Emotional Text-To-Speech Synthesis

Project Work

EE599: Deep Learning Lab for Speech Processing

University of Southern California

Aditya Mukewar, Siddharth Bhonge, Ishaan Vasant

1. Introduction

A traditional text-to-speech synthesis model consists of the following parts: -

1. Front-end text analysis
2. Acoustic Model
3. Audio Synthesis module

An acoustic model is used in automatic speech recognition to represent the relationship between an audio signal and the phonemes or other linguistic units that make up speech. The model is learned from a set of audio recordings and their corresponding transcripts. [1]

The above model results in a brittle model with various limitations to scope. This gave rise to the need for an end-to-end model that can generalize all these tasks. To address this issue, Google published a model called the Tacotron. Tacotron establishes a function to map input text to mel-scale cepstrum. Given (text, audio) pairs, the model can be trained completely from scratch with random initialization. Tacotron achieved a Mean Opinion Score (MOS) of 3.82 on US English, outperforming a production parametric system in terms of naturalness. What's more, since Tacotron creates speech at the frame level, it is significantly quicker than sample-level autoregressive techniques. [2]

2. Network Architecture

Tacotron is one of the first end-to-end text to speech (TTS) deep learning models. It consists of a complex encoder-decoder model that uses an attention mechanism to align the text with the corresponding audio. The decoder produces a spectrogram of the audio which is converted into a waveform by a technique called the Griffin-Lim Algorithm.

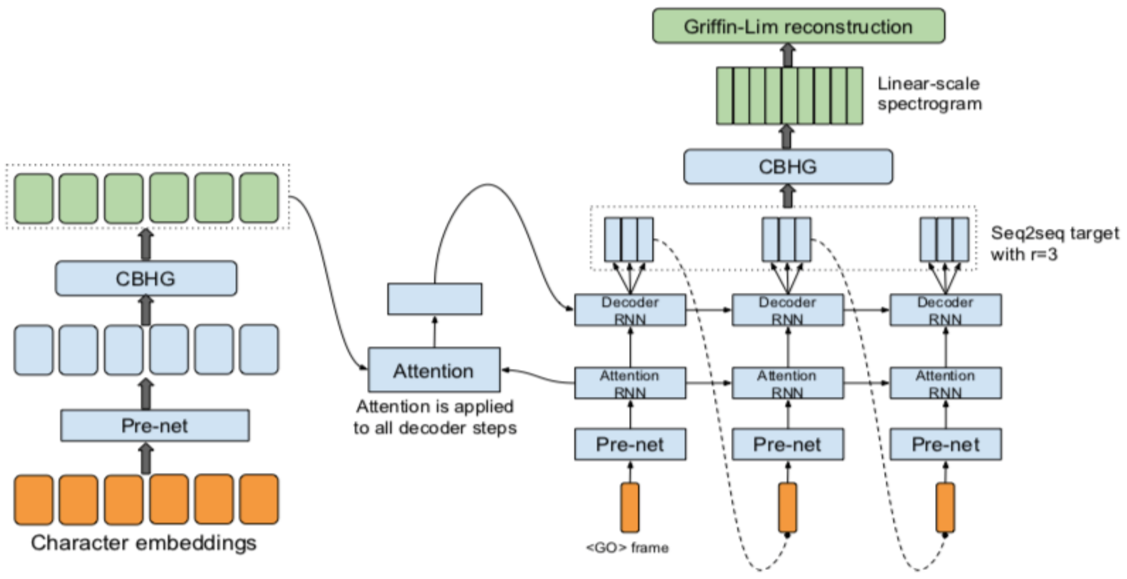

Figure 1: Model Architecture

2.1 Encoder

The encoder in the above architecture takes a sequence of characters as input, where each character is represented as a one-hot vector. An embedder is then used to project the input into continuous space. Due to their high dimensionality and sparsity, one-hot encoded vectors can lead to computational inefficiency if not used with techniques that exploit these characteristics. The embedder allows for significantly reducing the representation space. The embedder also allows us to learn about the different characters of our vocabulary. The embedding layer is then followed by a pre-net, which is a set of non-linear transformations. [2]

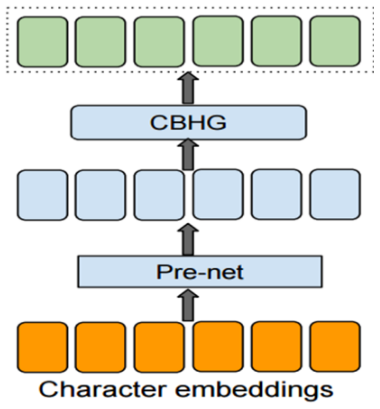

Figure 2: Encoder

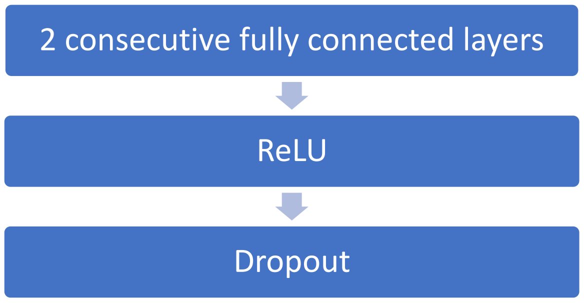

Figure 3: Pre-Net

In Figure 2 above, the number of neurons in the first fully connected layer is greater than 2 times the number of neurons in the second fully connected layer. The second fully connected layer is a bottleneck that helps in convergence and improves generalization.

