TTS conditioned on Speaker and Emotion

Gurparkash Singh 1600500112
Drumil Trivedi 170020016
Maitrey Gramopadhye 160050049

November 2019

Abstract

We present a model that could generate speech corresponding to an input text that is conditioned on speaker as well as emotion. The main contribution of our work is to introduce a model that could generate speech for any given speaker, which is an improvement over the existing methods which are limited to the speakers they're trained on. Our model consists of three main components: a speaker-embedding network, a text-to-speech module, and an emotion-conditioning network. All of these segments can be trained independently which obviates the need for massive parallel datasets. All the parts of our model are detailed below.

1 Introduction & Existing Work

Our initial goal was to develop a TTS model that could be conditioned on an arbitrary number of features, such as speaker, emotion, stress levels, etc. To develop this model, we decided to work on an instantiation of this idea by choosing two of these features, speaker and emotion. Theoretically speaking, the Wavenet model should be able to generate speech on an arbitrary number of features, however, that would require large parallel datasets, which have all these labels. Such datasets are not easily available. Therefore, we break our task into two parts: first to generate audio corresponding to a speaker, and then to impart emotion to that speech. For the former part, we can use the Wavenet model, while for the latter, we use a model based on GANs. Since we have divided our model into two separate tasks, it is possible to train them independently.

In the existing version of Wavenet, the speech is generated corresponding only to the speakers it was trained on. The model takes a one-hot input and generates speech corresponding to the mentioned speaker. However, our aim was to develop a model that could generate voice for arbitrary speakers. To satisfy this condition, we decided to condition our Wavenet module on the speaker embeddings which capture the characteristics of the voice of the speakers instead of the content of the speech utterances. For this part, we can use features from either a Speaker Recognition model or from a Speaker Verification model. More details have been delineated below. Other than the above approach, traditional approaches such as i-vectors can also be used.

2 Speaker Embedding

The speaker encoder is used to generate speaker embedding, to condition the wavenet on a reference speech signal from the desired target speaker. We use a speaker verification architecture and a speaker classification architecture to derive the speaker embedding.

In the original wavenet paper, the speaker conditioning vector used is a one-hot encoding vector specifying the speaker. However, we implement a generalised version of conditioning wavenet, where the speaker embeddings are dense vectors. This way our method would not be limited to a predefined set of speakers.

We use an LSTM encoder to generate the embeddings. To train this, we implement a discriminator network that takes as input two embeddings and outputs whether they belong to the same speaker or not. By this method, we are able to learn, from raw audio, just the speaker specific embedding that doesn't depend on the content of the audio. Also, since we choose an encoder-discriminator mechanism to train instead of a convectional encoder-decoder we get a clustering on the speakers, where ideally each speaker belongs to a different cluster. Thus, we get an encoding where signals from different speakers are encoded as different from each other as possible.

We first extract the MFCC features from two audio instances and feed them to the encoder. Then we concatenate their outputs and feed the joint output to the discriminator. We used a cross-entropy loss on the output of the discriminator, with the target being 0 (if the speakers are different) and 1 (if they are same) [Figure 1]. We use a subset of the LibriSpeech Dataset to train the network.

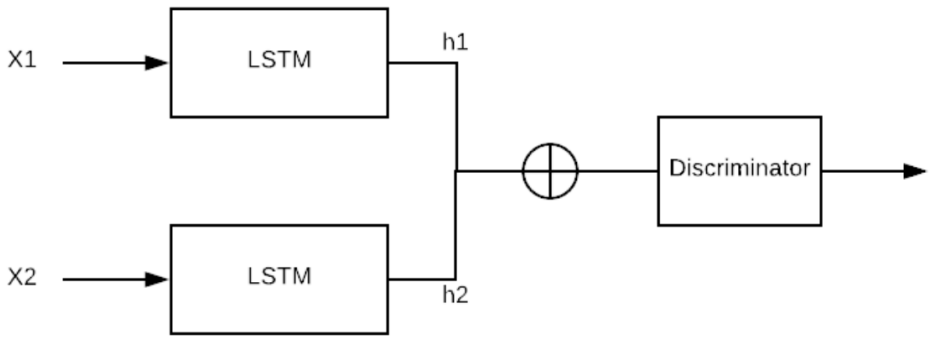

Figure 1

Another architecture we use for training the encoder is speaker classification. Where instead of a discriminator we use a classifier. In this setup we take a single audio instance, calculate its MFCC features and pass it through the encoder. The encoded vector is then passed through a classifier which outputs the speaker identity. The loss used in this case is also cross-entropy with speaker ids as targets.

