Developing TTS Model for Indian Languages

A Project Report

submitted by

SAHITH ALOORI

*in partial fulfilment of the requirements
for the award of the degree of*

BACHELOR OF TECHNOLOGY

**DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.
May 2019**

THESIS CERTIFICATE

This is to certify that the thesis entitled **Developing TTS Model for Indian Languages**, submitted by **SAHITH ALOORI** (**EE15B055**), to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelors of Technology** is a bona fide record of the research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

UMESH S
Research Guide
Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date:

ACKNOWLEDGEMENTS

I would like to thank everyone who helped me.

ABSTRACT

KEYWORDS: TTS - Text To Speech , Tacotron 2 , ESPNet , Rayhane-Mamah , Wavenet Vocoder , Feature prediction Network , mel-spectrogram , Griffin-Lim algorithm.

The aim of this project is to develop best TTS model for Indian languages with limited data. For the first part TTS model was built using ESPNet for data-sets LJSpeech, Indian English, Hindi .The synthesized outputs for all the three models were good. The only problem was that the synthesized audio doesn't sound like human speech rather it sounds like a robot. The reason may be that it was using linear mel-spectrogram for synthesizing audio with griffin-lim algorithm which is slightly outdated. In the next part Rayhane-Mamah code was used for the same data-sets. This code has wavenet vocoder which is a deep neural network for generating relatively realistic-sounding human-like voices by directly modelling waveforms using a neural network method trained with recordings of real speech. The results were pretty good and output was human like speech after training for 11 days for LJSpeech . But training feature prediction network was taking too long and the quality of this network was not very good. So ,the last consists of integrating ESPNet's feature prediction network and Rayhane's wavenet vocoder to get the best TTS training neural network.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

TTS	Text To Speech
IITM	Indian Institute of Technology, Madras

NOTATION

r	Radius, m
α	Angle of thesis in degrees
β	Flight path in degrees

CHAPTER 1

INTRODUCTION

1.1 TACOTRON-2

Modern text-to-speech (TTS) pipelines are complex . For example, it is common for statistical parametric TTS to have a text frontend extracting various linguistic features, a duration model, an acoustic feature prediction model and a complex signal-processing-based vocoder . These components are based on extensive domain expertise and are laborious to design. They are also trained independently, so errors from each component may compound. The complexity of modern TTS designs thus leads to substantial engineering efforts when building a new system.

1.1.1 Feature prediction network

There are thus many advantages of an integrated end-to-end TTS system that can be trained on ¡text, audio¿ pairs with minimal human annotation. First, such a system alleviates the need for laborious feature engineering, which may involve heuristics and brittle design choices. Second, it more easily allows for rich conditioning on various attributes, such as speaker or language, or high-level features like sentiment. This is because conditioning can occur at the very beginning of the model rather than only on certain components. Similarly, adaptation to new data might also be easier. Finally, a single model is likely to be more robust than a multi-stage model where each components errors can compound. These advantages imply that an end-to-end model could allow us to train on huge amounts of rich, expressive yet often noisy data found in the real world.

1.1.2 MODEL ARCHITECTURE

Our proposed system consists of two components, shown in Figure 1.1: (1) a recurrent sequence-to-sequence feature prediction network with attention which predicts a sequence of mel spectrogram frames from an input character sequence, and (2) a modified version of WaveNet which generates time-domain waveform samples conditioned on the predicted mel spectrogram frames.