2.2 CBHG Module

CB – 1D Convolutional Bank
H – Highway network
G – Bidirectional GRU

K sets of 1D convolutional filters are used to form the convolutional bank. The kth set contains C_k filters of width k (i.e. k = 1, 2, . . ., K). With this structure, we should be able to model the unigrams, bigrams and so on.

Max-pooling is used right after the convolutional bank. It makes learned features locally invariant. It is used with a stride of 1 to maintain the time resolution.

Batch normalization is utilized by all convolutional layers in CBHG. IT improves the performance and stability of neural networks. In this process, the input of the layer is modified so that the output given by the activation has mean zero and standard deviation one. It is known to help network converge faster, to allow for higher learning rates in deep networks alleviating the sensitivity of the network to weight initialization and add extra regularization by providing some noise. [3]

Following the max-pooling layer, two supplementary 1D convolutional layers are used (with ReLU). A residual connection binds the initial input with the output of the second convolutional layer. Residual connections allow for the better propagation of gradients. Instead of hoping each few stacked layers directly fit a desired underlying mapping, these layers are explicitly allowed to fit a residual mapping. Deep residual nets are easy to optimize, but the counterpart plain nets (that simply stack layers) exhibit higher training error when the depth increases. Deep residual nets can easily enjoy accuracy gains from greatly increased depth. [4] The next step involves a novel architecture that enables the optimization of networks with virtually arbitrary depth. This is accomplished through the use of a learned gating mechanism for regulating information flow which is inspired by Long Short Term Memory recurrent neural networks. Due to this gating mechanism, a neural network can have paths along which information can flow across several layers without attenuation. Such paths are called information highways, and such networks highway networks [5]. The highway network has a similar role to that of the residual connections.

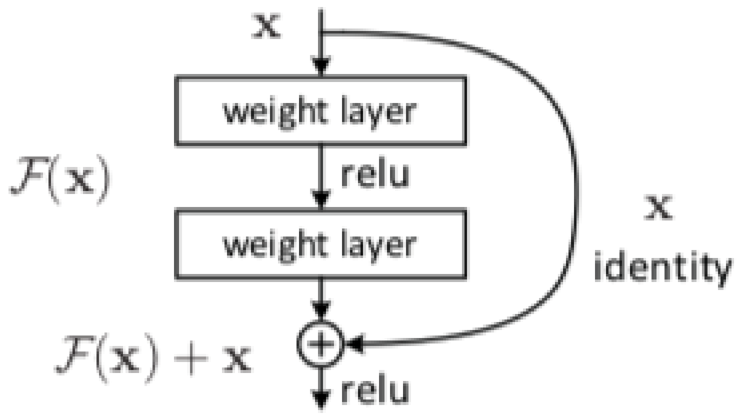
x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 4: Residual learning

To round off the CBHG and encoder, a bidirectional GRU is used to learn the long-term dependencies in the sequence from both the forward and backward context. The encoder output is used by the attention layer.

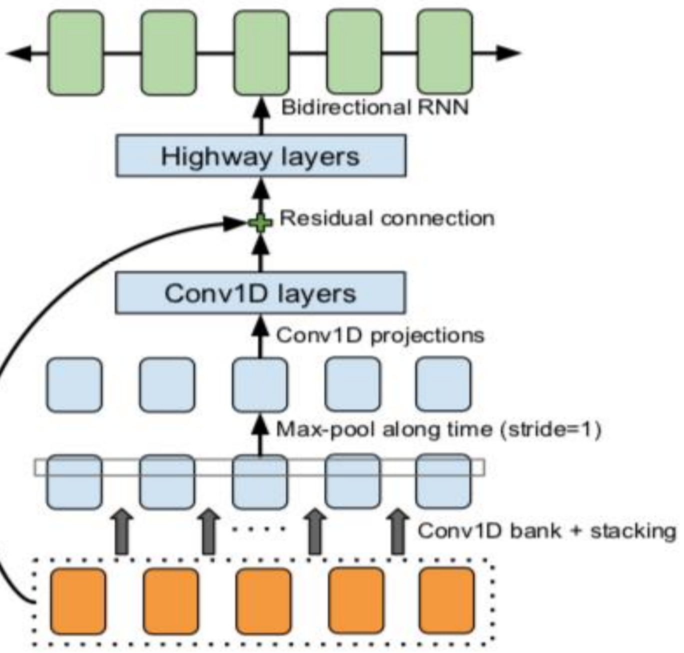

Figure 5: The CBHG (1-D convolution bank + highway network + bidirectional GRU) module

2.3 Decoder

The decoder uses an attention mechanism on the encoder's output to produce mel spectrogram frames. For each mini batch, the decoder is a given a <GO> frame which contains all zeros as input. Then, for every time step, the previously predicted mel spectrogram frame is used as input. The input fuels the pre-net that has the exact same architecture as the encoder's pre-net. The pre-net is followed by a 1-layer GRU whose output is concatenated with the encoder's output to produce the context vector through the attention mechanism. The GRU output is then concatenated with the context vector to produce the input of the decoder RNN block. The decoder RNN is a 2-layer residual GRU that uses vertical residual connections. [6]

