SpeakStream: Streaming Text-to-Speech with Interleaved Data

Anonymous submission to Interspeech 2025

Abstract

There has been an increasing integration of speech front-ends and large language models (LLM) with end-to-end models but cascaded models that stream LLM outputs to text-to-speech (TTS) systems remain surprisingly under-explored despite their simplicity. Using traditional TTS to convert LLM outputs to audio, however, poses a technical problem because entire utterances are needed to generate stylistic audio. In this paper we present a streaming TTS (SpeakStream) that can generate audio incrementally from streaming text using a decoder-only architecture. The model is trained using next-step prediction loss on force-aligned, interleaved text-speech data. During inference SpeakStream generates speech incrementally while absorbing streaming text, making it suitable for cascaded conversational AI agents where an LLM streams text to a TTS system. Our experiments show that SpeakStream matches batch TTS quality while enabling streaming capabilities.

Index Terms: text-to-speech, speech synthesis, streaming

1. Introduction

Recent years have witnessed a surge of interest in speech interfaces for large language models (LLMs). While substantial research has focused on end-to-end models where LLMs directly generate tokenized audio [1, 2], studies indicate that cascaded models—which stream text from LLMs to text-to-speech (TTS) systems—consistently outperform end-to-end approaches [3].

A primary challenge in cascaded models is reducing latency from two sources: (1) waiting for the LLM to generate a complete text segment (e.g. sentence) and (2) waiting for the TTS system to generate audio. Although some autoregressive TTS models [4, 5] can generate audio incrementally, most systems [4, 6–12] require complete text segments before producing audio—introducing significant latency that compromises interactive applications.

To address text-waiting latency, recent approaches [13, 14] have explored partial text context windows to balance responsiveness and quality. However, these streaming methods often struggle with long-range dependencies and require careful tuning of text-speech alignment [13, 15]. Furthermore, the separation between text encoding and speech generation processes can lead to suboptimal context utilization, particularly when modeling prosody across sentence boundaries.

To overcome these limitations, we propose SpeakStream—a novel decoder-only architecture that enables streaming TTS through interleaved text-speech modeling. Our approach offers three key advantages: (1) unified context modeling across modalities, (2) elimination of explicit alignment mechanisms during inference, and (3) efficient computation through kv-caching. We use a force-aligner to temporally align text and speech segments during training, creating interleaved text-speech sequences. The transformer decoder learns to predict the next speech segment conditioned on the current text segment, previous speech segments, and previous text segments. This creates a coherent, unified context that captures both modalities, allowing the model to maintain complete information about previously generated speech while processing incoming text.

By training SpeakStream using an autoregressive loss on the interleaved data, we can handle streaming input naturally without complex architectural modifications or explicit context management schemes. Our decoder-only design enables the model to use kv-cache to store all historical context, ensuring efficient inference with low latency. This eliminates the need for traditional encoder-decoder structures, resulting in a simpler, more efficient architecture while maintaining high-quality speech synthesis. Our empirical results demonstrate SpeakStream's effectiveness: automatic evaluation shows it achieves the lowest error rate across all latency configurations, while human evaluators rate its coherence comparable to non-streamed systems like RichTTS [4]. By deploying SpeakStream to a MacBook, we show it achieves 50ms TTS latency, making it suitable for real-time interactive applications.

2. Related Work

Traditional TTS systems [7–11] process complete text to generate complete audio. However, the rise of conversational AI demands reduced latency through dual streaming capabilities, i.e., streaming text input and streaming audio output.

Dual Streaming TTS mimics how humans read aloud a text stream as it unfolds. Modeling this behavior with neural networks presents several challenges. First, when synthesizing speech streamingly, creating smooth transitions between audio chunks while avoiding artifacts is difficult. Autoregressive speech generation offers a promising solution, as demonstrated in models such as RichTTS [4] and VALL-T [16].

Another significant challenge is that TTS models with text encoders struggle to handle streaming text input. These models typically need to re-encode the text sequence when new content arrives—a limitation affecting FastSpeech2's transformer encoder, Tacotron's LSTM encoder, and E3 TTS's BERT [17] encoder. All these architectures face difficulty synthesizing natural speech with limited context.