The classifier gave us a better result while training as in this case there is no imbalance between the classes (the speaker ids). Whereas while using the discriminator for training, there was a heavy class imbalance as one class (same speakers) had $O(N)$ entries where as the other (different speakers) had $O(N * N)$ entries, N being the number of speakers. However, using the classifier limits the number of speakers to the training set and it doesn't encourage the embeddings of different speakers to be different from each other.

3 Wavenet

Wavenet is a Neural Network Architecture which is used to generate raw audio waveforms. The model is fully probabilistic and auto-regressive, with the predictive distribution for each audio sample conditioned on all previous ones. Wavenet can be used to learn unconditioned density estimation $p(x)$ as well as conditioned density estimation $p(x|h)$, where x is an audio waveform and h is a feature vector which can be used to encode various attributes like content of the speech, speaker of the speech, emotion of the speech, etc.

The unconditioned Wavenet is a stack of 1D Convolutions over the input speech waveform. However, the outputs of the layers are shifted in order to make the output of a layer dependent only on the inputs of the previous timesteps. More specifically, output of the network at time t, O_t should only depend on the previous timesteps of the input, $I_1, I_2, ..., I_{t-1}$. To make this happen, we shift the output of the layer by the kernel size. This is more clear from the figure below.

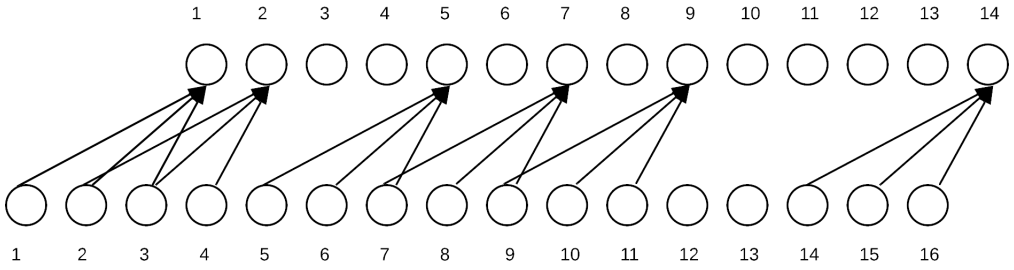

Figure 2

The causal layer of the Wavenet breaks the dependencies between input and output. It is clear from the figure that the output at time t depends only on the inputs before time t. After the first layer introduces the time independence, we can use a stack of dilated Convolution layers. The output at time t of this stack of layers depends on the input till time t, i.e., O_t depends on $I_1, I_2, ..., I_t$. We combine this stack of dilated Convolutional layers with the initial causal layer. The following figure shows this stack of layers.

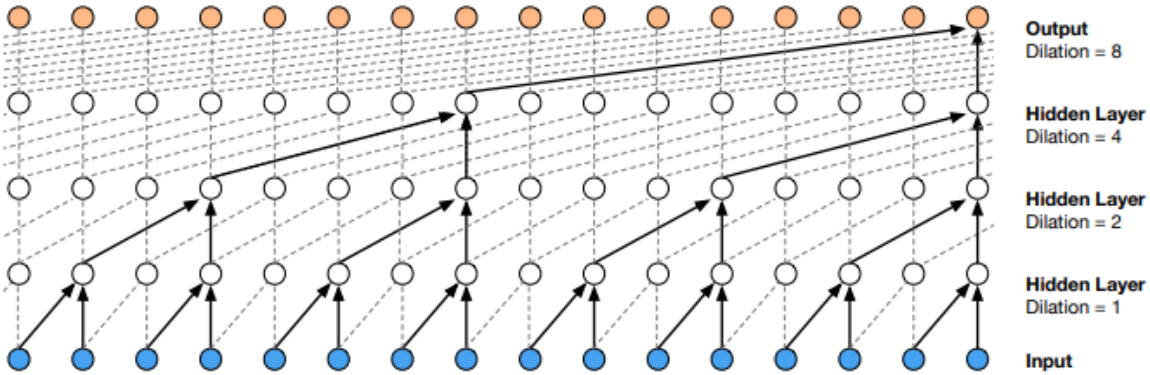

Figure 3

In general, this stack of layers can get very deep, which results in the problem of vanishing gradients. For a better gradient flow, we use residual layers and skip connections. Residual Layers and skip connections create a *gradient highway* that solves the problem of vanishing gradients.