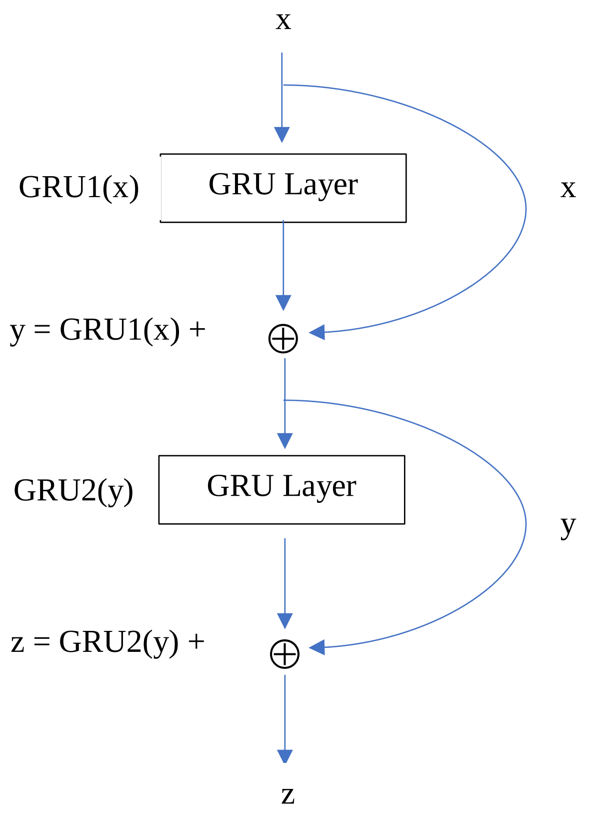

Figure 6: Decoder RNN

The decoder RNN module produces "r"-mel spectrogram frames and only the last one is used by the pre-net during the next time step. The choice of generating r frames, instead of one at each time step is motivated by the fact that one character in the encoder inut usually corresponds to multiple frames. Therefore, by outputting one frame, we force the model to attend to the same input element for multiple time steps – thus slowing down attention during training. Values of r=2 and r=3 are mentioned in the research paper. Increasing r also reduced the model size and decreases the inference time.

2.4 The Attention Mechanism

The Attention Mechanism used in this model is the Bahdanau Attention Mechanism. To explain this mechanism, consider α_{ij} to be a probability that the target word y_i is aligned to, or translated from, a source word x_j. Then, the ith context vector c_i is the expected annotation over all the annotations with probabilities α_{ij}. The probability α_{ij}, or its associated energy e_{ij} , reflects the importance of the annotation h_j with respect to the previous hidden state in deciding the next state and generating y_i. Intuitively, this implements the attention mechanism in the decoder. The decoder chooses parts of the source sentence to focus on. By giving the decoder a chance to have an attention mechanism, the encoder is relieved from the weight of encoding all data in the source sentence into a fixed length vector. With this new methodology the data can be spread all through the grouping of annotations, which can be specifically recovered by the decoder as needs be. [7]

2.5 The Griffin-Lim based Postprocessing Module

The Griffin-Lim based Postprocessing Module converts predicted mel spectrogram frames into the corresponding waveforms. A CBHG module is used right on the top of the predicted mel spectrogram frame; to extract both forward and backward features (bidirectional GRU) and to correct errors in the predicted frames. Thus, raw spectrogram is predicted. The spectrogram lacks information about phase. Using Griffin-Lim algorithm we can infer the speech waveform by estimating phase from the spectrogram. It iteratively attempts to find the waveform whose STFT magnitude is closest to the generated spectrogram. [8]

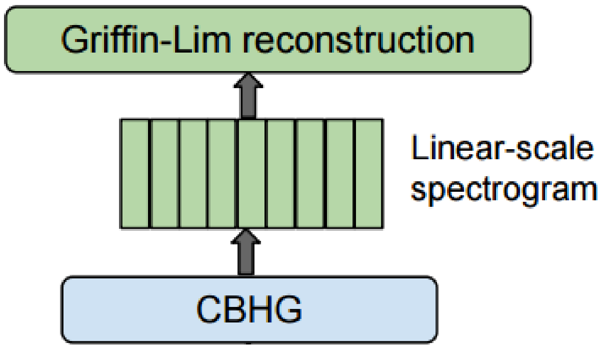

Figure 7: Postprocessing module

3. Dataset

There were 2 separate datasets used for this project – the L J Speech dataset and the NEU Emotional Speech Corpus based on the CMU Arctic [9] dataset.

3.1 L J Speech Dataset

This is a public domain speech dataset consisting of 13,100 short audio clips of a single speaker reading passages from 7 non-fiction books. A transcription is provided for each clip. Clips have a total length of approximately 24 hours. Metadata file consists of one record per line, delimited by the pipe character. The fields are: -

- ID: this is the name of the corresponding .wav file
- Transcription: words spoken by the reader
- Normalized Transcription: transcription with numbers, ordinals, and monetary units expanded into full words

Each audio file is a single-channel 16-bit PCM WAV with a sample rate of 22050 Hz. The audio clips range in length from approximately 1 second to 10 seconds. They were segmented automatically based on silences in the recording. Clip boundaries generally align with sentence or clause boundaries. The text was matched to the audio manually, and a quality assurance pass was done to ensure that the text accurately matched the words spoken in the audio. [10]

