Deep Learning Term Project Report

2015410008 Hyunsik Yoon
2015410112 Moonyoung Choi

1 Introduction

Today many people, industries, and academics want to generate the speech by text sequence. We can view so many examples such as Siri, which is an artificial intelligence of iOS, Clova, which is an artificial intelligence of Naver incorporation, and so on. However, as you may know, the quality of generated speech is still not easy on ears. They are far off from the human speech. Also, sometimes people want to generate the voice of someone we already know. For example, when we think about a Japanese animation named 'Detective Conan', the heroine named Conan can synthesize other one's voice with the same content that he speaks with his ribbon necktie. Like this, if we can make someone else's voice contain what we want, we can do so many things such as online lecture, sincere voicemail and so on. Hence, we want to generate a model which can generate the voice of a human, which has features of the person whose voices are used for training. There are so many works that were done before. Speech generation was traditionally done by 2 specific methods for Text to Speech conversion: Parametric TTS and Concatenative TTS. But parametric TTS is subject to generate muffled speech and noisy audio. Also, Concatenative TTS have to use huge databases, and also it takes very long train time. But the most important reason that these things are not used now commonly is that the voice features that these model catches are designed by humans. So deep learning comes into speech synthesis. There are several well-known models in this area such as WaveNet, deep voice series, SampleRNN and etc. But in this project, we will implement the model named Tacotron, which was suggested by Wang et al, because this model contains everything we learned in class: the concept of query, key, and value, attention, character embedding, GRU, highway network, convolutional 1D, Fully Connected layer and so on. In this paper, we will explain the whole model architecture of our implementation, the expected result of an experiment, conclusion, trouble, and future work.

2 Model architecture

The backbone of our model is the "Tacotron"[2]. Tacotron is seq2seq model with attention based model. Figure 1 is an overview of our model.

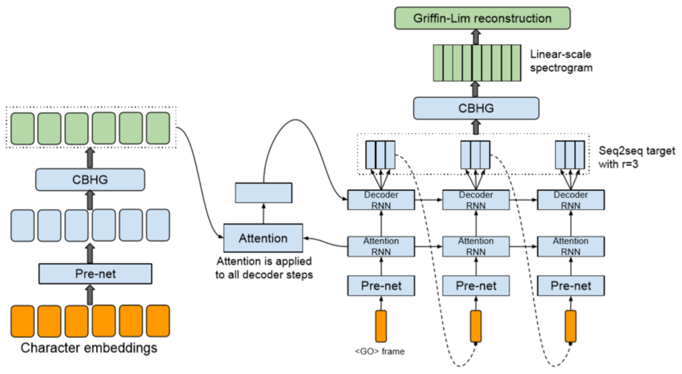


Figure 1: Model overview

The backbone of our model is the "Tacotron". Tacotron is seq2seq model with attention-based model. Figure 1 is an overview of our model. It includes an encoder, an attention-based decoder, and a post-processing net. Our model takes characters as input and produces spectrogram frames, which are then converted to waveforms. Our model takes characters as input and outputs raw spectrogram. We describe the main components of our model below.

2.1 Encoder

Figure 2 depicts the encoder part of our model. The encoder consists of Prenet and CBHG module.

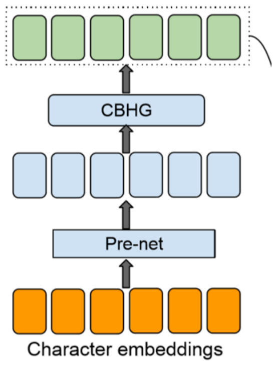


Figure 2: Encoder

2.1.1 Prenet

When character embeddings(256 dimensions) come into the encoder, they pass the Pre-net first. Prenet has FC(256)-ReLU-Dropout(0.5)-FC(128)-ReLU-Dropout(0.5) structure. It helps convergence and improves generalization.

2.1.2 CBHG module

After the Prenet, a Prenet output passes the CBHG module. CBHG module is a powerful module for extracting representations from sequences. CBHG module consists of a bank of 1-D convolution filters, followed by highway networks and a bidirectional GRU. An input sequence is first convolved with 16 sets of 1-D convolutional filters, where the kth set contains filters of width k(i.e. k = 1, 2, . . . , 16). These filters model contextual information using unigrams, bigrams, up to K-grams. The convolution outputs are stacked together and further max pooled along time to increase local invariances. We use stride 1 to preserve the original time resolution. We further pass the conv1D bank output to conv1D projection. Conv1D projection summarizes the contextual information. After that, we add original input sequence to conv1D projection outputs via a residual connection. We further pass the output to the highway network to extract high-level features. Finally, we stack a bidirectional GRU on top to extract sequential features. Batch normalization is used for all convolutional layers for zero-centered and unit variance data distribution.