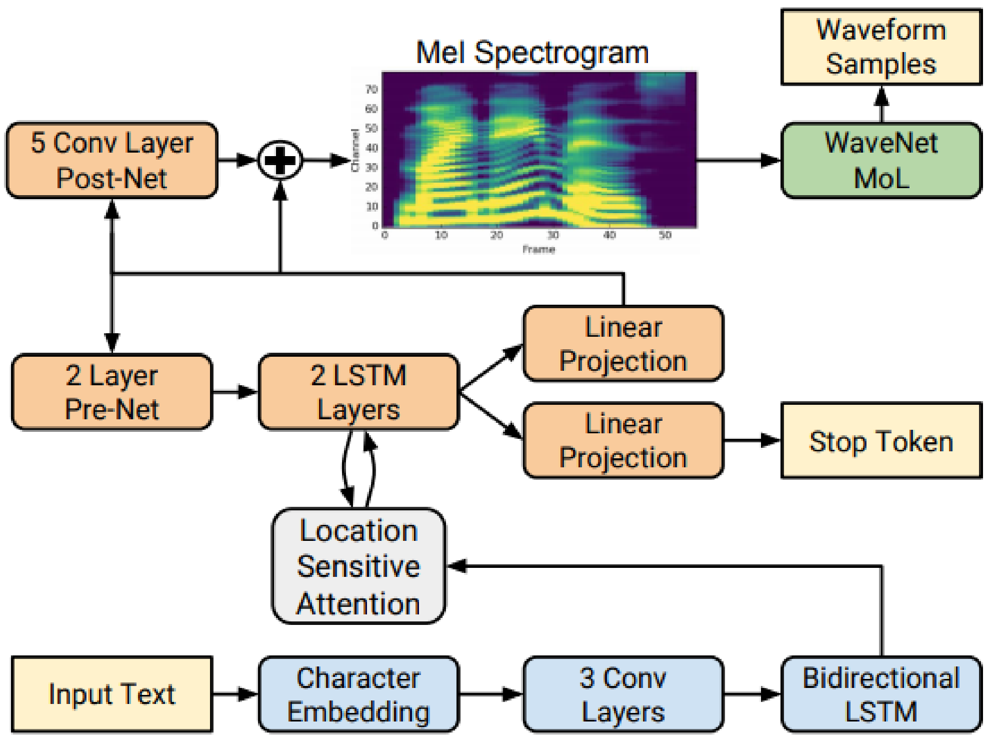


Figure 1.1: Block diagram of the Tacotron 2 system architecture.

As in Tacotron, mel spectrograms are computed through a shorttime Fourier transform (STFT) using a 50 ms frame size, 12.5 ms frame hop, and a Hann window function. We experimented with a 5 ms frame hop to match the frequency of the conditioning inputs in the original WaveNet, but the corresponding increase in temporal resolution resulted in significantly more pronunciation issues. We transform the STFT magnitude to the mel scale using an 80 channel mel filterbank spanning 125 Hz to 7.6 kHz, followed by log dynamic range compression. Prior to log compression, the filterbank output magnitudes are clipped to a minimum value of 0.01 in order to limit dynamic range in the logarithmic domain. The network is composed of an encoder and a decoder with attention. The encoder converts a character sequence into a hidden feature representation which the decoder consumes to predict a spectrogram. Input characters are represented using a learned 512-dimensional character embedding, which are passed through a stack of 3 convolutional layers each containing 512 filters with shape 5 1, i.e., where each filter spans 5 characters, followed by batch normalization [18] and ReLU activations. As in Tacotron, these convolutional layers model longer-term context (e.g., N-grams) in the input character sequence. The output of the final convolutional layer is passed into a single bi-directional [19] LSTM [20] layer containing 512 units (256 in each direction) to generate the encoded features. The encoder output is consumed by an attention network which summarizes the full encoded sequence as a fixed-length context vector for each decoder output step. We use the location-sensitive attention from [21], which extends the additive attention mechanism [22] to use cumulative attention weights from previous decoder time steps as an additional feature. This encourages the model to move forward consistently through the input, mitigating potential failure modes where some subsequences are repeated or ignored by the decoder. Attention probabilities are computed after projecting inputs and location features to 128-dimensional hidden representations. Location features are computed using 32 1-D convolution filters of length 31. The decoder is an autoregressive recurrent neural network which predicts a mel spectrogram from the encoded input sequence one frame at a time. The prediction from the previous time step is first passed through a small pre-net containing 2 fully connected layers of 256 hidden ReLU units. We found that the pre-net acting as an information bottleneck was essential for learning attention. The prenet output and attention context vector are concatenated and passed through a stack of 2 uni-directional LSTM layers with 1024 units. The concatenation of the LSTM output and the attention context vector is projected through a linear transform to predict the target spectrogram frame. Finally, the predicted mel spectrogram is passed through a 5-layer convolutional post-net which predicts a residual to add to the prediction to improve the overall reconstruction. Eachpost-net layer is comprised of 512 filters with shape 5 1 with batch normalization, followed by tanh activations on all but the final layer. We minimize the summed mean squared error (MSE) from before and after the post-net to aid convergence. We also experimented with a log-likelihood loss by modeling the output distribution with a Mixture Density Network [23, 24] to avoid assuming a constant variance over time, but found that these were more difficult to train and they did not lead to better sounding samples. In parallel to spectrogram frame prediction, the concatenation of decoder LSTM output and the attention context is projected down to a scalar and passed through a sigmoid activation to predict the probability that the output sequence has completed. This stop token prediction is used during inference to allow the model to dynamically determine when to terminate generation instead of always generating for a fixed duration. Specifically, generation completes at the first frame for which this probability exceeds a threshold of 0.5. The convolutional layers in the network are regularized using dropout [25] with probability 0.5, and LSTM layers are regularized using zoneout [26] with probability 0.1. In order to introduce output variation at inference time, dropout with probability 0.5 is applied only to layers in the pre-net of the autoregressive decoder. In contrast to the original Tacotron, our model uses simpler building blocks, using vanilla LSTM and convolutional layers in the encoder and decoder instead of CBHG stacks and GRU recurrent layers. We do not use a reduction factor, i.e., each decoder step corresponds to a single spectrogram frame.