3.2 NEU Emotional Speech Corpus based on the CMU Arctic Dataset

The NEU Emotional Speech Corpus dataset is built for the purpose of emotional speech synthesis. The transcript is based on the CMU Arctic database. It includes recordings for four speakers - two males and two females. For this particular project, recordings of one female speaker (Jenie) was used. The emotional styles are neutral, sleepiness, anger, disgust and amused. Each audio file is recorded in 16bits .wav format. [11] The CMU Arctic database was constructed at the Language Technologies Institute at Carnegie Mellon University as phonetically balanced, US English single speaker databases designed for unit selection speech synthesis research. The databases consist of around 1150 utterances carefully selected from out-of-copyright texts. The database includes US English male and female speakers (both experienced voice talent) as well as other accented speakers. The distributions include 16KHz waveform and simultaneous EGG signals. Full phonetic labelling was performed using the FestVox based labelling scripts. [12]

4. Complexity Analysis

- Total number of parameters in the short version of the network = 8,950,993
- We obtained the shorter version of the tacotron model by reducing the convolutional layers, and by using time distributed strategy.
- We were not able to construct the attention block for this custom version so decided to stick with original model.
- Total number of parameters in the original model = 14,120,450
- Following are the number of output units at each stage in the network:

Level	Number of output units
Embeddings	256
Prenet	128
Encoder	256
Attention	256
Concatenate Attention	512
Decoder Cell	256
Decoder (5 frames)	400
Decoder (1 frame)	80
Postnet	256
Linear Spectrogram units	1025

5. Preprocessing Pipeline

5.1 Audio Pre-processing

- For each audio file, numpy files for mel spectrum and linear spectrum are generated.
- The mel spectrum has 80 banks
- The number of frequencies in linear spectrum are 1025.
- These mel-scale and linear scale spectrogram are stored on to disc in '.npy' files.
- To use this extracted data during training, filenames of linear and mel spectrograms of the associated audio, respective texts and number of utterances are written as a metadata on train.txt file.
- This entire process is repeated for all the emotions.

5.2 Text Pre-processing

- Randomly some part of the sentence is converted to ARPABET representation using CMU dictionary, this sequence is enclosed in curly braces as an identifier for ARPAbet representation and process accordingly.
- Now, this string of text is converted into sequence of IDs corresponding to the symbols in the text.

- We adopted the methodology for incorporating ARPAbet representation from [13]

6. Demo Android Application

- We created an app for the demo.This app used simple “GET” commands to feed input to the model.
- The android app was connected to the server via Wi-fi. The Server in this case was our laptop.
- We hosted the same code on Amazon Sagemaker and plan to do the same in future.

7. Existing codebase

Following are some of the existing codebases from where we adapted some of preprocessing and model architecture:

- https://github.com/Rayhane-mamah/Tacotron-2
- https://github.com/Kyubyong/tacotron
- https://github.com/MycroftAI/mimic2

8. Challenges and Adaptation

- We faced challenges in finding the data (exact sentences) for the NEU speech corpus, but found its CMU arctic base form [12]

- The existing TensorFlow codebase was referred and an initial Keras model using the exact same architecture was developed. The total parameters were close to 14.5 million. Therefore, the number of convolutional layers and input size were reduced to reduce the number of parameters. We ended up with 8,950,993 trainable parameters.

- We tried to use given architecture as it is but were not able to build attention mechanism suitable for the custom architecture with the reduced parameters. Moreover, constantly ended up getting overfitting results with the NEU speech corpus dataset.

- The main reason for overfitting was the size of the dataset. After selecting speaker with the longest audio files from the NEU emotional corpus dataset, we effectively obtained the audio worth of 20 to 30 mins for each emotion. To tackle this issue with decided to go with finetuning the model weights obtained from larger dataset.

- We decided to work with LJ Speech dataset, which is single speaker and has neutral tone with enough number of samples. (LJ Speech dataset is explained in Sec 3.1)

- Training was done over the L J Speech dataset as it had recordings for a single speaker and had no emotion. Also, the dataset had enough samples. No silent regions at the beginning and end of audio and thus was well aligned for speech synthesis.

- From the L J Speech training which was done over 441,000 iterations, readymade weights were obtained. Those obtained weights were used to fine tune the model over the NEU Speech Corpus dataset which was a smaller dataset and had emotion.

- After effectively finetuning the LJ Speech weights, we obtained reliable results but were not able to synthesize out of dictionary texts. For this we randomly replaced some part of the input text with the underlying phoneme representation during training, this significantly improved the quality of the generated audio.

- We also tried wavenet vocoder instead of griffin lim to generate audio waveforms but that proved out to be much slow for our application.

