American Journal of Engineering and Technology Management
2025, Vol. 10, No. 6, pp. 94–100
https://doi.org/10.11648/j.ajetm.20251006.11

Research Article

A Study on the Incremental Text-to-speech Synthesis Taking into Account Intermediate Feature-level Context

Song-Yun Kim, Jin-Hyok Song, Dae-Hun Pak, Dong-Song Pak, Myong-Hyok Won, Hakho Hong*

Institute of Mathematics, State Academy of Sciences, Pyongyang, Democratic People's Republic of Korea

Abstract

While end-to-end text-to-speech (E2E TTS) has significantly improved speech quality compared to traditional TTS, the computational cost is very expensive due to the use of complex neural network architectures. The synthetic time has been much reduced by the efforts to reduce the computational cost. In TTS, the real-time factor in the device should be less than 1, and the latency is required to be possibly small. One way to reduce latency in E2E TTS system is incremental TTS. In incremental TTS, there is a disadvantage of the loss of naturalness at the boundary between the sentence segments, as speech is synthesized in units of the sentence segment. To improve naturalness at the boundary between the sentence segments, we take into account the context. Then, taking into account the context as text or the context as an intermediate feature of encoder and decoder containing attention, the amount of computation in acoustic model can increase and the synthetic speech can be broken at the boundary between the sentence segments. That is, incremental TTS is subject to a trade-off between the amount of computation and naturalness of synthetic speech. In this paper we propose an incremental Korean TTS method taking into account the intermediate feature-level context, which is based on the analysis of two-stage E2E TTS consisting of an acoustic model and a vocoder. We present experimental result conducted on the FastSpeech2 model, which shows the effectiveness of our approach.

Keywords

TTS, Incremental TTS, Encoder, Decoder, Transformer

1. Introduction

The study of incremental TTS has originated from the performance of tasks such as simultaneous interpretation, where the entire sentence is not given at a time, but is given step by step (c.f. [1-6]). Therefore, the input latency as well as the computation latency should be taken into account when one uses the general full-sentence-based TTS scheme.

In order to address this latency problem, the incremental TTS approach to synthesize speech with the text being given by one word or several words had been proposed. The incremental TTS significantly reduces the input latency and computation latency compared to the general TTS, and maintains a certain amount of latency regardless of the length of the sentence.

However, speech quality degradation arises in incremental TTS because the context information of the whole sentence is not available, but only local information is used. The in-

*Corresponding author: hhhong@star-co.net.kp (Hakho Hong)

Received: 19 November 2025; **Accepted:** 8 December 2025; **Published:** 29 December 2025

cremental TTS was initially applied to HMM-based statistical parametric TTS, and several studies [1-5] have been carried out to solve the problem of speech quality degradation. Nevertheless, traditional TTS such as HMM-based statistical parametric TTS trains all submodules (e.g., text parsing, acoustic models, and vocoders) separately, and the errors in the previous step are propagated to the subsequent step.

Furthermore, the quality of speech in incremental TTS still does not reach a satisfactory level for the reason that the overall contextual features can not be obtained from the limited sentence segment.

By developing E2E TTS (c.f. [12-16]), the complex and manual models of traditional TTS are simplified, and direct mapping from strings to acoustic features is trained using neural networks. As a result, the E2E TTS has shown better synthetic speech quality over traditional TTS.

A neural network-based incremental TTS method (c.f. [6, 11]) also showed high-quality synthetic speech. Since incremental TTS processes smaller units than sentences, the authors in [6] divided the sentence into a certain number of segments and followed by the addition of several symbols representing sentence onset, middle and end positions, to represent the position of the corresponding composite unit, and trained the composite model into those units. In addition, the sentence is divided into small units to synthesize the corresponding speech and then to combine it in a way that links them in generating speech.

However, the segments are synthesized in units only, regardless of the context [6]. The lack of context leads to lower quality in the synthetic speech.

To overcome these limitations, the context for selected segments is considered by applying a context encoder with the past segments before the current segment and the predicted sentences generated by the pre-trained language model GPT in [9]. However, the computational effort of [9] is increased by performing the computation for context in the encoder containing attention, and also the context information is not clear because it uses a sentence that is different from the given text such as the sentence generated by the language model.

