Prompt-to-Correct: Automated Test-Time Pronunciation Correction with Voice Prompts

Ayan Kashyap
IIIT Hyderabad, India
ayan.k@research.iiit.ac.in

Neil Shah
IIIT Hyderabad & TCS Research, India
neilkumar.shah@research.iiit.ac.in

Vineet Gandhi
IIIT Hyderabad, India
vgandhi@iiit.ac.in

***Abstract*—Pronunciation correction is crucial for Text-to-Speech (TTS) systems in production. Traditional methods, which rely on phoneme sequence manipulation, are often cumbersome and error-prone. To address this, we propose Prompt-to-Correct, an editing-based methodology for pronunciation correction in TTS systems using voice prompts. Our approach enables accurate, granular corrections at test-time without the need for additional training or fine-tuning. Unlike existing speech editing methods, we eliminate the need for external alignment to determine edit boundaries. By simply providing a correctly-pronounced reading of a word in any voice or accent, our system successfully corrects mispronunciations while maintaining continuity. Experimental results demonstrate that our method outperforms traditional baselines and state-of-the-art speech editing techniques. Speech samples are available at: https://prompt-to-correct.github.io/P2C**

***Index Terms*—speech synthesis, pronunciation control,**

I. Introduction

Correct pronunciation is critical for Text-to-Speech (TTS) systems, particularly when handling out-of-domain vocabulary such as proper nouns or code-mixed [1] text, which is increasingly common in today's multicultural environment. Modern TTS systems [2]–[4] typically follow a two-stage process: first, producing intermediate speech representations from preprocessed text, and second, generating raw waveforms conditioned on these representations. Given that most natural languages lack a one-to-one mapping between text (graphemes) and speech, these systems rely on a linguistic frontend to convert graphemes into phonemes using dictionary lookups or pretrained grapheme-to-phoneme models [5]. However, this often leads to incorrect pronunciations for out-of-domain or foreign words.

To address these mispronunciations, conventional methods involve manual manipulation of phoneme sequences. This approach is cumbersome, requires linguistic expertise, and is prone to errors, as demonstrated in our experiments. An alternative strategy could be to increase the variety of word pronunciations encountered during training or to develop a unified, multilingual TTS system. However, this introduces significant challenges. Training a large, multilingual TTS model capable of handling diverse pronunciations would require vast amounts of text-audio data, significantly complicating model deployment and inflating data collection costs to impractical levels. Moreover, typical TTS corpora have limited word coverage, making a data-centric approach inefficient and unlikely to provide practical improvements.

A more feasible solution is to directly edit the generated speech by replacing mispronounced segments with corrected ones. Recent advancements in text-based speech editing [6]–[9] allow users to modify speech recordings via text transcripts. However, these methods face similar challenges when applied to pronunciation correction because they rely on the text modality for edits. In our approach, we use speech as the source of pronunciation information, which is easier for end-users to provide compared to corrected phoneme sequences.

We introduce Prompt-to-Correct (P2C), a TTS editing system that allows users to correct pronunciations by providing the correct pronunciation as a speech prompt during inference, without requiring any model retraining or fine-tuning. We leverage discrete self-supervised speech codes to model the phonetic content of the speech prompt. Speech codes have been shown to correlate well with phonemes and also relatively free of speaker information. Our contributions include demonstrating that P2C can correct pronunciations using speech from a non-target speaker, regardless of accent, and automating the correction process by eliminating the need for external alignment tools to determine word boundaries. We show that Prompt-to-Correct consistently outperforms traditional phoneme-based correction methods and state-of-the-art speech editing techniques, particularly in multilingual and out-of-domain scenarios.