6. Individual Contributions

- Aditya – Preprocessing, Encoder, Attention, Data pipeline, Decoder, Report Compiling, Custom model, trained for amused and disgust
- Ishaan – Report Compiling and Presentation trained for sleepiness
- Siddharth – Custom Model, CBHG module, server, android application, Preprocessing, Attention Mechanism, tried sequence to sequence model

7. Results

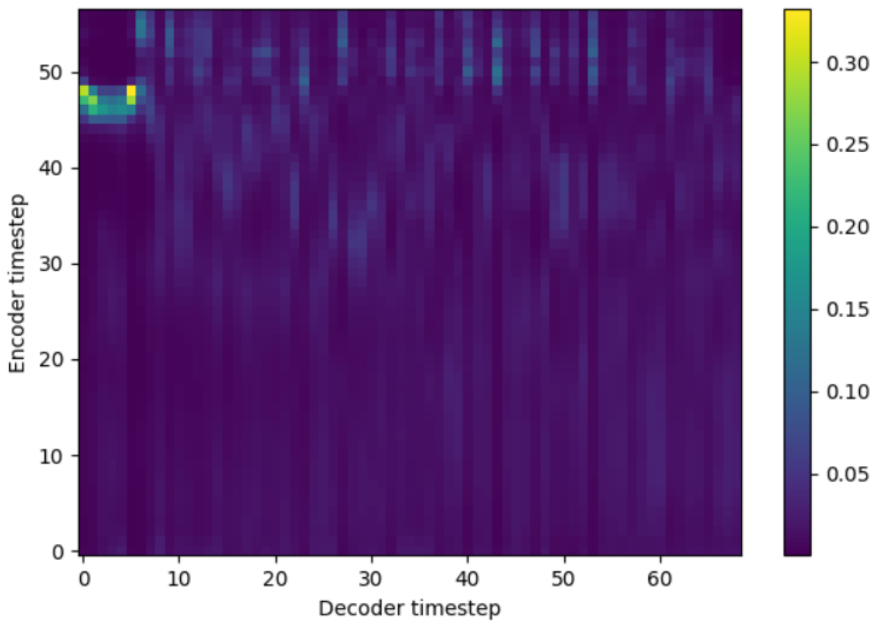

Figure 8: Step 147000 alignment in overfitted model

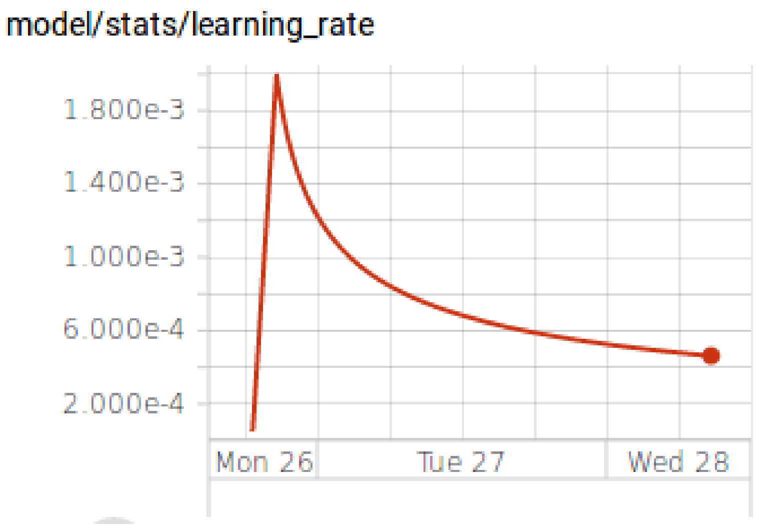

Figure 9: Loss curve after overfitting

Figure 8 shows the alignment with the overfitted model. We can hardly see and encoder decoder relationship and clearly there is a breach in establishing encoder-decoder relationship. In spite of this, even though the audio produced was unrelated to the text, it was legible.

Figure 9 shows the loss curve for the same, we can see loss reaching 2e-4 which is almost zero and clear evidence of model memorizing the target.
Later we finetuned the model that was first trained on LJ Speech dataset.