2.2 Decoder

Figure 3 depicts the decoder part of our model. The decoder consists of 3 parts: Prenet-Attention network-Decoder network. We use 80-mel spectrograms for our training inputs of the model. And also, we set some hyperparameters: r, epsilon, and so on. We will introduce these things later in this part. Also, when we predict the spectrograms for our encoder input (text embedding), we use a spectrogram segment that was made at the previous time step of GRU on the current time step.

2.2.1 Prenet

First, our 80-mel spectrogram input (train time) or a spectrogram segment that was made at the previous time step was put into this structure. The whole structure of decoder prenet is almost as same as the encoder prenet, so we will skip this part. The size of the output dimension is 128.

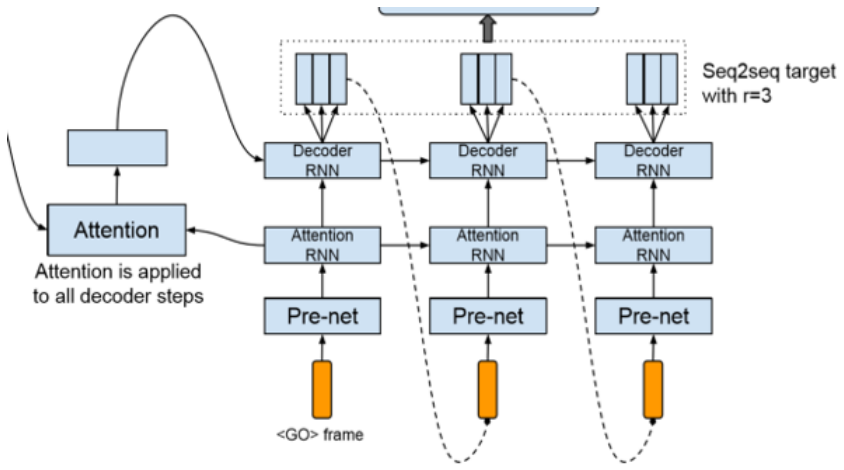
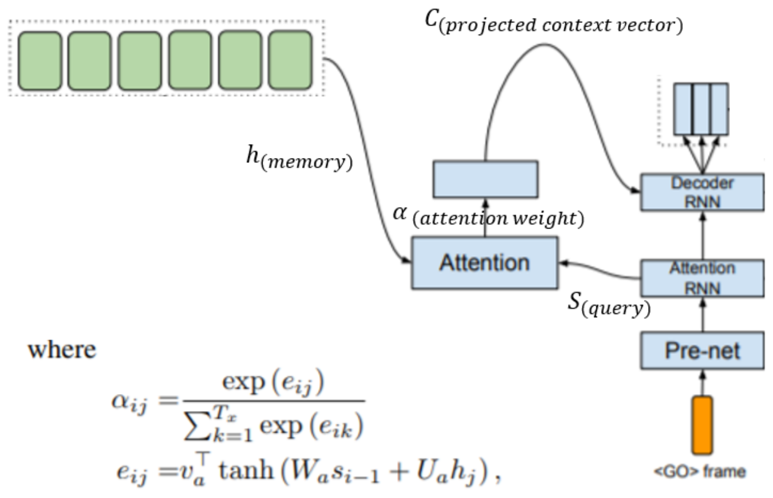


Figure 3: Decoder

2.2.2 Attention RNN

Second, the prenet output comes into this attention layer. To be exact, this attention 'cell'. this attention cell is composed of 2 parts: Baudanau attention-GRU that has the 256 sizes of the output dimensions. Figure 4 represents the whole structure of the attention module that was used in the Tacotron model. First, the first input (the prenet output) goes through GRU cell and query making layer. Then this vector combined with the memory (the encoder output). This data goes through tanh layer, fully connected layer and softmax layer sequentially, then the attention weight comes out. Then we do a weighted sum, generate a context vector, and concatenate this vector and query vector. The vector output goes out of this attention layer passing the final projection layer, which makes the size of the vector to 256 dimensions.



where

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

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

Figure 4: Attention RNN

2.2.3 Decoder RNN

The decoder RNN layer is composed of 2 GRU layers and residual connections. (GRU-residual connection-GRU-residual connection-projection) First layer output is connected with the residual connection, which adds the previous input vector to this output, and this whole vector goes through the second layer. The whole outcomes go through the final projection layer. This layer makes the size of the vector dimension into 80 * r. 'r' is the hyperparameter that the user can set before the training process. The model outputs not just one segment of the spectrogram, but r segments of the spectrogram. This is because one character is related not only a single segment but also a certain number of segments. So we produce 'r' segments per one step of the decoding process. And the whole output feeds into the next stage of the decoder. The reason why we use 80-mel spectrograms at the decoder stage is related to a sort of regularization issue. This model does not make the whole waveform at once. This model generates the spectrogram on the decode stage, and synthesize the whole waveform at the post-processing stage.

