Extraction of Facial Features from Speech

Saiteja Talluri 160050098
Neelesh Verma 160050062
Ankit 160050044
{saitejat,neelesh,ankitankit}@cse.iitb.ac.in

Abstract

In this project, the main motivation was to infer about a person's look from the way they speak. We design and train a deep neural network to perform this task using thousands of natural YouTube videos of people speaking. During training, our model learns voice-face correlations and then we used it for voice recognition to evaluate the efficiency of our model. The training is done in a self-supervised manner, by utilizing the natural co-occurrence of faces and speech in Internet videos, without the need to model attributes explicitly.

1 Introduction

There is a strong correlation between speech of a person and his/her appearance, part of which is a direct result of the mechanics of speech production: age, gender (which affects the pitch of our voice), the shape of the mouth, facial bone structure, thin or full lips — all can affect the sound we generate. In addition, other voice-appearance correlations stem from the way in which we talk: language, accent, speed, pronunciations. In this project, our goal is not to predict a recognizable image of the exact face, but rather to capture dominant facial traits of the person that are correlated with the input speech.

We design a neural network model that takes the complex spectrogram of a short speech segment as input and predicts a feature vector representing the face. More specifically, face information is represented by a 4096-D feature that is extracted from the penultimate layer (i.e., one layer prior to the classification layer) of a pre-trained face recognition network. To train our model, we use the AVSpeech dataset (Ephrat et al., 2018). Our model is trained in a self-supervised manner, i.e., it does not require additional information, e.g., human annotations.

2 SpeechToFace Model

The large variability in facial expressions, head poses, occlusions, and lighting conditions in natural face images makes the design and training of a SpeechToFace model non-trivial. A very straightforward approach of regressing from input speech to image pixels does not work because such a model has to learn to factor out many irrelevant variations in the data and implicitly extract a meaningful internal representation of faces — a challenging task by itself.

We used the same pipeline as the Speech2Face (Oh et al., 2019) as shown in Figure 1. comprising of two main components: 1) a voice encoder, which takes a complex spectrogram of speech as input, and predicts a low-dimensional face feature that would correspond to the associated face; and 2) a face decoder, which takes as input the face feature and produces an image of the face in a canonical form (frontal-facing and with neutral expression).

We trained only the SpeechToFace model that predicts the face feature and the face decoder model (Cole et al., 2017) was not available open source, so we decided to implement it as a future work. During training, the face decoder will be fixed, and the voice encoder that predicts the face feature is only trained. Moreover the complex spectrogram input and the 4096-D VGG face features (Parkhi et al., 2015) (used to compute loss function) are precomputed to speed up the training process.

3 Implementation Details

3.1 Preprocessing

We used the AVSpeech dataset (Ephrat et al., 2018) comprising of thousands of video segments from YouTube. Other libraries and tools that we used for pre-processing are described below :

- **youtube-dl** – download the videos from the csv files corresponding to start and end times.

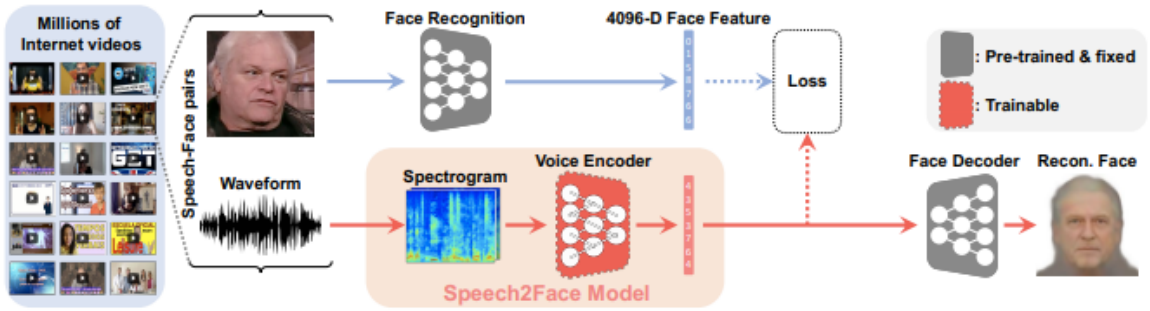

Figure 1: SpeechToFace model and training pipeline (Oh et al., 2019). The input to the network is a complex spectrogram computed from the short audio segment of a person speaking. The output is a 4096-D face feature that is then decoded into a canonical image of the face using a pre-trained face decoder network (Cole et al., 2017). The module we train is marked by the orange-tinted box. We train the network to regress to the true face feature computed by feeding an image of the person (representative frame from the video) into a face recognition network (Parkhi et al., 2015) and extracting the feature from its penultimate layer. We trained the model on around 5000 speech–face embedding pairs from the AVSpeech dataset (Ephrat et al., 2018)