Following are the alignments and loss curves for the same

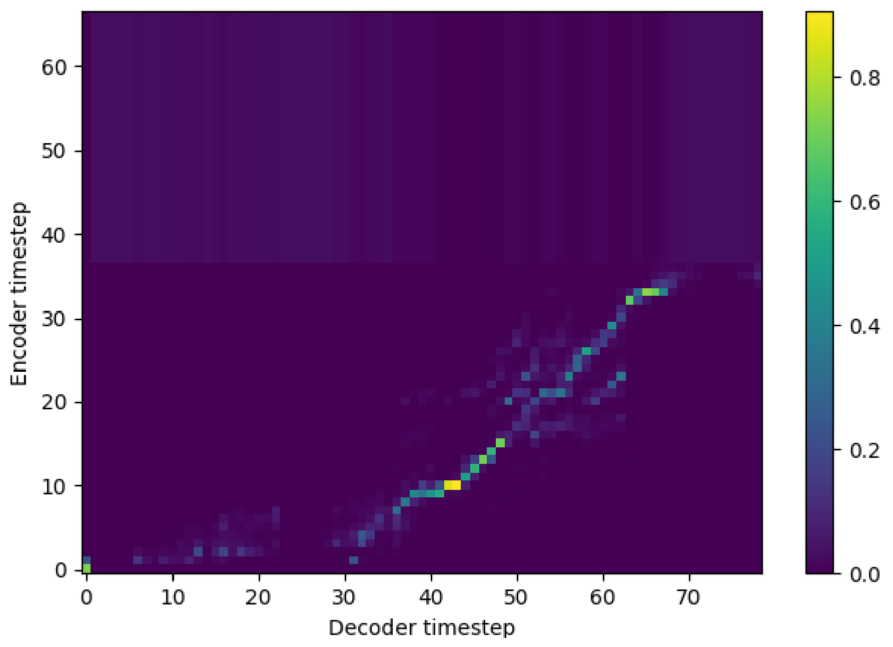

Figure 10: Step 441200 alignment

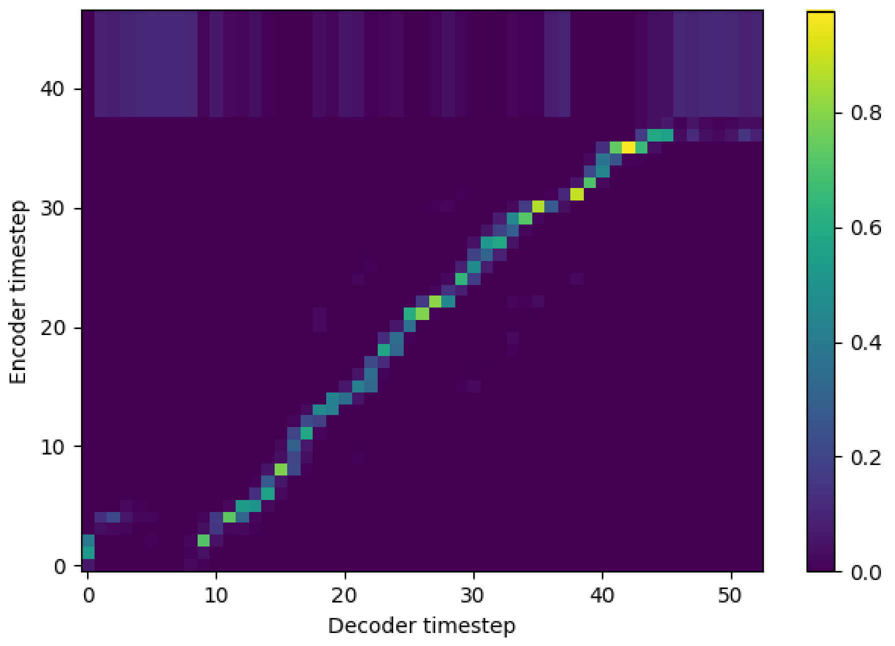

Figure 11: Step 441500 alignment

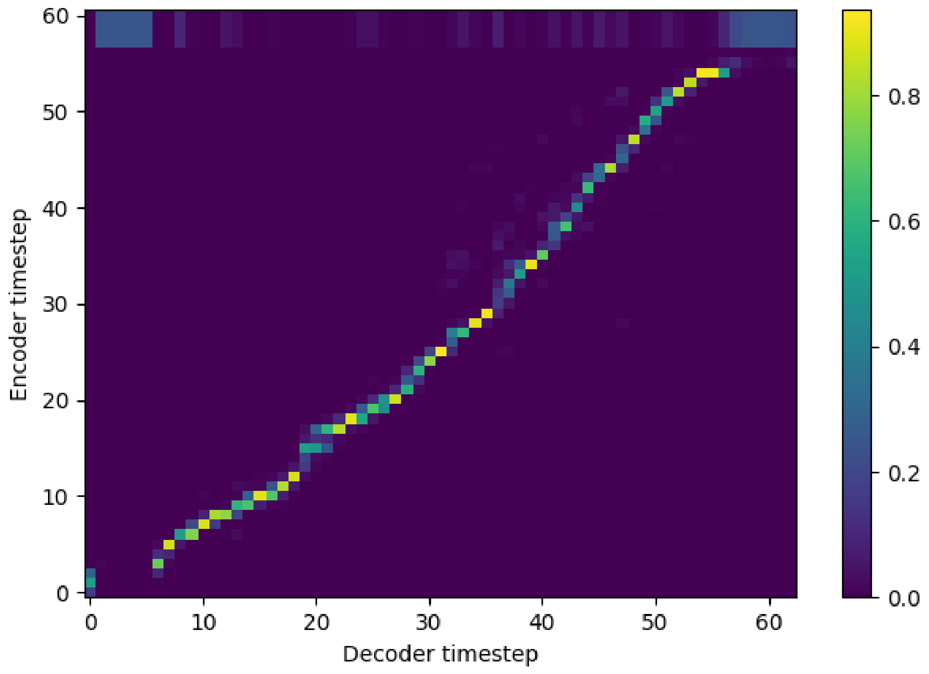

Figure 12: Step 453000 alignment

It is observed that the alignment gets better as the timestep increases during training.

The mel spectrogram of a generated audio was plotted. The learning rate and losses were plotted as well: -