[14] uses non-attentive Tacotron [18] for speech generation by distilling from a non-streaming TTS with limited access to future context. However, their architecture demonstrates limited zero-shot capability. [13] upgraded LiveSpeech [5] from full-text audio synthesis to text-chunk synthesis. However, their model encounters misalignment issues between speech gener-

ation and text chunks. Although they introduce a CTC-ASR model to generate graphemes and guide chunk generation, this approach complicates the overall architecture and potentially introduces train-test mismatching issues.

Transducer-based TTS approaches [19] offer improved alignment design, but their application in dual-streaming settings remains under-explored.

In this work, we propose SpeakStream, a decoder-only dual-streaming TTS model. By training a decoder-only transformer model with interleaved speech-text sequences, SpeakStream can store all generated chunks in its key-value cache, providing complete context without information loss. By predicting EOS tokens as chunk boundaries, our model avoids misalignment issues and eliminates the need for CTC aligners during inference. Furthermore, by controlling the interleaving window size, our model can attend to both current and future text chunks, ensuring minimal latency until the first audio chunk while maintaining smooth transitions for subsequent chunks.

Concurrently, [15] also found that decoder-only TTS models are suitable for dual-streaming synthesis. Unlike our model, they interleave text and speech with a fixed ratio rather than using alignment information, which could lead to complicated attention patterns when there are large variations in speaking rate. Further it might be difficult to precisely tie the streaming audio to the corresponding input text, which can be useful in interactive applications where a conversational agent is interruptible.

3. Method

Our approach enables streaming text-to-speech by introducing an interleaved representation of text and speech tokens in a decoder-only architecture. The model processes these interleaved sequences to generate speech output with low latency. This section describes our model architecture, token representations, interleaving schemes, and inference method.

3.1. Text and Speech Representation

Text Token Representation: We use character-level embeddings to capture fine-grained linguistic features. In this way, each word w_t consists of x character embeddings $c_{t_1}, c_{t_2}, \cdots, c_{t_x}$.

Speech Token Representation: We adopt the dMel [4] tokenization approach, which discretizes mel-filterbank channels into discrete intensity bins. This simple yet effective discretization method preserves both acoustic and semantic information, making it ideal for our streaming synthesis task. For each word w_t, its corresponding audio chunk s_t consists of y dMel embeddings $f_{t_1}, f_{t_2}, \cdots, f_{t_y}$.

3.2. SpeakStream Model

SpeakStream is built upon a vanilla transformer decoder architecture similar to RichTTS [4]. Compared to RichTTS, the key difference is how SpeakStream constructs the transformer input sequence. In RichTTS [4], the input to the model is:

$$[w_{\text{bos}}, w_1, \cdots, w_t, w_{\text{eos}}, s_{\text{bos}}, s_1, \cdots, s_t, s_{\text{eos}}]$$

where w_{bos} and w_{eos} are the beginning and end text embeddings, s_{bos} and s_{eos} are the beginning and end speech embeddings. t is the number of words.

SpeakStream, on the other hand interleaves the sequence above by inserting speech between text. A simplified illustration of the input to SpeakStream is:

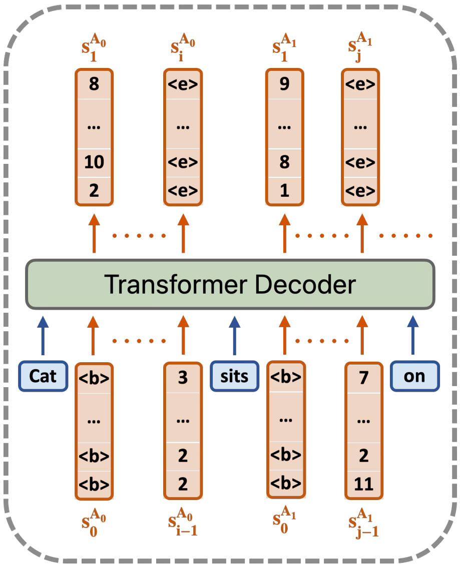

Figure 1: *SpeakStream Architecture.*

$$\begin{aligned} &[T_1, A_1, T_2, A_2, \cdots, T_x, A_x] \\ &T_i = w_i, \quad A_i = s_{\text{bos}}, s_i, s_{\text{eos}}, \quad x = t \end{aligned}$$

where text and speech are interleaved one by one. To establish precise temporal correspondence between words and speech frames, we utilize the A3T's alignment mechanism [6].

