Speech Understanding - text-to-speech

Subodh Kant (M23CSA531), SyamKrishnan Sakthidharan(M23CSA535)

January 2025

1 Overview

This report details the implementation and execution of a Tacotron2-based Text-to-Speech (TTS) pipeline using PyTorch and torchaudio. The pipeline converts text input into speech through multiple stages, including text encoding, spectrogram generation, and vocoder-based waveform synthesis.

1.1 State of the current SOTA models

The models evaluated in this study are

1. Tacotron2 (Spectrogram Generation) [1]
2. WaveRNN Vocoder (Waveform Generation)
3. Griffin-Lim Vocoder (Waveform Generation)
4. Waveglow Vocoder (Waveform Generation)

1.2 Importance in the Real World

Text-to-Speech (TTS) technology has emerged as a transformative tool with significant real-world applications, bridging the gap between human-computer interaction and accessibility. By converting written text into spoken language, TTS systems enable seamless communication for individuals with visual impairments, learning disabilities, or literacy challenges, thereby fostering inclusivity. Beyond accessibility, TTS plays a pivotal role in industries such as education, healthcare, and entertainment, where it enhances user experiences through voice-enabled assistants, audiobooks, and language learning tools. Moreover, advancements in neural TTS models have significantly improved the naturalness and expressiveness of synthesized speech, making it nearly indistinguishable from human speech. This progress underscores the importance of TTS in creating more intuitive and human-like interfaces, ultimately driving innovation and improving quality of life across diverse domains. As research in this field continues to evolve, the potential for TTS to revolutionize communication and accessibility remains vast, making it a critical area of study in both academic and industrial contexts.

2 Strengths and Limitations of SOTA models

2.1 Tacotron2

Tacotron2 is an end-to-end neural network model that converts text into a mel spectrogram, which is then used by a vocoder to generate speech waveforms.

Strengths:

- Can work with different vocoders (WaveRNN, WaveGlow, Griffin-Lim) for waveform generation.
- Generates realistic speech with smooth transitions and intonations.

Limitations:

- Can sometimes produce incorrect or unstable alignments, leading to stuttering or gibberish output.
- Requires large, high-quality datasets to perform well.
- It generates mel-spectrograms sequentially, limiting real-time applications.

2.2 WaveRNN Vocoder

WaveRNN is a neural vocoder that converts mel spectrograms into speech waveforms using a recurrent neural network (RNN).

Strengths:

- Can run on edge devices with lower computational resources.
- Produces clear and natural speech, comparable to WaveGlow.
- Faster to train compared to autoregressive models like Tacotron2.

Limitations:

- Autoregressive nature makes it slower than fully parallel models like WaveGlow.
- Struggle with high variability in pitch and prosody.

2.3 Griffin-Lim Vocoder

Griffin-Lim is a classical signal processing algorithm that reconstructs waveforms from spectrograms using iterative phase estimation.

Strengths:

- Faster than autoregressive vocoders like WaveRNN.
- Does not require deep learning, making it simple and lightweight.
- Often used for quick prototyping when high-quality synthesis isn't required.

Limitations:

- Struggles to recover realistic phase information, leading to less natural sound.
- Speech sounds robotic and lacks natural prosody compared to neural vocoders.

2.4 WaveGlow Vocoder

WaveGlow is a normalizing flow-based vocoder that converts mel spectrograms into high-quality speech waveforms.

Strengths:

- Produces clear and natural-sounding audio.
- Learns directly from data without requiring additional processing.
- WaveGlow synthesizes speech in a single forward pass, making it much faster.

Limitations:

- Requires significant GPU resources, making it impractical for mobile or low-power devices.
- Normalizing flow models like WaveGlow need careful tuning and large datasets.

3 Evaluation Metrics

3.1 Mean Opinion Score (MOS)

MOS is a subjective evaluation metric where human listeners rate the quality of synthesized speech on a scale of 1 (bad) to 5 (excellent).

Strengths: Captures human perception of naturalness and intelligibility and Evaluates prosody, intonation, and speaker similarity. It can be used for real-world applications where subjective quality matters.

Limitations: Expensive and time-consuming since it requires human evaluators. Prone to bias and inconsistency among raters and Hard to standardize across different studies.

Results:

- WaveGlow and WaveRNN tend to achieve higher MOS scores (4.0-4.6) compared to Griffin-Lim (which typically scores below 3.0 due to robotic quality).
- Tacotron2 + WaveGlow usually achieves the highest MOS due to high fidelity and natural prosody.

Conclusion

- WaveGlow + Tacotron2 performs best across MOS metrics, providing high MOS scores.
- WaveRNN + Tacotron2 is slightly behind but remains a good trade-off between efficiency and quality.
- Griffin-Lim + Tacotron2 is significantly worse in terms of perceived naturalness and spectral accuracy.