1.2 WaveNet Vocoder

We use a modified version of the WaveNet architecture from [8] to invert the mel spectrogram feature representation into time-domain waveform samples. As in the original architecture, there are 30 dilated convolution layers, grouped into 3 dilation cycles, i.e., the dilation rate of layer k (k = 0 . . . 29) is 2 k (mod 10). To work with the 12.5 ms frame hop of the spectrogram frames, only 2 upsampling layers are used in the conditioning stack instead of 3 layers. Instead of predicting discretized buckets with a softmax layer, we follow PixelCNN++ [27] and Parallel WaveNet [28] and use a 10- component mixture of logistic distributions (MoL) to generate 16-bit samples at 24 kHz. To compute the logistic mixture distribution, the WaveNet stack output is passed through a ReLU activation followed by a linear projection to predict parameters (mean, log scale, mixture weight) for each mixture component. The loss is computed as the negative log-likelihood of the ground truth sample.

CHAPTER 2

IMPLEMENTATION

2.1 ESPNET

ESPnet is an end-to-end speech processing toolkit, mainly focuses on end-to-end speech recognition and end-to-end text-to-speech. ESPnet uses chainer and pytorch as a main deep learning engine, and also follows Kaldi style data processing, feature extraction/format, and recipes to provide a complete setup for speech recognition and other speech processing experiments.

2.1.1 TTS1 RECIPE

tts1 recipe is based on Tacotron2 [1] (spectrogram prediction network) w/o WaveNet. Tacotron2 generates log mel-filter bank from text and then converts it to linear spectrogram using inverse mel-basis. Finally, phase components are recovered with Griffin-Lim.

