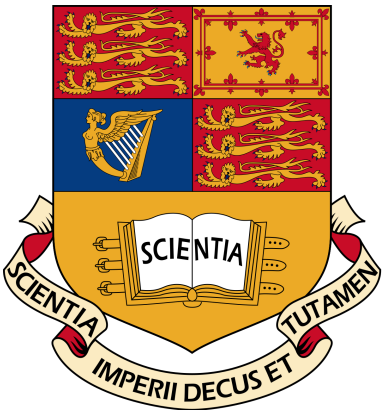
Reproduction & Improvement of State-of-the-art TTS model

Fei Xie

A Thesis Presented for the Degree of

Master of Science

Supervisor:
Chao Wu



Department of Computing
Imperial College London
London, UK

August, 2018

Reproduction & Improvement of State-of-art TTS model

Abstract

With the sharp increase in development of Deep learning in recent years, more computer speech and image problems consider using related approach to achieve a more positive performance, in different fields such as Object Detection, Classification Problems and Natural Language Processing. This project investigates the use of Deep Recurrent Neural Networks(RNNs), LSTM(Long Short-term Memory), Sensitive Attention, Convolutional Neural Networks(CNNs) with various kinds of optimization methods on the networks such as SGD(Stochastic Gradient Descent), Dropout, Zoneout, Data Regularization and Batch Normalization applied in this project. Building the whole structure of the networks with the language of python in TensorFlow was ultimately decided to be suitable for use due to TensorFlow's large community engagement and maintenance routine by Google, and its implementation was fully investigated and applied, resulting in a full guide to be able to fully implement and train this network approach in future research. To test this implementation, the whole project was applied to two problem domains using text and audio data. Firstly, we need to reproduce the whole network with two parts, which is Tacotron2, a kind of sequence to sequence network developed by Google TTS team for transforming the raw text data into Mel-Spactrogram data, based on this, the next step is to compress Tacotron2 model with removing the Post-Net and adding more dropout and zoneout techs to speed up the training process and inference time, and then the output of Tacotron2 was made as the input to WaveNet created by DeepMind team which is used to predict time-domain waveform samples. Here the big problem is the training of WaveNet, which cost me about 1 month to train, which held back the improvement step a lot. However, while testing and comparing the simplified model and the original model presented limited results, this approach was found to be suitable for further research.

Acknowledgements

Many thanks to Prof Yike Guo, Dr Hao Dong and Mr Yuanhan Mo's guidance and support throughout the whole year for this project, and I would like to express my deepest gratitude to them. In every major problem, Dr Hao Dong and Prof Yike Guo patiently provided almost all the help they can to assist me in finishing the objectives that was set. Working under their supervision was such a good experience and also provided me with many skills that are important in order to contribute effectively in a research process, and also a foundation into the next stage of my career. I also like to thank Prof Yike Guo for his support of server suggestions on improving the whole models under training huge data. Appreciations to Imperial College London, which has helped me foster a lot of essential skills needed to excel in my master studying and career path. Thanks to my father and all my friends for always helping and reminding about my project structure, schedule and deadlines, making sure that I was progressing as planned.

Contents

Chapter 1

Introduction

1.1 Sequence-to-sequence Model

Models of Text-to-speech(TTS) used to translate written language into human speaking, which can be applied to several scenes such as human-equipment interface, accessibility for visually impaired media and entertainment, has revolutionized in last few years. The classic technology were based on the concatenating TTS with unit selection, where to recombine large number of speech fragments from a large database containing the recordings from a single speaker. In other words, it is hard to modify the voice once finished [106]. Another multi-stages hand-engineered pipelines stated in [103] bring a widely used tool nowadays to transform text data into a compact audio representation with a synthesis method, which called vocoder [33]. Furthermore, smooth trajectories of speech features can also be generated by some statistical parametric speech methods such as Hidden Markov Model[104][27][118][105] then be synthesized by a vocoder. However, these methods are limited by the produced waveform which presents muffled and unnatural compared to human speech.[95]

However, with the boom of technology of Deep Learning, especially after the success of ImageNet[56], neural networks has been applied into TTS area to build as stated end-to-end neurons system to input the text of written sentences and then be transformed into kinds of audios as output with different voice.

In recent years, WaveNet present a really impressive result, which is a kind

Figure 1.1: An Example of Vocoder

of generative, fully probabilistic and autoregressive model to predict audio waveform frame by frame conditioned on the previous distribution[106], created by DeepMind in 2016. WaveNet has been applied to some TTS systems such as DeepVoice [5][7][82] developed by Baidu. Kinds of input features like linguistic features, predicted log fundamental frequency and phoneme durations are fed into WaveNet to train to demonstrate different performances, however, still in requirement of some other assistance involving test-analysis systems or robust lexicon.

Due to these reasons, Tacotron[114], a kind of sequence-to-sequence architecture[101] for producing magnitude spectrogram from a sequence of characters, with a single neural network to train the data alone, so as to replace the production of these linguistic and acoustic features, which replaced the traditional speech synthesis pipeline. Furthermore, Griffin-Lim algorithm [33] was used to act as vocoder to estimate the phase after the Fourier transform. Due to the reason that Griffin-Lim algorithm produce a worse quality of waveform than WaveNet, Tacotron2[95] produced a kind of sequence-to-sequence Tacotron-style model[114] that generates mel-spectrogram, followed by a modified WaveNet vocoder[7][102]. Tacotron2 can train the normalized character sequences directly and then to produce the corresponding speech waveform. The results present a natural sounding which is hard to distinguish from real one. Also, Deep

Voice3[82] and Char2Wav[97] describes a similar system but shows less competitive results compared to Tacotron2.

This project reproduced the work of Tacotron2 based on [95] and cut part of the network to compress the whole model to compare the relative results. The final results show that the performance can be as or even more positive than the one described in [95].

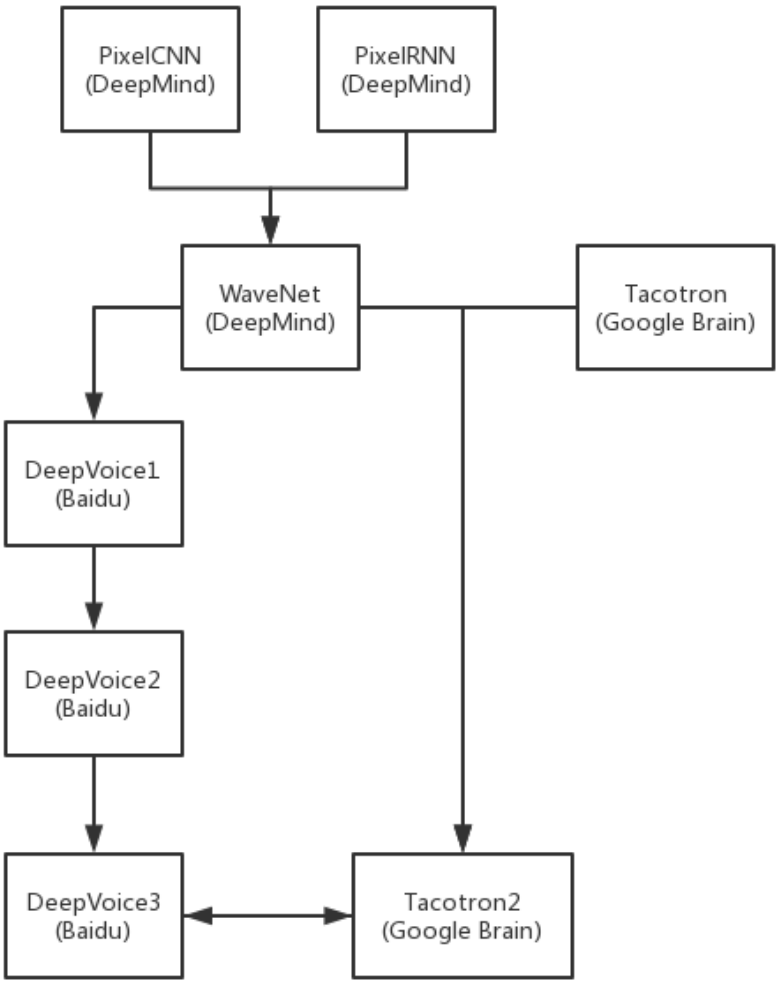


Figure 1.2: Development TTS System Based on Neural Networks

Figure 1.2 shows the tack of development of the main used technology in TTS problems at present.

1.2 Problem Statement

1.2.1 Text to Speech Problem

Text to speech problem is an essential component in many applications such as navigation systems and speech-enabled devices which synthesize human voice from text content.

The first aim of this research focused project is to use sequence-to-sequence model to encode and decode sentence features, and map audio and the learned text features to each other frame-by-frame. By this, it is meant that the text content features can be used to estimate equivalent audio features.

The text features learned with sequence-to-sequence model will be used as an input into a WaveNet as the vocoder for synthesizing the final waveform. This will follow the process outlined in [95].

However, as will be discussed later, the training process of WaveNet takes about one month to get a human-like performance, which results in lacking of time to improve and compress Tacotron2 model to achieve the goal of this project. However, the final result still shows a good demonstration for translating the sentence of input features produced from Tacotron2 and the final compressed model illustrates more improvement.

1.2.2 Improvement Problem

Improvement of Tacotron2 is a process of trying to drop some layers of the model and try to compress the whole model so as to speed up the processing time and also have the same or even better results. In the investigation performed in this thesis, we focus on three different kinds of drop process.

The second aim of this project is to drop some layers of the Tacotron2 to train the same data set, including Pre-Net and post-net and then try to compare the final results with the normal one.

The final results of comparison of the two results shows that the dropped layers does not perform really important role in Tacotron2 which means they are not

necessary to achieve a more human-like voice in the whole model structure.

1.3 Data Used

A speech data set called Lj-speech data set[46] was used in this project to for training process.

Lj-speech data set is a public domain speech data set, which includes 13,100 short audio clips of a single speaker, and the content of the reading is from 7 non-fiction books which are [51][20][63][38][89][35][8] respectively between 1884 and 196. In addition, the audio was recorded in 2016-17 by the LibriVox project. Each clip has a corresponding transcription provided in a text file. These clips vary in length from 1 to 10 seconds, and the total length is approximate 24 hours. In addition, each audio file is a single-channel 16-bit PCM WAV with a sample rate of 22050 Hz. All details are shown in the following table 1.2.

Table 1.1: Speech Data Statistics

Total Clips	13,100
Total Words	225,715
Total Characters	1,308,678
Total Duration	23:55:17
Mean Clip Duration	6.57 sec
Min Clip Duration	1.11 sec
Max Clip Duration	10.10 sec
Mean Words per Clip	17.23
Distinct Words	13,821

The file 'transcripts.csv' contains the metadata one record per line, delimited by the pipe character (0x7c). The fields are:

- **ID**: this is the name of the corresponding .wav file
- **Transcription**: words spoken by the reader (UTF-8)

- **Normalized Transcription**: transcription with numbers, ordinals, and monetary units expanded into full words (UTF-8).

Overall, the audio clips range randomly in length from approximately 1 second to 10 seconds. The silence is used as the position of segmentation to divide the audio waveform automatically.

The text was matched line by line to the recordings manually, and also to make sure the text can be accurately matched to the words, a QA pass was adopted.

Some abbreviations appear in the text as well. They need to be expanded as follows:

Table 1.2: Abbreviation and Expansion Mapping[49]

Abbreviation	Expansion	Abbreviation	Expansion
Mr.	Mister	Mrs.	Misess (*)
Dr.	Doctor	No.	Number
St.	Saint	Co.	Company
Jr.	Junior	Maj.	Major
Gen.	General	Drs.	Doctors
Rev.	Reverend	Lt.	Lieutenant
Hon.	Honorable	Sgt.	Sergeant
Capt.	Captain	Esq.	Esquire
Ltd.	Limited	Col.	Colonel
Ft.	Fort		

Chapter 2

Background Studying

Reproducing Tacotron2 and compressing the model to improve the final performance is the main focus of this project. Reproducing model process is based on the paper [95] to build up the sequence-to-sequence model from scratch. It uses encoder system to embedding the raw text into vectors and then try to train it with LSTM networks. Then the output features are seen as the input of attention model to pre-process so that the decoder part can train them with the weight which illustrates which features should be pay more attention to train. The final output of the S2S model is the the liner-mapping of Mel-spectrogram signal vectors, and then they are sent into the final training model of WaveNet Vocoder.

The compressing stage of Tacotron2 focus on the inside layers of S2S model which is Post-net in decoder part. Firstly we dropped the layer of Post-Net and then give the comparison with the original one to find the difference. The next step is to apply dropout and zoneout to get the final result and then compare them with the original one based on [95] to get the conclusion about the dropping operation.

2.1 Analysis of Different Structure

2.1.1 Wavenet

WaveNet[106] was created by DeepMind in 2016, which is the first autoregressive model to try to model the raw audio, which adapt 16,000 samples per second as the output which showed in figure2.1. Additionally, these samples depend on the previous ones which inspired using the similar structure of neural network as PixelCNN[78] and PixelRNN[75] where writers stated it possible to generate complex natural images one colour-channel at a time with a structure called dilated convolution networks showed in figure 2.2, which can enlarge the respective fields without increasing the layers of the network. Another structure called residual and skip connections showed in 2.3 was from [39] to accelerate the speed of training and ensure the gradient can be broadcast into a deeper layer.

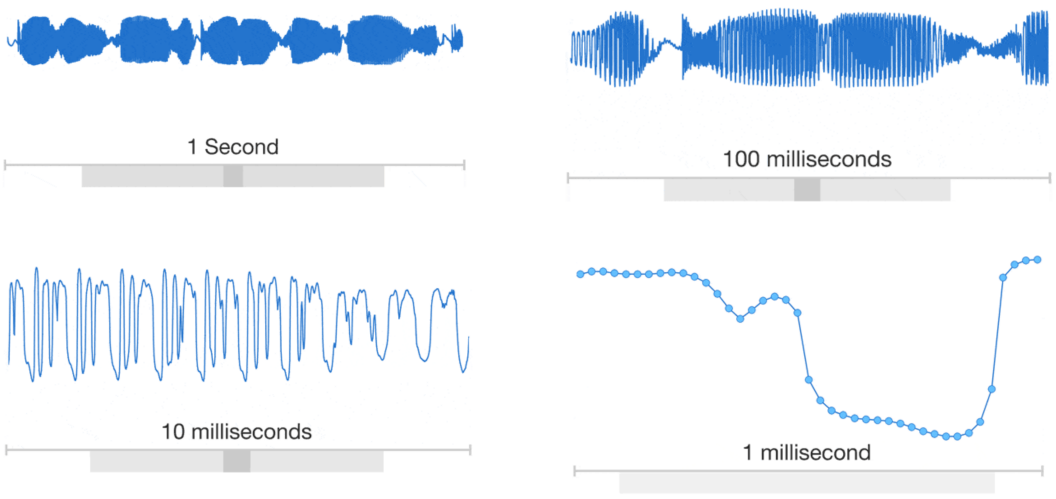


Figure 2.1: Audio Samples in Different Time Scale

At the training stage, the waveform recorded from human speakers is the input, and the output is the synthetic utterances after training. Then this output ise put back into the network to be a support to keep training the next value, which is implemented by the dilated networks. This process is very slow which is the main disadvantages of WaveNet [76], so DeepMind gave another more advanced network called Parallel WaveNet in[76] which use a similar structure like GANs to

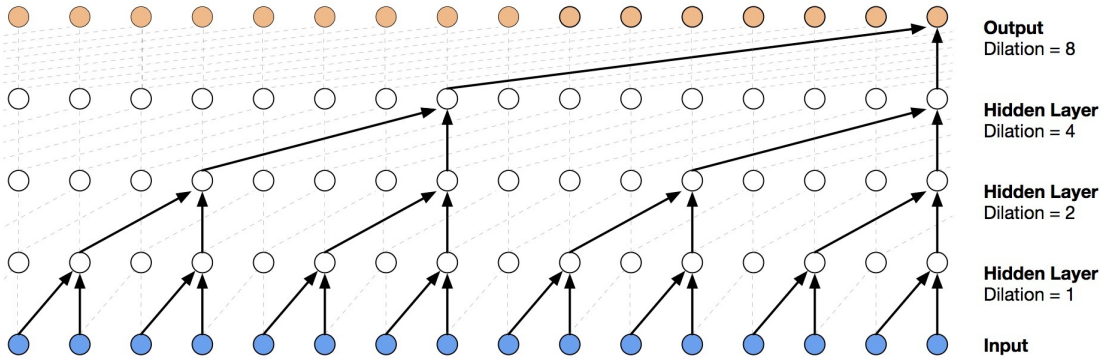


Figure 2.2: Structure of Dilated Convolution Networks

speed up the training process.

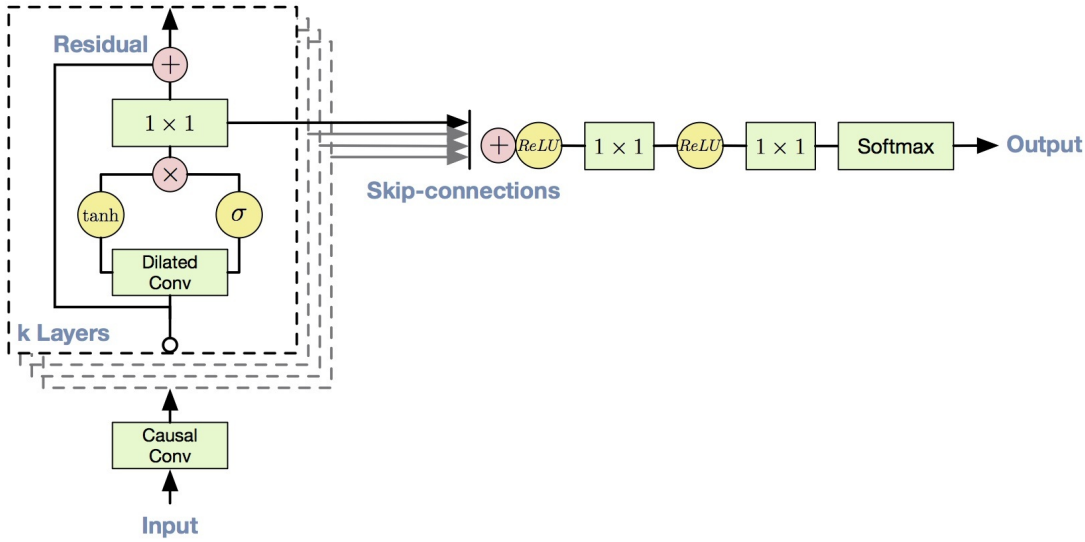


Figure 2.3: Residual And Skip Connections

2.1.2 Tacotron & Tacotron2

Two more structures were given by the group of Google Brain called Tacotron[114] present some other useful models to deal with TTS problems. Due to that the input to WaveNet need a lot of expertise to produce, involving elaborate text-analysis systems as well as a robust lexicon, Tacotron[114] applied a sequence-to-sequence(S2S)[101] architecture, which can produce magnitude spectrogram from a sequence of characters.

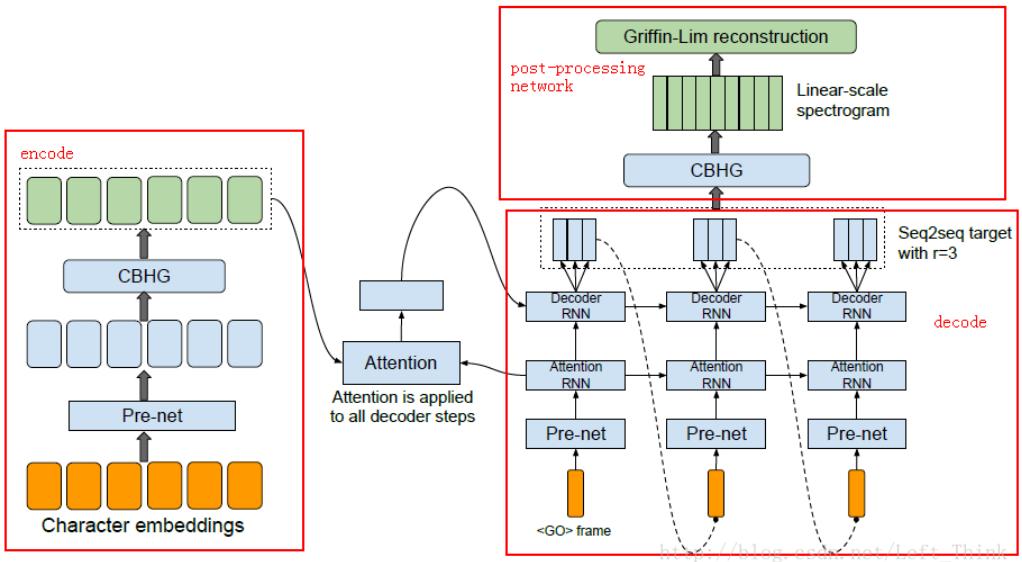


Figure 2.4: Structure of Tacotron

However, Tacotron, as showed in figure2.4 adapted the Griffin-Lim algorithm [34] for phase estimation which is vocoded from the results of the S2S network. As described above, WaveNet has been proved to produce higher audio quality than Griffin-Lim algorithm and even characteristic artifacts, in which case, Tacotron2 was made to combine the advantages of Tacotron and WaveNet with a entirely neural approach to speech synthesis based on architecture of Tacotron[114] S2S model and vocoder of WaveNet [106] which was demostrated in figure2.5.