The intermediate feature that is passed through the encoder is divided into a constant-size chunk [10]. It performs the decoder computation with the intermediate feature segmented by the chunk size, and performs the decoder computation by connecting the intermediate feature of the previous segment to each segment of the intermediate feature. Thus, the decoder computation for the current segment is done considering the context as a intermediate feature of the previous state. However, the computational complexity in the decoder [10] increases because we consider the context by combining the current segment with the previous state segment in the decoder.

The previous approaches take into account the context in the encoder and decoder which contains attention, and thus increase the computational effort.

For the incremental TTS, two problems should be taken into consideration:

1) Computational time: During the playback of a segment of speech, the TTS of the next segment must be completed.
2) Impaired speech quality: By synthesizing sentences in segments, the degradation of the resultant speech should not be significant.

Motivated by the above cited works, we are interested in incremental TTS. The purpose of this paper is to address these two problems. More precisely, in this paper we propose a method that takes context into account at the intermediate feature-level in variance adapter, not in the encoder and decoder that contains attention. In general, human prosody accents are mainly related to pitch and energy. In other words, the current speech is much affected by the pitch and energy of the previous speech. In this regard, we propose a method to calculate the pitch and energy of the current speech by considering the influence of pitch and energy on the previous speech.

We present experimental result to show the effectiveness of our approach. The experiments were conducted on the FastSpeech2 model. The experimental result demonstrates our method is superior to the state of art synthesis methods.

The rest of present paper is organized as follows. In Section 2, we shall present an incremental TTS taking into account intermediate feature-level context in detail, from its motivation, intuition, and formulation to the discussion. In Section 3, we experimentally demonstrate the effectiveness of the proposed method and the superiority of its performance compared to the previous synthesis methods. Section 4 is devoted to concluding the main contribution of this paper.

2. Method

2.1. Conditions for No Breaking of Synthetic Speech at the Boundary Between Segments

The flowchart for the two-step E2E TTS is shown in the following figure.

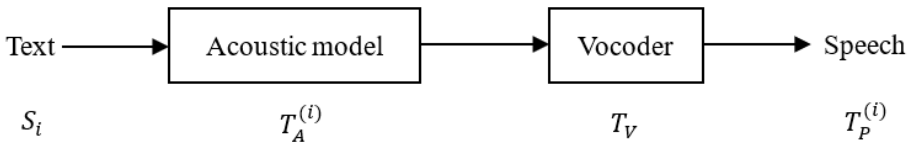

Figure 1. *Latent-time and Speech Playback Time in Two-stage E2E TTS.*

Figure 1 intuitively shows the latency that takes time during the *i*th segment S_i is entered and its speech is played. In the figure, $T_A^{(i)}$ is the computation time of the acoustic model for S_i, and T_V represents the generation time of one frame in the vocoder. $T_P^{(i)}$ represents the playback time of the generated speech for S_i.

Let T_F be the duration of a frame, N_i the number of frames in S_i, *RTF* the real-time factor, and $T_d^{(i)}$ the latency of S_i in the vocoder. Then we have

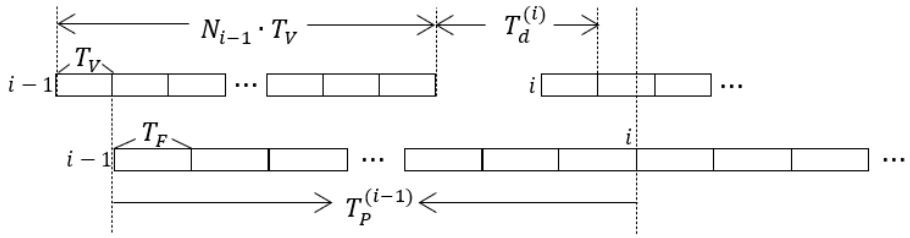

Figure 2. *In Vocoder, the Generation Time for the Previous Segment and the Latent-time When It Is Passed to the Next Segment.*

$$RTF = \frac{T_V}{T_F} \tag{1}$$

$$T_d^{(i)} = T_A^{(i)} + T_V = T_A^{(i)} + RTF \cdot T_F \tag{2}$$

$$T_P^{(i)} = N_i \cdot T_F \tag{3}$$

Figure 2 shows the computation time in the acoustic model for each segment sentence and the waiting time for the first frame of the next segment sentence in the vocoder. As shown in the Figure 2, one can verify that the following equation (5) is a necessary and sufficient for the continuous utterance of the synthetic speech at the boundary between the segments without breaking:

$$T_d^{(i)} < \left(T_P^{(i-1)} + T_V\right) - N_{i-1} \cdot T_V \tag{4}$$

$$T_A^{(i)} + T_V < N_{i-1} \cdot T_F + T_V - N_{i-1} \cdot T_V$$

$$T_A^{(i)} < N_{i-1} \cdot (T_F - T_V) = N_{i-1} \cdot (1 - RTF) \cdot T_F \tag{5}$$

That is, the computation time of S_i in the acoustic model must be shorter than $N_{i-1} \cdot (1 - RTF) \cdot T_F$ for a continuous utterance of a synthetic speech at the boundary between the segmens.