- **ffmpeg** – extract audio and frames separately from the video.
- **librosa and tensorflow libraries** – compute stft and power law compression
- **face recognition and keras vgg-facenet** – find face bounding boxes and compute 4096 dimensional face embedding vector.

We saved the audio spectrogram and the face embeddings as pickle files to speed up the training process.

3.2 Architecture

The speech encoder architecture is a convolutional neural network that turns the spectrogram of a short input speech into a pseudo face feature as shown in figure 4. The blocks of a convolution layer, ReLU, and batch normalization alternate with max-pooling layers, which pool along only the temporal dimension of the spectrograms, while leaving the frequency information carried over. This is intended to preserve more of the vocal characteristics, since they are better contained in the frequency content, whereas linguistic information usually spans longer time duration.

At the end of these blocks, we apply average pooling along the temporal dimension. This allows us to efficiently aggregate information over time and makes the model applicable to input speech of varying duration. The pooled features are then fed into two fully-connected layers to produce a 4096-D face feature.

3.3 Data

We divided the entire dataset that we downloaded into 3 parts : Training Data (that is 80% of the entire data), Validation Data (10% of the entire data), and Test Data (10% of the entire data) as shown in figure 3. We had 6100 of entire data, thus training, test and validation data are as follows :

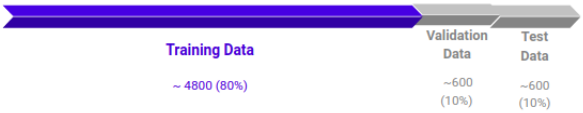

Figure 3: Train/Dev/Test Split

- Training Data - 4880 videos
- Validation Data - 610 videos
- Test Data - 610 videos

3.4 Training

Our voice encoder is trained in a self-supervised manner, using the natural co-occurrence of a speaker's speech and facial images in videos. To this end, we use the AVSpeech dataset, a large-scale "in-the-wild" audiovisual dataset of people speaking. A single frame containing the speaker's

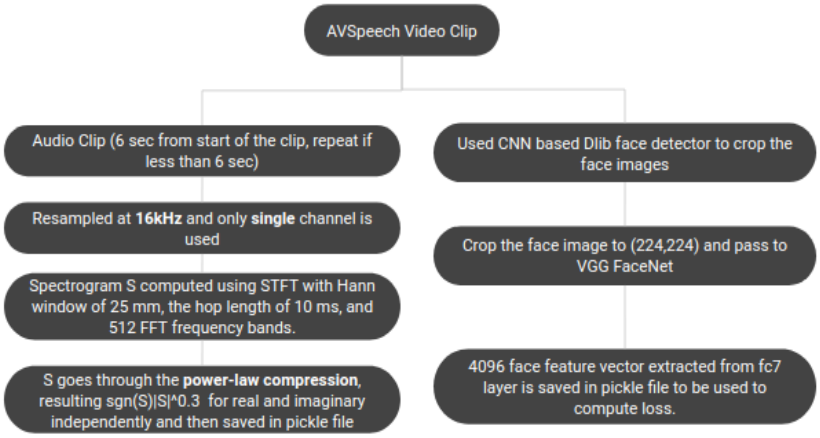

Figure 2: Pre-processing Pipeline. We used up to 6 seconds of audio taken from the beginning of each video clip in AVSpeech. The audio waveform is then resampled at 16 kHz and only a single channel is used. Spectrograms are computed by taking STFT with a Hann window of 25 mm, the hop length of 10 ms, and 512 FFT frequency bands. Each complex spectrogram S subsequently goes through the power-law compression, resulting $sgn(S)|S|^{0.3}$ for real and imaginary independently, We run the CNN-based face detector from Dlib, crop the face regions from the frames, and resize them to 224×224 pixels. The VGG-Face features are computed from the resized face images. The computed spectrogram and VGG-Face feature of each segment are collected in pickle files and then used for training.