4 The text-to-speech pipeline goes as follows:

4.1 Text preprocessing

First, the input text is encoded into a list of symbols. In this tutorial, we will use English characters and phonemes as the symbols.

4.2 Spectrogram generation

From the encoded text, a spectrogram is generated. We use the Tacotron2 model for this.

4.3 Time-domain conversion

The last step is converting the spectrogram into the waveform. The process to generate speech from spectrogram is also called a Vocoder. In this tutorial, three different vocoders are used, WaveRNN, GriffinLim, and Nvidia's WaveGlow.

The following figure illustrates the whole process.

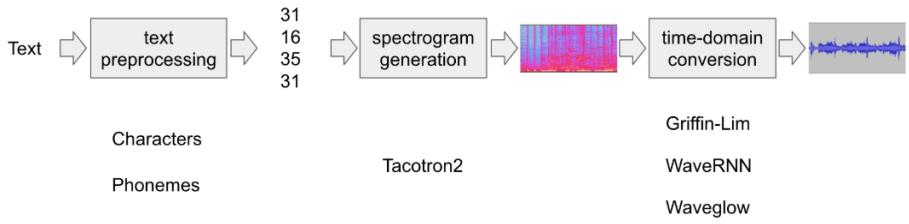

Figure 1: text-to-speech pipeline

5 Setup and Environment

First, we install the necessary dependencies. In addition to torchaudio, DeepPhonemizer is required to perform phoneme-based encoding.

Setting the Seed and cuda - The script initializes PyTorch with a random seed for reproducibility and determines whether a GPU (CUDA) is available

6 Text Processing

6.1 Character-based encoding

The pre-trained Tacotron2 model is designed to work with a specific set of symbols (letters, punctuation, and special characters). These symbols must be converted into numeric IDs that the model can process. While libraries like torchaudio provide this functionality, manually implementing the encoding process helps you understand the underlying mechanics.

We start by defining the set of symbols _!'(),.:;?

abcdefghijklmnopqrstuvwxyz. Next, each character in the input text is mapped to the index of its corresponding symbol in the table, while any characters not present in the table are ignored.

As noted above, the symbol table and indices must align with the requirements of the pretrained Tacotron2 model. Fortunately, torchaudio includes this transformation alongside the pretrained model. We can create and use this transformation.

Note: The output from our manual encoding matches the result from torchaudio's text-processor, confirming that we correctly reimplemented the functionality of the library. The text-processor ac-

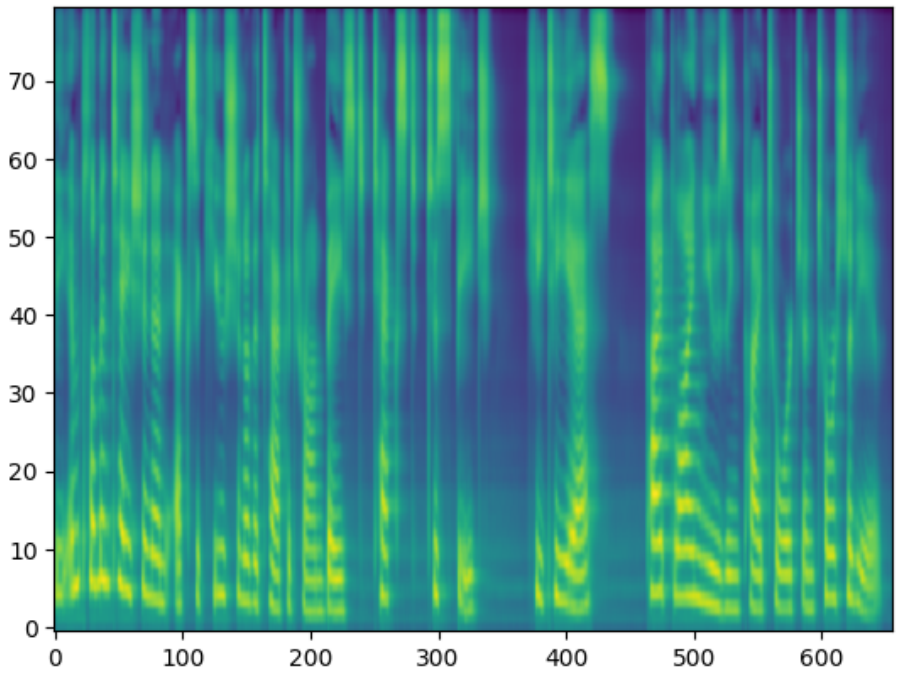

Figure 2: Spectrogram

cepts either a single text or a list of texts as input. When a list of texts is provided, the returned lengths variable indicates the valid length of each processed token in the output batch.