2.1.3 DeepVoice3

In recent years, Baidu offered several kinds of architectures of networks to solve TTS problem. Starting from Deep Voice 1[6], Deep Voice 2[7] and till now the Deep Voice 3 [83] has shown the highest performance. Deep Voice 1 and 2 kept the traditional structure of TTS pipelines such as separating grapheme-to-phoneme conversion at first, then perform duration and frequency prediction, and the last step is to produce waveform synthesis. Compared to Deep Voice 1 and Deep Voice 2, Deep Voice 3 employs an attention based sequence-to-sequence model, which is similar to the architecture of Tacotron[114], yielding a more compact architecture. However, Deep Voice 3 avoided Recurrent Neural Networks (RNNs) to speed up

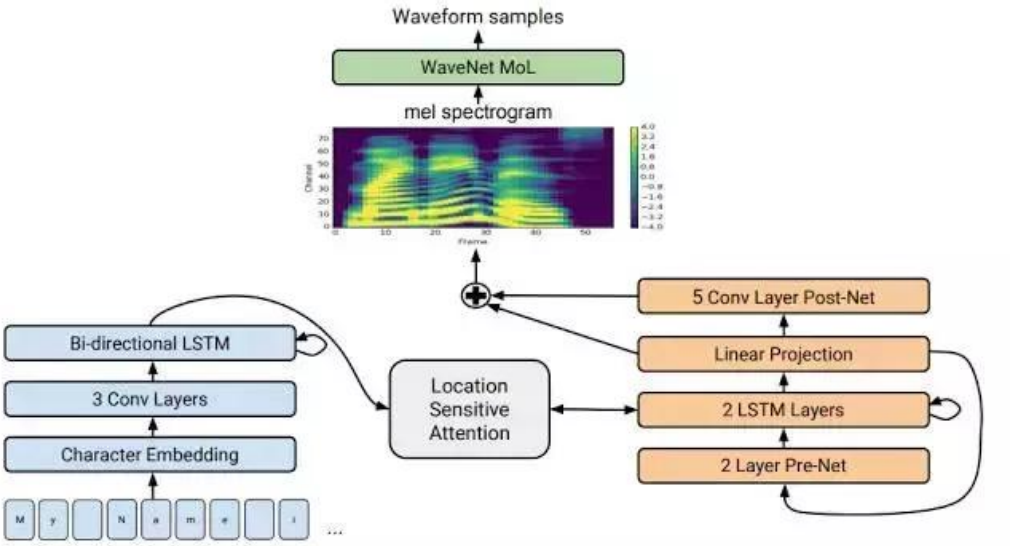


Figure 2.5: Structure of Tacotron2

the whole training process, which also makes attention-based TTS feasible for a production TTS system with no compromise on accuracy by avoiding common attention errors.

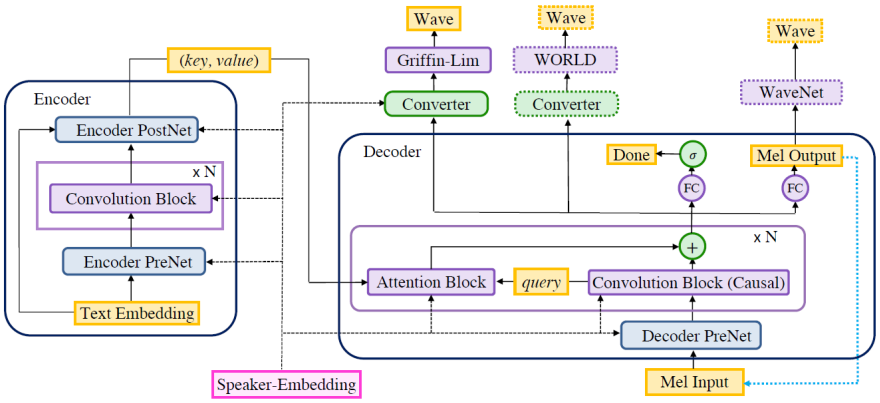


Figure 2.6: Structure of DeepVoice3 Networks

As shown in the figure2.6, the whole network contains 3 parts:

- Encoder: Here the encoder is a kind of fully-convolutional network, which converts textual features to an internal learned representation.
- Decoder: This is another kind of fully-convolutional causal network, which decodes the learned representation from encoder with a convolutional atten-

tion mechanism and then transformed into a low-dimensional audio representation in an autoregressive manner.

- Converter: Converter is a fully-convolutional network as well with a post-processing structure, which is used to do prediction about final vocoder parameters (depending on the vocoder choice) from the decoder hidden states. However, unlike the decoder, converter is a non-causal network so as to depend on future context information.

2.1.4 Summary

The overall aim of this research focused project is to use deep learning technology to learn hand-writing text features, and then produce corresponding audios frame-by-frame, and then try to compress the whole model so as to speed up the model on training and inferencing, in which case it can run with a simpler device such as mobile phone and so one. On this condition, it means that the text features can be used to estimate relative audio features. One approach to do this is reproducing state-of-the-art Text-to-Speech models, which were trained in end-to-end style to make TTS translation, which will aid greatly to develop basic deep learning skills.

Table 2.1: Comparison The Performance of TTS Models

Model Name	MOS(Mean Opinion Score)
WaveNet	4.341 ± 0.051
Tactron	4.001 ± 0.087
Tactron2	4.341 ± 0.051
DeepVoice2	3.69 ± 0.23
DeepVoice3	4.331 ± 0.051
Ground truth	4.582 ± 0.053

According to the table2.1 shown above, DeepVoice3 and Tactron2 will be the most candidates to be reproduced and optimized. Considering the useful materials and code available online, Tacotron2 illustrated a better performance and

a simpler architecture which can be easier to be reproduced and improved than DeepVoice3. As a result, we pay the main attention on Tactron2.

2.2 Convolution Neural Network

Convolution Neural Network is used mainly in the encoder system in the S2S model to play the role of N-grams which can make correlation between each characters. A Convolution Neural Network consists of many neurons that have learnable weights and biases. Each neuron receives some inputs from the input or last layer, runs a dot product which takes two sequences of numbers with equal-length (usually coordinate vectors) and returns a single number as the result [32], and optionally follows it with a non-linearity whose change of the output is not proportional to the change of the input [28]. CNNs express a single differentiable score function: they receive the input of raw data and output scores. A loss function (SVM/Softmax[64]) on the full-connected layer will be used and all the optimization approaches for normal Neural Networks still apply [48] such as regularization[110] on weights and batch normalization[45].

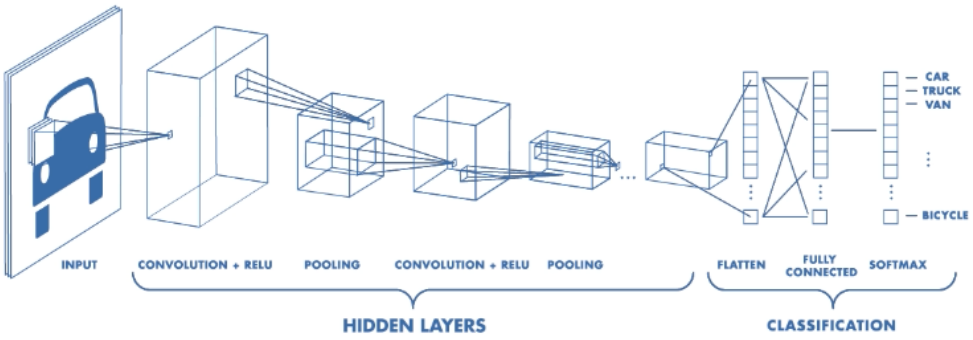


Figure 2.7: Convolutional Neural Networks

2.3 Basic Recurrent Neural Network

RNN will be the main network to train the whole model of Tacotron2. A recurrent neural network (RNN) is a type of artificial neural network, and the connections between the nodes make a directed graph along a sequence, which presents the temporal dynamic behavior related to a time sequence[112]. Unlike feedforward neural networks such as CNN(Convolutional Neural Networks), the blocks inside RNN can be seen as memory so as to process sequences of inputs. Under these conditions, RNN can be applicable to some tasks such as vision problems like unsegmented, connected handwriting recognition[13] or natural language process problems like speech recognition.[90][62]

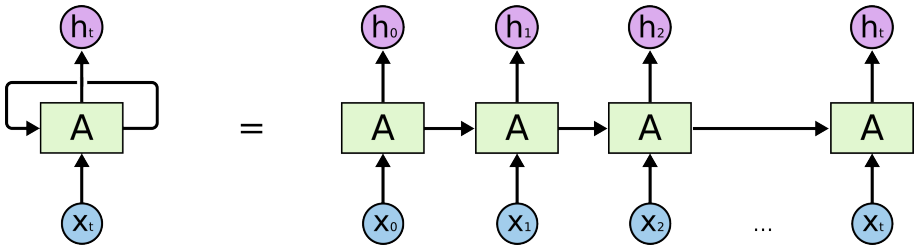


Figure 2.8: Recurrent Neural Networks

There are two broad classes of RNN with a similar general structure, which one is finite impulse, the other is a directed acyclic graph that can be unrolled and replaced with a strictly feedforward neural network, and the other is infinite impulse, which is a directed cyclic graph that can not be unrolled. However, these two classes of networks present a temporal dynamic behavior.[73] Additionally, both of these two RNN can have extra storage state which can be under control of networks as well, and the storage can be replaced by another networks and graph. These types of storage state are also referred as gate state or gated memory.[42]

The structure of basic RNN is a network of neuron-like nodes linked together to be organized into several successive layers, each node in a given layer is connected to all the other nodes in the next successive layer with a fixed connection.[72] The basic functionality of these neurons are:

- Each neuron has an activation function to deal with time-varying data.

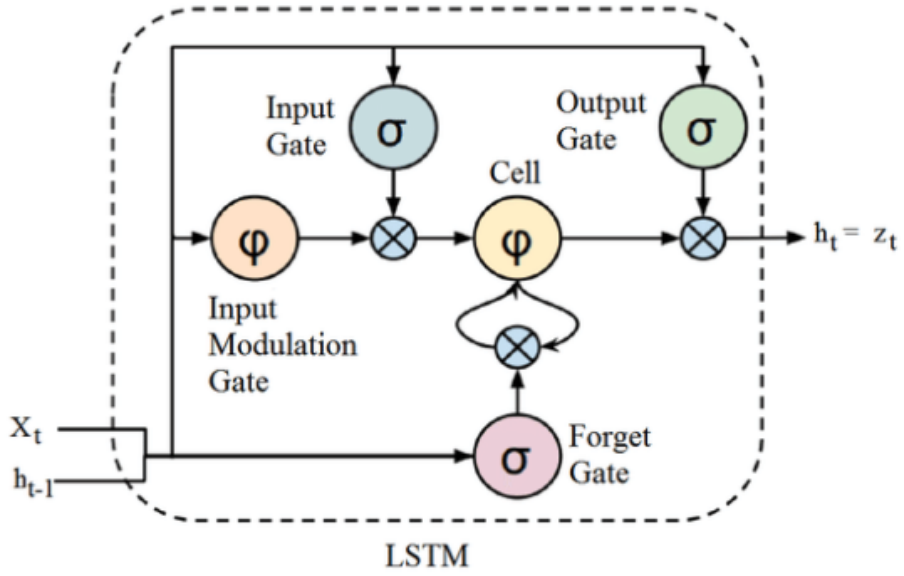


Figure 2.9: LSTM Gate Diagram

- Each connection has a modifiable real-valued weight.
- Nodes are either input nodes which can receive data from outside the network, output nodes which will give the final results, or hidden nodes to modify the processing data.

With the setting of discrete time, each vector will arrive at the input nodes in sequence at a time. At each time step, one unit computes the result with activation function then to sum all of these units results which connect to the specific one neuron. At the same time, for each data training, a target activation need to be supplied for the output unit. For example, a speech classification problem, if the input sequence is a speech signal vectors corresponding to some spoken digit, the output label will be the id number of some speaker.

The total error of the final result is the sum of the errors computed by all individual sequences from the input layer, where each sequence makes an error as the sum of the deviations of all target signals.

2.4 LSTM(Long Short Memory Machine)

2.4.1 General Description

Long Short Term Memory networks(LSTM) are a special RNN, which is capable of learning long-term dependencies. LSTM were introduced in [42], and were improved and popularized by many scholar in recent 10 years. They work well on a large variety of problems such as machine translation and audio classification, and are widely used especially in natural language processing at present.

The original purpose of LSTM are explicitly to avoid the long-term dependency problem, which is to remember information for long periods of time. Basically most of recurrent neural networks have the form of a chain of repeating modules of neuron sequences. In basic RNN, these repeating modules have a simple structure, such as a single tanh layer which is described above. LSTM also have the similar chain structure like RNN do, but with a different structure of repeating modules inside. Instead of only a single neural network layer, there are four different gates interacting in a very special way, which are Update gate, Forget Gate and Output Gate accordingly, which are showed in figure2.10.

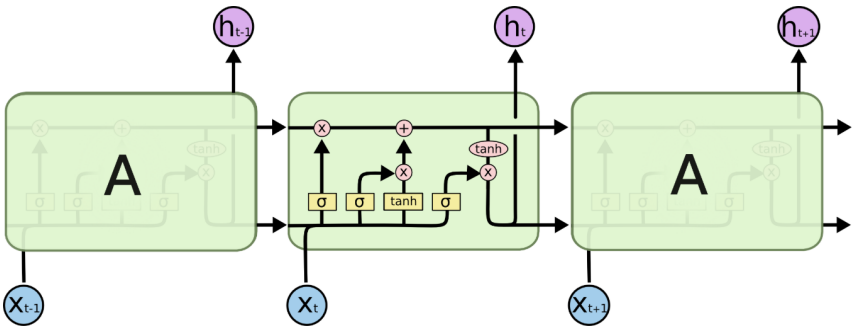


Figure 2.10: Repeating Module in an LSTM

2.4.2 The Core Idea Behind LSTM

The key part of LSTM is the cell state, which is the black horizontal line going through the top of the diagram showed in 2.11. The cell state can be seen as a conveyor belt, where only with some minor linear interactions, it runs straight

down the entire chain. Information can just flow along the cell state with data unchanged.

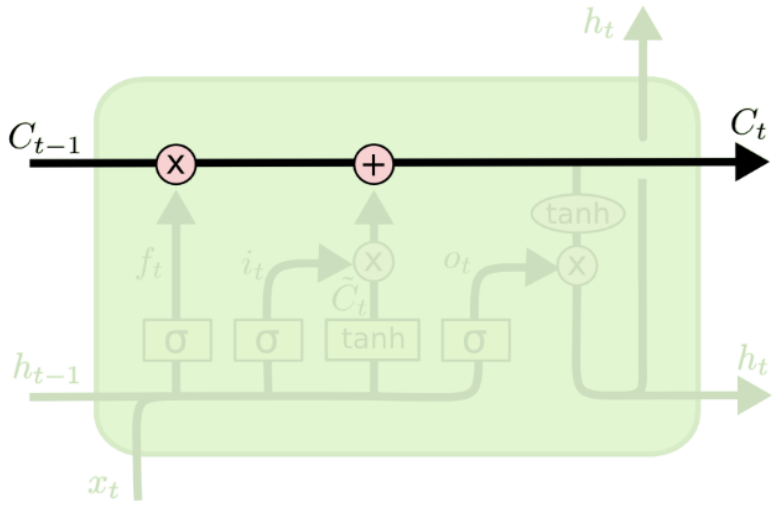


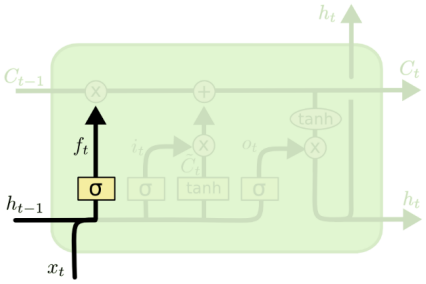
Figure 2.11: LSTM Cell States

LSTM can remove or add information to the cell states by structures called gates carefully. They are composed of a sigmoid layer and a pointwise multiplication operation to compute data in. The sigmoid layer outputs a number between zero and one, deciding the percentage of each information should be let go through. Zero means nothing through while a value of one means let everything go through. These three gates are used to protect and control the cell state.

2.4.3 Step-by-Step LSTM Walk Through

The first step in LSTM is to decide which data should run through the cell state. The decision is made by a sigmoid layer called Forget Gate Layer. It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each state in the cell state of C_{t1}. 1 represents completely keep the information while 0 represents completely drop the information.

The second step is to decide what new data should be stored in the cell state for further used in future. This has two parts: Firstly, a sigmoid layer called update gate layer decides which values to update. Secondly, a tanh layer creates a

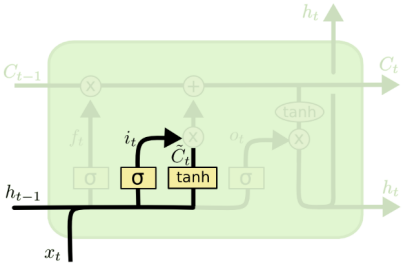


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figure 2.12: LSTM Forget Gate

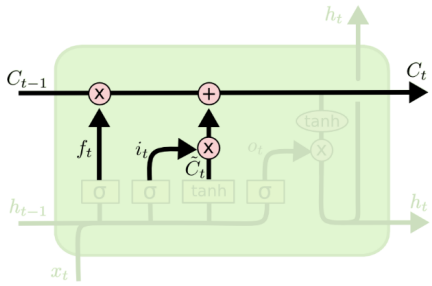
vector of new candidate values C_t that could be added to the state. At last step, combining these two values to create an new data to cell state with updating old cell state of C_{t1} into the new cell state C_t.

The old state then be multiplied by f_t, forgetting the data decided to forget earlier. Then adding $i_t C_t$. This is the new candidate values which scaled by how much to update each state value.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figure 2.13: LSTM Update Gate: Step 1

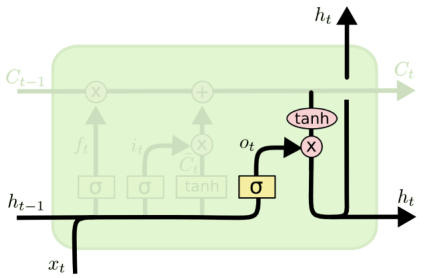


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 2.14: LSTM Update Gate: Step 2

Finally, network need to decide what need to be output according to the data.

This output will be in filtered version based on cell state. First, the sigmoid layer decides what parts of the cell state to output, then putting the cell state through tanh where to rescale the values between 1 and 1 and multiply it to the output of sigmoid gate, so that to only output the parts the netwrok decided to.

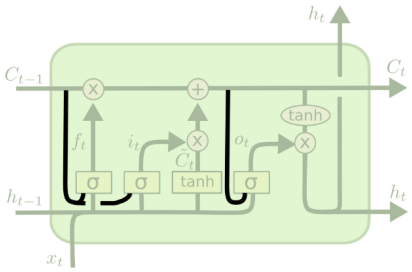


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 2.15: LSTM Output Gate

2.4.4 LSTM Variant

What have been described so far is a the normal structure of LSTM. However, not all LSTMs are the same as the one shown above. One popular LSTM variant, introduced by [42], is adding "peephole connections" to let the gate layers look at the cell state.



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t]\ +\ b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t]\ +\ b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t]\ +\ b_o\right)$$

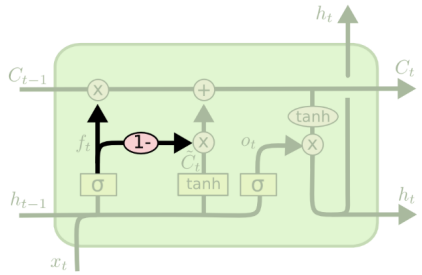
Figure 2.16: LSTM Peephole Connections

As shown in above diagrams, all the gates are added Peephole Connections.

Another variation is to use several forget and input gates. Instead of separately deciding what to forget and what should be added onto the new information, the decision is made together. Two rules need to be fulfilled:

- only forget when we're going to input something in its place
- only input new values to the state when we forget something older

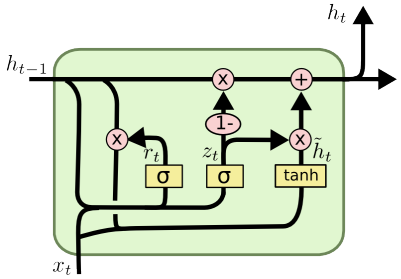
The rules described are shown in the following figure2.17:



$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$

Figure 2.17: LSTM Forget Gate to Delete Old Memories

Another useful and popular variation on the LSTM is the Gated Recurrent Units(GRU), introduced by [19]. GRU combine the forget and input gates into a single update gate along with merging the cell state and hidden state together, and then make some other changes. GRU model is simpler than standard LSTM models which results in more usages in recent years.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 2.18: GRU Networks

There are lots of others useful and popular variants such as Depth Gated RNN in [117]. Some different approaches to tackling long-term dependencies like Clockwork RNN by [55] is used in some models as well. [99] made a comparison of these variants and found that these variants share the similar performance. [47] also tested about ten thousand RNN architectures and found some that worked better than LSTM but on certain tasks.

2.5 Sequence-to-sequence Model

Despite the flexibility and power of traditional deep neural networks, they are useful only to solve problems with their inputs and targets can be encoded in fixed dimensionality of vectors. This is a significant limitation due to that many important problems are necessary to be expressed in sequences whose lengths are not known in advance. For example, speech recognition and machine translation are traditional sequential problems.[101] Additionally, the problem of question answering can also be assumed as to map the question to the answer, where both of them are in a sequence of words representing. This is the reason why a domain-independent method which can learn to map sequences to sequences would be useful. However, sequences give another challenge that they need the dimensionality of the inputs and outputs be known and fixed.

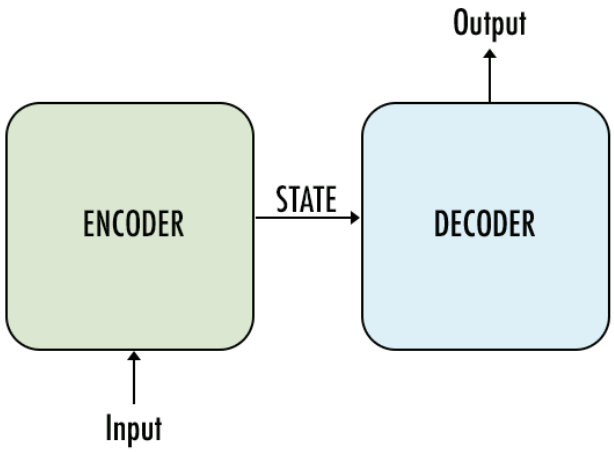


Figure 2.19: Basic Sequence to Sequence Model

A typical sequence to sequence model has two parts: an encoder and a decoder showed in figure2.19. Both of these parts are two different models such as RNN and LSTM which are combined to form one giant network. The function of an encoder network is to understand the input sequence, and transform the input data into a smaller dimensional representation. This representation is then forwarded to a decoder network which generates a sequence of its own that represents the output. Under this condition, the size of the input and output can be ignored due

to that the encoder will produce a fixed length of output features for decoder and then be sent into decoder network which will produce result conditioned on the previous output states, which is described in the following equation:

$$P(Y^1, Y^2, Y^3, ...Y^t) = P(Y^1) * P(Y^2|Y^1) * P(Y^3|Y^2, Y^1) * ... * P(Y^t|Y^1, ..., Y^{t-1}) \tag{2.5.1}$$

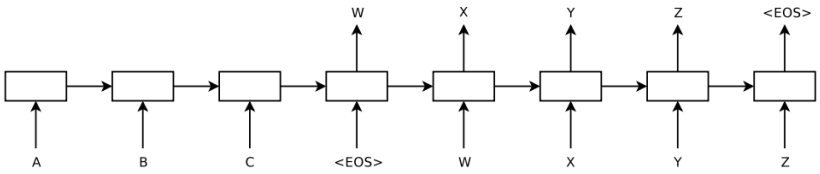


Figure 2.20: Sequence to Sequence Model in Translation Problem

The model in figure2.20 is from [101] which reads an input sentence of "ABC" and then produces then corresponding string of "WXYZ" as the output sentence. While outputting the end-of-sentence token, the model stops making predictions shortly. Note that the input sentence is read by LSTM in reverse, this results from that doing so introduces many short term dependencies in the data, which makes the optimization much easier.