II. Background

A. Discrete SSL representations for Speech

Self Supervised Learning (SSL) methods for speech representation have shown remarkable results in tasks like Automatic Speech Recognition(ASR) [10], [11], spoken language modeling [12], [13] and speech resynthesis [14]. Wav2Vec2.0 [10] employs a convolutional transformer-based encoder architecture to encode raw audio samples into discrete latent representations, optimizing a contrastive loss. HuBERT [11] is trained with a masked prediction task similar to BERT [15] on masked audio signals. The discrete units are the cluster IDs obtained by clustering the learned features from intermediate layers of the encoder network. In [14], the authors compared various discrete speech representation methods for speech synthesis, assessing their ability to model lexical content and filter out speaker information. They found

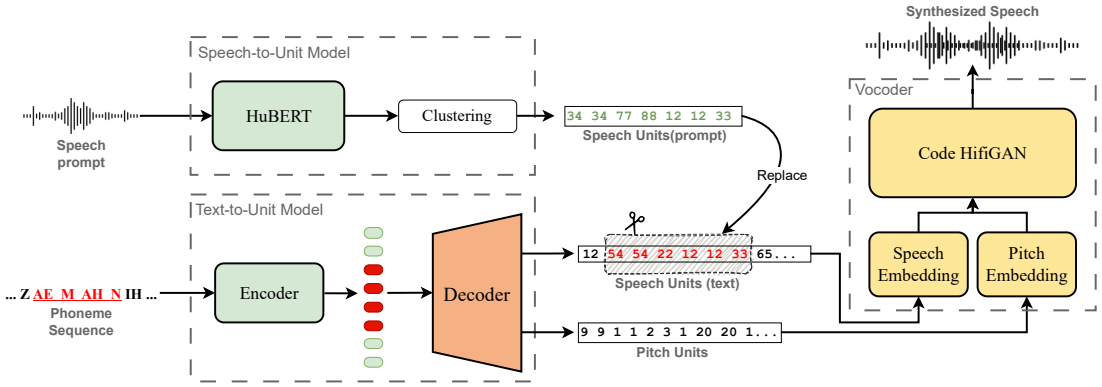

Fig. 1. Overview of the editing pipeline

empirically that HuBERT displays superior disentanglement properties, making it the preferred choice for acoustic representation in our pipeline. This is corroborated by a similar study conducted in [12], where HuBERT scored the highest in the speech generation task.

B. SSL based TTS architectures

Traditional TTS pipelines are cascade systems [2], [3], [16], [17] consisting of an acoustic model for translating text or phoneme sequences into intermediate acoustic representations, and a vocoder [18], [19] for synthesizing the waveform based on these representations. Recently, self-supervised speech representations have emerged as alternatives to mel-spectrograms in these two-stage TTS models. For example, WavThruVec [20] employs wav2vec2.0 embeddings as the intermediate speech representation. It utilizes a transformer-based encoder to map text input to wav2vec embeddings which are then synthesized by HiFi-GAN [19]. Similarly, VQTTS [21] follows a similar setup but uses vector quantized acoustic features [22]. ParrotTTS [4] opts for HuBERT units as the intermediate units, leveraging their disentangled properties [12], [14].

Incorporating a shared embedding space for both speech and text facilitates the potential for seamless multimodal editing. Consequently, the TTS design of Prompt-to-Correct is inspired by the non-autoregressive version of ParrotTTS, adding a pitch encoder and decoder to predict per-phoneme pitch tokens for maintaining prosodic continuity after editing.

C. Speech editing

Speech editing involves manipulating specific words or phrases in original speech while maintaining the naturalness of the overall audio. In recent years, text-based editing systems have seen significant development, enabling editors to modify the text transcript of speech and apply changes to the waveform accordingly. Voco [23], one of the early methods combines a single-speaker TTS model with a voice conversion model to generate the desired speech segment, which is then concatenated with the input speech. More recent approaches have improved by conditioning speech generation on the surrounding context of the edit region. For instance, EditSpeech [6] uses two LSTM decoders to produce acoustic frames from the left and right boundaries of the edit region, which are then fused using a bidirectional fusion module. Campnet [7] and A^{3}T [8] adopt a masked reconstruction objective to capture more complex contextual information. FluentSpeech [9], a diffusion-based speech editing model, achieves state-of-the-art performance on speech editing tasks using the LibriTTS [24] and VCTK [25] datasets.

III. Prompt-To-Correct

This section presents the overarching architectural design and editing pipeline of Prompt-to-Correct (P2C), as illustrated in Figure 1. P2C comprises of two pre-trained and fixed audio encoders i) Speech-to-Unit(S2U), ii) Pitch-to-Unit(P2U); Text-to-Unit(T2U) model and the unit-based vocoder.