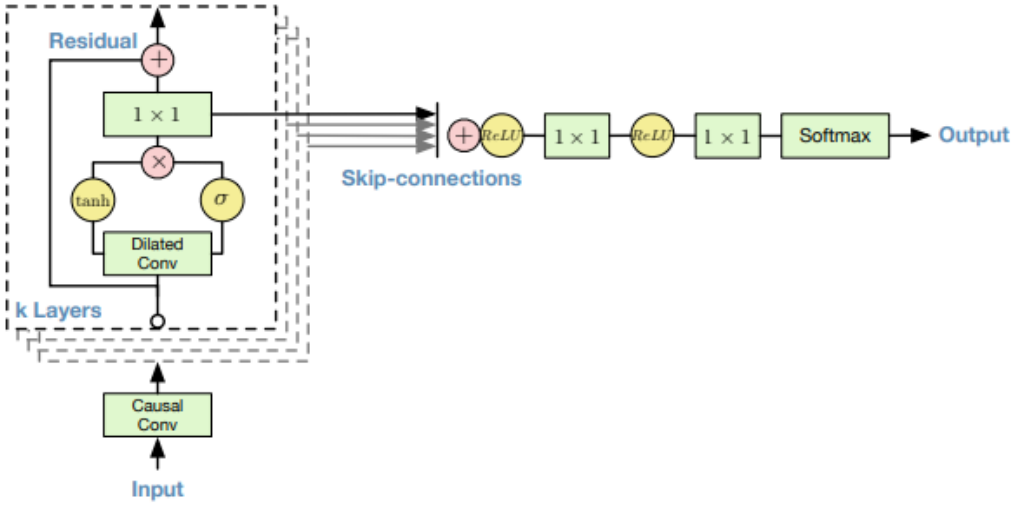

Figure 4

In the unconditioned version, the output of the Gated Activation unit only depends on the speech input at that time, whereas for the conditioned Wavenet, the residual connections also account for the feature vector h. Following are the unconditioned and conditioned equations that govern the output of each layer.

$$\mathbf{z} = \tanh\left(W_{f,k} * \mathbf{x}\right) \odot \sigma\left(W_{g,k} * \mathbf{x}\right),$$

$$\mathbf{z} = \tanh\left(W_{f,k} * \mathbf{x} + V_{f,k}^{T}\mathbf{h}\right) \odot \sigma\left(W_{g,k} * \mathbf{x} + V_{g,k}^{T}\mathbf{h}\right).$$

For the conditioned Wavenet, there are two types of dependencies, global and local. Global dependencies are the ones that stay same throughout the speech, for example speaker identity or emotion. Local dependencies vary with time. This can include text, the amount of stress on the words, etc. For local conditioning, the length of the conditioning

vector should be the same as that of the output speech waveform. In our case, we have to up sample the features from the input text such that they're equal to length of the generated speech. The most common approach to this is to use an attention based module to generate local features at each time step from the input text.

4 Emotion Conditioning using Auto-Encoder and GAN

4.1 Assumptions

The key assumptions involved are:

1. The data received from the previous Wavenet model is speech that we will process on directly.
2. The speech involved can be mapped to a latent space that has 2 sections c and s.
3. The section c represents the space mapping the emotion independent part of speech.
4. The section s represents the space mapping the emotion dependent part of speech.

4.2 Model

Fig.5 shows the generative model of speech with a partially shared latent space. For any emotional speech x_i, we have a deterministic decoder $x_i = G_i(c_i, s_i)$ and its inverse encoders $c_i = E_i^c(x_i)$, $s_i = E_i^s(x_i)$. To convert emotion, we just extract and recombine the content code of the source speech with the style code of the target emotion. Where each s_i is trained over all samples of the emotion available to make it more robust.