2.2. Proposed Method

In this subsection, we propose an improved incremental TTS method that can reduce computational time while producing natural speech. More precisely, we add a concatenation and a separate structure of intermediate features to reflect context information to the variance adapter used in the FastSpeech2 model. When a person utters a sentence, the prosody is mainly affected by phoneme duration, pitch, and energy. Here, pitch and energy have a great influence on the connection between the previous and the next segment. Therefore, we propose a method that takes context into account in pitch and energy features. The details of the proposed method are as follows.

Suppose that the sentence *S* is divided into *I* segments, and S_1, S_2, …, S_I denotes each segment, respectively. Each segment S_i is converted to a hidden feature h_i through a grapheme-to-phoneme conversion, a phoneme layer, and an encoder. Let x_i be the feature after the hidden feature h_i passes through the duration predictor and length regulation. Let p_i, e_i be the intermediate features after this feature x_i passes through the pitch predictor and energy predictor, respectively. Let further m_i be the input of the decoder corresponding to the *i*th segment. To synthesize the speech of the *i*th segment S_i, we synthesize the speech corresponding to the *i*th segment, taking into account the context as the previous segment, *i*-1 segment. That is, the following conditional probabilities are modeled:

$$P(V_i|S_{i-1}, S_i), i = 2, \ldots, I \tag{6}$$

Then, unlike in [9, 10], we take the context into account using the intermediate feature x_i of the variance adapter. That is

$$P(V_i|x_{i-1}, x_i), i = 2, \ldots, I \tag{7}$$

Let us consider the details of the improved variance adapter. If the hidden feature h_i passes through the duration predictor, it outputs d_i and outputs the feature x_i via length regulation. Then it concatenates the feature x_{i-1} and x_i and input the concatenated feature (x_{i-1}, x_i) into the pitch predictor and energy predictor. When this concatenated feature passes through the pitch predictor and the energy predictor, the intermediate features (p_{i-1}, p_i) and (e_{i-1}, e_i)

are obtained. By separating the intermediate features obtained above, we obtain p_i and e_i (Figure 3).