A. Audio Encoders

The proposed approach consists of two pretrained and frozen self-supervised audio encoders. The input to both encoders is a raw audio signal, $X = (x_1, ..., x_T)$, where T is the number of samples and X is sampled at 16 kHz. Both encoders are decoupled modules trained separately to predict speech and pitch units respectively.

Speech-to-Unit: The S2U model is a pretrained HuBERT-Base model [11] that encodes speech into a downsampled sequence of embeddings, which are subsequently clustered using the k-means algorithm to obtain discrete units[1], denoted as, $z^{(s)} = (z_1^{(s)}, ..., z_{T'}^{(s)})$, where $T' = T/320$ and each $z_i \in \{1, ..., K\}$ for $1 \leq i \leq T'$ with K being the number of clusters. The resulting $z^{(s)}$ represents the phonetic content of the input speech.

Pitch-to-Unit: We use the fundamental frequency (F0, or pitch) to represent prosody. Previous work by [14] demonstrates that pairing HuBERT units with discrete F0 representations enhances speech re-synthesis by incorporating more prosodic information. The YAAPT algorithm [26] is used to extract the F0 sequence $(f_1, ..., f_{L'})$ from the input signal X.

[1]Throughout this paper, when referring to the S2U model, we encompass both the encoding stage and the clustering stage, unless explicitly stated otherwise

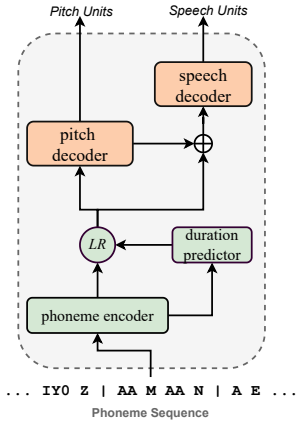

Fig. 2. Detailed architecture of the Text-to-Unit model.

Subsequently, the pitch sequence encoded by P2U to obtain discretized representations, denoted as $z^{(f)} = (z_1^{(f)}, ..., z_{L'}^{(f)})$.

P2U is a VQ-VAE-based network [27], consisting of a convolutional encoder, a bottleneck with a fixed-sized codebook, and a decoder. The encoder extracts latent vectors from the F0 sequence, mapped to the nearest codebook vector within the bottleneck. The decoder then reconstructs the original F0 signal given these mapped latent vectors. The indices of the mapped latent vectors act as pitch units.

B. Text-to-Unit model

The T2U model is a text encoder that maps input phoneme sequence $P = (p_1, ..., p_N)$ to corresponding speech and pitch units $(z^{(s)}, z^{(f)})$. Illustrated in Figure 2, the architecture closely resembles the Non-Autoregressive FastSpeech2 [2] framework, comprising Feed-Forward Transformer Encoder and Decoders. Initially, the input phonemes are encoded into a sequence of phoneme embeddings. The duration predictor, a two-layer 1D-convolutional network, and length regulator (LR) modules ensure alignment between the length of the phoneme embedding sequence and the output unit sequence. Following the expansion of the phoneme embedding sequence, it is first directed to the pitch decoder for decoding corresponding pitch units $z^{(f)}$. The output representation from the pitch decoder is then added back to the encoder output and fed into the speech decoder to generate speech units $z^{(s)}$. These units are then concatenated and fed to the vocoder for waveform synthesis.

C. Unit-based vocoder

For unit-to-speech conversion, we use Code HiFi-GAN, introduced in [14] which takes speech and pitch units $(z^{(s)}, z^{(f)})$ as input. In this setup, each discrete unit corresponds to an entry in their respective embedding look-up tables (LUT) within the generator. These embeddings are upsampled and transformed into the corresponding waveform. The vocoder is also trained separately from the other modules.

D. Editing Pipeline

Figure 1 illustrates our editing pipeline. Given an input text sequence $W = (w_1, ..., w_N)$ where each w_i represents a word, we convert W to a phoneme sequence $P = (p_1, ..., p_{N'})$ using an off-the-shelf phonemizer. P is passed to the T2U model to generate speech units $z^{(s)}$ and pitch units $z^{(f)}$. Now, suppose w_e is the mispronounced word and S_p is a correctly pronounced speech utterance by any speaker. S2U encodes S_p into speech units compatible with the vocoder. We then identify the boundaries of w_e in $z^{(s)}$ using the per-phoneme durations provided by the duration predictor and replace the incorrect speech units with those from the prompt. Finally, the edited speech units along with the originally predicted pitch units are passed to the vocoder to synthesize the corrected speech. The pitch units help ensure that the synthesized speech is continuous and does not contain breaks.