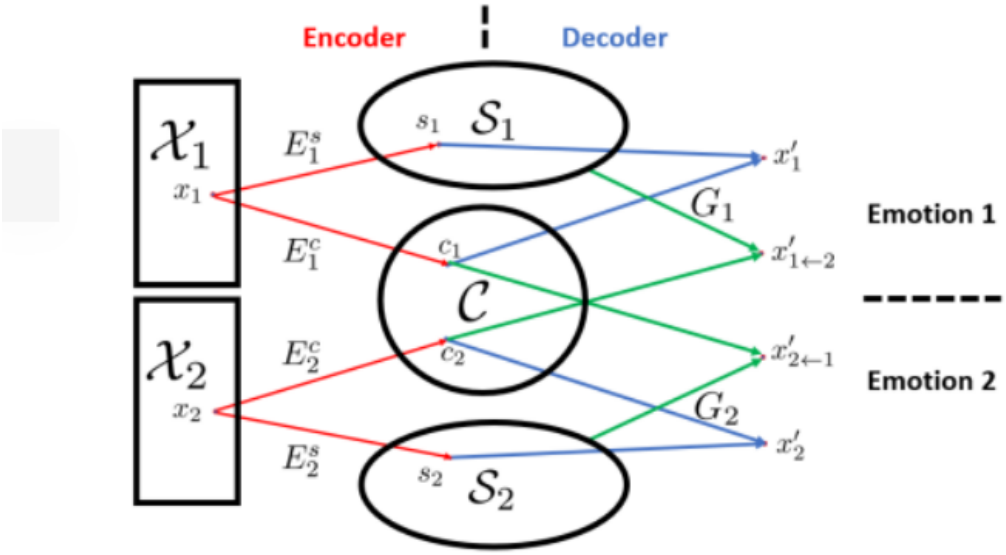

Figure 5: *Speech auto encoder model with partially shared latent space.Speech with emotion i is decomposed into an emotion-specific space S_i and a shared content space C. Corresponding speech $(x1, x2)$ are encoded to the same content code*

Fig. 6 shows an overview of our nonparallel emotional speech conversion system. The features are extracted and recombined by WORLD and converted separately. A speech analysis-synthesis tool STRAIGHT was used to extract fundamental frequency (F0) and power envelope from raw audio. We modify F0 by linear transform to match statistics of the fundamental frequencies in the target emotion domain. The conversion is performed by log Gaussian normalization

$$f2 = exp((logf_1 - \mu_1)\frac{\sigma_2}{\sigma_1} + \mu_2)$$

where μ_i, σ_i are the mean and variance obtained from the source and target emotion set. Aperiodicity(AP) is mapped directly since it is independent of emotion.

For spectral sequence, we use low-dimensional representation in mel-cepstrum domain to reduce complexity, the 50 MFCC are enough to synthesize full bandspeech without quality degeneration. Spectra conversion is learnt by the autoencoder model in Fig. 5. The encoders and decoders are implemented with gated CNN.In addition a GAN module is used to generate realistic spectral frames. This component has 4 subnetworks E_c, E_s, G, D, in which D is the discriminator in GAN to distinguish between real samples and machine-generated samples.

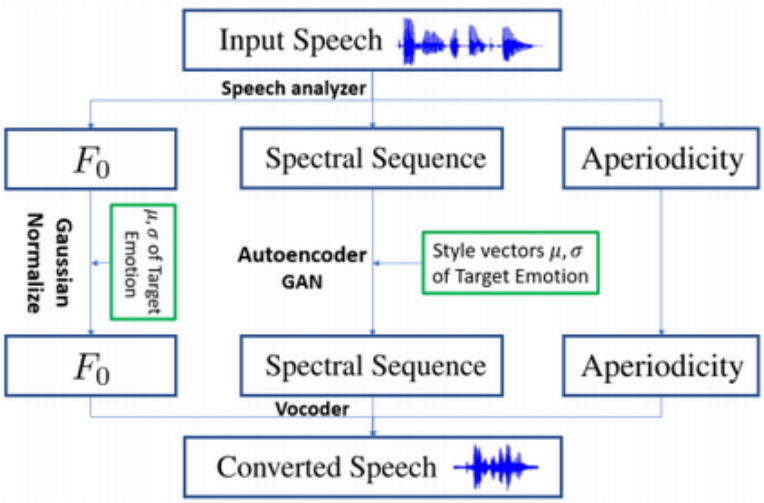

Figure 6: *Overview of nonparallel emotion conversion system*

4.3 Loss functions

We jointly train the encoders, decoders and GAN's discriminators with multiple losses displayed in Fig.6 .To keep encoder and decoder as inverse operations, we apply reconstruction loss in the direction $x_i \rightarrow (c_i, s_i) \rightarrow x_i^{'}$. The spectral sequence should not change after encoding and decoding.