2.6 Attention Mechanism

2.6.1 Functionality of Attention

An attention mechanism can avoid encoding the full source sentence into a fixed-length vector. Rather, decoder attends to different parts of the source sentence at each step of the output generation. Importantly, what the data training process need to do is to let the model learn what part of the input features to attend based on the sentence input before and what it has produced so far. Consequently, for some languages that are pretty well aligned such as English and German, decoder prefer to choose to attend to things sequentially. That is what was done in Neural Machine Translation by Jointly Learning to Align and translate. This is shown in figure2.21.

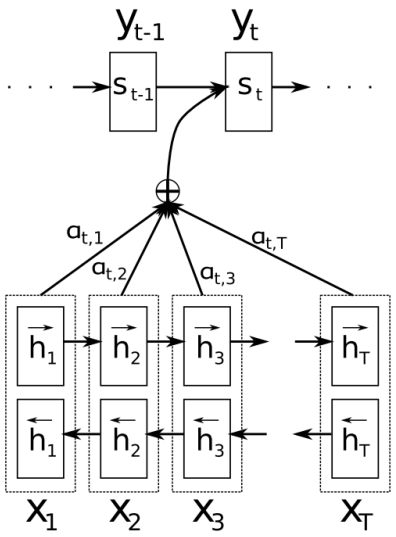


Figure 2.21: Attention Used

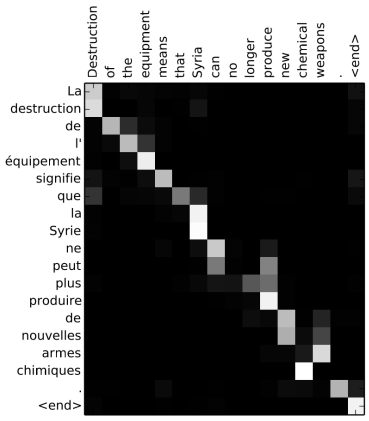


Figure 2.22: Attention Weight Matrix

An advantage of attention is that it can offer us the good performance of interpreting and visualize what the model has done after training. For example, by visualizing the attention weight matrix when a sentence is translated, we can understand how the model is translating:

It can be seen that while translating from French to English, the network attends sequentially to each input state, but sometimes it attends two words at a time while producing an output, for example, “la Syrie” to “Syria” in the figure2.22.

2.6.2 Types of Attentions

For the purpose of the decoder mapping each decoding step to an input token, some context vector need to be computed, which is also called attention vector at each decoding step denoted as c_i at step i. In the original Attention introduced by [10], the context vector c_i is computed as weighted sum of the output from encoder annotated as h_j. Mathematically speaking:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{2.6.2}$$

where α_{ij} is called alignments or attention weights, which are need to be tuned during the training process. From the above equation2.6.2, it can be noticed that the context vector consists of all encoder outputs to determine which is the most important one. Several mechanisms such as soft attention [116], hard attention [4] and monotonic attention[3] are described as useful for translation problems. In this documentation only soft attention will be covered.

As suggested by name, soft alignments comes from softmax layer that being used in the processing of attention weights:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \tag{2.6.3}$$

where e_{ij} is a score usually called energy.

Content Based Attention

The most basic attention model is content-based attention:

$$e_{ij} = score(s_{i-1}, h_j) = {v_a}^T \, tanh(W_a s_{i-1} + U_a h_j) \tag{2.6.4}$$

s_{i-1} is the hidden outputs inside decoder network from previous time step, h_j is hidden output of encoder, and j, W_a, U_a and v_a are the weights matrices which need to be learned. Due to that U_a and h_j is independent from the decoding step i, it needs to be computed in advance.

Global attention has been proved to offer the ability to link different outputs with the corresponding input tokens with ignoring their location[69].

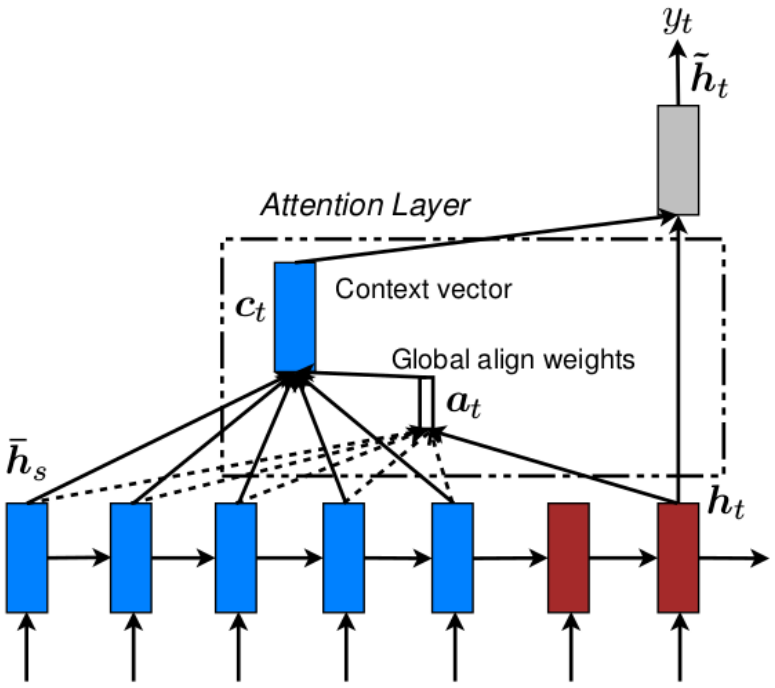


Figure 2.23: Content Based Attention Network

Location Based Attention

Local based attention was first introduced in [30]. The difference between location-based attention and content-based attention is the way how the attention network compute scoring of e_{ij}:

$$e_{ij} = score(\alpha_{i-1}, h_j) = {v_a}^T \, tanh(W h_j + U f_{i,j}) \tag{2.6.5}$$

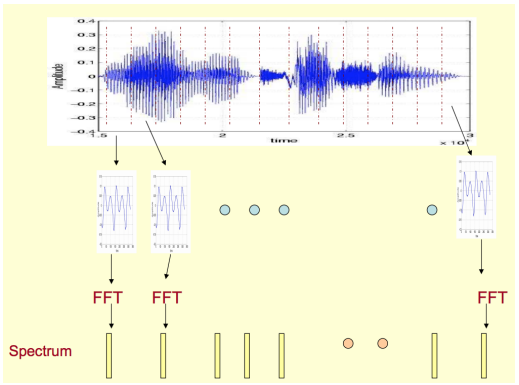
where $f_{i,j}$ is location feature computed by previous alignments α_{i-1} with convolution filters F, which means the features of previous alignments need to go through a convolution layer. The rest parameters, v_a, W, U and F, are weights to be learned.

Location-based attention ignores the actual content of the input and only pay attention to the locations and distances between these input tokens.

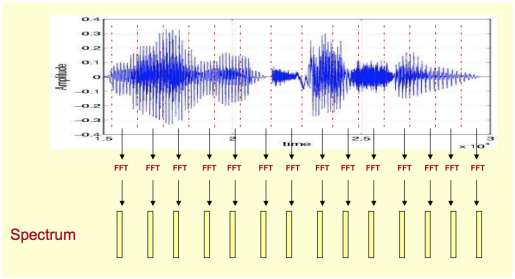
2.7 Mel-Spectrogram

A spectrogram is a kind of representation to demonstrate the spectrum of frequencies of sound in version style because of their attributes of varying with time. Spectrograms also have some other names such as sonographs, voiceprints and voicegrams[31]. If the data is in the form of a 3D plot then they have the name of waterfalls[36]. Spectrograms are widely used in the several domains such as music, sonar, radar and speech processing[26]. Spectrograms of audio can be applied to kinds of fields like identifying spoken words phonetically[50][84][21] and analyzing the various calls of animals[70][111].

The process of transforming the normal speech signal to spectrogram is as shown below: Firstly, slice the whole speech signal into several smaller chunk of voice signal, then do Fast Fourier Transform(FFT)[107] to samples these signals over a period of time or space and divides it into its frequency components.[9] After doing all of the original signals, we can have the spectrum of transformed frequency diagram for each signal segment.



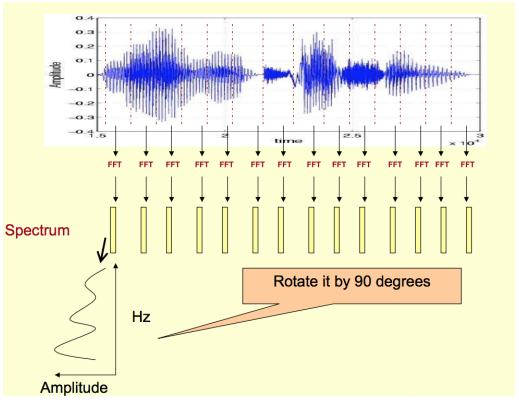
(a) Step 1: Slicing



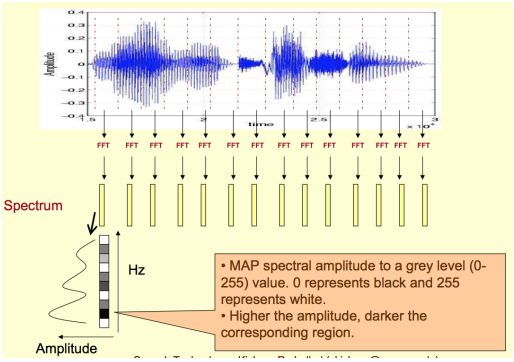
(b) Step 2: FFT All Segments

Secondly, rotating each frequency diagram by 90 degrees and then mapping spectral amplitude to a grey level between 0 and 255 value. Here 0 represents black and 255 represents white, and higher the amplitude, darker the corresponding region.

Finally, after transforming all the signal segments we achieve the diagram with the x-axis of Time and y-axis of Frequency, which is the representation of a speech signal, referred as spectrogram, the Dark regions indicate peaks (formants) in the

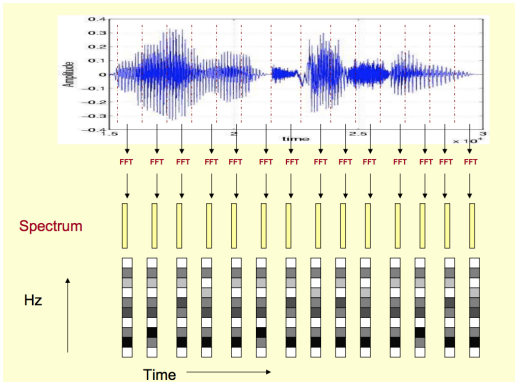


(a) Step 3: Rotate Spectrum

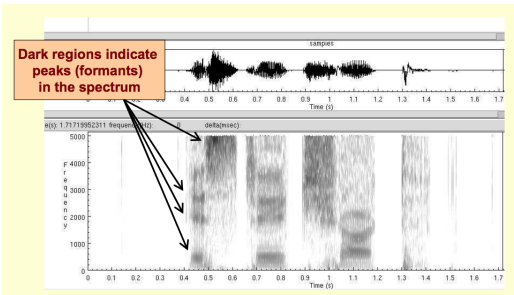


(b) Step 4: Map to Gray Level

spectrum. In the diagram of spectrogram, phones and their properties are better observed. Additionally, sounds can be identified much better by the formats and transitions.



(a) Step 5: Transform All



(b) Step 6: Analysis of Result

Mel-spectrogram is the normal spectrogram being mapped to mel-scale[65] from hertz with the equation:

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{2.7.6}$$

where f is the Hertz-scale and m is the mel-scale.

A high quality text to speech system should produce synthesized speech whose spectrogram need to nearly match with the natural sentences[85]. At the same time, mel frequency tries to approximates the mapping of frequencies to patches of nerves in the cochlea[79], under which condition, it has relative importance of different sounds to humans hearing. As a result, binning a spectrum into approximately mel frequency spacing widths will make the spectral information produced

by networks act the same way as human hearing, and this is the reason why mel-spectrogram is the most suitable choice to be the intermediate data for a TTS system.

As shown in figure2.27, a speech signal is transformed into mel-spectrogram where the x-axis is in frequency and y-axis is in time.

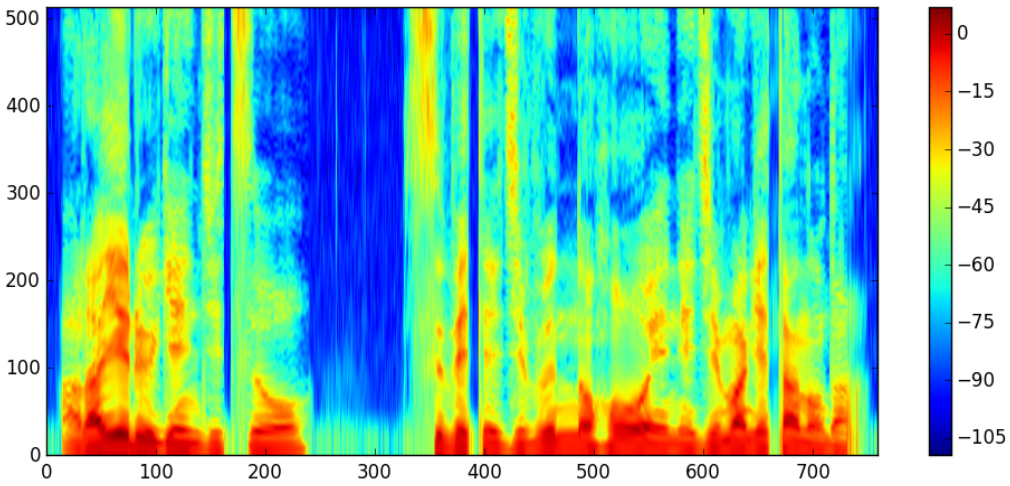


Figure 2.27: An Example of Mel-Spectrogram Diagram

Chapter 3

Project Design

3.1 Overall General Design

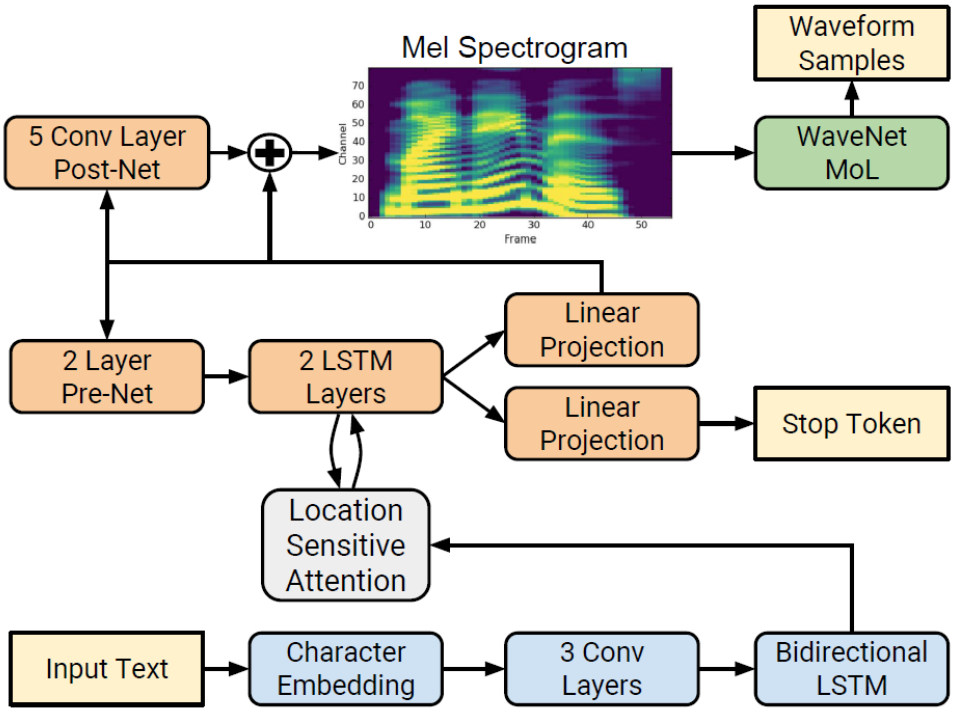


Figure 3.1: Data Flow Diagram[95]

As shown in figure 3.1, the whole structure is designed to implement an end-to-end model to make the input as text, and output as according waveform. The

whole structure will include four components which are

1. The encoder part of a recurrent Sequence-to-sequence network used to encode the text features and processed to output a fixed-length of vectors to send to attention model.

2. The decoder part of a recurrent Sequence-to-sequence network used to decode the processed output of the encoder and attention model then output mel-spectrogram.

3. A location attention model is posited between the encoder and decoder networks to deal with the ouput of encoder networks so as to improve the performance of the Tacotron2 network.

4. WaveNet which has been modified with a faster and smaller architecture is used to generate time-domain waveform samples from the output of sequence-to-sequence network, which is mel-spectrogram.

3.2 System Components

3.2.1 Encoder

Structure Description

Table 3.1: Architecture of Encoder

Layer	Size	layer
Character Embedding	512-dimension	1
Convolutional Layers	512-filters	3
Bidirectional LSTM	512-units	1

The input of characters sequence will be converted into a hidden feature representation by encoder. Here the input character are represented by a learning character embedding with size of 512-dimension, then it is passed into a 3 stacked

convolution networks each containing 512 filters with shape of 5x1. Batch Normalization [45] and ReLU[2] function is applied to speed up the training process. The output of the final layer is passed into another structure of network called bi-directional[93] LSTM[42] which has 512 units to produce features in need. The whole structure are listed in table3.1.

Function Details

An input sentence $X = [X_1, X_2, ..., X_{T_x}]$ is mapped to some encoder hidden states which is a series of annotations as $H = [H_1, H_2, ..., H_{T_x}]$. Note that both of the input sequence and the hidden states have the exact same length, which means that the encoder outputs a series of annotations, where each annotation H_i contains information about the corresponding X_i token in input sequence, where the information is conditioned on its neighboring tokens.

In Tacotron2, each input token X_i is encoded by one character of the text. Tacotron2 encoder consists of 3 stack of Convolutional Networks followed by a bi-directional LSTM Network. Similarly to image process where convolution networks are useful to extract local correlations between each pixels, CNN can be applied to character embedding matrix to find correlations between characters. Using CNN layers gives Tacotron2 a long-term independent context, which is similar to N-grams [15]. For example, given two words, Floor and Flour, though they have the same first characters, their pronunciations vary a lot, due to the reason that humans take more consideration on the pronunciation of u in 'Flour'. However, using RNN can be enough to capture described details above since RNN networks have been proved effective in capturing time correlation among text data. The reason here to keep CNN layers is that for short range correlations, RNN performs usefully but poorly in long term dependency. In addition, convolutional layers can make model robust in the situation of some words with silent characters such as 'k' in 'know' and 'd' in 'django'.

Overall, for capturing the longer-term dependent characters, instead of feeding embedded characters into RNN directly, CNN should be applied firstly to extract N-grams and then fed to a RNN. As a result, Tacotron2 encoder should perform

effectively in long-term dependency characters.

In a mathematical point of view, below are the necessary equations where input sequence noted as X and the corresponding encoder outputs as H at one time step:

$$X = [X_1, ..., X_{T_x}], X_i \in \mathbb{R}^{K_x}$$

$$H = [H_1, ..., H_{T_x}], H_i \in \mathbb{R}^{K_x}$$

where K_x representing total size of input tokens in whole vocabulary, and T_x is the total length of the input sequence.

For convolutional layers, f_e express the results after convolving embedded characters with consecutive convolution filters F_1, F_2 and F_3, where each is applied with a non-linear ReLU activation function. Here $\overline{E}$ stands for embedding vectors.

$$f_{e,i} = ReLU(F_3 * ReLU(F_2 * ReLU(F_1 * \overline{E}X))) \tag{3.2.1}$$

Then the output features are fed into a bi-directional LSTM block for generating the hidden encoder outputs H:

$$H = EncoderRecurrency(f_e) \tag{3.2.2}$$

Using of bi-directional LSTM makes sure that the model process the input sequence not only from left to right, but also from right to left, meaning that all the past and future information are included under consideration of each input character. In other words, hidden state H_i is made of the summaries of both the preceding following characters around input X_i. Because of the features of RNN which is obliging in denoting recent inputs, H_i pays more attention on the characters around X_i.

A bidirectional LSTM is composed of two independent LSTM, a forward LSTM $\overrightarrow{f}$ reads the input sequence in the order from time step 1 to final step, and a backward RNN $\overleftarrow{f}$ reads these input in reverse. The corresponding outputs are ($\overrightarrow{H_1}, ..., \overrightarrow{H_{T_x}}$) and ($\overleftarrow{H_1}, ..., \overleftarrow{H_{T_x}}$).

The final encoder outputs $H = (H_1, ..., H_{T_x})$ should be the concatenation of the forward and backward hidden states described as followed:

$$H_i = \begin{bmatrix} \overrightarrow{H_i} \\ \overleftarrow{H_i} \end{bmatrix} \tag{3.2.3}$$

3.2.2 Attention Model

Attention network will receive these features to summarize the full encoded sequence to be fixed-length vector which is the same with the size of output of decoder. It should be noted that the attention mechanism [18] used here, which adapted cumulative attention weights to add up the feature from previous decoder steps, extends the traditional one called additive attention mechanism[10]. In this condition, some sub-sequence contained within some potential failure modes are repeated or neglected by decoder so as to promote the network moving forward consistently. After projecting inputs and location features into some hidden representations with 129-dimensional, the attention probabilities will then be calculated.