IV. Experimental Setup

A. Implementation Details

We use the LJSpeech [28] single speaker dataset for all our training purposes. Following [2], [16], we convert the text into phoneme sequences using an open-source grapheme-to-phoneme tool [5]. All speech recordings and datasets were resampled to 16 kHz sample rate and preprocessed to remove silences.

Speech-to-Unit: S2U is a HuBERT-BASE model pre-trained for two iterations on 960 hours of the LibriSpeech [29] corpus. Similar to [12], [14], the representations were extracted from the sixth layer. The k-means algorithm is trained on LibriSpeech clean-100h dataset using $K = 100$ centroids. Both the HuBERT and k-means models were publicly available.

Pitch-to-Unit: The P2U model was trained on the LJSpeech dataset for 100k steps, following the methodology described in [14]. F0 values were extracted from the audio using a frame length of 20 ms and a hop size of 5 ms, resulting in a sampling rate of 200Hz. The VQVAE quantization downsamples the signal 16 times, resulting in a sampling rate of 12.5 Hz.

Text-to-Unit: We extract discrete speech and pitch units from LJSpeech using the Unit Encoders to train the T2U model. For most of the model configuration, we follow [4]. We use Montreal Forced Aligner (MFA) to get phoneme-to-unit alignment for the duration predictor. To ensure alignment between speech units and pitch units during training and editing, we dynamically adjust pitch unit sequences by either padding them to match longer speech unit sequences or truncating them symmetrically from the middle to match shorter ones, preserving the temporal coherence between the two modalities. Both decoders are trained using a cross-entropy loss over the discrete units.

Vocoder: We trained a single-speaker Code-HiFiGAN model on LJSpeech for 300K steps. Speech and pitch units for the vocoder are generated using the same S2U and P2U models, respectively.

B. Datasets

Pronunciation-Correction: We curated a diverse dataset for the pronunciation correction study primarily comprising four thematic categories: Indian, French, German, and Medical. We include 20 words from each category with

TABLE I
MOS(↑) EVALUATION FOR PRONUNCIATION QUALITY ON THE PRONUNCIATION-CORRECTION DATASET

Category	Model	MOS(↑)
TTS	FastSpeech2-Std	1.86
	FastSpeech2-Alt	2.82
	ParrotTTS-Std	1.90
	ParrotTTS-Alt	2.92
Editing	FluentSpeech-Std	2.24
	FluentSpeech-Alt	3.12
	Prompt-to-Correct (Ours)	**4.11**

accompanying speech recordings from online pronunciation dictionaries to guarantee precise phonetic representations. The selected words are either nouns or commonly used code-mixing phrases. We construct two sentences for each word, resulting in a total corpus size of 160. Our deliberate choice of Indian, French, and German languages, in addition to English, allows for a gradient of linguistic divergence, enabling a thorough evaluation of the method across varying degrees of linguistic dissimilarity and arbitrary speakers and accents. Furthermore, including medical terminology, known for its compound nature, enriches the dataset and enhances the complexity of the study.