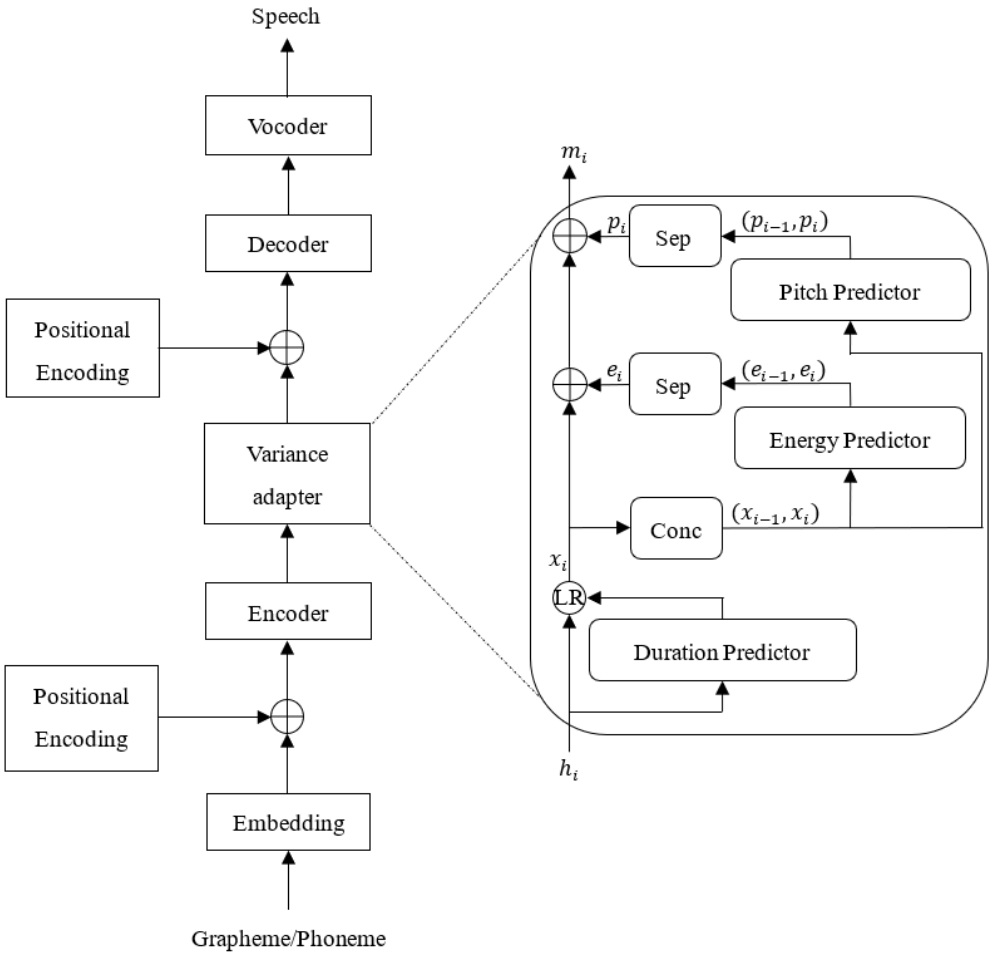

Figure 3. *The Entire Framework of a Model with Improved Variance Adapter.*

For example, suppose the sentence is given by:

e.g. The important thing is that space is not a good place for human beings to live and it's much too cold for us.

Now let us assume that the sentence is divided as follows.

The important thing/is that space is not a good place for human beings to live/and it's much too cold for us.

Then the segments are as follows:

S_1*= The important thing.*

S_2*= is that space is not a good place for human beings to live.*

S_3*= and it's much too cold for us.*

Now, let us synthesize the speech corresponding to S_2. First, when synthesizing the speech corresponding to the first segment S_1, we calculate the intermediate features p_2 and e_2 with the computed features x_1 and the feature x_2 computed from the second segment S_2, and then pass through the decoder and the vocoder to synthesize the speech.

Thus, by taking into account the context of a selected segment as an intermediate feature corresponding to the previous segment, we synthesize the speech corresponding to the current segment.

The encoder contains a self-attention layer, and its complexity is the square of the input length n: $O(n^2)$, so that the computational cost is greatly increased by taking into account the context in the encoder and decoder. However, the proposed method takes into account the context as an intermediate feature of the variance adapter without the self-attention layer, this leads to reducing the computational cost. The problem of reducing the computational cost is important for incremental TTS applications. If the computational cost is very large, the latency will be longer than the time of speech playback in the device, and the disconnection of speech at the boundary between the segments will occur.

However, taking into account context as an intermediate feature, the amount of operation in the acoustic model is less than in the text, and consequently the amount of operation in the acoustic model is reduced. Therefore, reducing the computational cost by taking into account context as an intermediate feature is not merely a problem of reducing the computational time, but also very important for practical applications of incremental TTS.

The overall structure of the proposed TTS model is shown

in Figure 3. Given a grapheme or phoneme sequence, it is entered into the encoder through the embedding layer. To be able to take the order into account, the output of the positional encoding is added to the output of the embedding layer and then input. The decoder produces a mel-spectrogram. The decoder has the same structure as the encoder, only the hyperparameters are different. The last layer of the decoder is the forward layer, whose output dimension is equal to the mel-spectrogram dimension. The generated mel-spectrogram is converted to speech via a vocoder.

3. Experiments and Results

3.1. Experiment Environment

Dataset

The proposed method is trained and evaluated on a custom dataset constructed for Korean TTS. First, we obtain the model by training with the undivided text-speech pair. Then, we segment the text, and then segment of the speech correspondingly, obtaining the intermediate features $x_{i,j}$ corresponding to the segmented sentences. At this time, we obtain the intermediate features $x_{i,j}$ corresponding to the segment in the intermediate feature x_i corresponding to the text before segmenting the intermediate features. That is, the intermediate feature corresponding to the text before sentence is divided according to the duration of the sentence to obtain the intermediate feature corresponding to the sentence. We train the intermediate features corresponding to the segment thus obtained, the segmented sentence and the segmented speech pair. That is, we train the pair $S_{i,j}, V_{i,j}, x_{i,j-1}$.