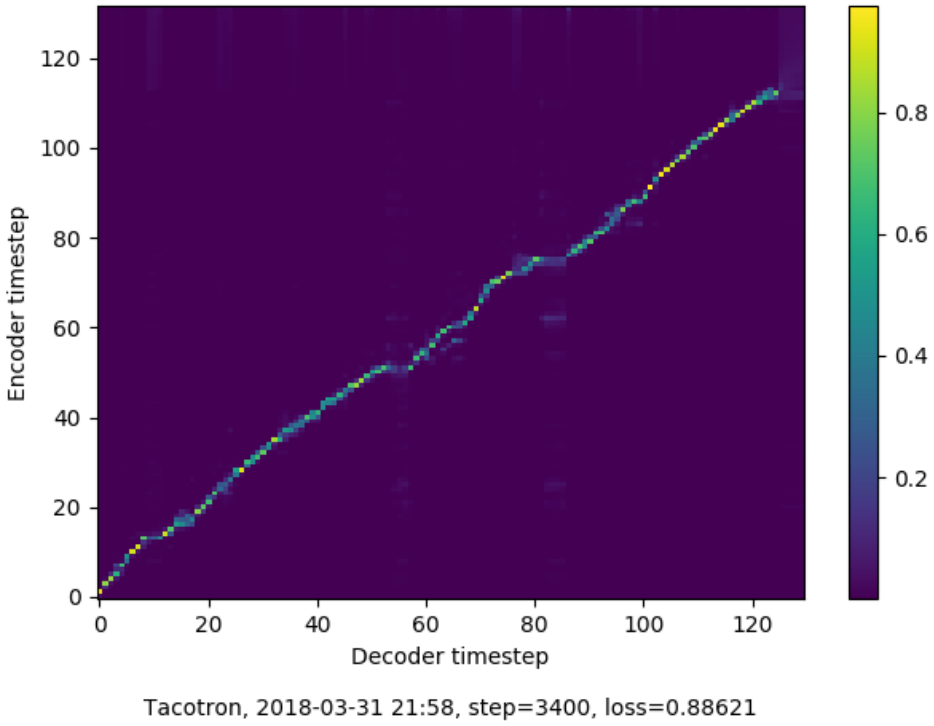


Figure 3.2: Training Example Result of Attention Model[69]

Function Details

Different from the two discussed attention model above, which are content-based attention and location based-attention, Tacotron2 adapted a type of model of hybrid attention, which is, as its name described, the mix of two previous attention models. Here the only difference is still about how to compute the energy:

$$e_{ij} = score(s_{i-1}, \alpha_{i-1}, h_j) = {v_a}^T \, tanh(W s_{i-1} + V h_j + U f_{i,j}) \tag{3.2.4}$$

where s_{i-1} represents hidden states inside decoder, α_{i-1} denotes alignments from previous time step and h_j is the j-th hidden state inside of encoder. W, V, U and v_a are the trainable parameters which are required to be tuned during training process. Furthermore, $f_{i,j}$ are location features:

$$f_i = F * \alpha_{i-1} \tag{3.2.5}$$

Usually, the score computation can be bias free which means no bias needed for liner transformation, however adding biases makes the attention more adjustable:

$$e_{ij} = {v_a}^T \, tanh(W s_{i-1} + b_1 + V h_j + b_2 + U f_{i,j} + b_3) \tag{3.2.6}$$

For reducing the number of model parameter, these 3 bias terms can be combined into a single b vector of bias term, as a result, model training process can learn to adjust value of b corresponding to the summed biases. So the energy computation equation then is:

$$e_{ij} = {v_a}^T \, tanh(W s_{i-1} + V h_j + U f_{i,j} + b) \tag{3.2.7}$$

Both of the content and the location of inputs tokens are considered in hybrid attention model, in other words, hybrid attention adapts both the advantages of previous discussed two attentions and overcome their limitations. The attention described in [95] called Location Sensitive Attention cited from [18], which extends the additive attention described in [10] to adapt cumulative attention weights from previous decoder time steps as an additional feature. The computation of context vector should be similar to all previously discussed attentions, but

the only difference is how to compute the energy. By researching, thinking and training, the computation of attention in Tacotron2 should be as follows:

$$e_{ij} = score(s_i, c\alpha_{i-1}, h_j) = {v_a}^T \, tanh(Ws_i + Vh_j + Uf_{i,j} + b) \tag{3.2.8}$$

where s_i denote the current hidden state instead of previous one's decoded from LSTM networks. The bias b should be initialized to zeros in vector forms for the model to tune during training process if required, and actually it will converge close to 0 after training. Location features are calculated using the cumulative alignments $c\alpha_{i-1}$ with time as follows:

$$f_i = F * c\alpha_{i-1}$$
$$c\alpha_{i-1} = \sum_{j=1}^{i-1} \alpha_j$$

As illustrated in the above equations, alignments cumulation is done additively. The reason why not choosing multiplicative one is that multiplicative cumulation can result in loosing location information which the network attempt to seek: As

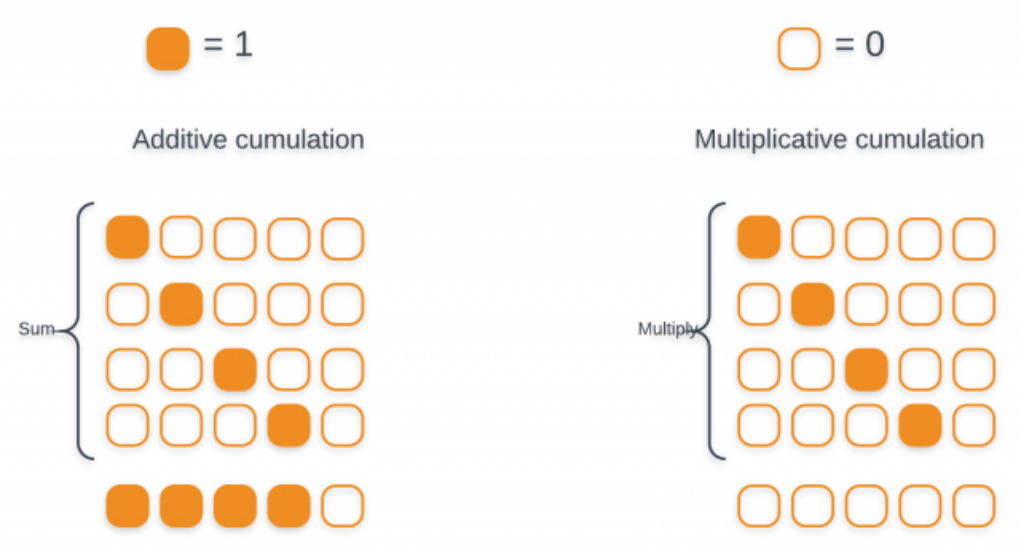


Figure 3.3: Two Different Cumulation Attention[69]

demonstrated above, the multiplicative cumulation can lead to 0 due to possible value of 0 in the process. With adding all previous alignments, the attention

network can find some useful information about input characters which is already attended, then it can constantly keep moving forward in the sequence, at the same time, to avoid being trapped in undesirable voices to repeat that.

Additionally, there are many different attention variants are described in the tutorial of Tensorflow[67].

The scoring function, attention function and whether use the previous state h_{i-1} or h_i in the scoring function as originally suggested in [10] can decide the form of attention variants. Empirically, only some certain choices really matter. Firstly, it is required to show that the basic form of attention which make connections between target and source. Secondly, feeding the attention vector to the next time-step, in order to inform the network some information about past attention decisions made as demonstrated in, is important for the the attention training[68]. After training some basic models, choosing different scoring function can often lead to a different performance.

3.2.3 Decoder

Structure Description

Table 3.2: Architecture of Decoder

Layer	Size	layer
Uni-directional LSTM	512-dimension	2
Pre-Net	256-ReLU-Units	2
Post-Net	512-filters	5
Liner Projection	512-units	1

Decoder will use this information to make prediction about the spectrogram. For the decoder, which is an autoregressive recurrent neural network(RNN)[109], it is used to predict a mel-spectrogram per frame at a time by the sequence of input which is from encoder. Additionally, a small network called PreNet containing 2 fully connected layers of 256 hidden ReLU units will process the prediction at first, then the output is concatenated with the context vector made by attention

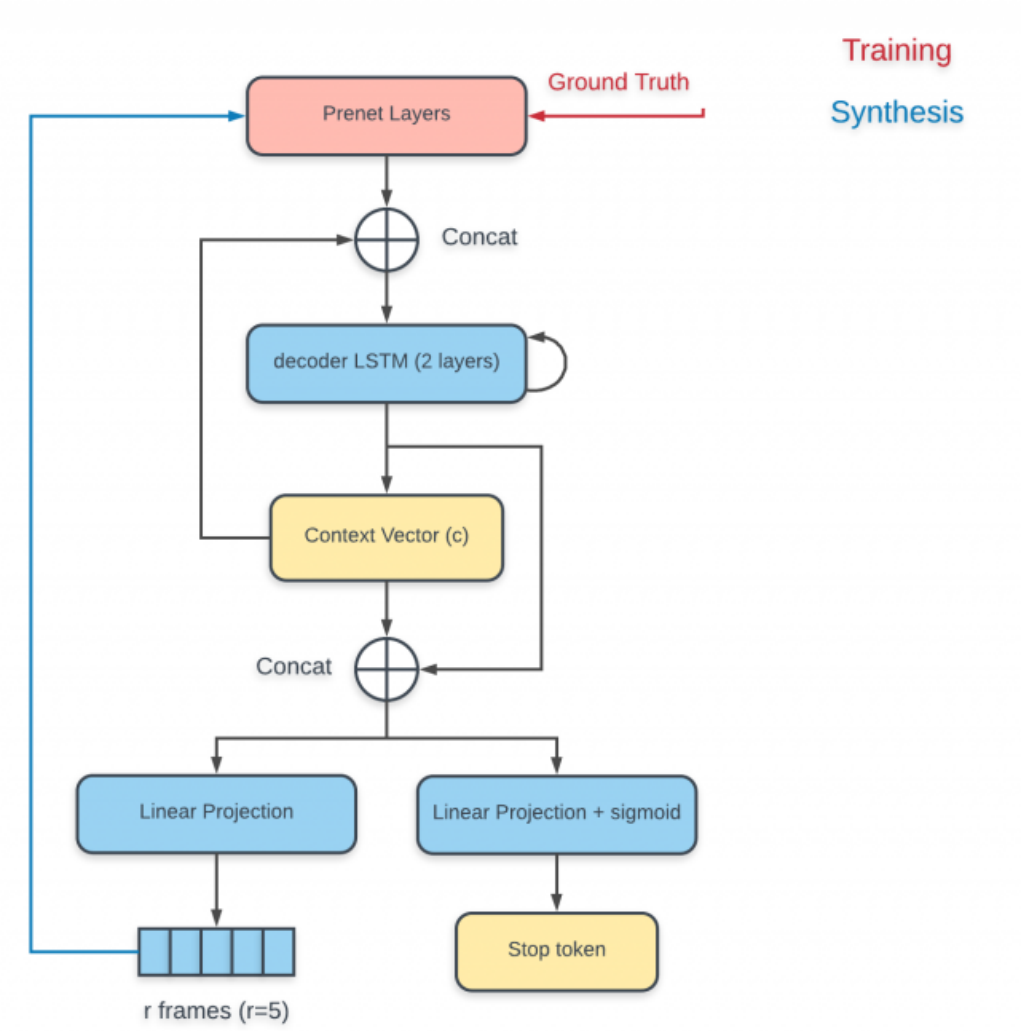


Figure 3.4: Decoder Structure[69]

to feed to a stack of 2 uni-directional LSTM layers. The concatenation of the LSTM output and the context vector from attention net then is projected within a liner transform to make prediction for the spectrogram frame. Finally, the made mel spectrogram will be sent into a 5-layer convolutional post-net which is used to make improvement onto the whole networks with adding to the prediction. Furthermore, each PostNet layer contains 512 filters with shape of 5x1 with batch normalization.

Function Details

The start step of decoder function is fed with the previous output spectrogram frame in synthesis procedure and ground truth of the previous spectrogram frame in training process. PreNet consisting of 2 fully-connected layer processes these input data at first. Then the output from PreNet is concatenated with the context vector which is produced by the previous decoder step. Then the combined new vector data is sent to a 2 layers of LSTM network. The output from LSTM decoder is used as an query vector which is stored for computing the next context data. Finally, the up-to-date computed context vector is concatenated with the output from decoder and then fed to a 512-units liner projection layers to make prediction about spectrogram frames and the probability of stop_token which determines whether the decoding should stop or not.

So, from mathematical view, following series of equations present the i-th step of decoder could do:

$$
\begin{aligned}
py_i &= Prenet(y_{i-1}) = ReLU(W_2\ ReLU(W_1 y_{i-1} + b_1) + b_2) \\
s_i &= LSTM(LSTM(s_{i-1}, py_i, c_{i-1}))(*) \\
y_i &= Linear([s_i; c_i]) = W_p[s_i; c_i] + b_p \\
y_{s,i} &= SigmLinear([s_i; c_i]) = \sigma(W_s[s_i; c_i] + b_s)
\end{aligned}
$$

Some projection layers are widely used in the process of decoders to make projection for the hidden states in LSTM to the output space. In [95], Pre-Net was described as the information bottleneck which is essential for learning attention. In this implementation, the use of a reduction factor was employed so as to typically allow the decoder to predict Mel-spectrogram frames in one decoding step with some reduction factors. This proved to speed up computation and reduce memory usage by [69] and even experienced better alignments with a reduction factor than without one.

3.2.4 WaveNet Vocoder

WaveNet is used to invert the mel-spectrogram features produced by the prediction network stated above into some time-domain waveform samples. However, due to that it is slow and hard to train, which use 3 dilation cycles made from 30 dilated convolution layers, a new version which only adapt 2 upsampling layers in the conditioning stack instead of 3. In addition, PixelCNN++[91] and Parallel WaveNet [76] inspired a new idea with a 10 component mixture of logistic distribution to generate 16-bit samples at 24 kHz. The output of WaveNet is passed to a ReLU activation followed by a linear projection, in which case, to predict parameters for each mixture component. Furthermore, the loss function will be negative log-likelihood of the ground truth sample.

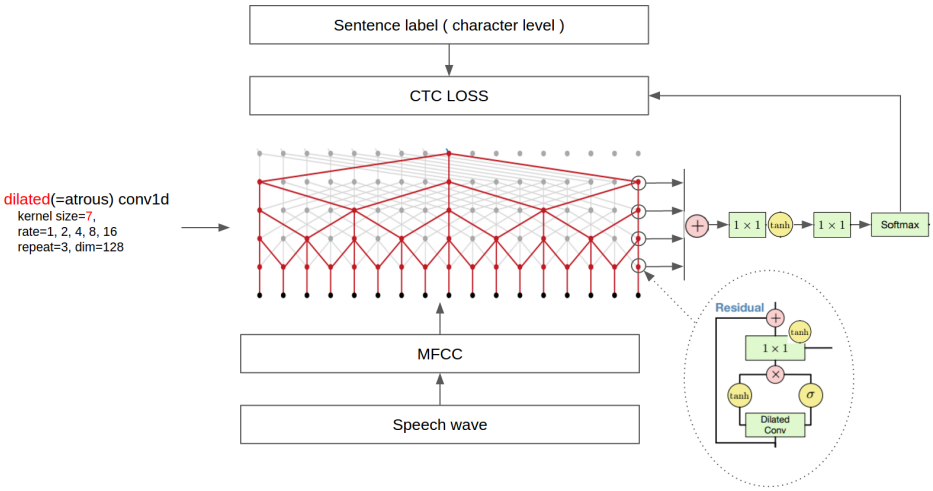


Figure 3.5: Structure of Wavenet[88]

3.3 Intermediate Feature Representation

For this project, a low-level representation called mel-frequency spectrogram[44] is chosen to be the data transformed between the two systems stated above. The reason for choosing this data type is due to that firstly it is easily computed from time-domain waveform, which helps us train the two basic systems separately, and

secondly it is smoother than waveform samples and its loss can be measured by squared error method. Here the mel-frequency spectrogram has some relationship with linear-frequency spectrogram which is also called short-time Fourier transform (STFT).[34] With using a non-linear transform to the frequency axis of the STFT, mel-frequency spectrogram is easy to achieved. The advantages of applying this auditory frequency scale can result in some effect which can be emphasized details in lower frequencies. This is critical to speech intelligibility especially deenohasizing high frequency details. In this situation, features derived from the mel-scale has been the representation of speech recognition [22]. Compared to the data used in WaveNet, mel-spectrogram is a simpler and lower level features, so it can be straightforward to be an input of a similar WaveNet model to generate audio, which has the similar function as vocoder[33].

3.4 Measurement

In this project, a measurement method called Mean Opinion Score(MOS)[86][87][100], which is widely used in the domain of quality of experience and telecommunications engineering[108], is used to measure the performance of reproduced Tacotron2 Model. These ratings can be usually collected in some subjective quality evaluation tests, but can be algorithmically estimated as well[113].

Nowadays, MOS is a commonly used measurement for video, audio, audiovisual quality evaluation and so on. MOS contains a range of 1 to 5 to represent the quality of the audio, where 1 is the lowest and 5 represents the highest perceived quality. However, some other MOS ranges are also useful if the rating scale need to be scaled to a large range based on the test. In this project, the Absolute Category Rating scale is used, which maps ratings between Bad and Excellent to numbers between 1 and 5 which is shown in table3.3.

Within a subjective quality evaluation test, MOS will be computed as the arithmetic mean over single ratings performed by human subjects for a given stimulus.

$$MOS = \frac{\sum_{n=1}^{N} R_n}{N}$$

Table 3.3: Rating scales and mathematical definition

Rating	Label
5	Excellent
4	Good
3	Fair
2	Poor
1	Bad

3.5 Model Implemented in Tensorflow

All the code with training and testing of CNN, LSTM and Fully-connected networks will be written in open source software library called TensorFlow coded in Python. Tensorflow is a scalable distributed training and inference system which is developed from DistBelief [23], the first generation system created by Google Brain project. TensorFlow computations are expressed as stateful dataflow graphs. The name of TensorFlow is derived from the operations which such neural networks perform on multidimensional data arrays [1]. TensorFlow uses a dataflow-like model to do computations and is supported by a wide variety of different hardware platforms, ranging from running inference on mobile device platforms such as Android and iOS to training and inference systems with single computers with one or many GPU cards to large-scale systems running on many specialized machines with thousands of GPUs [1].

Chapter 4

Realization

The training process consists of training feature prediction networks, which is Tacotron2, firstly and independently, followed by the WaveNet training which depends on the output features from Tacotron2. The whole implementation process follows these steps listed below:

1. Acquire audio and text data from Lj-speech [46]
2. Set up TensorFlow[1]
    I Install TensorFlow on Linux
    II Install CUDA on Linux[92]
    III Install cuDNN[17] in library of CUDA
    IV Configure TensorFlow to link to cuDNN
3. Do data pre-processing in TensorFlow
4. Training Tacotron2 model
5. Test Tacotron2 model
6. Training WaveNet model
7. Test WaveNet model
8. Training compressed Tacotron2 Model

9. Test compressed Tacotron2 Model
10. Test compressed Tacotron2 Model with trained WaveNet
11. Evaluation the final results

4.1 Data Preprocessing in TensorFlow

4.1.1 Audio Data Preprocess

For the audio data preprocessing, the raw data are required to be transformed into the form of liner-spectrogram and then to mel-spectrogram which is the same as the output form of tacotron2. Firstly we need to use SFFT(Short Fast Fourier

Table 4.1: Hyper-parameters of Audio Data Preprocessing

Variables	Value	Explanation
num_mels	80	Number of mel-spectrogram channels and local conditioning dimensionality
num_freq	513	Used for adding linear spectrograms post processing network
rescaling_max	0.999	Rescaling value
trim_silence	True	To clip silence in Audio at beginning and end of audio
max_mel_frame	900	Length of clipped frames
silence_threshold	2	silence threshold used for sound trimming for WaveNet preprocessing

Transform) to transform the raw audio data so that all the frequency and time domains are easy to be understood and used by the network. Next, librosa[71] is used to transform the liner-spectrogram into mel-spectrogram.

```
D = _stft(wav, hparams)
#raw audio to liner-spectrogram
S = _amp_to_db(np.abs(D), hparams) - hparams.ref_level_db
```

```
D = _stft(wav, hparams)
#liner to mel-spectrogram
S = _amp_to_db(_linear_to_mel(np.abs(D), hparams), hparams) - hparams.
    ref_level_db
```

Normalization step is important for speeding up the training process. So here clipping and re-scaling the audio data are applied to all the data frames so as to make the training label in the same length and is normalized into symmetric around 0.

```
np.clip((2 * hparams.max_abs_value) * ((S - hparams.min_level_db) / (-
    hparams.min_level_db)) - hparams.max_abs_value
wav = wav / np.abs(wav).max() * hparams.rescaling_max
```

Then we do trim silences step to remove the silence frame in the beginning and end of the waveforms.

```
start, end = audio.start_and_end_indices(out, hparams.silence_threshold
    )
wav = wav[start: end]
out = out[start: end]
```

For clipping all the training data with the same size, zero padding is necessary for quantized the signal

```
out = np.pad(out, (l, r), mode='constant', constant_values=
    constant_values)
```

The last step is writing the spectrogram produced by above stated steps and the raw audio to disk for the basic comporison.

4.1.2 Text Data Preprocess

After finishing the label data production, we need to preprocess the raw input text into the vector which can be identified by networks. Word embedding[61] is the most used method to transform the raw text data into vector data, some famous embedding libraries such as Word2Vec[29] and GloVe[80] which are pre-trained embedding libraries have been applied into several famous fields such as machine translation and speech classification. However, for this project, ARPAbet[11] and

CMU Pronouncing Library[115] are used to translate each character to the pronouncing symbols.