C. Baselines

We compare P2C with the following methods: FastSpeech2 (Std/Alt), ParrotTTS (Std/Alt), and FluentSpeech (Std/Alt). FastSpeech2 and ParrotTTS are well-established TTS architectures trained on the LJSpeech dataset. In the standard (Std) setting, the target word is phonemized using a conventional G2P model, while in the alternate (Alt) setting, the phoneme sequence of the target word is predicted using a pretrained phoneme prediction model (Wav2Vec2Phoneme [30]) on the reference pronunciation. These TTS baselines allow us to evaluate the effectiveness of phoneme manipulation within traditional TTS pipelines. FluentSpeech is a state-of-the-art speech editing method originally designed to modify grapheme sequences for editing. Although FluentSpeech was pretrained on LibriTTS, which differs from the LJSpeech dataset used for training the TTS models, this discrepancy is not critical as the pronunciation dataset we are evaluating on includes random speakers and diverse accents. Since FluentSpeech also relies on a G2P model, we adapted the method with a custom wrapper to enable direct phoneme sequence modifications, ensuring a fair comparison. For evaluation, we simulate the correction process by first generating TTS samples using a speaker from the set of LibriTTS speakers and then attempting to correct the pronunciation errors using FluentSpeech. FluentSpeech-Std refers to its original grapheme-based correction, while FluentSpeech-Alt refers to our phoneme-level adaptation. The inclusion of FluentSpeech in both Std and Alt settings allows us to demonstrate that even a sophisticated editing approach struggles to achieve reliable pronunciation correction.

D. Pronunciation correction evaluation setup

To evaluate the efficacy of P2C's pronunciation correction, we conducted a reference-based listening test in which 40 participants were presented with synthesized speech of sentences from the Pronunciation-Correction dataset. Each sentence included a specific word that participants were instructed to focus on for pronunciation accuracy. A reference pronunciation for each target word was established to serve as the benchmark for evaluation. Participants were presented with one sentence at a time and to prevent order bias, the order of the samples were randomized for each participant. Each method was ranked on a 5-point Likert scale, with 1 indicating a poor match with the reference pronunciation and 5 indicating a close match.

V. RESULTS AND DISCUSSION

Table I presents the results of the MOS evaluation for pronunciation quality on the Pronunciation-Correction dataset. P2C significantly outperforms all other baselines, achieving a MOS score of 4.11. In contrast, the "standard" TTS models, FastSpeech2-Std and ParrotTTS-Std, struggle with accurately pronouncing complex and out-of-domain words, yielding low MOS scores of 1.86 and 1.90, respectively. The "alternate" versions of these models perform better with scores of 2.82 for FastSpeech2-Alt and 2.92 for ParrotTTS-Alt. However, their improvements are marginal, and they still face challenges with pronunciations from languages distant from English, such as Indian and French.

FluentSpeech-Alt, performs better than the other baseline methods, achieving a MOS score of 3.12. However, it remains limited by its grapheme/phoneme editing approach, which hinders its ability to correct pronunciation effectively across diverse linguistic contexts. Additionally, FluentSpeech requires an extra alignment step using a forced aligner to determine the boundaries of the edit. In contrast, P2C benefits from automatic boundary determination through its duration predictor, enhancing its flexibility and accuracy in pronunciation correction.

VI. CONCLUSION

In this work, we introduced Prompt-to-Correct (P2C), a TTS system designed to correct pronunciations using speech inputs. Our experiments show that P2C can effectively correct pronunciations from non-target speakers and various accents without model retraining or fine-tuning. By Leveraging advancements in modular TTS architectures and self-supervised representations, P2C harnesses a shared embedding space to facilitate seamless editing across both text and speech modalities. We address prevalent challenges associated with phonetic control in existing TTS and editing systems by offering a more intuitive and user-friendly approach. P2C facilitates multilingual code-switching and ensures accurate pronunciations for out-of-distribution words without extensive retraining. Future work will focus on addressing limitations in external alignment methods, such as duration prediction and prosodic variability.

REFERENCES

[1] C. Myers-Scotton, *Duelling languages: Grammatical structure in codeswitching*. Oxford University Press, 1997.

[2] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech 2: Fast and high-quality end-to-end text to speech," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=piLPYqxtWuA

[3] E. Casanova, J. Weber, C. D. Shulby, A. C. Junior, E. Gölge, and M. A. Ponti, "Yourtts: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone," in *International Conference on Machine Learning*. PMLR, 2022, pp. 2709–2720.

[4] N. Shah, S. Kosgi, V. Tambrahalli, N. Sahipjohn, N. Pedanekar, and V. Gandhi, "Parrottts: Text-to-speech synthesis by exploiting self-supervised representations," *arXiv preprint arXiv:2303.01261*, 2023.

[5] J. Park, Kyubyong Kim, "g2pe," https://github.com/Kyubyong/g2p, 2019.