$S_{i,j}$ is the jth segment of the ith sentence, $V_{i,j}$ is the speech corresponding to $S_{i,j}$, and $x_{i,j-1}$ is the intermediate feature corresponding to $S_{i,j-1}$. The intermediate features are updated with the parameters during training.

This custom dataset consists of 20,010 text-speech pairs of about 25 hours. The sampling frequency is 22.05 kHz, the quantization bits number is 16, and the single-channel data. The dataset is divided into three sub-datasets: 16,010 training datasets, 2,000 test datasets, and 2,000 validation datasets. After segmenting, the dataset consists 70,530 pairs of segmented sentence, segmented speech and intermediate feature. The dataset is divided into three sub-datasets: 56,530 training datasets, 7,000 test datasets, and 7,000 validation datasets. We transform text sequences into phoneme sequences using a grapheme-to-phoneme conversion tool [8].

The speech data are converted into 80-dimensional mel-spectrograms using short-time Fourier transform (STFT). When performing STFT, we used a Hanning window, and the frame size and moving size were set to 1,024 and 256, respectively.

Training and Inference

The proposed model is trained on a single NVIDIA Tesla P100 GPU computer. Adam optimizer is used. The inference is done on a mobile phone equipped with an AArch64 (1.8 GHz) processor, and the vocoder uses the MB-MelGAN model. The model structure used is the same as in [7].

3.2. Model Overview

Table 1 shows the hyperparameters of the proposed model. In the proposed model, the encoder consists of five basic blocks.

Table 1. *Hyperparameters of the Proposed Model.*

Hyperparameter	Value
Phoneme embedding size	256
Number of layers in encoder	5
Number of attention heads in encoder	2, 2, 2, 2, 2
Attention dimension	96, 96, 96, 96, 96
Encoder dimension	128, 128, 128, 128 128
Filter sizes	31, 31, 31, 31, 31
Number of layers in decoder	4
Number of attention heads in decoder	2, 2, 2, 2
Dropout rate	0.1

The hyperparameters of the variance adapter (duration predictor, pitch predictor, energy predictor) are set as in the FastSpeech 2 model.

3.3. Result

We compare our method with the methods of [9, 10]. Table 2 evaluates the proposed method by comparing with the previous models using the mean opinion score.

Table 2. *Mean Opinion Score Evaluation of the Previous and Proposed Methods.*

method	Mean Opinion Score (MOS)
FastSpeech 2	4.32 ±0.084
Method of [9]	3.77 ±0.053
Method of [10]	4.12 ±0.079
Proposed method	4.26 ±0.064

As can be seen from the table, our method outperforms the previous methods.

Next, we evaluate the continuity of the synthesized speech. We evaluate the sentence from the previous example.

The following table shows the comparison of the above text synthesis in the mobile phone equipped with AArch64 (1.8GHz) processor and the mobile phone equipped with MT6580 (1.3GHz) processor, whether the synthetic speech of the previous method and our proposed method is broken or not. We evaluate the continuity of the synthetic speech at the boundary between the segment sentences S_1 and S_2.

Table 3. *Comparison of the Synthetic Tone Continuity of the Proposed Method with the Previous Methods.*

Processor Method	AArch64 (1.8GHz)	MT6580 (1.3GHz)
Method of [9]	○	×
Method of [10]	○	×
Our proposed method	○	○

In the table, "○" denotes the non-breaking of the synthetic speech, and "×" denotes the breaking of the synthetic speech.

As can be seen from the table, the previous methods are able to decouple the synthetic speech in low performance devices, but our method is able to perform continuously without decoupling the synthetic tone even in low performance devices.

4. Conclusion

In this paper, we propose an incremental TTS method that takes into account the intermediate feature-level context to reduce latency for arbitrary length sentences. Through experiments, we have demonstrated that our method improves the synthetic speech quality and computational time over the previous incremental TTS methods.

In the future, we are going to investigate the incremental TTS that can further reduce the latency and provide human-like level of speech quality.