Layer	Input	CONV RELU BN	CONV RELU BN	CONV RELU BN	MAXPOOL	CONV RELU BN	MAXPOOL	CONV RELU BN	MAXPOOL	CONV RELU BN	MAXPOOL	CONV RELU BN	CONV RELU BN	CONV	AVGPOOL RELU BN	FC RELU	FC
Channels	2	64	64	128	–	128	–	128	–	256	–	512	512	512	–	4096	4096
Stride	–	1	1	1	2×1	1	2×1	1	2×1	1	2×1	1	2	2	1	1	1
Kernel size	–	4×4	4×4	4×4	2×1	4×4	2×1	4×4	2×1	4×4	2×1	4×4	4×4	4×4	$\infty \times 1$	1×1	1×1

Figure 4: **Speech Encoder Architecture** (Oh et al., 2019) : The input spectrogram dimensions are 598×257 (time $\times$ frequency) with two input channels in the table corresponding to the spectrogram's real and imaginary components

.

face is extracted from each video clip and fed to the VGG-Face model (Parkhi et al., 2015) to extract the 4096-D feature vector, v_f . This serves as the supervision signal for our voice encoder—the feature, v_s, of our voice encoder is trained to predict v_f .

3.5 Loss Function

$$L1 = \|\mathbf{V_f} - \mathbf{V_s}\|_1 \quad (1)$$

$$L2_{norm} = \left\| \frac{\mathbf{V_f}}{\|\mathbf{V_f}\|} - \frac{\mathbf{V_s}}{\|\mathbf{V_s}\|} \right\|_2^2 \quad (2)$$

$$L_{Total} = \left\| \frac{\mathbf{V_f}}{\|\mathbf{V_f}\|} - \frac{\mathbf{V_s}}{\|\mathbf{V_s}\|} \right\|_2^2 + \lambda_2 L_{distill}(f_{VGG}(\mathbf{v_f}), f_{VGG}(\mathbf{v_s})) \quad (3)$$

$$L_{distill}(\mathbf{a}, \mathbf{b}) = -\sum_i p_{(i)}(a) * \log p_{(i)}(b) \quad (4)$$

$$p_{(i)}(a) = \frac{\exp(a_i/T)}{\sum_j (\exp(a_j/T))} \quad (5)$$

A natural choice for the loss function would be the L1 loss. But, the training undergoes slow and unstable progression with this loss. So we have used $L2_{norm}$ loss for training where we first normalize the feature vector and then calculate the L2 loss. Speech2Face paper (Oh et al., 2019) mentions another interesting loss L_{Total} which additionally penalises the difference in activation of the last layer of VGG Facenet (Parkhi et al., 2015). This uses knowledge distillation which encourages the output of a network to approximate the output of another. This requires the fc7 to fc8 layer weights of VGG Facenet during training which could not be sustained due to GPU memory constraint.

Model	R@1	R@5	R@10	R@25	R@50	R@75	R@100
Train Data	45	52	55	58	62	64	66
Test Data	51	61	66	70	75	77	81

Table 1: SpeechToFace → Face retrieval performance. We measure retrieval performance by recall at K (R@K, in %), which indicates the chance of retrieving the true image of a speaker within the top-K results. Train Data contains a database of 4800 images on which the model was trained and Test Data contains around 600 completely new images.