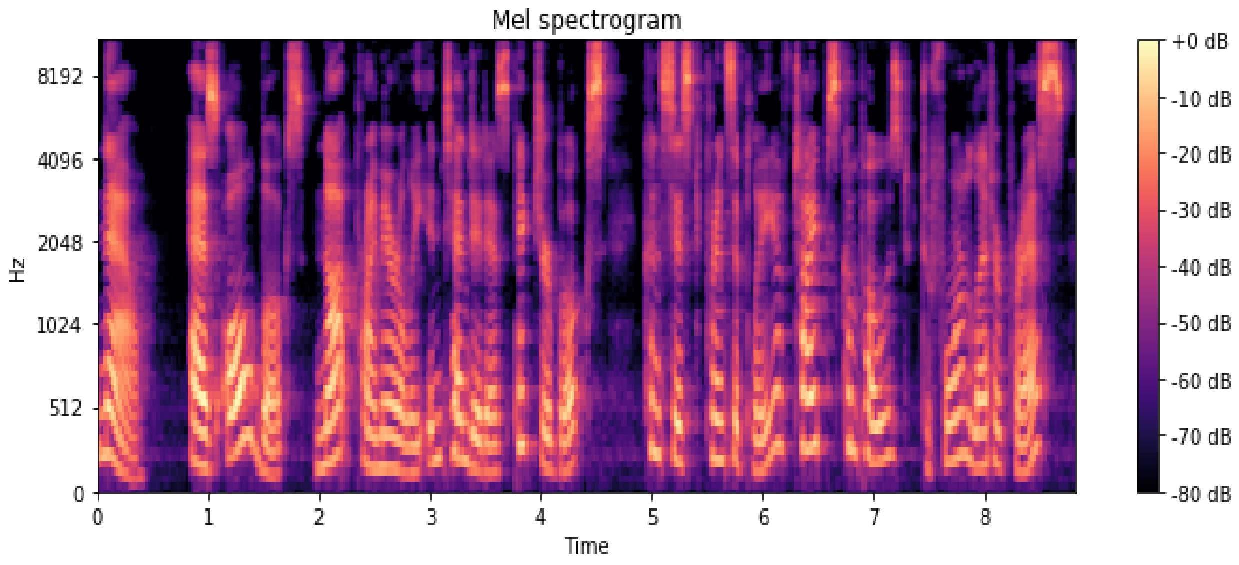

Figure 12: Mel spectrogram of generated audio

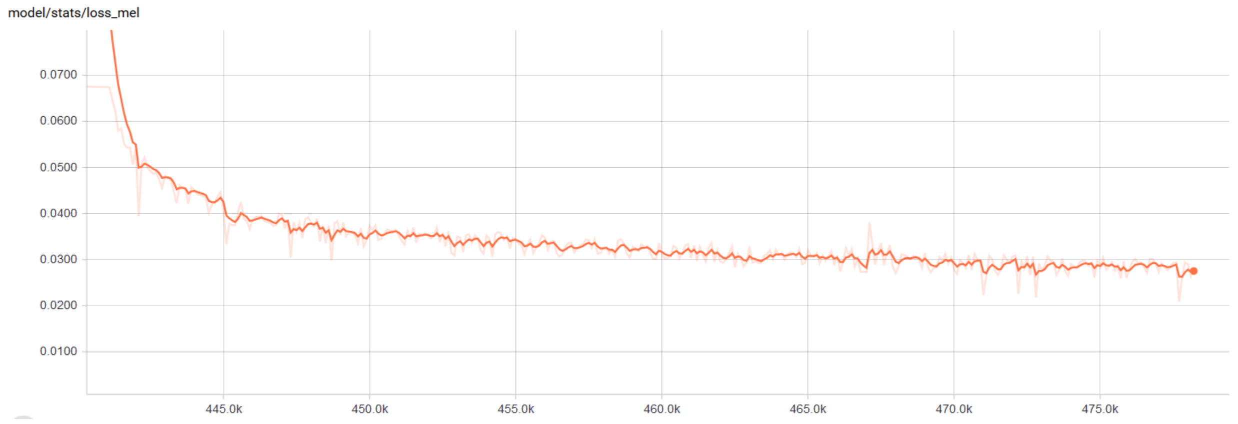

Figure 12: Loss – Mel

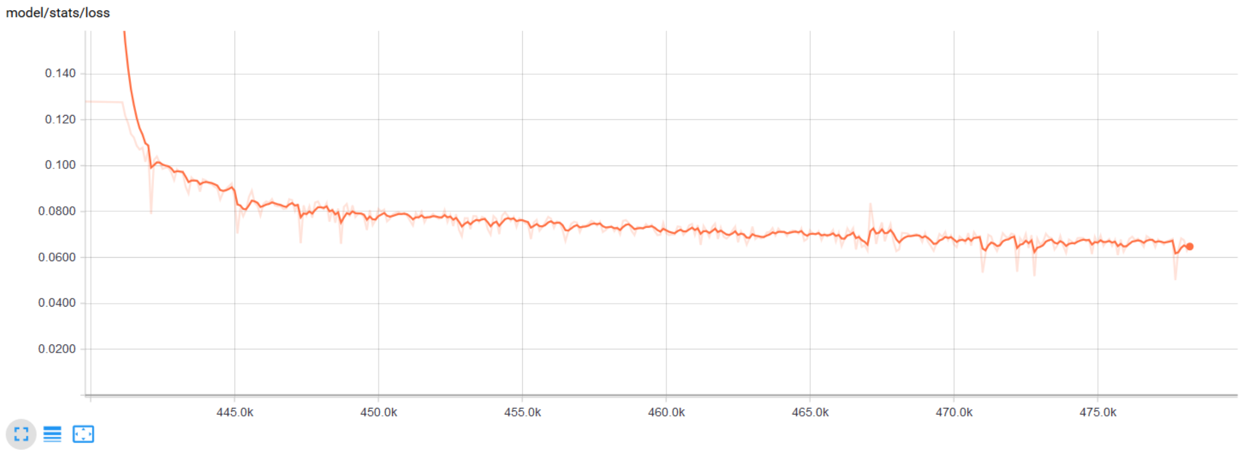

Figure 13: Loss – Post Mel

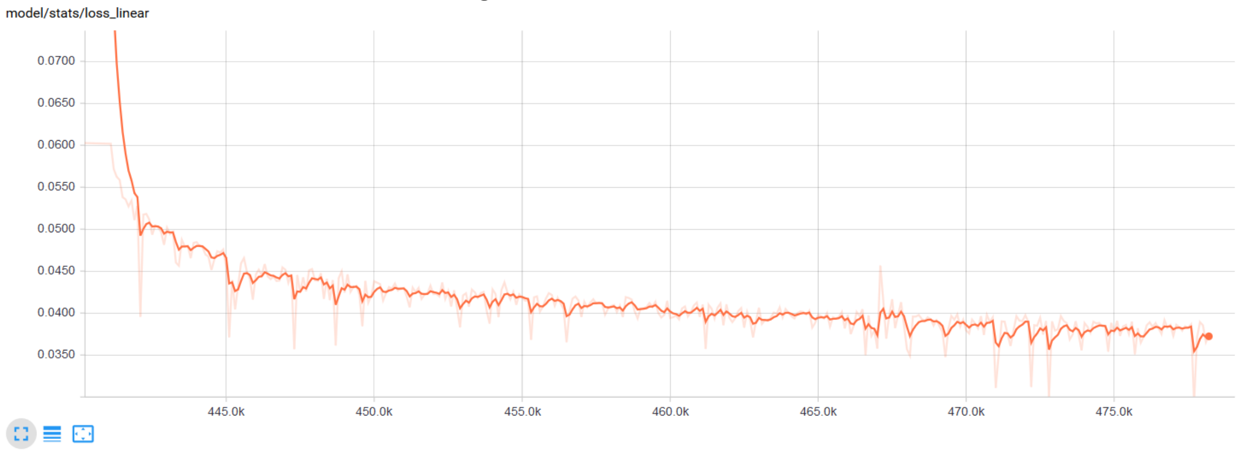

Figure 14: Loss – Post Linear

We finally deployed the model into an android app “OK Trojan!” for the demo.

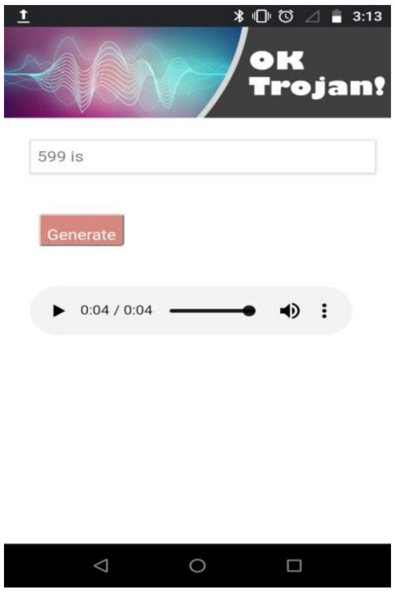

Figure 15: OK Trojan! Layout

Current samples in amused, sleepy and disgust for the same sentence can be checked on following link:

https://soundcloud.com/aditya-mukewar-801416775/sets/emotional-speech-synthesis-samples

All the preprocessed audio and respective metadata and the best weights for the model for each of these emotions are uploaded to following drive folder:

https://drive.google.com/open?id=1gs12BDqg2eJbrEiHQ_KZRp1scmj3xkmv

9. Future Work

- We are planning to launch a service-based keyboard application, which can generate emotional audio as per the emoticons in the text.
- One of the first things that we are currently trying is parallel wavenet vocoder which is faster than the vanilla wavenet vocoder and more suitable for our application.
- Since current approach is extremely raw and computationally heavy due to different model for each emotion, first thing that we have already started working on is fixing the encoder and adding information related to emotion separately into single model.