Abbreviations

TTS	Text-To-Speech
E2E	End-To-End
HMM	Hidden Markov Model
RTF	Real Time Factor
Sep	Separate
Conc	Concatenate
MOS	Mean Opinion Score

Funding

The authors declare that no fund and no support were received during the preparation of the research paper.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Pouget, M., Hueber, T., Bailly, G., Baumann, T., "Hmm training strategy for incremental speech synthesis," in 16th Annual Conference of the International Speech Communication Association (Interspeech 2015), pp. 1201–1205, 2015. https://doi.org/10.21437/Interspeech.2015-304

[2] Pouget, M., Nahorna, O., Hueber, T., Bailly, G., "Adaptive latency for part-of-speech tagging in incremental text-to-speech synthesis," in 17th Annual Conference of the International Speech Communication Association, pp. 2846–2850, 2016. https://doi.org/10.21437/Interspeech.2016-165

[3] Baumann, T., Schlangen, D., "Evaluating prosodic processing for incremental speech synthesis," in Thirteenth Annual Conference of the International Speech Communication Association, 2012. https://doi.org/10.21437/Interspeech.2012-152

[4] Baumann, T., "Decision tree usage for incremental parametric speech synthesis," in Proc. of ICASSP, pp. 3819–3823, 2014. https://doi.org/10.1109/ICASSP.2014.6854316

[5] Yanagita, T., Sakti, S., Nakamura, S., "Incremental TTS for Japanese language," in Proc. Interspeech, pp. 902–906, 2018. https://doi.org/10.21437/Interspeech.2018-1561

[6] Yanagita, T., Sakti, S., Nakamura, S., "Neural iTTS: Toward Synthesizing Speech in Real-time with End-to-end Neural Text-to-Speech Framework", In Proc. 10th ISCA Speech Synthesis Workshop, 2019. https://doi.org/10.21437/SSW.2019-33

[7] Yang, G., Yang, S., Liu, K., Fang, P., Chen, W., & Xie, L. (2020). Multi-band melgan: Faster waveform generation for high-quality text-to-speech. In Proceedings of Spoken Language Technology Workshop (SLT). https://doi.org/10.1109/SLT48900.2021.9383551

[8] Ren Y., Hu C., Tan X., Qin T., Zhao S., Zhao Z., & Liu T.-Y. (2021). Fastspeech 2: Fast and high-quality end-to-end text to speech. In Proceedings of international conference on learning representations (ICLR 2021). https://doi.org/10.48550/arXiv.2006.04558

[9] Saeki, T., Takamichi, S., Saruwatari, H. "Incremental Text-to-Speech Synthesis Using Pseudo Lookahead with Large Pretrained Language Model." arXiv preprint arXiv: 2012.12612v2 [cs. SD], 2021. https://doi.org/10.1109/LSP.2021.3073869

[10] Muyang D., Chuan L., Junjie L. "Incremental FastPitch: Chunk-Based High Quality Text To Speech." arXiv preprint arXiv: 2401.01755v1 [cs. SD], 2024. https://doi.org/10.48550/arXiv.2401.01755

[11] Yanagita, T., Sakti, S., Nakamura, S., "Japanese Neural Incremental Text-to-speech Synthesis Framework with an Accent Phrase Input", Volume 11, 22355-22363, IEEE Access, Mar. 2, 2023.

[12] Kayyar, K., Dittmar, C., Pia, N., Habets, E., “Low-resource text-to-speech using specific data and noise augmentation,” in Proc. IEEE-SPS European Signal Processing Conf., pp. 61-65., 2023.

[13] Bataev, V., Ghosh, S., Lavrukhin, V., Li, J., "TTS-Transducer: End-to-End Speech Synthesis with Neural Transducer," arXiv preprint arXiv: 2501.06320v1, 2025.

[14] Chen, Y., Niu, Z., Ma, Z., Deng, K., Wang, C., Zhao, J., Yu, K., Chen, X. "F5-tts: A fairytaler that fakes fluent and faithful speech with flow matching." arXiv preprint arXiv: 2410.06885, 2024.

[15] Shen, K., Ju, Z., Tan, X., Liu, E., Leng, Y., He, L., Qin, T., Zhao, S., Bian, J., "NaturalSpeech 2: Latent Diffusion Models are Natural and Zero-Shot Speech and Singing Synthesizers." In Proc. Intl. Conf. Learning Representations (ICLR), 2024.

[16] Peng, P., Huang, P., Li, D, Mohamed, A., Harwath, D., " Voice-Craft: Zero-Shot Speech Editing and Text-to-Speech in the Wild." arXiv preprint arXiv: 2403.16973, 2024.