Face Reconstruction from Speech

Pranjal Saini
203050014@iitb.ac.in
IIT Bombay

Shubham Nemani
203050011@iitb.ac.in
IIT Bombay

Ashwani Kumar Jha
20305R001@iitb.ac.in
IIT Bombay

Abstract

In this project, we perform the task of reconstructing a facial image of a person from a short audio recording of that person speaking. We design and train a deep neural network to perform this task using thousands of natural YouTube videos of people speaking. We aim our model to learn the voice-face correlations that allow it to produce images that capture various physical attributes of the speakers such as age, gender and ethnicity. We also perform a qualitative and quantitative analysis of how well the predicted image resembles the true image. Code is available at [1].

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 to predict a recognizable image of the speaker's face and 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 [1] .We decode the predicted face feature into an image of the person's face using a separately trained reconstruction neural network model.

[1] https://github.com/nshubham655/Speech2Face

2. Speech2Face 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 [2] 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.

We trained the voice encoder module that predicts the face feature in a supervised manner where the ground truth features are generated from corresponding face image of the person using a pre-trained VGG-Face model and the face decoder model proposed by [3] was not available open source, so we implemented our own decoder.

During training of encoder, the face decoder is 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) [4] (used to compute loss function) are precomputed to speed up the training process.

3. Implementation

3.1. Preprocessing

We used the AVSpeech dataset Ephrat et al., 2018 [1] comprising of thousands of video segments from YouTube. Other libraries and tools that we used for pre-processing are described below :

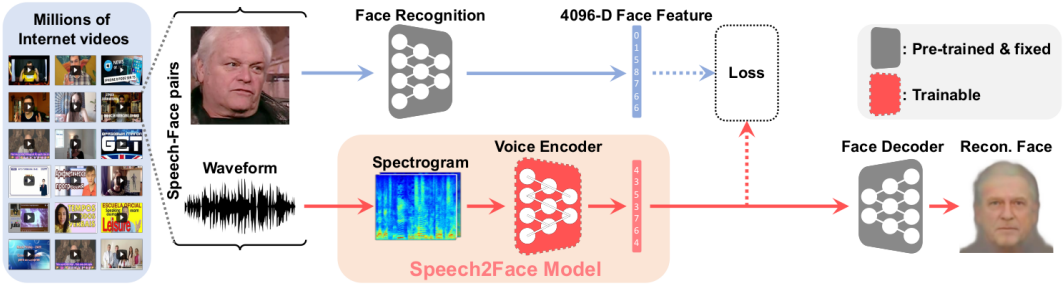

Figure 1. SpeechToFace model and raining pipeling

- **youtube-dl** download the videos from the csv files corresponding to start and end times.
- **moviepy** - extract audio and frames separately from the video.
- **librosa and tensorflow** -compute stft and power law compression
- **dlib and face recognition** - find faces from a particular video frame
- **vgg16-facenet** - extract vgg features i.e. 4096 vector form faces.

The pre-processing steps are described in Figure 2 We saved the audio spectrograms and the vgg face features as pickle files to speed up the training process

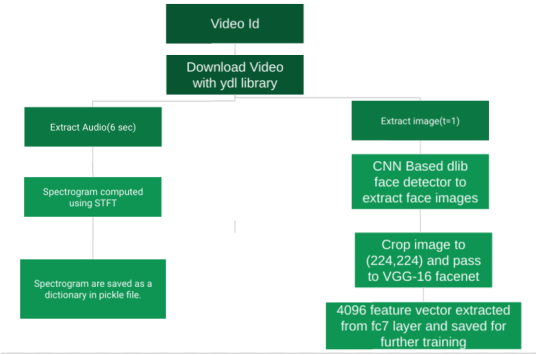

Figure 2. Preprocessing Pipeline

3.2. Architecture

The voice 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 3. The blocks of a convolution layer, ReLU, and batch normalization alternate with maxpooling 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.

The goal of the face decoder is to reconstruct the image of a face from a low-dimensional face feature. Our face decoder consists of 4 transpose convolution layers, thus upsampling a 4096 face feature vector into a 224*224*3 RGB face image.

3.3. Dataset

We have used AVSpeech Dataset [1] for Voice Encoder training, details are as follows :

- Training Data : 8400 videos
- Validation Data : 395 videos
- Test Data : 402 videos

We have used UTKFace Dataset [5] for Face Decoder training, details are as follows :

- Training Data : 5900 images
- Validation Data : 300 images
- Test Data : 280 images

3.4. Training

Our voice encoder is trained using the natural co-occurrence of a speaker's speech and facial images

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	$\infty\times1$	1×1	1×1

Total Trainable Parameters = 46593664

Figure 3. Voice Encoder Architecture

in videos. To this end, we use the AVSpeech dataset, a largescale "in-the-wild" audiovisual dataset of people speaking. A single frame containing the speaker's face is extracted from each video clip and fed to the VGG-Face model Parkhi et al., 2015 [4] 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 .

Our Decoder is trained on UTKFace dataset. It contains face images from all age groups. Face features are extracted using VGG-16 facenet and fed into the Decoder to reproduce original face image of size 224*224*3. It is then fine-tune on small set of samples from AVSpeech Dataset.