$$L_{recons}^{x_i} = \mathbb{E}_{x_i}(||x_i - x_i^{'}||_1), \;\; x_i^{'} = G_i(E_i^c(x_i), E_i^s(x_i))$$

Due to partially shared latent space, we have to use semi-cycle loss as:

$$L_{cycle}^{c1} = \mathbb{E}_{c_1,s_2}(||c_1 - c_{2\leftarrow 1}^{'}||_1), \;\; c_{2\leftarrow 1}^{'} = E_2^c(x_{2\leftarrow 1}^{'})$$

$$L_{cycle}^{s2} = \mathbb{E}_{c_1,s_2}(||s_2 - s_{2\leftarrow 1}^{'}||_1), \;\; s_{2\leftarrow 1}^{'} = E_2^s(x_{2\leftarrow 1}^{'})$$

Finally we use the GAN to make the speech indistinguishable from normal speech hence the discriminator loss.

$$L_{GAN}^{i} = \mathbb{E}_{c_j,s_i}[log(1 - D_i(x_{i\leftarrow j}^{'}))] \;\; + \;\; \mathbb{E}_{x_i}[logD_i(x_i)]$$

Finally the use to minimise the weighted sum of all the loss functions stated above

$$\min_{E_1^c,E_1^s,E_2^c,E_2^s,G_1,G_2} \max_{D_1,D_2} L(E_1^c, E_1^s, E_2^c, E_2^s, G_1, G_2, D_1, D_2) \;\; = \;\; \lambda_s(L_{cycle}^{s1} + L_{cycle}^{s2}) +$$

$$\lambda_c(L_{cycle}^{c1} + L_{cycle}^{c2}) \;\; + \;\; \lambda_x(L_{recons}^{1} + L_{recons}^{2}) \;\; + \;\; \lambda_g(L_{GAN}^{1} + L_{GAN}^{2})$$

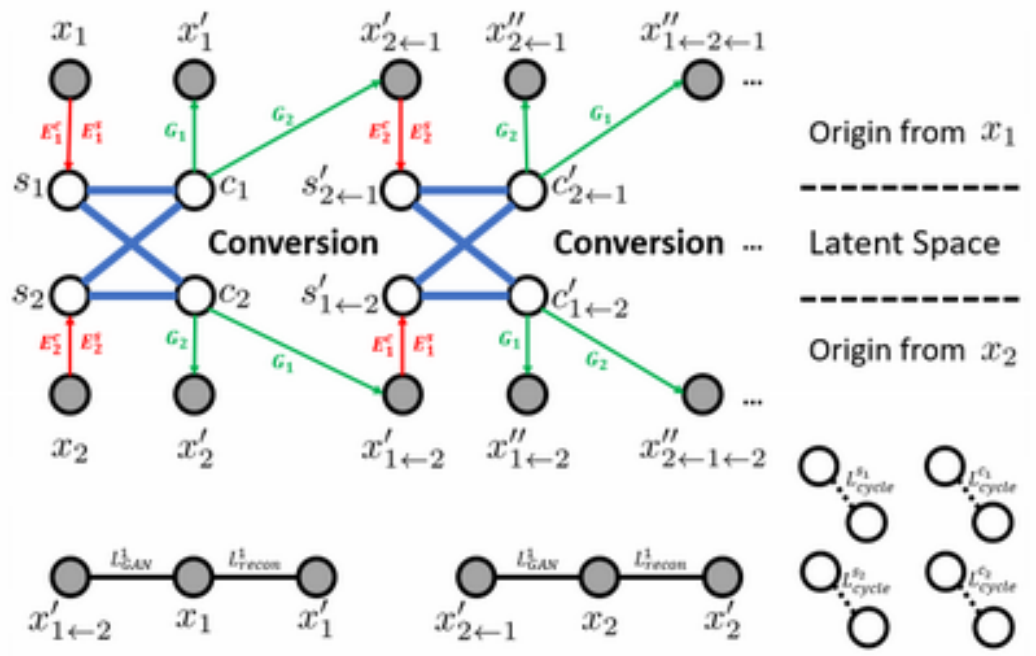

Figure 7: *Train on multiple loss functions*

5 Results

Since the Wavenet and the GAN modules take a long time to train, we were not able to train them on large datasets and stitch them together. However, we were able to train some parts of our model on smaller datasets to show that they will be able to scale to larger datasets provided the time and sources to train.