2.1.2 RESULTS

THE DATASETS USED FOR TRAINING and their loss curves versus epochs:-

1.) LJSpeech (US English) - 24hrs

2.) Indian English - 8hrs

3.) Hindi - 8hrs

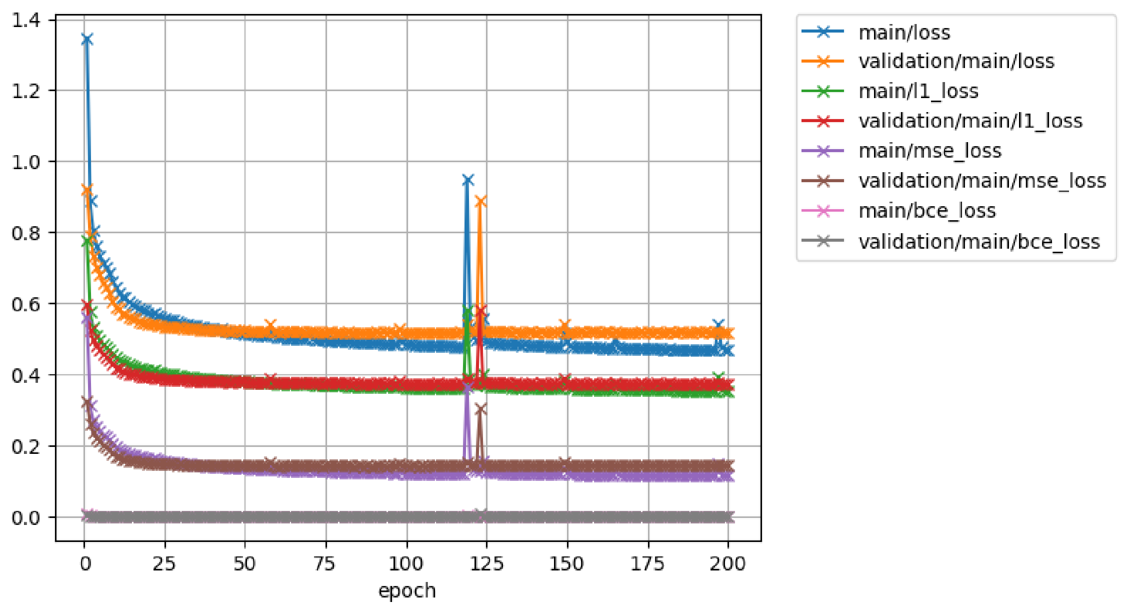


Figure 2.1: loss vs epoch curve for LJSpeech.

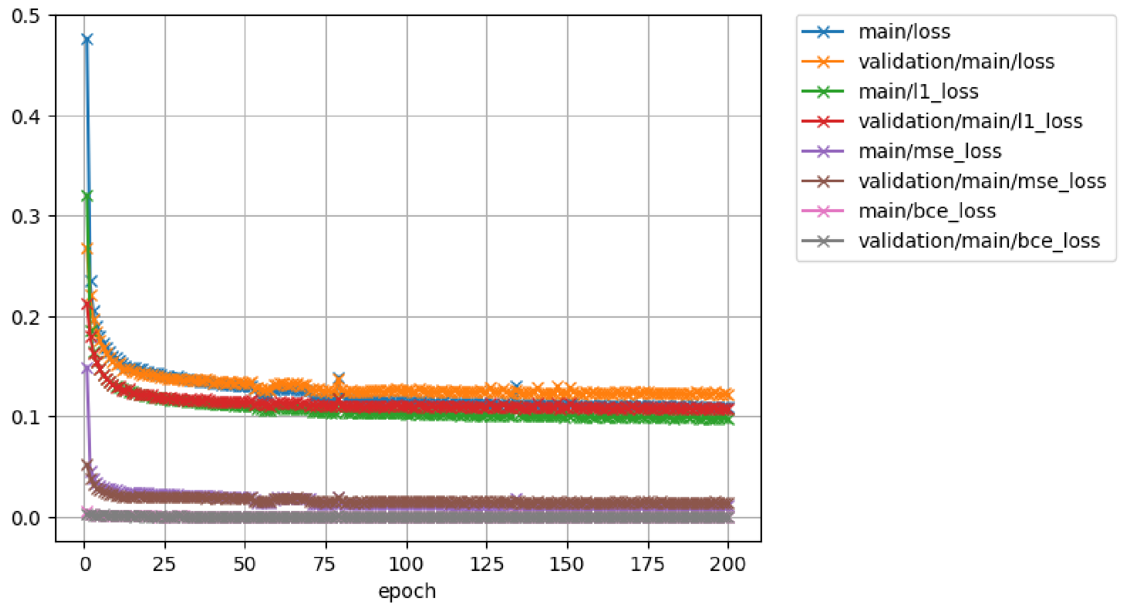


Figure 2.2: loss vs epoch curve for Indian English.