[6] D. Tan, L. Deng, Y. T. Yeung, X. Jiang, X. Chen, and T. Lee, "Editspeech: A text based speech editing system using partial inference and bidirectional fusion," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2021, pp. 626–633.

[7] T. Wang, J. Yi, R. Fu, J. Tao, and Z. Wen, "Campnet: Context-aware mask prediction for end-to-end text-based speech editing," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 2241–2254, 2022.

[8] H. Bai, R. Zheng, J. Chen, M. Ma, X. Li, and L. Huang, "A ̋3 t: Alignment-aware acoustic and text pretraining for speech synthesis and editing," in *International Conference on Machine Learning*. PMLR, 2022, pp. 1399–1411.

[9] Z. Jiang, Q. Yang, J. Zuo, Z. Ye, R. Huang, Y. Ren, and Z. Zhao, "Fluentspeech: Stutter-oriented automatic speech editing with context-aware diffusion models," in *Findings of the Association for Computational Linguistics: ACL 2023*, 2023, pp. 11 655–11 671.

[10] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in neural information processing systems*, vol. 33, pp. 12 449–12 460, 2020.

[11] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.

[12] K. Lakhotia, E. Kharitonov, W.-N. Hsu, Y. Adi, A. Polyak, B. Bolte, T.-A. Nguyen, J. Copet, A. Baevski, A. Mohamed *et al.*, "On generative spoken language modeling from raw audio," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 1336–1354, 2021.

[13] E. Kharitonov, A. Lee, A. Polyak, Y. Adi, J. Copet, K. Lakhotia, T.-A. Nguyen, M. Rivière, A. Mohamed, E. Dupoux *et al.*, "Text-free prosody-aware generative spoken language modeling," *arXiv preprint arXiv:2109.03264*, 2021.

[14] A. Polyak, Y. Adi, J. Copet, E. Kharitonov, K. Lakhotia, W.-N. Hsu, A. Mohamed, and E. Dupoux, "Speech Resynthesis from Discrete Disentangled Self-Supervised Representations," in *Proc. Interspeech 2021*, 2021, pp. 3615–3619.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[16] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 4779–4783.

[17] Y. Ren, J. Liu, and Z. Zhao, "Portaspeech: Portable and high-quality generative text-to-speech," *Advances in Neural Information Processing Systems*, vol. 34, pp. 13 963–13 974, 2021.

[18] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[19] J. Kong, J. Kim, and J. Bae, "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis," *Advances in neural information processing systems*, vol. 33, pp. 17 022–17 033, 2020.

[20] H. Siuzdak, P. Dura, P. van Rijn, and N. Jacoby, "WavThruVec: Latent speech representation as intermediate features for neural speech synthesis," in *Proc. Interspeech 2022*, 2022, pp. 833–837.

[21] C. Du, Y. Guo, X. Chen, and K. Yu, "VQTTS: High-Fidelity Text-to-Speech Synthesis with Self-Supervised VQ Acoustic Feature," in *Proc. Interspeech 2022*, 2022, pp. 1596–1600.

[22] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," *arXiv preprint arXiv:1910.05453*, 2019.

[23] Z. Jin, G. J. Mysore, S. Diverdi, J. Lu, and A. Finkelstein, "Voco: Text-based insertion and replacement in audio narration," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–13, 2017.

[24] H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen, and Y. Wu, "Libritts: A corpus derived from librispeech for text-to-speech," *arXiv preprint arXiv:1904.02882*, 2019.

[25] C. Veaux, J. Yamagishi, K. MacDonald *et al.*, "Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit," *University of Edinburgh. The Centre for Speech Technology Research (CSTR)*, vol. 6, p. 15, 2017.

[26] K. Kasi and S. A. Zahorian, "Yet another algorithm for pitch tracking," in *2002 ieee international conference on acoustics, speech, and signal processing*, vol. 1. IEEE, 2002, pp. I–361.

[27] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[28] K. Ito and L. Johnson, "The lj speech dataset," https://keithito.com/LJ-Speech-Dataset/, 2017.

[29] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[30] Q. Xu, A. Baevski, and M. Auli, "Simple and effective zero-shot cross-lingual phoneme recognition," *arXiv preprint arXiv:2109.11680*, 2021.