3.5. Loss Function

We have used various loss functions to train Voice Encoder. These are as follows :

L1 Loss :

$$L_1 = \left\| \frac{\mathbf{v}_f}{\|\mathbf{v}_f\|} - \frac{\mathbf{v}_s}{\|\mathbf{v}_z\|} \right\|$$

L2 Loss :

$$L_2 = \left\| \frac{\mathbf{v}_f}{\|\mathbf{v}_f\|} - \frac{\mathbf{v}_J}{\|\mathbf{v}_s\|} \right\|_2^2$$

Combination of L1 and L2 loss :

$$L_{combined} = \left\| \frac{\mathbf{v}_f}{\|\mathbf{v}_f\|} - \frac{\mathbf{v}_s}{\|\mathbf{v}_z\|} \right\| + \lambda \left\| \frac{\mathbf{v}_f}{\|\mathbf{v}_f\|} - \frac{\mathbf{v}_J}{\|\mathbf{v}_s\|} \right\|_2^2$$

Contrastive Loss:

$$L_{contrastive} - \frac{1}{N} \sum_{i=1}^{N} \log \left(\frac{\exp\left(q_i^\top \cdot v_i\right)}{\sum_{j \neq i} \exp\left(q_i^\top \cdot v_j\right)} \right)$$

We got best results on L_1 loss between normalized features predicted by Voice Encoder and Vgg-16 facenet.

For Decoder, we have used L_1 loss between original image and predicted image for training.

4. Results

We test our model both qualitatively and quantitatively on the AVSpeech dataset [1].

For evaluation of our Voice Encoder, we measured the average of l_1 distance, l_2 distance and cosine similarity between VGG16 facenet features and feature embedding predicted by Voice Encoder over the test dataset of around 400 samples. The results are shown in Table 1.

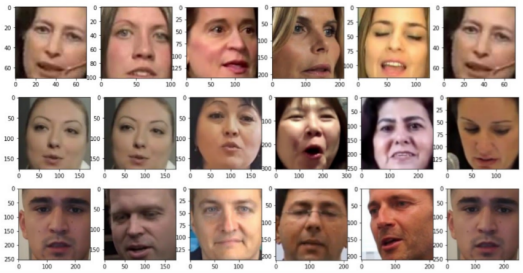

Figure 4. Face Retrival Examples

.

We further evaluated how accurately we can retrieve the true speaker from a database of face images (test dataset is used for it). To do so, we take the speech of a person to predict the feature using our Voice Encoder, and query it to the database by computing its distances to the VGG face features of all face images in the database. We report the retrieval performance by measuring the **recall @ K**, i.e., the percentage of time the true face is retrieved within the rank of K. The results are shown in Table 2. The top 5 retrievals of some images are show in Figure 4

5. Analysis

- **Face Retrival Performance** We query a database of 400 face images by comparing our Voice Enocder prediction of input audio to all VGG-16 face features in the database. For each query, we show the top-5 retrieved samples. It can be seen from the retrieved images in Figure 4 that model

has learned various physical attributes such as age and gender since in all the rows, the retrieved images have same gender and similar age as the query image. In row 2: the first 3 matching images have same expression as the query image.

Figure 5. Speech2Face Model Output

- **S2F Output** From figure 5, the first column shows original image of speaker, second column image is generated by actual VGG16 face features fed to decoder and third column image is output of our Speech2Face model on input audio waveform. It can be observed that images generated by decoder on audio features are quite similar to that generated using VGG features except the landmarks, but very significant difference can be seen between images generated using VGG features and original image of speaker.

6. Limitations

The data preprocessing step for the task is very time consuming for the AVSpeech Dataset Ephrat et al., 2018 [1] because of the downloading and computing audio spectrograms and the face features.

We preprocessed around 9000 videos (compared to 1.7 million by original paper) and it took around 3-4 hrs per 1K videos. Also the original Voice Encoder proposed by Oh et al. [2] contains around 760 million parameters, with which it was not possible to train it on google colab so we decreased the voice encoder complexity to around 46 million parameters. We trained the model on google colab, it took around 20 min per epoch and we trained for best epoch.

7. Conclusion and Future Work

From the outputs of our S2F model 5 and features similarity on Voice Encoder output, we can infer that feature extraction from speech is not done properly, this is because of very less training data i.e. 8400 as compared to 1.7 million. With little more training, the decoder can work more effectively.

Evaluation / Loss Metrics	L1 Distance	L2 Distance	Cosine Similarity
L1 Loss	20.23	0.77	0.68
L2 Loss	34.12	1.42	0.62
L1 loss + lambda * L2 loss	28.45	1.32	0.63
Contrastive Loss	43.67	2.35	0.49

Table 1: Feature Similarity

	Train set(8K)					Test set(400)				
Similarity Measure	R@1	R@5	R@10	R@20	R@50	R@1	R@5	R@10	R@20	R@50
L1 Distance	11.96	29.26	41.73	55.47	72.26	0.51	1.77	3.03	6.83	14.18
L2 Distance	11.70	29.52	41.48	55.72	73.03	0.76	2.03	3.15	6.33	14.68
Cosine similarity	11.20	27.23	39.44	54.45	73.03	0.25	1.01	2.53	4.56	14.94

Table 2: Face Retrieval Performance

References

[1] 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," *ACM Transactions on Graphics*, vol. 37, p. 1–11, Aug 2018.

[2] T.-H. Oh, T. Dekel, C. Kim, I. Mosseri, W. T. Freeman, M. Rubinstein, and W. Matusik, "Speech2face: Learning the face behind a voice," pp. 7539–7548, 2019.

[3] F. Cole, D. Belanger, D. Krishnan, A. Sarna, I. Mosseri, and W. T. Freeman, "Synthesizing normalized faces from facial identity features," 2017.

[4] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," pp. 41.1–41.12, September 2015.

[5] "Utkface dataset,"