- After hitting the product worth reliability in the generated audio with discrete emotions we are planning on adding valence scale parameter which would enable us to mix 2 or more emotions as well as continuous emotional content in the audio.

- One more application that we are currently thinking of is generating audio for audiobooks. Our most baseline approach is to use sentiment analysis through nlp and incorporate that into synthesis of speech.

9. References

1. https://en.wikipedia.org/wiki/Acoustic_model
2. https://arxiv.org/abs/1703.10135 “Tacotron: Towards End-to-End Speech Synthesis”
3. https://arxiv.org/abs/1502.03167 “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”
4. https://arxiv.org/abs/1512.03385 “Deep Residual Learning for Image Recognition”
5. https://arxiv.org/abs/1505.00387 “Highway Networks”
6. https://arxiv.org/abs/1609.08144 “Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation”
7. https://arxiv.org/pdf/1409.0473 “Neural Machine Translation by Jointly Learning to Align and Translate”
8. https://ieeexplore.ieee.org/document/1164317 “Signal estimation from modified short-time Fourier transform”
9. Shao-Yen Tseng, PhD - USC Electrical Engineering, for pointing us in the direction of the NEU speech corpus-CMU Arctic dataset
10. https://keithito.com/LJ-Speech-Dataset/
11. http://www.coe.neu.edu/Research/AClab/Speech%20Data/README.md
12. http://www.festvox.org/cmu_arctic/index.html
13. https://github.com/keithito/tacotron/tree/master/text