We used a Speaker Classification model to generate speaker-embeddings. We trained the network on a dataset containing audio clips corresponding to 6 different speakers. To visualize our embeddings, we projected the 128 dimensional features to a plane using Principal Component Analysis (PCA). Following are the results.

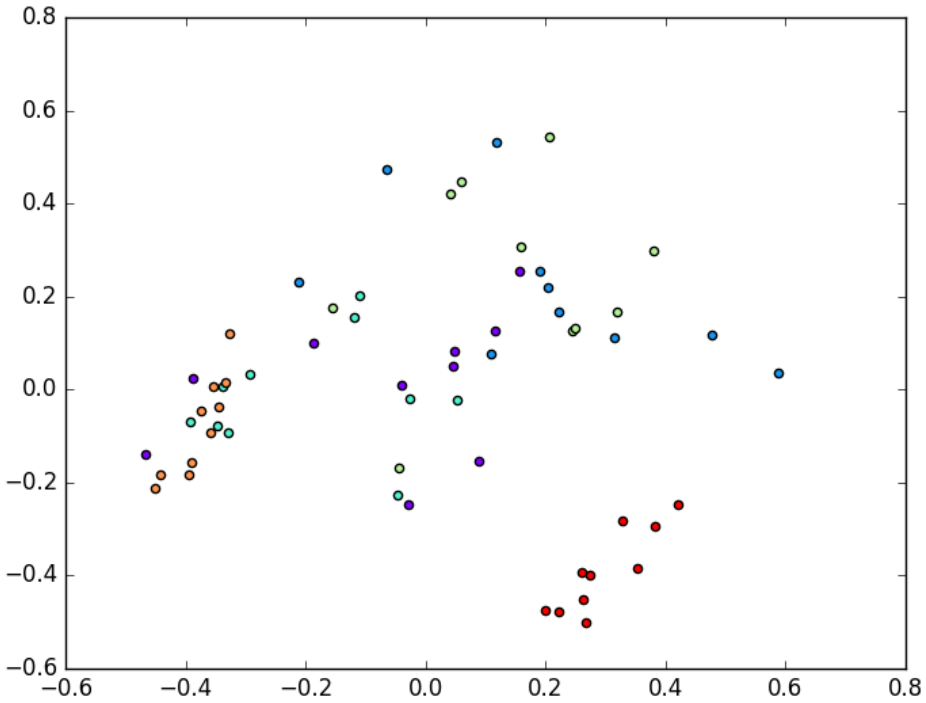

Figure 8: *2D Visualization of the features (from 128D embeddings)*

As we can see from the above figure, different speakers appear to form clusters. This behavior is most clearly visible for the red and orange data-points, the data-points for which are closest to each other. Even though we are drastically decreasing the dimensionality, the fact that the data-points for different speakers lie in clusters and away from the other speakers is clear to see. This proves our hypothesis that a speaker classification network can be used to generate feature embeddings.

Even though we could not generate human-like speech outputs, visualizing the generated waveforms does reveal an interesting pattern. For reference, following are the two waveforms. First corresponds to one of the audio samples from the dataset and the second is the waveform generated by the Wavenet.