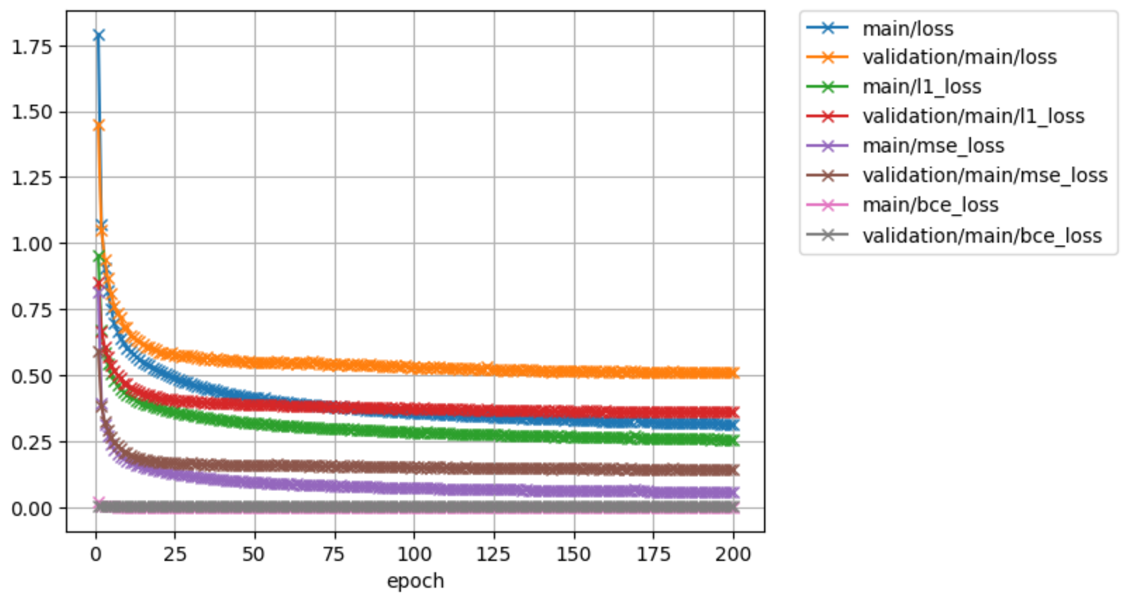


Figure 2.3: loss vs epoch curve for Hindi

All of the datasets didn't take more than 2 days for training. the pronounciation was really good. But the only problem was that the synthesized voice was robotic probably the reason being that lim-griffin algorithm converts mel-spectrograms to linear-spectrograms which amplifies the error.

2.2 Rayhane-Mamah

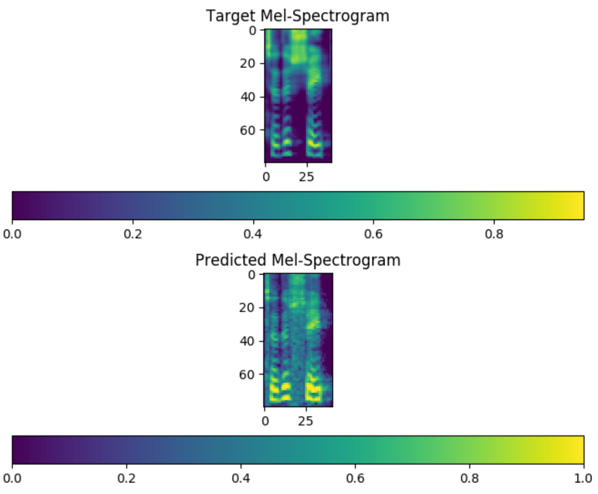
Rayhane - Mamah code uses wavenet vocoder to synthesise audio from mel-spectrograms instead of Griffin-Lim algorithm.