6.2 Phoneme-based encoding

Similar to the case of character-based encoding, the encoding process is expected to match what a pretrained Tacotron2 model is trained on. torchaudio has an interface to create the process.

Notice that the encoded values are different from the example of character-based encoding.

6.3 Spectrogram Generation

Tacotron2 is the model we use to generate spectrogram (Figure 2,3) from the encoded text. torchaudio.pipelines.Tacotron2TTSBundle bundles the matching models and processors together so that it is easy to create the pipeline.

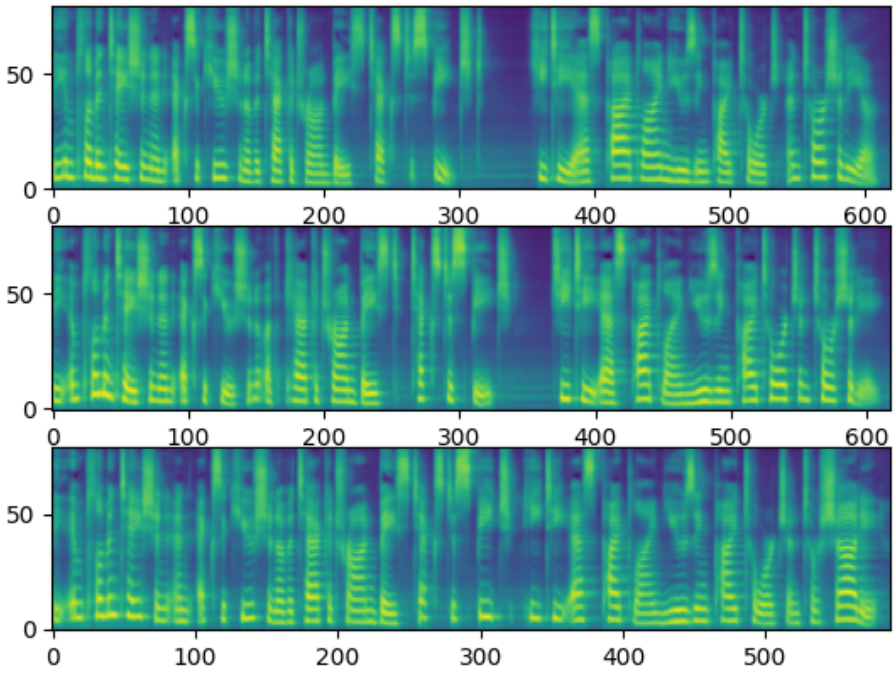

Figure 3: Enter Caption

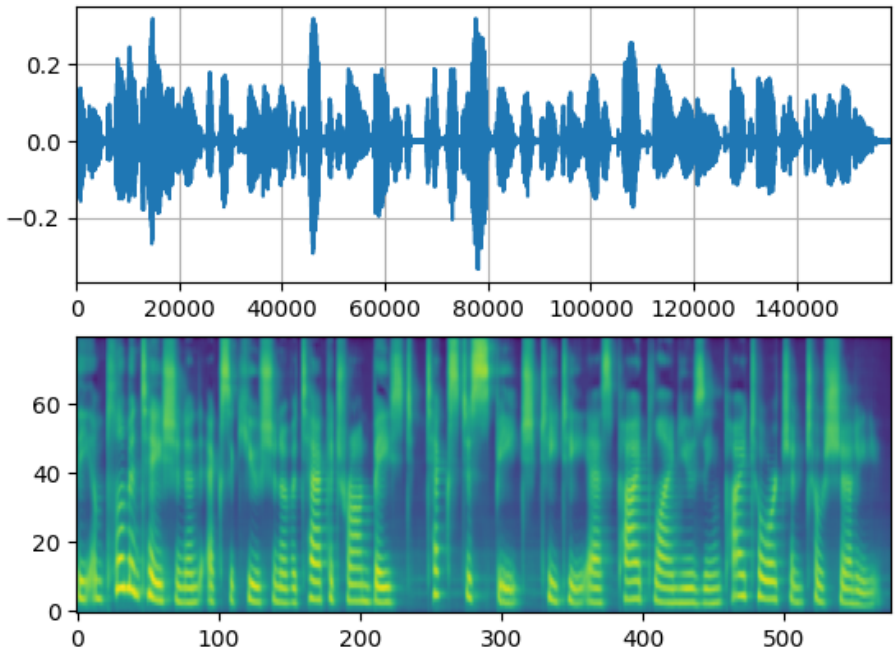

Figure 4: WaveRNN Vocoder

Note that Tacotron2.infer method perfoms multinomial sampling, therefore, the process of generating the spectrogram incurs randomness.