2.3 Post-processing net

The final stage is this post-processing net. The input is the 80-mel spectrogram that was generated at the decode stage, and the output is the whole waveform. This layer does not use a neural network entirely. This layer is composed of 2 parts: CBHG module, fully connected layer, and Griffin-Lim reconstruction. The CBHG module is almost the same as the encoder CBHG module, so we will not explain this part in detail.

3 Experiment

We couldn't do training since we don't have enough time. But we checked our model doing proper forward pass and backward pass. Checking task was carried out using test_tacotron.ipynb file. If we train our model properly, we can achieve an MOS of 3.82, which outperforms the parametric system. And also this results implies that Griffin-Lim synthesis works quite well. But if we heard the output

	mean opinion score
Tacotron	3.82 ± 0.085
Parametric	3.69 ± 0.109
Concatenative	4.09 ± 0.119

Table 1: Mean opinion score evaluation

that was generated by other people, we can recognize that there is a little bit phase distortion, so it sounds artificial. Furthermore, using the 'attention_weights' in the decoder.py, we can visualize the attention strengths per a character embedding vector like figure 5.

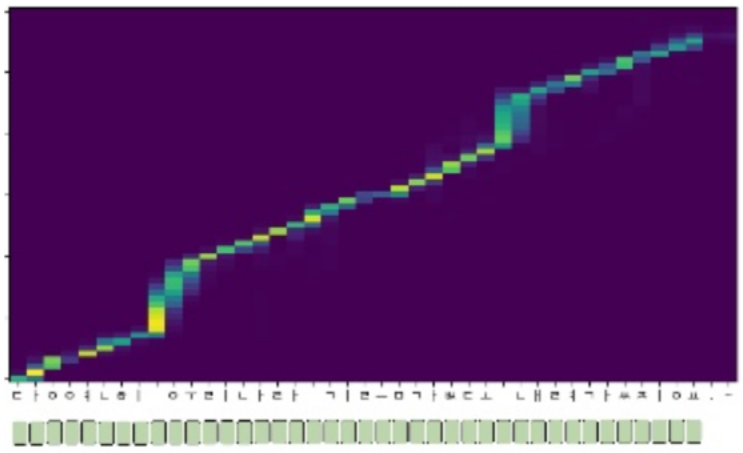


Figure 5: Visualized attention

4 Conclusion

We have implemented Tacotron, an integrated end-to-end generative TTS model that takes a character sequence as input and outputs the corresponding spectrogram. Although we couldn't do training, our model does forward pass and backward pass properly. If we trained our model, our model will synthesize Prof.choo's voice.

5 Trouble

1. Detailed implementation was different with or not mentioned in the paper of Tacotron (ex, the dimension size of the layer, input concatenation, CBHG module at the decoder..), so we have to read a lot of articles and papers about the text to sequence model, attention, speech synthesis and so on. Also, we referred to the implementations that were done by other people, but their implementations also different from each other.

2. We had so many problems at the decoder attention layer. We tried the different sizes of input dimensions by concatenating the query and the memory, or adding two things, or passing through the fully connected layer.

3. We used PyTorch library for the first time, so we had to get used to using the various methods and layers.

4. Decoder stage is not parallel.

6 Future work

Future work should train our model using Prof.Choo's voice data and make model can learn multiple voices. Furthermore, using WaveNet synthesis model, Griffin-Lim reconstruction can be replaced by a neural network.

References

[1] Ping, W., Peng, K., Gibiansky, A., Arik, S. O., Kannan, A., Narang, S., ... & Miller, J. (2017). Deep voice 3: 2000-speaker neural text-to-speech. arXiv preprint arXiv:1710.07654.

[2] Wang, Y., Skerry-Ryan, R. J., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., ... & Le, Q. (2017). Tacotron: Towards end-to-end speech synthesis. arXiv preprint arXiv:1703.10135.