ARPABET is a set of phonetic transcription codes developed by Advanced Research Projects Agency (ARPA) as a part of their Speech Understanding Research project in the 1970s. It represents phonemes and all phones of General American English with distinct sequences of ASCII characters.[96][58][94] CMU Pro-

IPA and ARPAbet Symbols

IPA	ARPAbet	Example	IPA	ARPAbet	Example
ɑ	aa	*b*o*t*	ɨ	ix	d*e*bit
æ	ae	*b*a*t*	i	iy	b*ee*t
ʌ	ah	*b*u*t*	j	jh	*j*oke
ɔ	ao	*b*ough*t*	k	k	*k*ey
aʊ	aw	*b*ou*t*	k̚	kcl	k closure
ə	ax	*a*bout	l	l	*l*ay
əˈ	ax-h	potat*o*	m	m	*m*om
ɚ	axr	butt*er*	n	n	*n*oon
aɪ	ay	b*i*te	ŋ	ng	si*ng*
b	b	*b*ee	ɾ	nx	wi*nn*er
b̚	bcl	b closure	o	ow	b*oa*t
ʧ	ch	*ch*oke	ɔɪ	oy	b*oy*
d	d	*d*ay	p	p	*p*ea
d̚	dcl	d closure	ɒ	pau	pause
ð	dh	*th*en	p̚	pcl	p closure
ɾ	dx	mud*d*y	ʔ	q	a*tl*as
ɛ	eh	b*e*t	ɹ	r	*r*ay
l̩	el	bott*le*	s	s	*s*ea
m̩	em	bott*om*	ʃ	sh	*sh*e
n̩	en	butt*on*	t	t	*t*ea
ŋ	eng	Washi*ng*ton	t̚	tcl	t closure
ɨ̞	epi	epinthetic silence	θ	th	*th*in
ɝ	er	b*ir*d	ʊ	uh	b*oo*k
e	ey	b*ai*t	u	uw	b*oo*t
f	f	*f*in	ʉ	ux	t*oo*t
ɡ	g	*g*ay	ʌ	v	*v*an
ɡ̚	gcl	g closure	w	w	*w*ay
h	hh	*h*ay	j	y	*y*acht
ɦ	hv	a*h*ead	z	z	*z*one
ɪ	ih	b*i*t	ʒ	zh	a*z*ure
-	h#	utterance initial and final silence			

Figure 4.1: AEPABET and IPA Mapping[74]

nouncing Library is composed of about 134,000 words and their corresponding pronunciations. It is an pronunciation dictionary by transforming the word in text to which can be read by machine for North American English. It have been maintained and expanded since 2002. There are 39 phonemes included nowadays, without varia due to lexical stress. This phone set is based on the ARPAbet symbol set described above. Some examples in CMU pronouncing library are listed in below table 4.2. And the whole phone set are listed in table4.3:

Table 4.2: Example of Phoneme Set

Phoneme	Example	Translation
AA	odd	AA D
AE	at	AE T
AE	at	AE T
AH	hut	HH AH T
AH	hut	HH AH T
AO	ought	AO T
AW	cow	K AW
AY	hide	HH AY D
B	be	B IY
CH	cheese	CH IY Z
D	dee	D IY
DH	thee	DH IY
EH	Ed	EH D

Table 4.3: All CMUDict Phoneme in Order

'AA'	'AA0'	'AA1'	'AA2'	'AE'	'AE0'	'AE1'	'AE2'	'AH'	'AH0'	'AH1'	'AH2'
'AO'	'AO0'	'AO1'	'AO2'	'AW'	'AW0'	'AW1'	'AW2'	'AY'	'AY0'	'AY1'	'AY2'
'B'	'CH'	'D'	'DH'	'EH'	'EH0'	'EH1'	'EH2'	'ER'	'ER0'	'ER1'	'ER2'
'EY'	'EY0'	'EY1'	'EY2'	'F'	'G'	'HH'	'IH'	'IH0'	'IH1'	'IH2'	'IY'
'IY0'	'IY1'	'IY2'	'JH'	'K'	'L'	'M'	'N'	'NG'	'OW'	'OW0'	'OW1'
'OW2'	'OY'	'OY0'	'OY1'	'OY2'	'P'	'R'	'S'	'SH'	'T'	'TH'	'UH'
'UH0'	'UH1'	'UH2'	'UW'	'UW0'	'UW1'	'UW2'	'V'	'W'	'Y'	'Z'	'ZH'

According to the dictionary above, we than can start the preprocessing about the text information. Firstly, we need to expand all the abbreviations in text which is shown in table1.2 so as to get the exact readings which can be found in CMU Dictionary. The next step is to change the number in digit form into string form.

```
num = int(m.group(0))
```

```
if num > 1000 and num < 3000:
    if num == 2000:
      return 'two thousand'
    elif num > 2000 and num < 2010:
      return 'two thousand ' + _inflect.number_to_words(num % 100)
    elif num % 100 == 0:
      return _inflect.number_to_words(num // 100) + ' hundred'
    else:
      return _inflect.number_to_words(num, andword='', zero='oh', group
   =2).replace(', ', ' ')
else:
    return _inflect.number_to_words(num, andword='')
```

Additionally, we need to change some special and also meaningful symbols into the corresponding string forms such as '$' to 'dollars'. Furthermore, in order to avoid encoding problems about the text, we need to transform all the texts into ASCII[40] form. The last step is then removing extra space and make all the text characters in lower case.

```
text = convert_to_ascii(text)
text = lowercase(text)
text = collapse_whitespace(text)
```

Since we have achieved the dictionary of CMU Dictionary, what we need to do is just according to the dictionary, translating all the text content into the the corresponding ID numbers to achieve the final input vector, which is demonstrated in figure4.2:

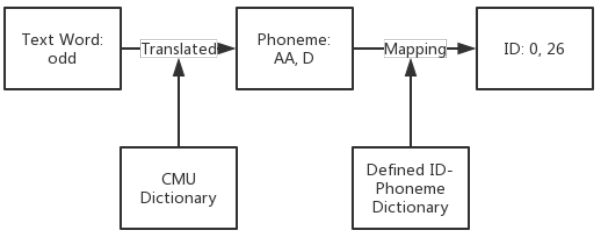


Figure 4.2: Phoneme Translation Process

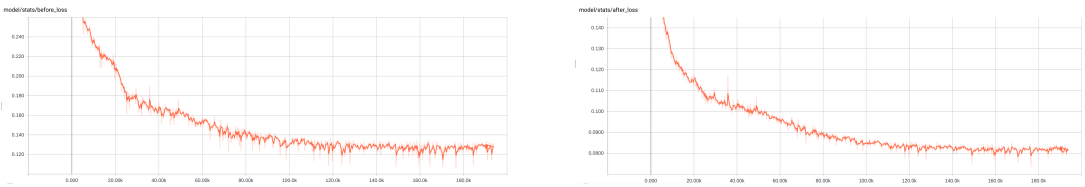
4.2 Training Tacotron Model

Table 4.4: Hyperparameters of Tacotron2 Networks

Variables	Value	Explanation
outputs_per_step	2	number of frames to generate at each decoding step
stop_at_any	True	Determines whether the decoder should stop when predicting <stop> to any frame or to all of them
embedding_dim	512	dimension of embedding space
enc_conv_num_layers	3	number of encoder convolutional layers
enc_conv_kernel_size	(5,)	size of encoder convolution filters for each layer
enc_conv_channels	512	number of encoder convolutions filters for each layer
encoder_lstm_units	256	number of lstm units for each direction (forward and backward)
attention_dim	128	dimension of attention space
attention_filters	32	number of attention convolution filters
attention_kernel	(31,)	kernel size of attention convolution
prenet_layers	[256, 256]	number of layers and number of units of prenet
decoder_layers	2	number of decoder lstm layers
decoder_lstm_units	1024	number of decoder lstm units on each layer
max_iters	900	Max decoder steps during inference
postnet_num_layers	5	number of postnet convolutional layers
postnet_kernel_size	(5,)	size of postnet convolution filters for each layer
postnet_channels	512	number of postnet convolution filters for each layer

Table 4.5: Settings of Tacotron2 Training

summary_interval	250	Steps between running summary ops
checkpoint_interval	5000	Steps between writing checkpoints
eval_interval	10000	Steps between eval on test data
tacotron_train_steps	190000	total number of tacotron training steps
tacotron_batch_size	24	number of training samples on each training steps
tacotron_reg_weight	1e-6	regularization weight of L2 regularization
tacotron_test_batches	24	number of test batches



(a) Tacotron Before Loss (b) Tacotron After Loss

Figure 4.3: Tacotron Training Before and After Loss

The whole training process takes about 3 days to reach 190,000 steps. The value of total loss, calculated by mean squared error(MSE)[25], is consisted of before_loss and after_loss, where before and after means before and after the PostNet. Adam optimizer [52] with $\beta_1 = 0.9$, $\beta_2 = 0.999$,$\epsilon = 10^6$ was adapted, along with a exponentially decaying of 10^3 to 10^5 which begins after 50,000 steps of the learning rate. L2 regularization with 10^6 was applied as well.The change of before and after loss are shown in figure4.3, while the total loss showed in figure 4.4.

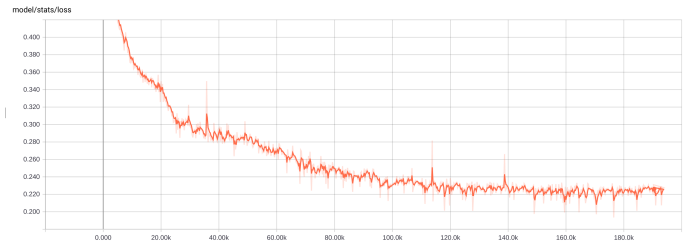
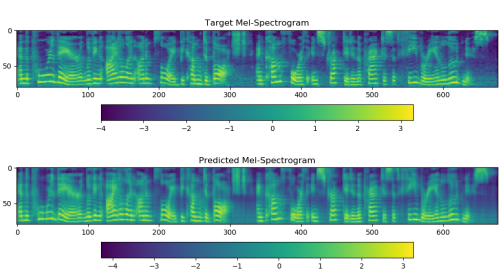


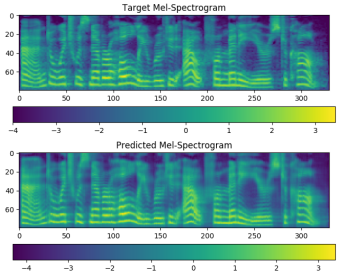
Figure 4.4: Tacotron2 Total Loss

As demonstrated in these figures, the total loss converges quickly for the 100,000 steps, and tendency stayed almost unchanged after 150,000 steps, which stays with the loss about 0.22. This similar for before_loss and the after_loss in figure of before_loss and after_loss, which converge fast for the first 100,000 steps and maintain the value of 0.082 for the after_loss and 0.122 for the before_loss.

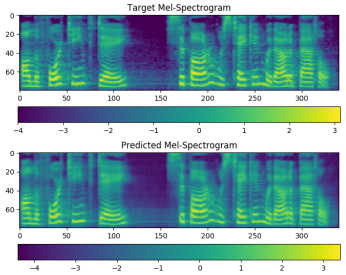
The quality of the produced mel-spectrogram compared to the ground truth based on the training data set are shown in figure 4.5. These are step 5000, 50000, 100000 and 190000 accordingly.



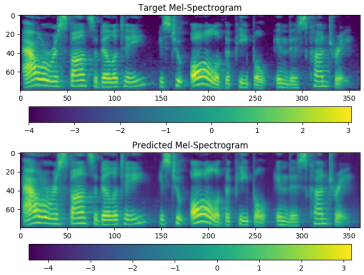
(a) Step 5000



(b) Step 50000



(c) Step 100000

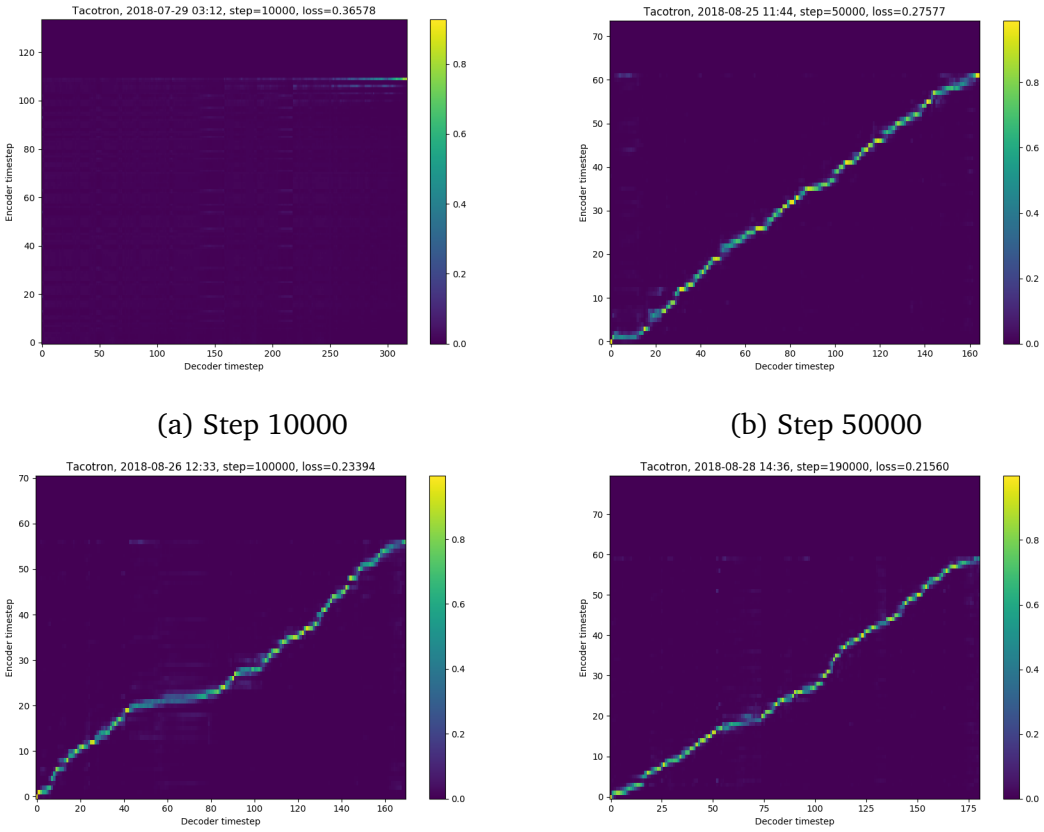


(d) Step 190000

Figure 4.5: Tacotron Training Prediction

From these figures, we can see that the predicted mel-spectrogram is similar to the target one even at the step of 5000, where the loss has declined to about 0.434, which indicated that the features prediction networks performs very effective on the training data even at the start stage of the training process. Furthermore, the

figure of 190,000 shows the network predicted almost the same mel-spectrogram diagram with the real one. Then the attention part of the training aligns are shown



(a) Step 10000

(b) Step 50000

(c) Step 100000

(d) Step 190000

Figure 4.6: Tacotron Training Aligns

in figure4.6. As we can see, for the first attention align diagram at step of 10000, the network did not converge at all since no bright lines come up on the diagram, but it gets more slim, continuous, and bright which means the higher probability of matching between the encoder features and decoder outputs when the situation comes to step of 50,000. The last step of 190,000 give us the best result with a pretty shinning and slime lines on the diagram which presents the attention model converges effectively.

The next training step is the stop_token loss which is produced by concatenation of decoder LSTM output. The results of the attention context is then projected to a scalar, which is passed through a sigmoid activation function to predict the probability whether to stop outputting sequence or not. If not stopping, the last

output would be seen as the next input and pass through the state. The network would stop when the model outputs the probability of p larger than 0.5 for all frames between normalized frame.

```
if self.stop_at_any:
  finished = tf.reduce_any(finished) #Recommended
else:
  finished = tf.reduce_all(finished) #Safer option
```

The stop_token loss is present in figure4.7

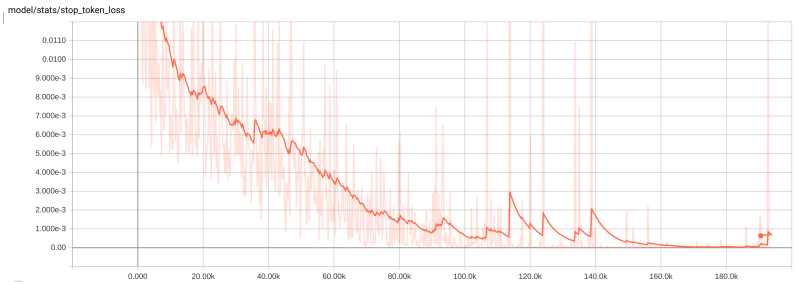


Figure 4.7: Tacotron Training StopToken Loss

4.3 WaveNet Training

WaveNet training is much more time-consuming which took about one month to train and the whole steps reached about 1.16 million so to make the final result get better. As shown in figure4.8, the whole loss declines moderately from about 9 to around 6, with the steps reached about 1.165 million steps. The prediction

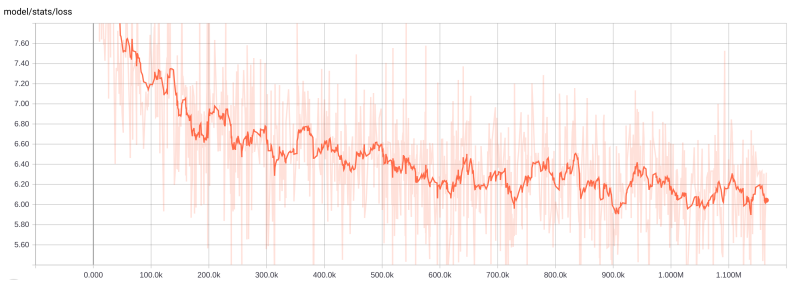
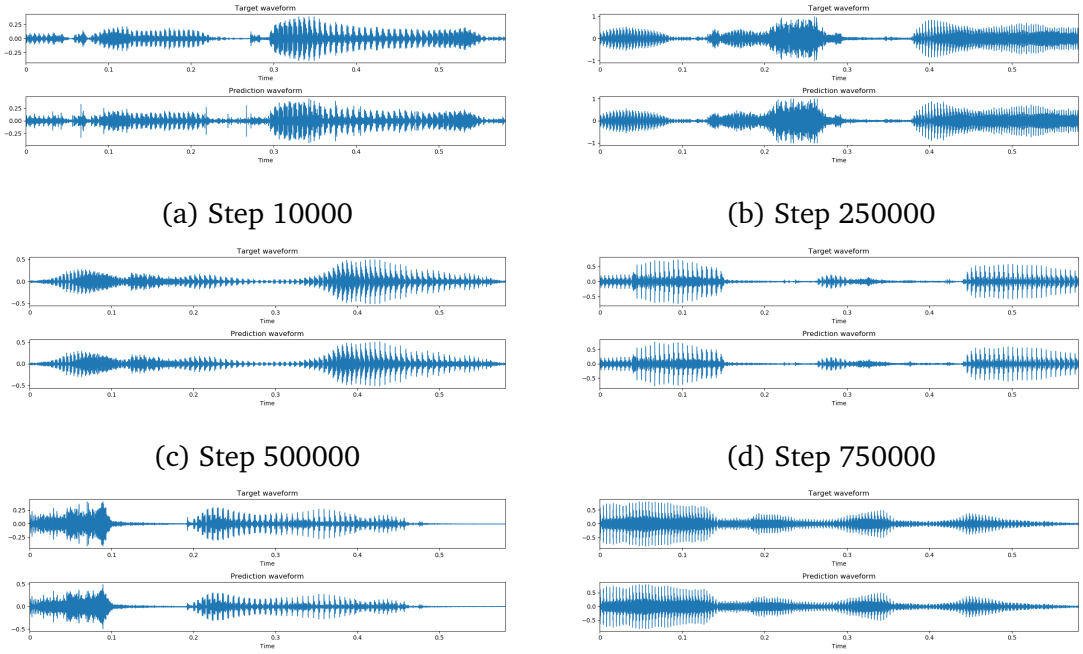


Figure 4.8: WaveNet Train Loss

of the waveform based on the training data set are shown in figure4.9. As we can

see, the predicted waveform is very similar or even the same as the ground truth of data, which means the networks are trained effective based on the training data set. However, the reason why we need to continue training WaveNet with the data



(a) Step 10000

(b) Step 250000

(c) Step 500000

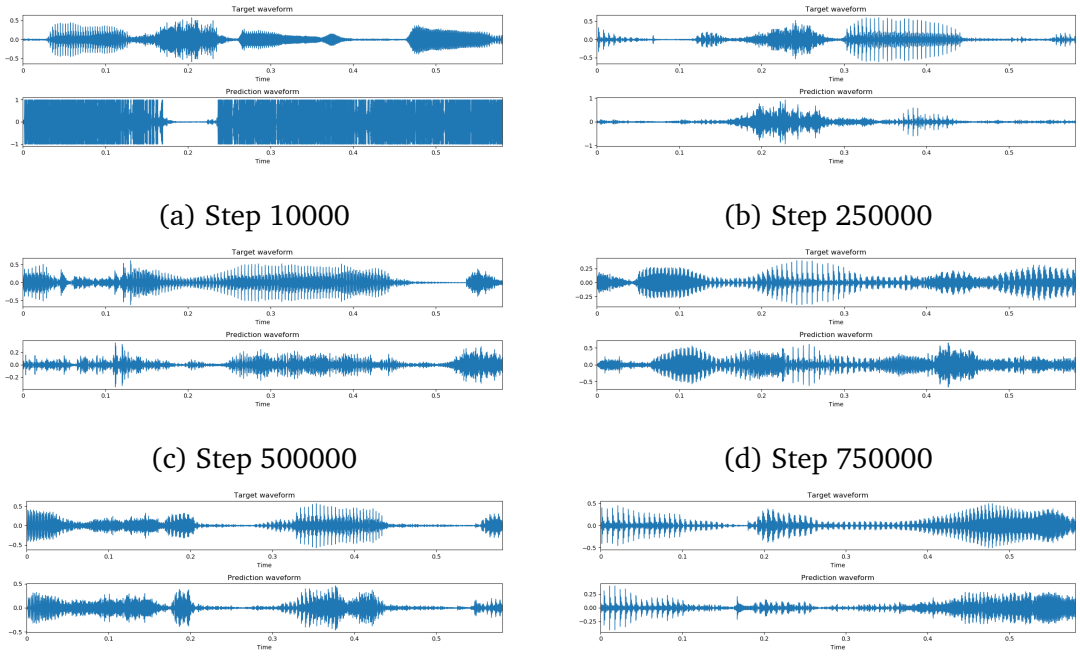
(d) Step 750000

(e) Step 1000000

(f) Step 1160000

Figure 4.9: WaveNet Training Prediction

set is because it performs negatively on the evaluation data set it never see before, the comparison diagram are shown in figure4.10. As these figures illustrated, the prediction on the first 10,000 steps, the waveforms is really noisy with an unclear waveform. However, the quality is getting positive with the training steps increasing. For example, although the speech on step of 250,000 is not completely the same with the ground truth, some necessary peak parts of the prediction waves start showing correspondingly to the real one. Furthermore, the step of 500,000 and 750,000 demonstrate that all the peak domain produced by WaveNet is related to the ground truth, which are even though not totally correlated with the true one. Finally, the step of 1 million and 1.16 million present almost the same waves with the ground truth. Under this condition, the WaveNet has been proved to completely trained positively, which is ready to use as vocoder to translate the output of mel-spectrogram of Tacotron2 into human understandable waveform.