2.2.1 WaveNet

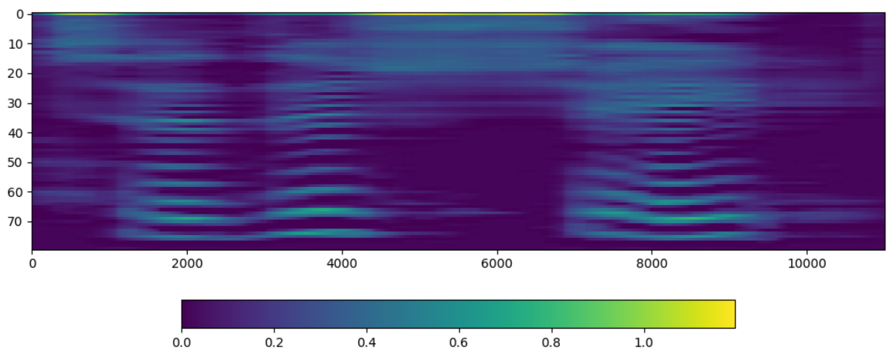
WaveNet is a deep neural network for generating relatively realistic-sounding human-like voices by directly modelling waveforms using a neural network method trained with recordings of real speech.

2.2.2 Results

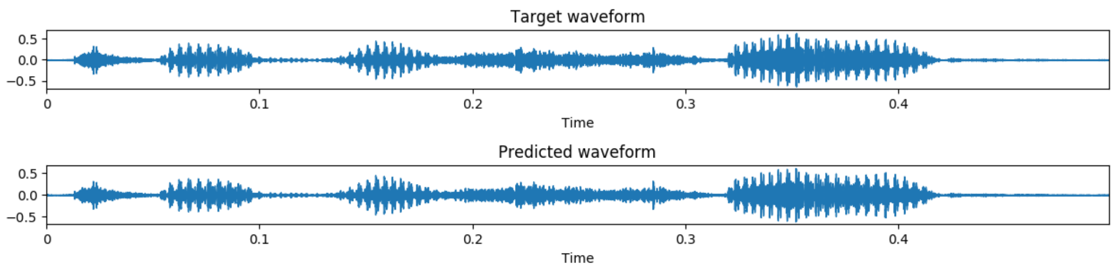
LJSpeech



Local Condition vs Reconst. Mel-Spectrogram, step=500000, loss=-4.18441

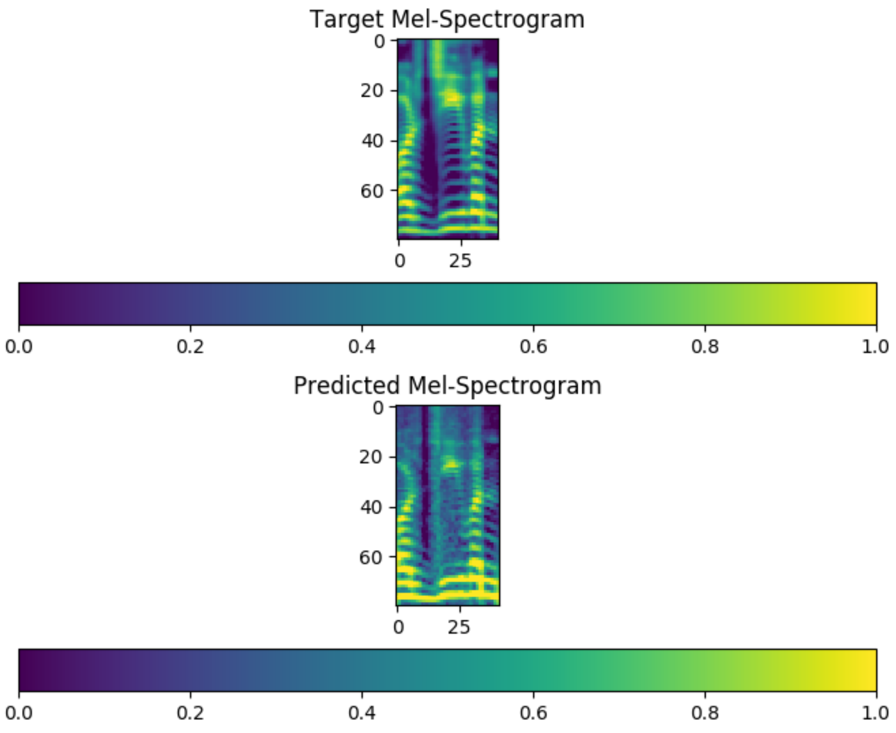


Upsampled Local Condition features, step=500000, loss=-4.18441

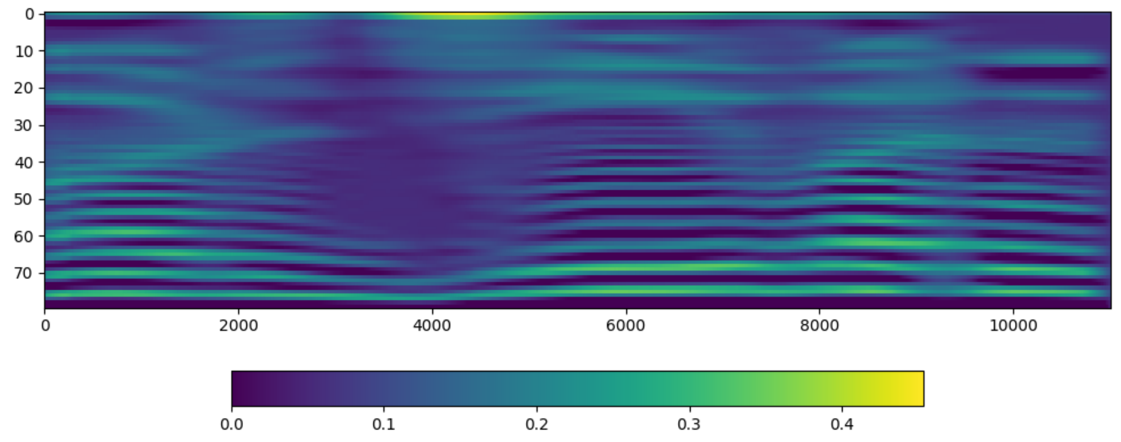


WaveNet, 2019-05-25 15:03, step=500000, loss=-4.18441

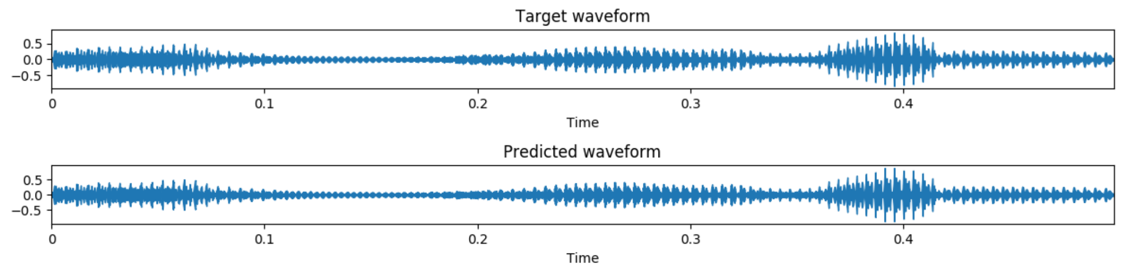
Hindi



Local Condition vs Reconst. Mel-Spectrogram, step=987500, loss=-4.58440



Upsampled Local Condition features, step=987500, loss=-4.58440



Tacotron-2, 2019-06-24 23:39, step=987500, loss=-4.58440

The results were amazing with more human like speech but, the problem was it takes more than 12 days to train and the generated mel-spectrograms were not as good as ESPNet's generated mel-spectrograms . So I've combined ESPNet feature generation network and wavenet vocoder to yield best results which are more human like. Currently I'm working on it.

Publications

1. https://arxiv.org/pdf/1703.10135.pdf
2. https://arxiv.org/pdf/1703.10135.pdf

REFERENCES