7 Waveform Generation

Generate the waveform from the spectrogram using a vocoder - Griffin-Lim, WaveRNN and Waveglow.

7.1 WaveRNN Vocoder

[Figure 4] Once the spectrogram is generated, the last process is to recover the waveform from the spectrogram using a vocoder, torchaudio provides vocoders based on GriffinLim and WaveRNN.

7.2 Griffin-Lim Vocoder

[Figure 5]

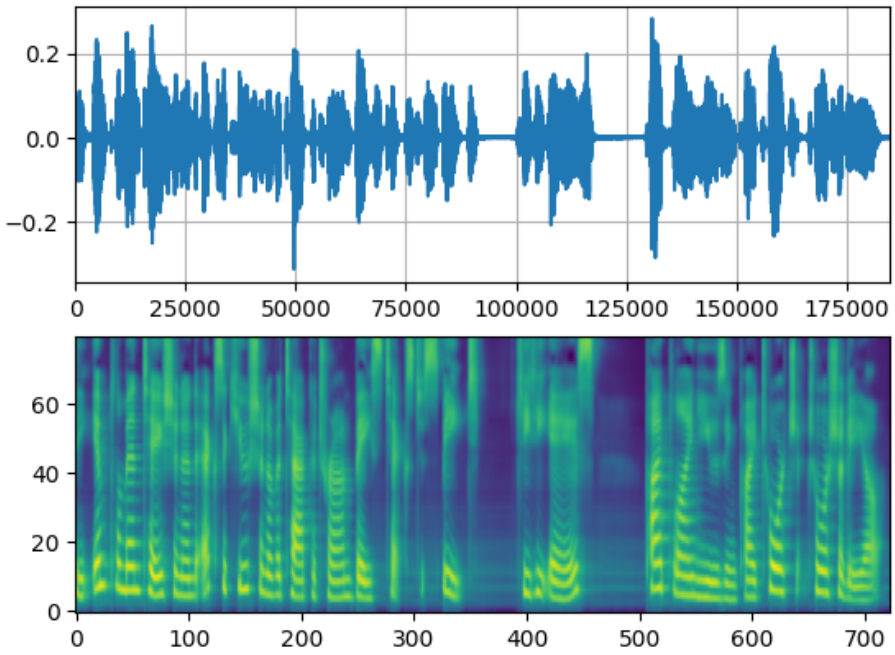

Figure 5: Griffin-Lim Vocoder

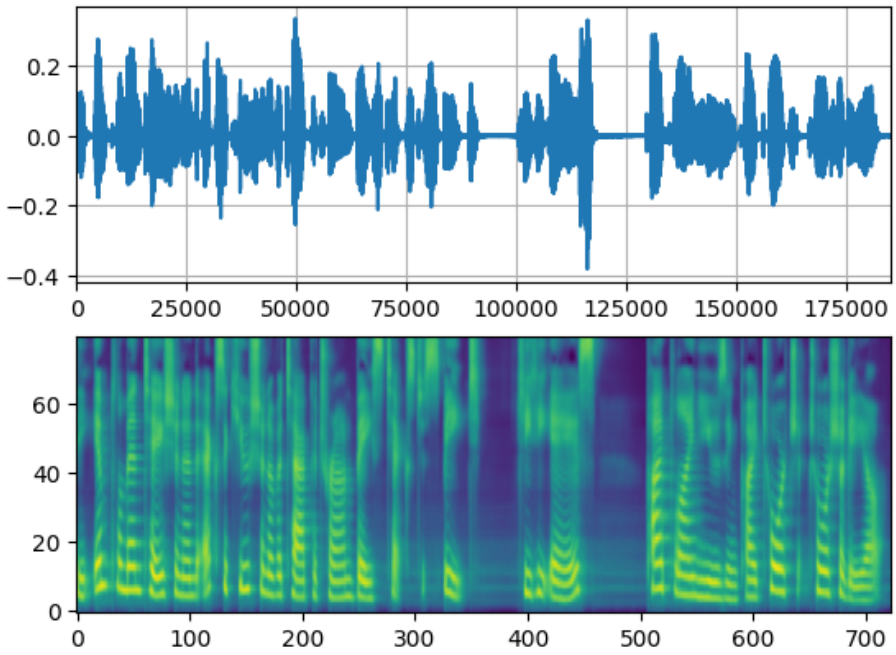

Figure 6: Waveglow Vocoder

7.3 Waveglow Vocoder

Waveglow is a vocoder published by Nvidia. The pretrained weights are published on Torch Hub. One can instantiate the model using torch hub module [Figure 6].

References

[1] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, Rif A. Saurous, Yannis Agiomvrgiannakis, and Yonghui Wu. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. *None*, pages 4779–4783, 2018.