(a) Step 10000

(b) Step 250000

(c) Step 500000

(d) Step 750000

(e) Step 1000000

(f) Step 1160000

Figure 4.10: WaveNet Evaluation Prediction

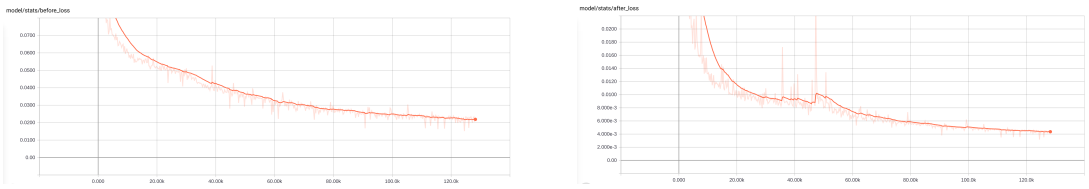
4.4 Data Reconstruction

After finishing the training process, we need to reconstruct the audio data from mel-spectrogram. LWS(Local Weighted Sums)[59][60] and librosa[71] functions are used for STFT to recovery phase from mel-spectrogram to audio. Firstly we need to denormalization on the data to recover it to the original waveform, and then LWS is used to transform the data with STFT to be the input into the function called 'istft' in librosa.

```
1    np.clip(D, -hparams.max_abs_value,
2         hparams.max_abs_value) + hparams.max_abs_value) * -hparams.
   min_level_db / (2 * hparams.max_abs_value))
3         + hparams.min_level_db)
4  S = _db_to_amp(D + hparams.ref_level_db) #Convert from mel back to
   linear
5  D = processor.run_lws(S.astype(np.float64).T ** hparams.power) # use
   lws as STFT function
6  y = processor.istft(D).astype(np.float32) #
```

4.5 Training Original Tacotron Model

For the original Tacotron2 model traning, the number of total steps is only about 125,000 due to it is getting harder to train after 120,000 steps. The reason here is because the total loss is in very small number which is only about 0.03. This



(a) Original Tacotron2 Before Loss (b) Original Tacotron After Loss

Figure 4.11: Original Tacotron2 Training Before and After Loss

tacotron model is totally the same model described in [95] with the same hyper-parameters. The model does not use any dropout and zoneout methods to improve the whole structures. As shown in figure4.11, both before_loss and after_loss declines to 0.02 and 0.004 respective within steps of 125,000. This is much smaller than the loss data of the compressed Tacotron2 network. Additionally, the total loss is similarly with a really sharp slope of decreasing to about 0.03 and maintaining this loss value until the training process finished with step 125,000 showed in figure4.12. Compared to the compressed model, this network is with much smaller total loss at the final steps of 125,000, where the compressed model is with total loss of 0.22.

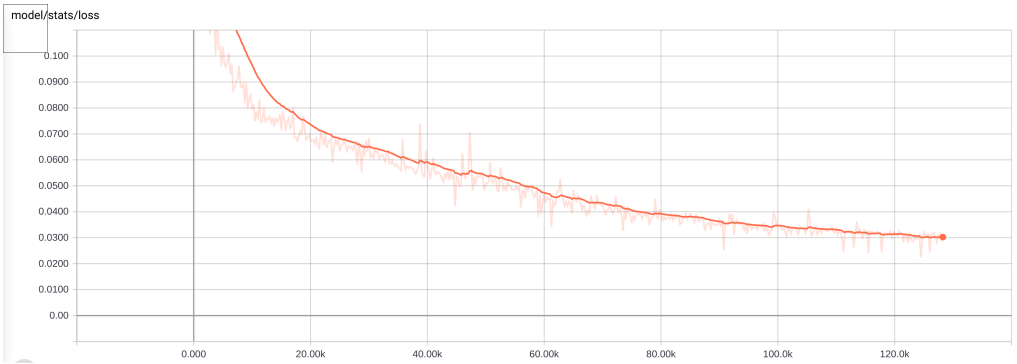
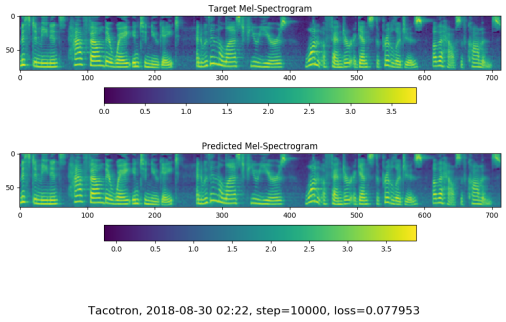


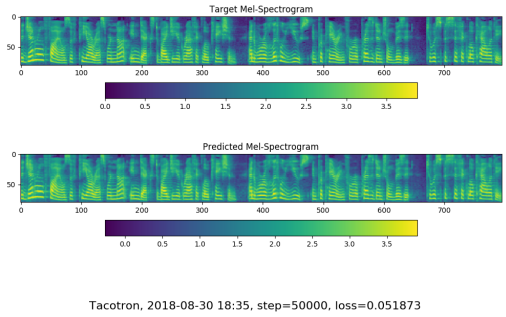
Figure 4.12: Original Tacotron2 Total Loss

Additionally, mel-spectrogram of the original model showed in figure4.13

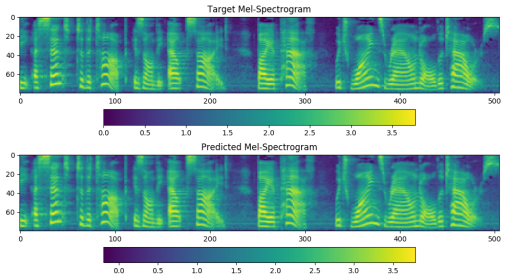
present a similar results as the compressed one. Starting with the very first steps of 10,000, the predicted one is very similar with the ground truth. Also, the mel-spectrogram keeps almost the same waves with the real one through all the steps showed in 4.13. In other words, original Tacotron2 model gives us very similar results with the compressed one which indicates that these two models perform effectively on the training data. Here the align diagrams present the similar re-



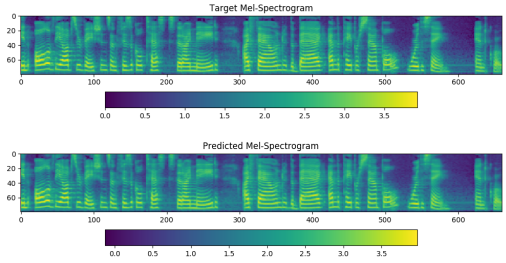
(a) Step 10000



(b) Step 50000



(c) Step 100000

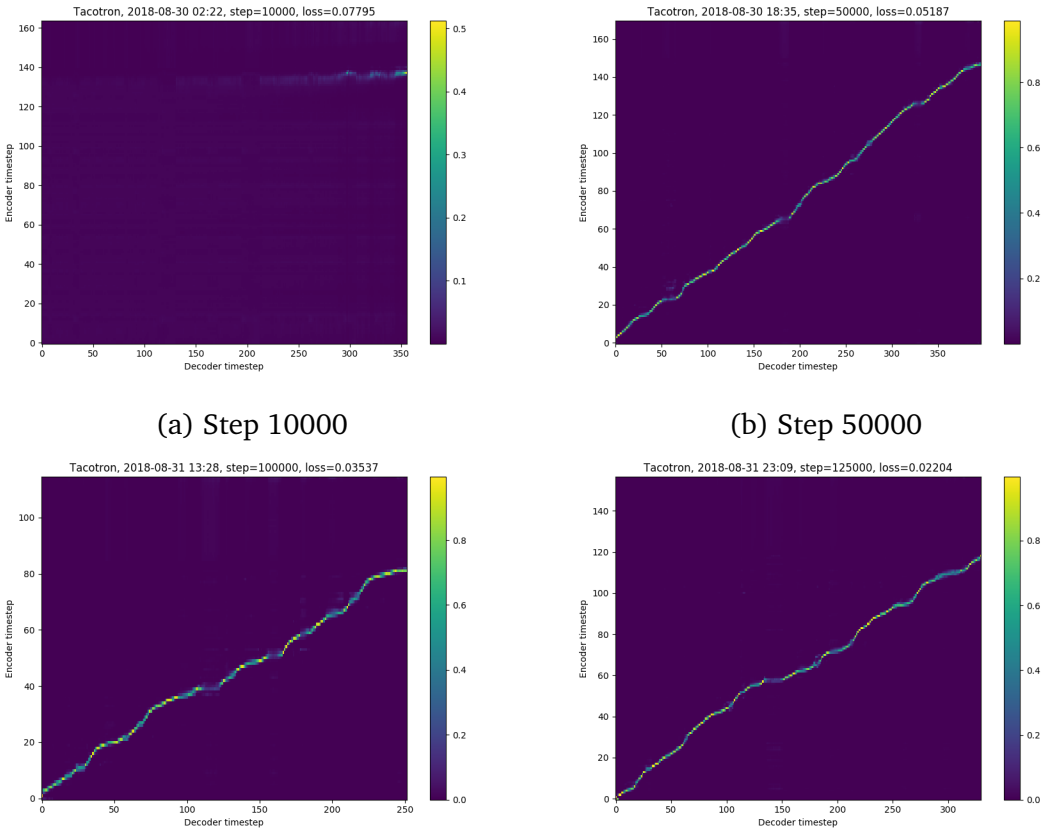


(d) Step 190000

Figure 4.13: Original Tacotron2 Training Prediction

sults as well as demonstrated in figure4.14. Here we can find that the first 10,000 steps gives a merely negative results, where the connection between encoder and decoder has not been established. However, the diagram of 50,000 steps offer us a really positive results where the slope lines present a slim and bright connection. When the steps increased onto 100,000 and 125,000, the slope of the connection increased, which means the output of encoder can specifically map to the output

of decoder, which conforms to the length of text as the input into the encoder is longer than the length of output in decoder which is corresponding speech waves. Additionally, the stop_token of original Tacotron2 is nearly about 0 after all the



(a) Step 10000

(b) Step 50000

(c) Step 100000

(d) Step 125000

Figure 4.14: Original Tacotron2 Training Aligns

training process of 125,000 steps, which is shown in figure5.6. These results provide the evidence that the model can strictly predict all the necessary stop token at the end of each sentence. These are pretty efficient results to present the model taht containing the attributes of making prediction on the sentence stopping for each training example.

These results of original Tacotron2 model showed above present nearly the similar consequence in loss tendency, making prediction on training mel-spectrogram and the aligns of attention mechanism, with lower loss and possible even better results. In other words, the compressed Tacotron2 and original Tacotron2 achieved an effective and useful mechanism on the Lj-speech data with

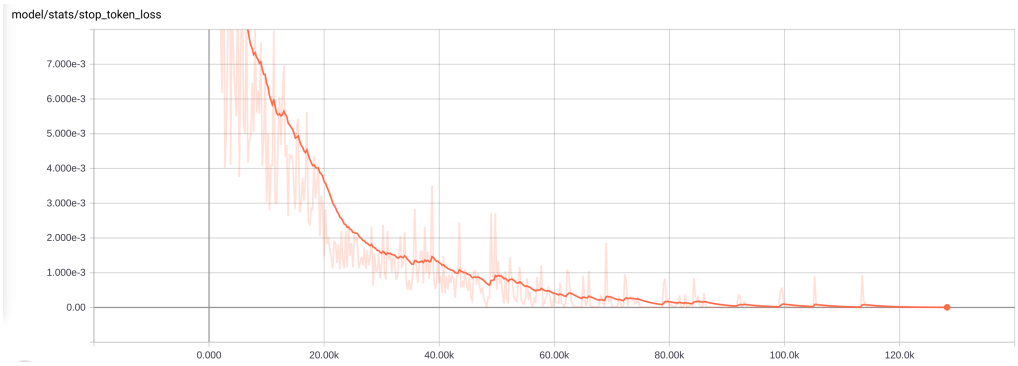


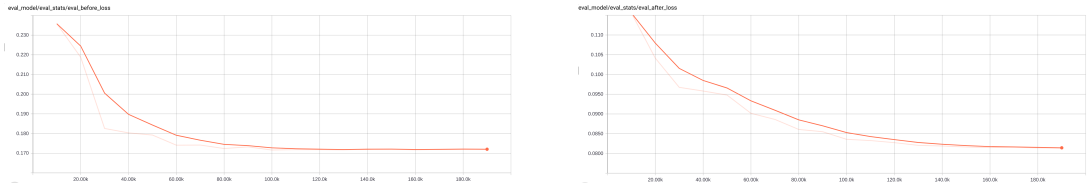
Figure 4.15: Original Tacotron2 Stop Token Loss

enough training steps, but from the perspective of loss value and training time, the result of original model present a more efficient result than the one of compressed model, which seems that the steps of zoneout and dropout cannot show a huge improvement only based on the training results.

Chapter 5

Evaluation

5.1 Tacotron2 Evaluation

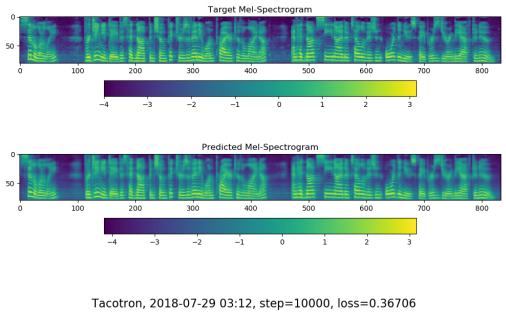


(a) Before Loss (b) After Loss

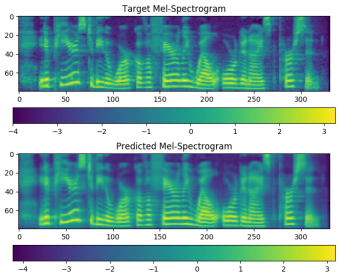
Figure 5.1: Tacotron Evalutation on Before and After Loss

As shown in the figure5.1, it presents the before_loss and after_loss of evaluation on Tacotron2. The before_loss declines sharply during the first 100,000 steps but then stay merely unchanged with 0.172. However, the after_loss of the PostNet continues decreasing slightly during the whole process and stay about 0.082 at the last several steps. The after_loss of evaluation of Tacotron is the same with the after_loss of training data. In addition, the before_loss of evaluation is larger than the training which is 0.122. For this point, although WaveNet contains several convolutional layers, the PostNet of Tacotron, which is convolution networks as well, still performs a significant role in the training process. The quality of prediction quality can be found in figure5.2. It shows that the first step of 10,000 can achieve a really positive outcome compared to the real one, whose loss is about 0.36706. Furthermore, the outcome keeps positive with steps increasing

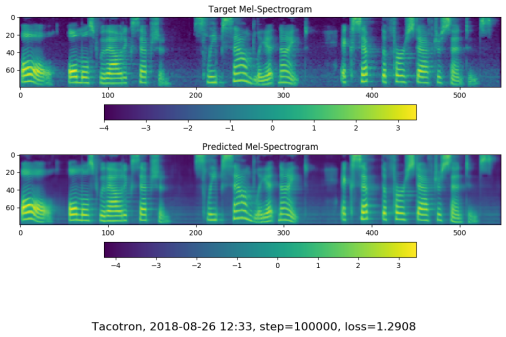
to 50,000. Additionally, the spectrogram of steps of 100,000 and 190,000 is still correlated to the ground truth. However, the value of the loss on that time is increasing which means the network is probably overfitting.



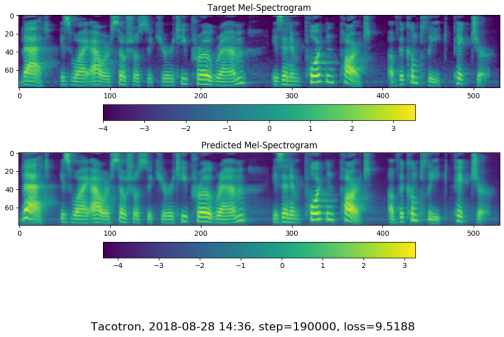
(a) Step 10000



(b) Step 50000



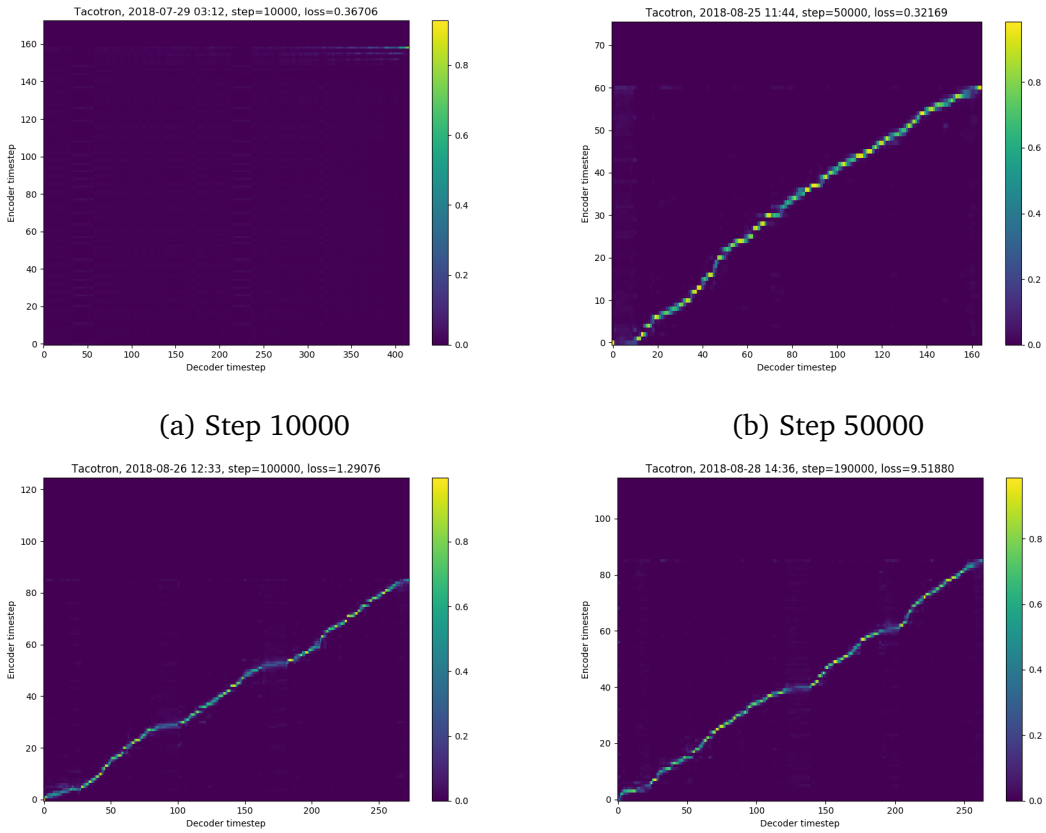
(c) Step 100000



(d) Step 190000

Figure 5.2: Tacotron Evaluation Prediction

The aligns of evaluation shows in figure5.3 presents training process keeping improving with the steps increasing from 10,000 to 50,000, which also shows that no aligns about the attention mechanism appears. The lines are getting slimmer and brighter during the process. Additionally, the slope of the lines is increasing which indicates that each encoder output can specifically map to one output of decoder, but still some encoder outputs cannot find the corresponding one, which means these could be empty space. Basically, text is always longer than mel-spectrogram in size, so the results are getting more positive for the training process continuing.



(a) Step 10000

(b) Step 50000

(c) Step 10000

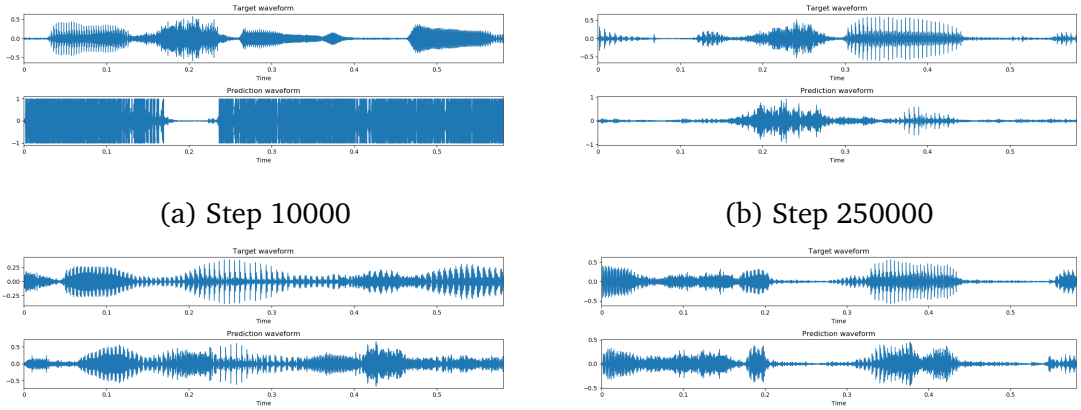
(d) Step 190000

Figure 5.3: Tacotron Evaluation Aligns

5.2 WaveNet Evaluation

As shown in figure4.10, the evaluation of WaveNet prediction is becoming more positive with training steps growing. On step of 10,000, the output is really negative with large areas of oscillation, which indicates some noise with no information about the speech produced by WaveNet. Then the outcome is better with the necessary peak demonstrating in the diagram, which represents the reflection of speech pressure showing in the spectrogram. Step of 250,000 shows corresponding two peaks with the ground truth, but step of 500,000 shows all the same voice peak in spectrogram with the real speech, but not total the same amplitude with the basic one. Furthermore, when the step reach 750,000, the number of peaks and the shape of the waveform demonstrate almost the same with the actual one. Additionally, the step of 1,000,000 and 1,160,000 demonstrate merely the

same with the real speech waveform. These processes show the evidence that the WaveNet structure is effective enough for being used as the vocoder of Tacotron2 to produce the final waveform.



