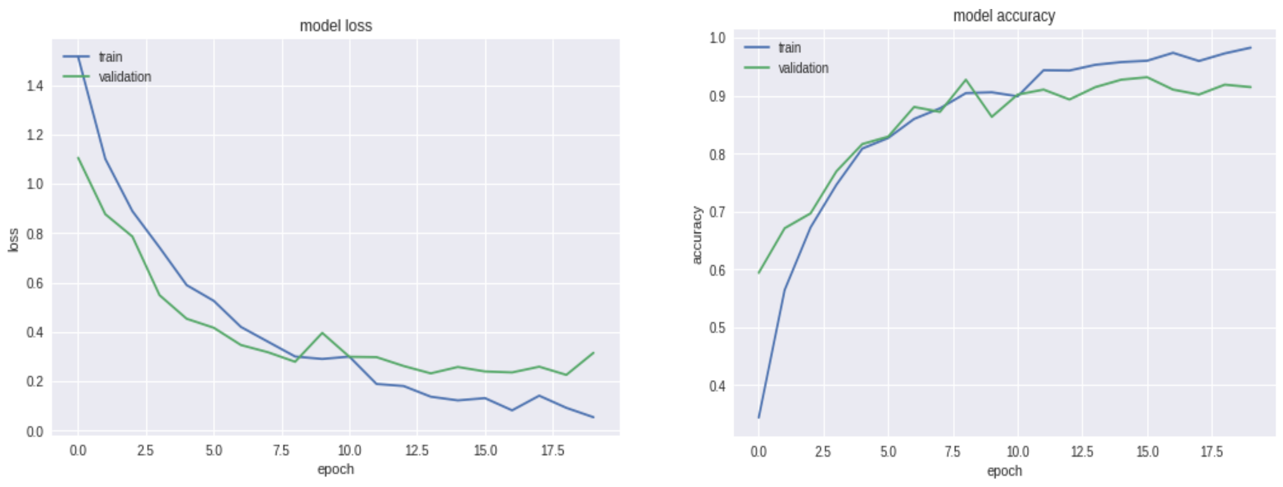


Fig:Results for Using spectrograms with autoencoders

Summary:

Despite trying 6 models and two ways (Visual and audio) to tackle this problem, only voice recognition is giving satisfactory results. Visual methods are overfitting to training data. Visual data is so redundant (no variation in frames) with same backgrounds, same dress colors in many frames. We used only 750 frames from a video even though accuracies are not satisfactory. Facial recognition with pretrained FaceNet might work if we fine-tuned FaceNet instead of just using top layer features for recognition.

What make this problem challenging is dataset. If there is enough variation in dataset, one could easily solve this problem with CNN. Voice recognition can be improved more if we were able to remove noise like audience laughing, audience asking questions etc. with the help of some filters.

With regard to this problem, it can be concluded that it is better to use voice recognition than visual recognition because audio is less redundant than visual data.

References:

1) http://minhdo.ece.illinois.edu/teaching/speaker_recognition/speaker_recognition.html
2) https://www.analyticsvidhya.com/blog/2017/08/audio-voice-processing-deep-learning/
3) https://blog.keras.io/building-autoencoders-in-keras.html
4) https://keras.io/applications/
5) Facial recognition course on Coursera- Andrew ng
6) https://github.com/iwantooxxoox/Keras-OpenFace
7) https://www.pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/