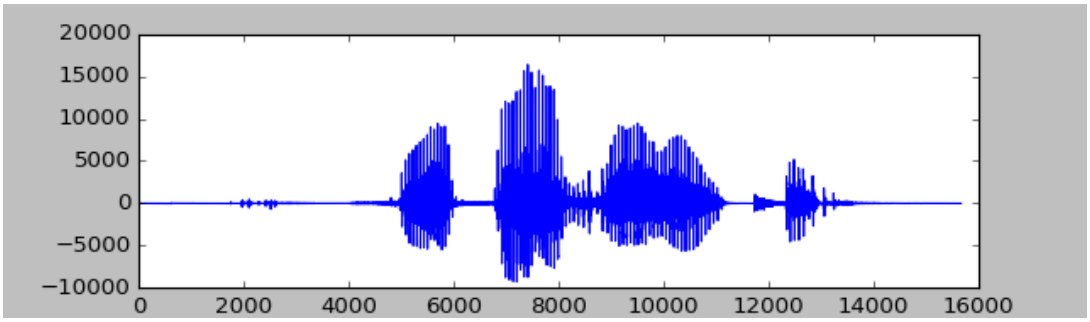

Figure 9: *Waveform from the input dataset*

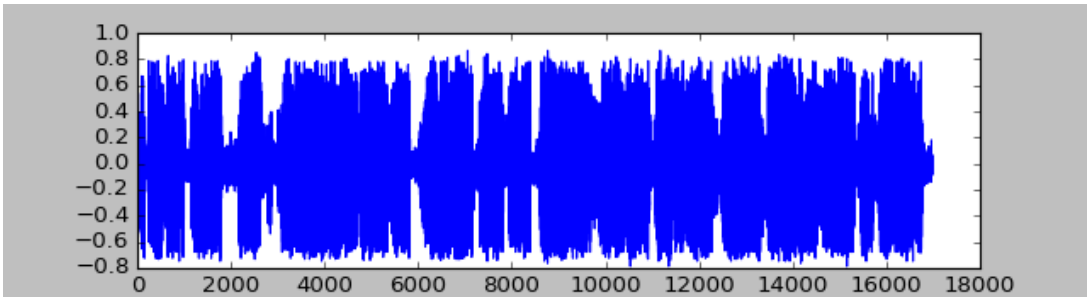

Figure 10: *Generated Waveform*

As we can see from Figure 10, the output generated by the Wavenet is not completely random. There are periods of crest and trough, which emulate the human speech to some extent. This shows that our model is capable of learning from the input sequences. We believe that training on a bigger dataset will solve this issue.

6 Challenges Faced

1. In speaker embedding, we first tried to train the encoder using a discriminator module. So that we could encourage the embeddings to be different from each other. However due to imbalance in classes we didn't get good results. So, we had to switch to using a classifier for training the encoder.

2. For the Auto encoder and GAN ,we required massive training time and lack of resources for the same. Also there were no pre-trained model to be directly used.

7 Contribution

- Gurparkash - Coded Wavenet, Speaker Verification and Classification Networks
- Drumil Trivedi - Programmed The Non-Parallel Emotion Speech Conversion
- Maitrey Gramopadhye - Trained the Speaker Verification Network and figured out the datasets

References

[1] Tacotron 2: Generating Human-like Speech from Text
https://ai.googleblog.com/2017/12/tacotron-2-generating-human-like-speech.html

[2] Natural TTS Synthesis by conditioning wavenet on mel spectrogram predictions *Jonathan Shen1, Ruoming Pang1, Ron J. Weiss1, Mike Schuster1, Navdeep Jaitly1, Zongheng Yang2,Zhifeng Chen1, Yu Zhang1, Yuxuan Wang1, RJ Skerry-Ryan1, Rif A. Saurous1, Yannis Agiomyrgiannakis1,and Yonghui Wu1*

[3] Tacotron: Towards End-to-End Speech Synthesis *Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, Rif A. Saurous*

[4] WaveNet: A generative model for raw audio
https://deepmind.com/blog/article/wavenet-generative-model-raw-audio

[5] Wavenet: A generative model for raw audio *Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu*

[6] Conditional Image Generation withPixelCNN Decoders *Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, Koray Kavukcuoglu*

[7] Pixel Recurrent Neural Networks *Aaron van den Oord, Nal Kalchbrenner, Koray Kavukcuoglu*

[8] Deep Voice: Real-time Neural Text-to-Speech *Sercan O. Arik, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Andrew Ng, onathan Raiman, Shubho Sengupta, Mohammad Shoeybi*

[9] Transfer Learning from Speaker Verification toMultispeaker Text-To-Speech Synthesis *Ye Jia, Yu Zhang, Ron J. Weiss, Quan Wang, Jonathan Shen, Fei Ren, Zhifeng Chen, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, Yonghui Wu*

[10] End-to-End Text-Dependent Speaker Verification *Georg Heigold, Ignacio Moreno, Samy Bengio, Noam Shazeer*

[11] Generalized End-to-End Loss for speaker verification *Li Wan, Quan Wang, Alan Papir, Ignacio Lopez Moreno*

[12] Nonparallel Emotional Speech Conversion *Jian Gao1, Deep Chakraborty2, Hamidou Tembine1, Olaitan Olaleye3*

[13] LibriSpeech Dataset *http://www.openslr.org/12*

[14] IEMOCAP Dataset *https://sail.usc.edu/iemocap/*