(a) Step 10000

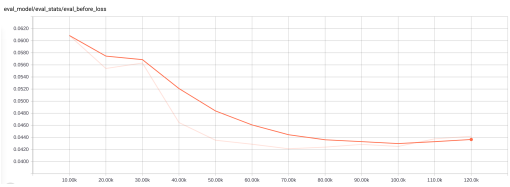
(b) Step 250000

(c) Step 750000

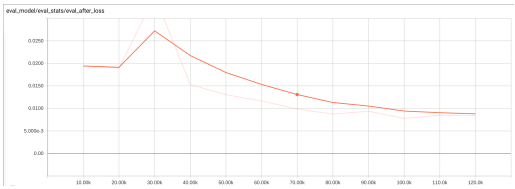
(d) Step 1000000

Figure 5.4: WaveNet Evaluation Prediction

5.3 Original Tacotron2 Evaluation



(a) Before Loss



(b) After Loss

Figure 5.5: Original Tacotron Evalutation on Before and After Loss

As shown in figure5.5, the before_loss and after_loss of evaluation on original Tacotron2 demonstrate a decline of loss over 120,000 steps overall. The loss had a slight increase on steps of 30,000 steps in the after_loss diagram but change to decrease again and keep this tendency till the end of the training. Due to the evaluation step is only processed per 10,000 steps, the evaluation steps are only performed on 120,000 but not 125,000, which is different from the training steps. The final loss value of before_loss keeps at 0.044 and the value of after_loss

is 0.008, which is still lower than the values of compressed model on training process, which are 0.082 for after_loss and 0.122 for before_loss respectively.

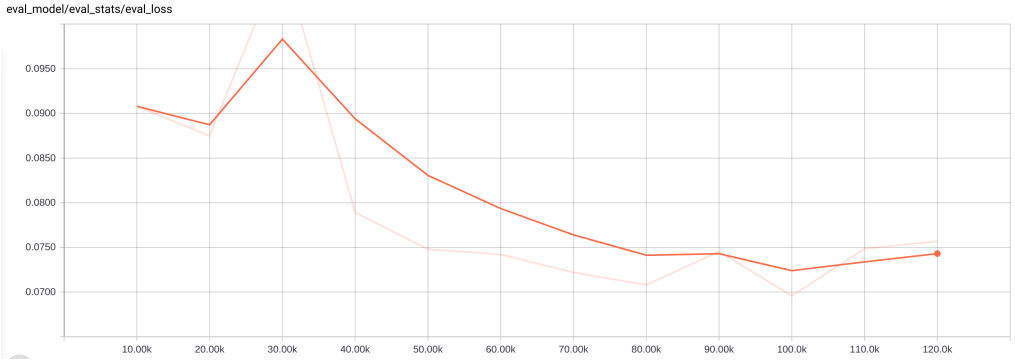
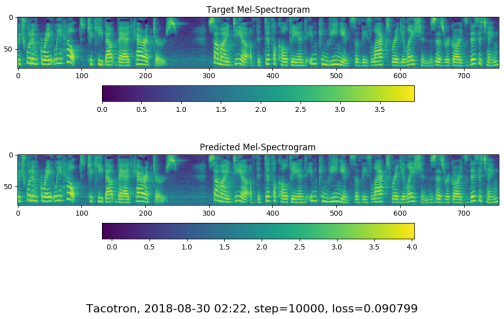


Figure 5.6: Original Tacotron2 Evaluation Total Loss

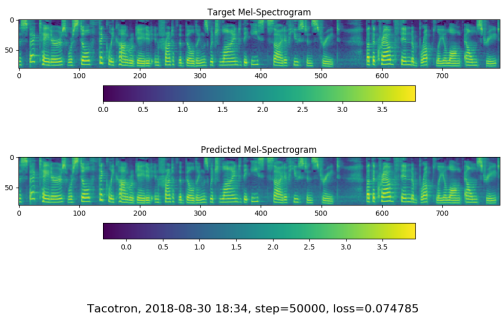
The total loss of the evaluation process in figure5.6 present a decline to about 0.075 overall. On step of 30,000, the loss value has an increase to 0.098 which is a small peak, and then keeps decreasing until the final step of 120,000. Here the reason should be due to the increase at the same steps of after_loss. The tendency of evaluation and training process keeps the same, which both have a continuous decrease, means the whole training process dose not experience overfitting problems like the compressed Tacotron2 model does.

Similarly, the mel-spectrogram of evaluation presents almost the same distribution of the highlights in the diagram since step of 10,000 to the end of the training process, which is 120,000 steps. This proves that the model performs pretty efficient on the evaluation data set which the model never see before. Along with the training loss presentation in figure4.12, the original model is proved to have a really positive performance on Lj-speech data set.

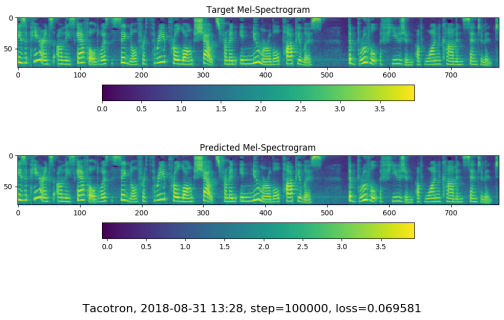
Again, the next is the diagrams showing the evaluation attention mechanism aligns in figure5.8. Similarly, for the first 10,000 steps, there is no obvious lines can be seen due to that the model have not been trained enough to make specific connection between encoder and decoder outputs. The performance improves with the step rising. We can see that diagram of 50,000 steps offers a merely slime and bright lines to demonstrate the relationship between encoder and decoder. The slope changes a lot with the number of steps increasing, which also indicates the same results that the encoder output can be mapped to the exact output of de-



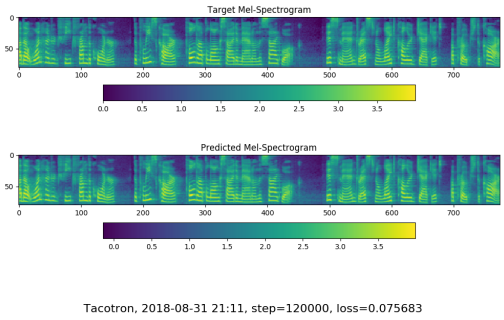
(a) Step 10000



(b) Step 50000



(c) Step 100000



(d) Step 120000

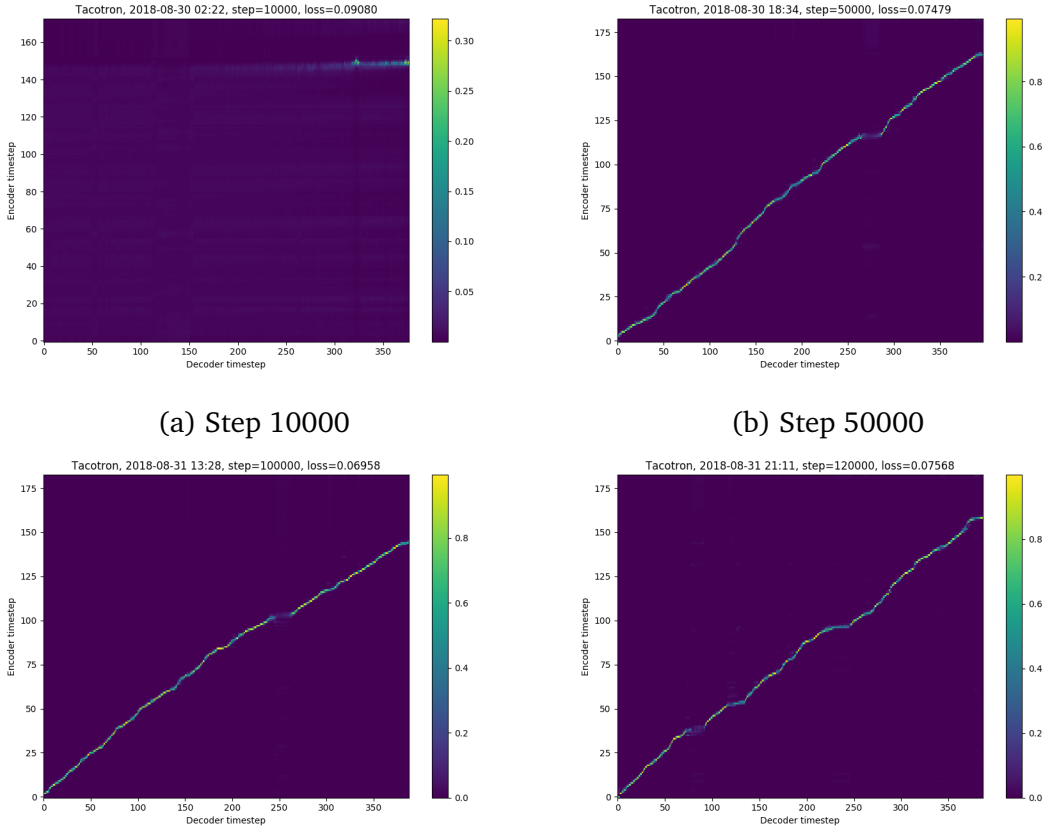
Figure 5.7: Original Tacotron Evaluation Prediction

coder, which has shorter length than the encoder. All these prove that the original model has been trained enough for synthesizing final speech waveform.

5.4 End-to-end Test

For testing and comparing the final performance of the two networks, we use three sentences to be as the test input showed as following:

- Scientists at the CERN laboratory say they have discovered a new particle.
- There's a way to measure the acute emotional intelligence that has never gone out of style.
- President Trump met with other leaders at the Group of 20 conference.



(a) Step 10000

(b) Step 50000

(c) Step 10000

(d) Step 120000

Figure 5.8: Original Tacotron Evaluation Aligns

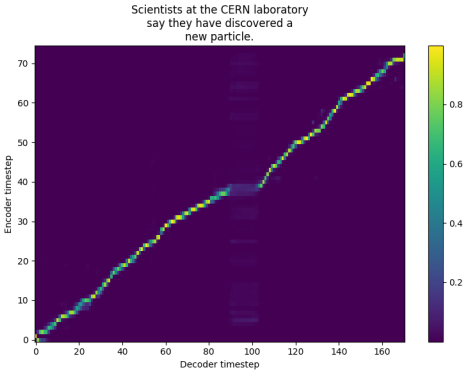
Firstly we need to use Tacotron2 to generate corresponding mel-spectrogram which will be used as the input to WaveNet. The time for generating audio of the 3 sentences of the original and compressed model is 15 seconds and 14 seconds separately. However, the generation of WaveNet to output the final waveform is much longer than the Tacotron2 model. For sentence 1, the original model spent 12 minutes, 46 seconds, however, the time of the compressed model is only 10 minutes 54 seconds. For sentence 2, the time is similar for original model, which are 15 minutes 20 seconds, but shorter time of 15 minutes 8 seconds for compressed model. The last sentence is various a lot for 12 minutes 24 seconds and 10 minutes 47 seconds respectively. All the data shown in table5.1.

During the process of the generating mel-spectrogram, the change of aligns of attention mechanism is demonstrated in the figure5.9 and figure5.10 below. As we can see, the line in original model processing on sentence 1 and sentence 2 shows

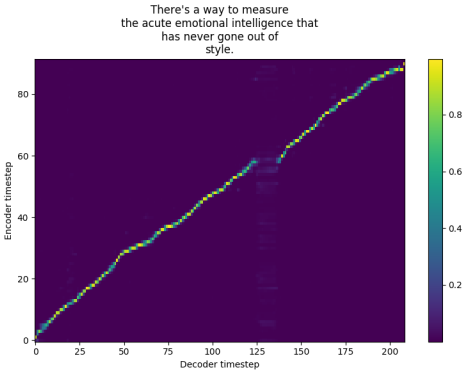
Model	Mel Generation	Sentence 1	Sentence 2	Sentence 3
Original Tacotoron2	00:15	12:46	15:20	12:24
Compressed Tacotron2	00:14	10:54	15:08	10:47

Table 5.1: Time of Generation Comparison

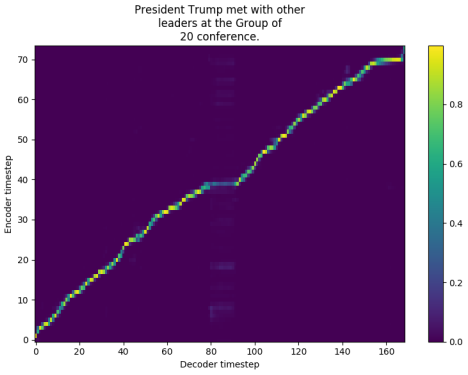
a broken(or dark with low probability) in the middle part, which means the model predict negatively results during that time steps. However, the lines in compressed model present a continuous and bright line in the diagram which indicates that the prediction of the compressed model is positive, and the generated information is useful for the WaveNet Using as well.



(a) Sentence 1



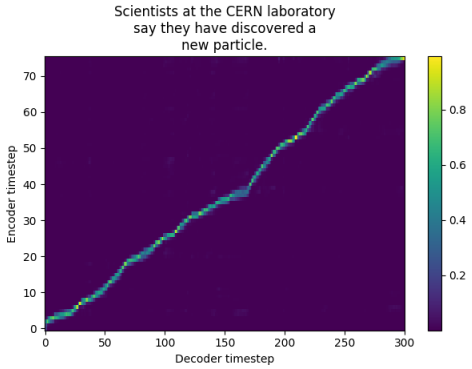
(b) Sentence 2



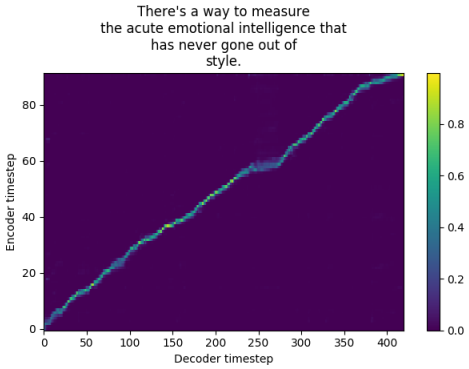
(c) Sentence 3

Figure 5.9: Original Model End-to-end Aligns

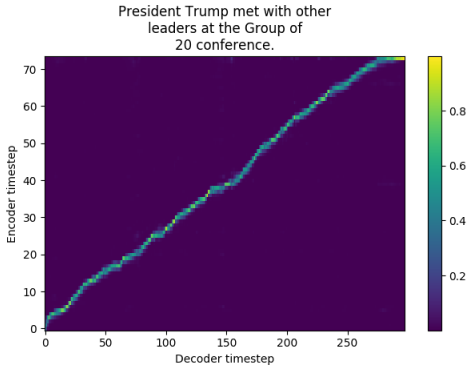
For the final result of the output, we use the method called MOS(Mean Opinion Score) which described in table1.2 to measure the quality of the final result. The



(a) Sentence 1



(b) Sentence 2



(c) Sentence 3

Figure 5.10: Compressed Model End-to-end Aligns

results and comparison are showed below in table5.2.

Model	Score 1	Score 2	Score 3	Score 4	Score 5	Average
Original Tacotoron2	2	2	3	2	1	2
Compressed Tacotron2	3	4	4	4	4	3.8

Table 5.2: MOS of Final Results

From the table above we can summary that the results of original network is not positive, which is more like a speech from machine but not human speech. However, the result of the compressed model is much better than the original one that reach the score of 3.8, which is much more like natural human speaking. The result present that the compressed model is successfully improved based on the

final results, with a short training time. The reason here are as follows:

1. The original model is not trained enough which is only 125,000 steps, but the steps of compressed one is 195,000.

2. The original model is overfitting due to the lower loss but negative tests.

3. This is due to that the loss of original one decline too quickly which make the training become even harder with the steps increasing.

From these point of views, we can say that the compressed model using Dropout and Zoneout to avoid the overfitting problems and also deleting the final duplicated layers of convolution related to the WaveNet part make the inference much more faster and better.

5.5 Weakness & Limitations

The most significant problem here is the inference time of WaveNet which is too long. Here we test two long sentences, the introduction of DSI(Data Science Institute) in Imperial College London[66], through Tacotron2 and WaveNet to generate the final test waveform which are:

- Data science is by nature at the core of all modern transdisciplinary scientific activities, as it involves the whole life cycle of data, from acquisition and exploration to analysis and communication of the results.

- Data science is not only concerned with the tools and methods to obtain, manage and analyse data: it is also about extracting value from data and translating it from asset to product. Launched on 1st April 2014, the Data Science Institute at Imperial College London aims to enhance Imperials excellence in data driven research across its faculties by fulfilling the following objectives.

- EOS

The inference time on Tacotron2 is only 38 seconds, which is showed in figure5.11, however, it take too much time when going through WaveNet which is 38 miniutes and 1 hour 10 minutes demonstrating in figure 5.12.

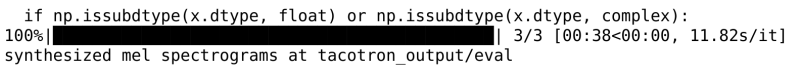


Figure 5.11: Test with Two Long Sentence on Tacotron

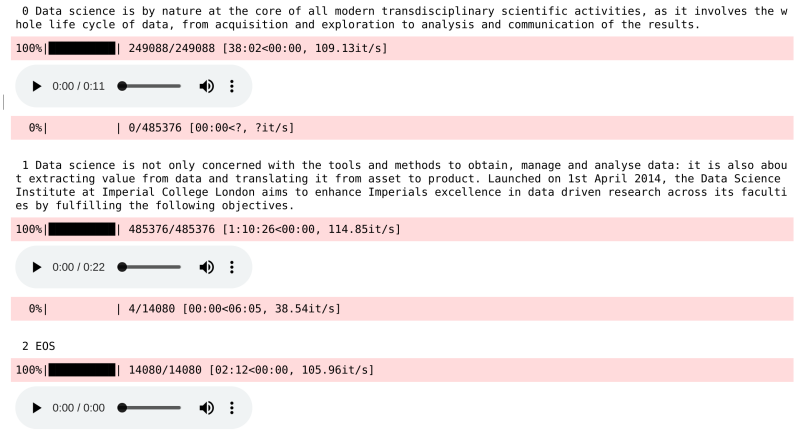


Figure 5.12: Test with Two Long Sentence on WaveNet

These results indicate that the inference on Tacotron2 is positive enough to even fulfill the real-time application requirement. However, it presents a too slow performance on WaveNet which is far away from the real-application, this is due to the attributes of WaveNet which is autoregressive, in other words, WaveNet can only generate the audio samples frame by frame depending on the previous one, which is that it cannot skip some time slice but only obey the order of time sequence.

The other problems should be the end-to-end problem. As the paper [95] said, Tacotron2 is a kind of end-to-end model, which is to say, inputting the text then outputting the corresponding speeches. However, the whole process still abides by the process that the text should be transformed into some kind of spectrogram(liner or mel-spectrogram), then be sent into some vocoders such as Griffin-Lim[114] [5]or WaveNet[95][83], which is separately trained in different stages. This is not the real end-to-end system for text-to-speech(TTS) problem.

5.6 Future Work

Due to the limitation of WaveNet, in the future, some more advanced WaveNet structures can be tested for improving the time of inferencing of the WaveNet discussed in this project, such as Parallel WaveNet[76], which uses a fully pre-trained WaveNet model as a teacher to teach a smaller and compressed network called student, which is smaller and more parallel, in which condition it is more suitable to modern computational hardware[77], which also showed in figure 5.13. This student network discussed above equipped with a smaller dilated convolutional neural network that is similar to the basic WaveNet. However, the generation of each sample is totally different because the smaller dilated network does not depend on the previously samples producing, which means it is possible that the network can generate any sample of the speech at any time. This property can solve the problem of original WaveNet, which is the autoregressive limitation.

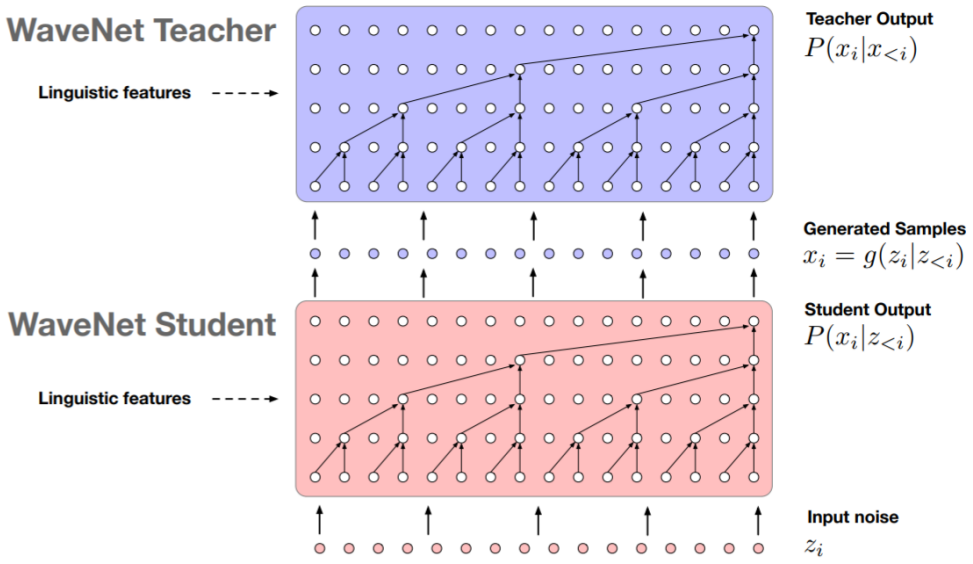
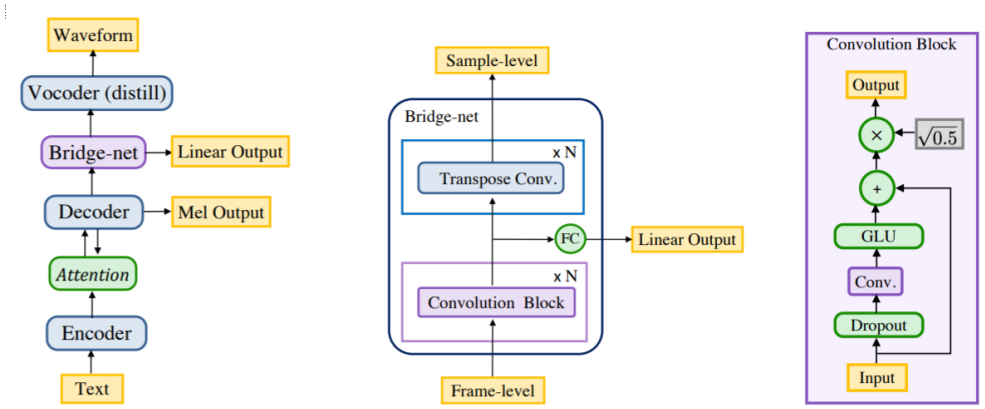


Figure 5.13: Structure of Parallel WaveNet

In addition, Baidu proposed a new structural network called ClariNet[81] this July, 2018. The structure of ClariNet is presented in figure5.14. ClariNet minimizes a novel regularized KL divergence, which is computed in closed-form, between their output distributions, to extract a Gaussian inverse autoregressive flow

from WaveNet. This kind of method make the training process simpler and distillation more effective. Additionally, this network can synthesize speech with fully convolutional layers fast and totally end-to-end without any intermediate features like spectragram[83]. [81] also shows that this network outperforms a lot than the DeepVoice and Tactotron. Under these conditions, the future work can reproduce ClariNet and use a similar method stated above to improve the whole structure.