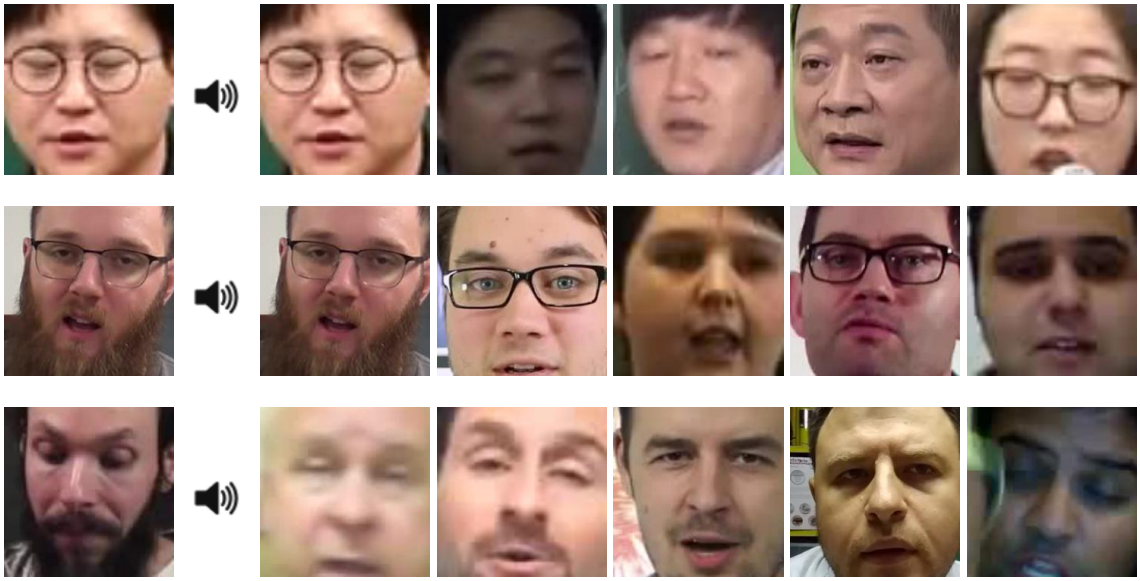

Figure 5: SpeechToFace → Face retrieval examples. We query a database of 600 face images by comparing our SpeechToFace prediction of input audio to all VGG-Face face features in the database. For each query, we show the top-5 retrieved samples. First row (Perfect match i.e, top 1) : Speech suggests that the person is Chinese and all our predicted faces are Chinese, however there is a case of gender mismatch in one of the top 5 results. Second row (Perfect match) - Most of the predicted persons match in ethnicity and gender. Last row is an example where the true face was not among the top results, this may be attributed to **too much beard** (which model didn't learn properly owing to less such data), **poor quality of the cropped images** due to which face features are not proper. However most of the predicted faces have their eyes looking downwards which is strikingly noticeable and may be related to the voice, though it is little debatable.

4 Results

We test our model both qualitatively and quantitatively on the AVSpeech dataset (Ephrat et al., 2018). Our goal is to gain insights and to quantify how closely our SpeechToFace model predicts the facial features compared to the true facial features.

5 Limitations and Challenges

The data preprocessing step for the task is very time consuming for the AVSpeech Dataset (Ephrat et al., 2018) because of the downloading and computing audio spectrograms and the face features. We preprocessed around 6000 videos (compared to 2 million by original paper) and it took around 40-50 hrs. We trained the model on GTX 1080 Ti, it took around 20 min for very epoch and we trained for 10 hrs. We couldn't implement the distillation loss as it requires large amount of GPU memory because the model was huge and on top of that we require fc7 to fc8 layer VGG facenet weights during training. We are very sure that increasing dataset to around 2 million, using multiple GPU's, more training time and fine tuning the hyper parameters can increase the accuracy multi-fold.

6 Future Work

We didn't implement the Face Decoder Model, which takes the face features predicted by SpeechToFace model as input and produces an im-

age of the face in a canonical form (frontal-facing and with neutral expression). The Speech2Face paper (Oh et al., 2019) had used by the pretrained model by (Cole et al., 2017), but the pretrained model was not available open source. We tried to implement the model but it required huge amount of data as the results were not so satisfactory. As the main aim of the project was to implement the Speech Model, we postpone this vision task as a future work.

References

Forrester Cole, David Belanger, Dilip Krishnan, Aaron Sarna, Inbar Mosseri, and William T. Freeman. Synthesizing Normalized Faces from Facial Identity Features. *arXiv e-prints*, art. arXiv:1701.04851, Jan 2017.

A. Ephrat, I. Mosseri, O. Lang, T. Dekel, K Wilson, A. Hassidim, W. T. Freeman, and M. Rubinstein. Looking to listen at the cocktail party: A speaker-independent audio-visual model for speech separation. *arXiv preprint arXiv:1804.03619*, 2018.

Tae-Hyun Oh, Tali Dekel, Changil Kim, Inbar Mosseri, William T. Freeman, Michael Rubinstein, and Wojciech Matusik. Speech2Face: Learning the Face Behind a Voice. *arXiv e-prints*, art. arXiv:1905.09773, May 2019.

Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *BMVC*, 2015.