(a) Text-to-wave architecture (b) Bridge-net (c) Convolution block

Figure 5.14: Structure of ClariNet

Chapter 6

Learning Points

Embarking on this project has enabled acquisition of a much better understanding on how to conduct a research project and thinking of feasible solutions to solve problems in the field of computer science and technology. By taking advice from the project supervisor and reading many related paper, most risks involved with carrying out a project that involves lacking of hardware for training and new problem proposed for strengthen this project were eliminated.

In this project, I learned a lot from several courses, such as Sequence Models by Andrew Ng [24], Neural networks for machine learning [41] taught by Geoffrey Hinton on the website of Coursera and CS231n: Convolutional neural networks for visual recognition [48] on Youtube[16]. Through these courses, I learned a lot about the basic models such as RNN, GRU[19], LSTM, word embedding[61], Beam Search[53], Attention Mechanism to build up a basic sequence-to-sequence model, and also different kinds of Text-to-Speech(TTS) systems such as Deep-Voice, Tacotron and ClariNet and their structure in details. By comparing their performance on different data set such as Lj-speech and CMU ARCTIC[54], to choose which to reproduce and try to compress the model and improve the final performance.

Additionally, some useful technology also have been adapted in this project such as dropout [98], Zoneout[57] and network regularization to avoid overfitting problems, along with batch normalization [45], stochastic gradient descent [14] and Adam optimization[12] to optimize the performance of neural networks.

Then I read two books [43] and [37] and some tutorials to learn how to use Tensorflow to construct and train CNNs on several GPUs. Also I learned how to set up the framework of Tensorflow on Linux Operation Systems. Overall, the experience of setting up the hardware for training, building the whole structure in TensorFlow and proper testing and evaluation of the project has helped me to achieve essential skills required to chase further studies in the field of computer science.

Chapter 7

Professional Issues

According to the code of practice and conduct issued by the British Computer Society, my project was finished punctually and I can confirm the purpose that, the whole system I designed is to benefit society. During the process of this project I always following the “Key IT practices”, for example:

1. Offering complete information on the project’s milestones and outcomes;

2. Regularly doing revision for avoiding risks and reporting any key issues that influence the design and implementation of project;

3. Finding out relative projects to study better researching methods and trying to eliminate issues they have encountered before;

4. Timely and frankly summarizing problems encountered, and knowledge learned.

As for the code of wrote, public’s safety, wellness, and environment are all considered. As stated in the code of conduct, I shall: 1. Report all the possible risks and results;

2. Regard the legal rights of third parties, project supervisor, and professional peers;

3. Carry out activities in a professional manner without distinctions;

4. Preserve secret information and not expose it for personal benefit unless the requirement or permission from the involved authority or in a court of law.

Bibliography

[1] Martın Abadi et al. “Tensorflow: Large-scale machine learning on heterogeneous distributed systems”. In: *arXiv preprint arXiv:1603.04467* (2016).

[2] Forest Agostinelli et al. “Learning activation functions to improve deep neural networks”. In: *arXiv preprint arXiv:1412.6830* (2014).

[3] Roee Aharoni and Yoav Goldberg. “Morphological inflection generation with hard monotonic attention”. In: *arXiv preprint arXiv:1611.01487* (2016).

[4] Roee Aharoni and Yoav Goldberg. “Sequence to sequence transduction with hard monotonic attention”. In: (2016).

[5] Sercan O Arik et al. “Deep voice: Real-time neural text-to-speech”. In: *arXiv preprint arXiv:1702.07825* (2017).

[6] Sercan O Arik et al. “Deep voice: Real-time neural text-to-speech”. In: *arXiv preprint arXiv:1702.07825* (2017).

[7] Sercan Arik et al. “Deep voice 2: Multi-speaker neural text-to-speech”. In: *arXiv preprint arXiv:1705.08947* (2017).

[8] Arts and Crafts Exhibition Society. *Arts and crafts essays*. Rivington, Percival, & Company, 1893.

[9] NTi Audio. *Fast Fourier Transformation FFT*. URL: https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft (visited on 08/30/2018).

[10] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014).

[11] Alan Bell et al. “Effects of disfluencies, predictability, and utterance position on word form variation in English conversation”. In: *The Journal of the Acoustical Society of America* 113.2 (2003), pp. 1001–1024.

[12] Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. “A maximum entropy approach to natural language processing”. In: *Computational linguistics* 22.1 (1996), pp. 39–71.

[13] R Bertolami et al. “A Novel Connectionist System for Improved Unconstrained Handwriting Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.5 (2009).

[14] Léon Bottou. “Large-scale machine learning with stochastic gradient descent”. In: *Proceedings of COMPSTAT’2010*. Springer, 2010, pp. 177–186.

[15] Peter F Brown et al. “Class-based n-gram models of natural language”. In: *Computational linguistics* 18.4 (1992), pp. 467–479.

[16] Jean Burgess and Joshua Green. *YouTube: Online video and participatory culture*. John Wiley & Sons, 2013.

[17] Sharan Chetlur et al. “cudnn: Efficient primitives for deep learning”. In: *arXiv preprint arXiv:1410.0759* (2014).

[18] Jan K Chorowski et al. “Attention-based models for speech recognition”. In: *Advances in neural information processing systems*. 2015, pp. 577–585.

[19] Junyoung Chung et al. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. In: *arXiv preprint arXiv:1412.3555* (2014).

[20] Peter A Clayton and Martin Price. *The seven wonders of the ancient world*. Routledge, 2013.

[21] Anne Cutler. *Native listening: Language experience and the recognition of spoken words*. Mit Press, 2012.

[22] Steven B Davis and Paul Mermelstein. "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences". In: *Readings in speech recognition*. Elsevier, 1990, pp. 65–74.

[23] Jeffrey Dean et al. "Large scale distributed deep networks". In: *Advances in neural information processing systems*. 2012, pp. 1223–1231.

[24] Deepleaning.ai. *Sequence Models*. 2018. URL: https://zh.coursera.org/learn/nlp-sequence-models (visited on 09/02/2018).

[25] Yariv Ephraim and David Malah. "Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator". In: *IEEE Transactions on acoustics, speech, and signal processing* 32.6 (1984), pp. 1109–1121.

[26] James L Flanagan. *Speech analysis synthesis and perception*. Vol. 3. Springer Science & Business Media, 2013.

[27] Mark Gales, Steve Young, et al. "The application of hidden Markov models in speech recognition". In: *Foundations and Trends® in Signal Processing* 1.3 (2008), pp. 195–304.

[28] Vadas Gintautas and Alfred W Hübler. "Resonant forcing of nonlinear systems of differential equations". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 18.3 (2008), p. 033118.

[29] Yoav Goldberg and Omer Levy. "word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method". In: *arXiv preprint arXiv:1402.3722* (2014).

[30] Alex Graves. "Generating sequences with recurrent neural networks". In: *arXiv preprint arXiv:1308.0850* (2013).

[31] Steven Greenberg and Brian Kingsbury. "The modulation spectrogram: In pursuit of an invariant representation of speech". In: *icassp*. IEEE. 1997, p. 1647.

[32] Werner H Greub. *Linear algebra*. Vol. 23. Springer Science & Business Media, 2012.

[33] Daniel W Griffin and Jae S Lim. "Multiband excitation vocoder". In: *IEEE Transactions on acoustics, speech, and signal processing* 36.8 (1988), pp. 1223–1235.

[34] Daniel Griffin and Jae Lim. "Signal estimation from modified short-time Fourier transform". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 32.2 (1984), pp. 236–243.

[35] Arthur Griffiths. *The chronicles of Newgate*. 1884.

[36] Catalin Grigoras. "Digital audio recording analysis–the electric network frequency criterion". In: *International Journal of Speech Language and the Law* 12.1 (2005), pp. 63–76.

[37] Siyu Gu, Zeyu Zheng, and Caicloud. *TensorFlow: Practice in Google Framwork of Deep Learning*. Beijing, China: Publishing House of Electronics Industry, 2017.

[38] Marion Harland. *Marion Harland's complete cook book.*, 2000.

[39] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[40] James L Hieronymus. "ASCII phonetic symbols for the world's languages: Worldbet". In: *Journal of the International Phonetic Association* 23 (1993), p. 72.

[41] Geoffrey Hinton, Nitsh Srivastava, and Kevin Swersky. "Neural networks for machine learning". In: *Coursera, video lectures* 264 (2012).

[42] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[43] Wenjian Huang and Yuan Tang. *TensotFlow in Action*. Beijing, China: Publishing House of Electronics Industry, 2017.

[44] Satoshi Imai. "Cepstral analysis synthesis on the mel frequency scale". In: *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'83*. Vol. 8. IEEE. 1983, pp. 93–96.

[45] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167* (2015).

[46] Keith Ito et al. *The LJ speech dataset*. 2017.

[47] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. “An empirical exploration of recurrent network architectures”. In: *International Conference on Machine Learning*. 2015, pp. 2342–2350.

[48] Andrej Karpathy. “Cs231n: Convolutional neural networks for visual recognition”. In: *Online Course* (2016).

[49] Keithito. *The LJ Speech Dataset*. 2018. URL: https://keithito.com/LJ-Speech-Dataset/ (visited on 09/04/2018).

[50] Spencer Kellis et al. “Decoding spoken words using local field potentials recorded from the cortical surface”. In: *Journal of neural engineering* 7.5 (2010), p. 056007.

[51] John Fitzgerald Kennedy, Lee Harvey Oswald, et al. “Report of the President’s Commission on the Assassination of President John F. Kennedy”. In: (1964).

[52] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[53] Philipp Koehn. “Pharaoh: a beam search decoder for phrase-based statistical machine translation models”. In: *Conference of the Association for Machine Translation in the Americas*. Springer. 2004, pp. 115–124.

[54] John Kominek and Alan W Black. “The CMU Arctic speech databases”. In: *Fifth ISCA workshop on speech synthesis*. 2004.

[55] Jan Koutnik et al. “A clockwork rnn”. In: *arXiv preprint arXiv:1402.3511* (2014).

[56] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[57] David Krueger et al. “Zoneout: Regularizing rnns by randomly preserving hidden activations”. In: *arXiv preprint arXiv:1606.01305* (2016).

[58] Peter Ladefoged and Keith Johnson. *A course in phonetics*. Nelson Education, 2014.

[59] Jonathan Le Roux et al. “Fast Signal Reconstruction from Magnitude STFT Spectrogram Based on Spectrogram Consistency”. In: *Proc. International Conference on Digital Audio Effects (DAFx)*. Sept. 2010, pp. 397–403.

[60] Jonathan Le Roux et al. “Phase Initialization Schemes for Faster Spectrogram-Consistency-Based Signal Reconstruction”. In: *Proc. Acoustical Society of Japan Autumn Meeting (ASJ)*. 3-10-3. Mar. 2010.

[61] Omer Levy and Yoav Goldberg. “Neural word embedding as implicit matrix factorization”. In: *Advances in neural information processing systems*. 2014, pp. 2177–2185.

[62] Xiangang Li and Xihong Wu. “Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE. 2015, pp. 4520–4524.

[63] Eric V Linder. “Exploring the expansion history of the universe”. In: *Physical Review Letters* 90.9 (2003), p. 091301.

[64] Weiyang Liu et al. “Large-Margin Softmax Loss for Convolutional Neural Networks.” In: *ICML*. 2016, pp. 507–516.

[65] Beth Logan et al. “Mel Frequency Cepstral Coefficients for Music Modeling.” In: *ISMIR*. Vol. 270. 2000, pp. 1–11.

[66] Imperial College LOndon. *About Data Science Institute*. 2018. URL: https://www.imperial.ac.uk/data-science/about-the-institute/ (visited on 08/30/2018).

[67] Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. *Neural machine translation (seq2seq) tutorial*. 2017.

[68] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. "Effective approaches to attention-based neural machine translation". In: *arXiv preprint arXiv:1508.04025* (2015).

[69] Rayhane mamah. *Spectrogram Feature prediction network*. 2018. URL: https://github.com/Rayhane-mamah/Tacotron-2/wiki/Spectrogram-Feature-prediction-network (visited on 08/30/2018).

[70] Peter Marler. "Characteristics of some animal calls". In: *Nature* 176.4470 (1955), p. 6.

[71] Brian McFee et al. "librosa: Audio and music signal analysis in python". In: *Proceedings of the 14th python in science conference*. 2015, pp. 18–25.

[72] Tomáš Mikolov et al. "Recurrent neural network based language model". In: *Eleventh Annual Conference of the International Speech Communication Association*. 2010.

[73] Milos Miljanovic. "Comparative analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction". In: *Indian Journal of Computer Science and Engineering* (2012), pp. 180–191.

[74] MIT. *PA and ARPAbet Symbols*. URL: http://courses.csail.mit.edu/6.345//notes/IPA/P001.html (visited on 2002).

[75] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks". In: *arXiv preprint arXiv:1601.06759* (2016).

[76] Aaron van den Oord et al. "Parallel WaveNet: Fast High-Fidelity Speech Synthesis". In: *arXiv preprint arXiv:1711.10433* (2017).

[77] Aäron van den Oord. *High-fidelity speech synthesis with WaveNet*. 2018. URL: https://deepmind.com/blog/high-fidelity-speech-synthesis-wavenet/ (visited on 08/30/2018).

[78] Aaron van den Oord et al. "Conditional image generation with pixelcnn decoders". In: *Advances in Neural Information Processing Systems*. 2016, pp. 4790–4798.

[79] Tanvina B Patel and Hemant A Patil. "Combining evidences from mel cepstral, cochlear filter cepstral and instantaneous frequency features for detection of natural vs. spoofed speech". In: *Sixteenth Annual Conference of the International Speech Communication Association*. 2015.

[80] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[81] Wei Ping, Kainan Peng, and Jitong Chen. "ClariNet: Parallel Wave Generation in End-to-End Text-to-Speech". In: *arXiv preprint arXiv:1807.07281* (2018).

[82] Wei Ping et al. "Deep voice 3: 2000-speaker neural text-to-speech". In: *arXiv preprint arXiv:1710.07654* (2017).

[83] Wei Ping et al. "Deep voice 3: Scaling text-to-speech with convolutional sequence learning". In: *Proc. 6th International Conference on Learning Representations*. 2018.

[84] Louis CW Pols. "Real-time recognition of spoken words". In: *IEEE Transactions on Computers* 100.9 (1971), pp. 972–978.

[85] Kishore Prahallad. "Speech technology: A practical introduction, topic: Spectrogram, cepstrum and mel-frequency analysis". In: *URl: http://www. speech. cs. cmu. edu/15-492/slides/03_mfcc. pdf* (2011).

[86] ITUT Rec. "P. 800.1, Mean opinion score (MOS) terminology". In: *International Telecommunication Union, Geneva* (2006).

[87] Flávio Ribeiro et al. "Crowdmos: An approach for crowdsourcing mean opinion score studies". In: *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE. 2011, pp. 2416–2419.

[88] Buribu Risuri. *Speech-to-Text-WaveNet : End-to-end sentence level English speech recognition using DeepMind's WaveNet*. URL: https://github.com/buriburisuri/speech-to-text-wavenet (visited on 2017).

[89] Franklin Delano Roosevelt. *Fireside Chat*. 1941.

[90] Haşim Sak, Andrew Senior, and Françoise Beaufays. "Long short-term memory recurrent neural network architectures for large scale acoustic modeling". In: *Fifteenth annual conference of the international speech communication association*. 2014.

[91] Tim Salimans et al. "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications". In: *arXiv preprint arXiv:1701.05517* (2017).

[92] Jason Sanders and Edward Kandrot. *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional, 2010.

[93] Mike Schuster and Kuldip K Paliwal. "Bidirectional recurrent neural networks". In: *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681.

[94] Stephanie Seneff and Victor Zue. "Transcription and alignment of the TIMIT database". In: *TIMIT CD-ROM Documentation* (1988).

[95] Jonathan Shen et al. "Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions". In: *arXiv preprint arXiv:1712.05884* (2017).

[96] June E Shoup. "Phonological aspects of speech recognition". In: *Trends in speech recognition* (1980), pp. 125–138.

[97] Jose Sotelo et al. "Char2wav: End-to-end speech synthesis". In: (2017).

[98] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting." In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[99] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Highway networks". In: *arXiv preprint arXiv:1505.00387* (2015).

[100] Robert C Streijl, Stefan Winkler, and David S Hands. "Mean opinion score (MOS) revisited: methods and applications, limitations and alternatives". In: *Multimedia Systems* 22.2 (2016), pp. 213–227.

[101] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

[102] Akira Tamamori et al. "Speaker-dependent WaveNet vocoder". In: *Proc. Interspeech*. Vol. 2017. 2017, pp. 1118–1122.

[103] Paul Taylor. *Text-to-speech synthesis*. Cambridge university press, 2009.

[104] Keiichi Tokuda et al. "Speech parameter generation algorithms for HMM-based speech synthesis". In: *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*. Vol. 3. IEEE. 2000, pp. 1315–1318.

[105] Keiichi Tokuda et al. "Speech synthesis based on hidden Markov models". In: *Proceedings of the IEEE* 101.5 (2013), pp. 1234–1252.

[106] Aäron Van Den Oord et al. "WaveNet: A generative model for raw audio." In: *SSW*. 2016, p. 125.

[107] Charles Van Loan. *Computational frameworks for the fast Fourier transform*. Vol. 10. Siam, 1992.

[108] Mahesh Viswanathan and Madhubalan Viswanathan. "Measuring speech quality for text-to-speech systems: development and assessment of a modified mean opinion score (MOS) scale". In: *Computer Speech & Language* 19.1 (2005), pp. 55–83.

[109] Eleni Vlahogianni and Matthew Karlaftis. "Testing and comparing neural network and statistical approaches for predicting transportation time series". In: *Transportation Research Record: Journal of the Transportation Research Board* 2399 (2013), pp. 9–22.

[110] Li Wan et al. "Regularization of neural networks using dropconnect". In: *International Conference on Machine Learning*. 2013, pp. 1058–1066.

[111] Hanbiao Wang et al. “Target classification and localization in habitat monitoring”. In: *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*. Vol. 4. IEEE. 2003, pp. IV–844.

[112] Jun Wang. “Analysis and design of a recurrent neural network for linear programming”. In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 40.9 (1993), pp. 613–618.

[113] Shihua Wang, Andrew Sekey, and Allen Gersho. “An objective measure for predicting subjective quality of speech coders”. In: *IEEE Journal on selected areas in communications* 10.5 (1992), pp. 819–829.

[114] Yuxuan Wang et al. “Tacotron: A fully end-to-end text-to-speech synthesis model”. In: *arXiv preprint* (2017).

[115] Robert L Weide. “The CMU pronouncing dictionary”. In: *URL: http://www. speech. cs. cmu. edu/cgibin/cmudict* (1998).

[116] Kelvin Xu et al. “Show, attend and tell: Neural image caption generation with visual attention”. In: *International conference on machine learning*. 2015, pp. 2048–2057.

[117] Kaisheng Yao et al. “Depth-gated recurrent neural networks. arXiv preprint”. In: *arXiv preprint arXiv:1508.03790* 9 (2015).

[118] Heiga Ze, Andrew Senior, and Mike Schuster. “Statistical parametric speech synthesis using deep neural networks”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE. 2013, pp. 7962–7966.

Appendix A

Ethics Checklist

Sections	Yes	No
Section 1: HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?		✓
Does your project involve the use of human embryos?		✓
Does your project involve the use of human foetal tissues / cells?		✓
Section 2: HUMANS		
Does your project involve human participants?		✓
Section 3: HUMAN CELLS / TISSUES		
Does your project involve human cells or tissues? (Other than from "Human Embryos/Foetuses" i.e. Section 1)?		✓
Section 4: PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?	✓	
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?		✓
Does it involve processing of genetic information?		✓
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.		✓

Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?		✓
Section 5: ANIMALS		
Does your project involve animals?		✓
Section 6: DEVELOPING COUNTRIES		
Does your project involve developing countries?		✓
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?		✓
Could the situation in the country put the individuals taking part in the project at risk?		✓
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?h		✓
Does your project deal with endangered fauna and/or flora /protected areas?		✓
Does your project involve the use of elements that may cause harm to humans, including project staff?		✓
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?		✓
Section 8: DUAL USE		
Does your project have the potential for military applications?		✓
Does your project have an exclusive civilian application focus?		✓
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		✓
Does your project affect current standards in military ethics – e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?		✓

Section 9: MISUSE		
Does your project have the potential for malevolent/criminal/terrorist abuse?		✓
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?		✓
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?		✓
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		✓
SECTION 10: LEGAL ISSUES		
Will your project use or produce software for which there are copyright licensing implications?		✓
Will your project use or produce goods or information for which there are data protection, or other legal implications?		✓
SECTION 11: OTHER ETHICS ISSUES		
Are there any other ethics issues that should be taken into consideration?		✓