By training a decoder-only transformer model on such interleaved sequences, SpeakStream learns to synthesize the current speech segment A_i conditioned on the current text segment T_i, previous speech segments $A_{<i}$, and previous text segments $T_{<i}$. Compared to existing streaming solutions that synthesize each text chunk independently, SpeakStream offers several advantages:

1. Each speech segment A_i is conditioned on the complete speech history $A_{<i}$, ensuring acoustic coherence and consistency;
2. Each speech segment A_i is conditioned on the complete text history $T_{<i}$, maintaining semantic precision;
3. With access to comprehensive speech and text history, our model achieves high-quality streaming synthesis even with short text segments T_i, enabling lower latency;
4. The model's kv-cache architecture efficiently maintains all historical context during inference, significantly reducing computational overhead and enabling fast generation;
5. The model eliminates the need for a force aligner during inference by predicting the EOS token for each speech segment and processing text segments accordingly.

3.3. Interleaving Schemes

There is a trade-off between latency and accuracy of streaming TTS. To lower the latency, the length of each T_i should be short. However, given the existence of polyphonic words, the speech synthesis of certain words must consider not only the preceding words but also the subsequent words in the context sequence. Therefore, $T_{<=i}$ should have additional words beyond A_i's corresponding words to maintain synthesis accuracy.

To address this trade-off, we design two interleaving schemes, with each scheme having multiple variants by adjusting the **text window length** m and **speech hop length** n, where $1 \leq n \leq m$ and $m \geq 1$. This ensures the first n words of T_i correspond to A_i, while the remaining $(m-n)$ words provide future context.

Scheme 1

$$[T_1, A_1, T_2, A_2, \cdots, T_x, A_x]$$

$$T_i = w_{n(i-1)+1}, \cdots, w_{min(t,n(i-1)+m)}$$

$$A_i = s_{\text{bos}}, s_{n(i-1)+1}, \cdots, s_{min(t,n \cdot i)}, s_{\text{eos}}, \qquad x = \left\lceil \frac{t}{n} \right\rceil$$

Example (m=3, n=2, t=8):

$$[w_1, w_2, w_3, s_{\text{bos}}, s_1, s_2, s_{\text{eos}}, w_3, w_4, w_5, s_{\text{bos}}, s_3, s_4, s_{\text{eos}}, w_5, w_6, w_7, s_{\text{bos}}, s_5, s_6, s_{\text{eos}}, w_7, w_8, s_{\text{bos}}, s_7, s_8, s_{\text{eos}}]$$

As shown in Scheme 1, we repeat text tokens to ensure the first n words of T_i correspond to A_i, while the remaining $(m-n)$ words provide future context.

Scheme 2

$$[T_1, A_1, T_2, A_2, \cdots, T_x, A_x, A_{x+1}, \cdots, A_y]$$

$$T_i = \begin{cases} w_1, \cdots, w_m & i = 1 \\ w_{n(i-2)+m+1}, \cdots, w_{min(t,n(i-1)+m)} & i > 1 \end{cases}$$

$$A_i = s_{\text{bos}}, s_{n(i-1)+1}, \cdots, s_{min(t,n \cdot i)}, s_{\text{eos}}$$

$$x = \left\lceil \frac{t-m}{n} \right\rceil + 1, \quad y = \left\lceil \frac{t}{n} \right\rceil$$

Example (m=3, n=2, t=8):

$$[w_1, w_2, w_3, s_{\text{bos}}, s_1, s_2, s_{\text{eos}}, w_4, w_5, s_{\text{bos}}, s_3, s_4, s_{\text{eos}}, w_6, w_7, s_{\text{bos}}, s_5, s_6, s_{\text{eos}}, w_8, s_{\text{bos}}, s_7, s_8, s_{\text{eos}}]$$

Scheme 2 maintains the same $(m-n)$ contextual words but avoids token repetition, creating a more compact representation at the cost of more complex attention patterns.

3.4. Streaming Inference

During inference, our model enables true streaming generation through an autoregressive process that maintains the interleaved structure. Given a streaming text sequence, the model:

1. Waits until receiving the first segment of text words and generates their corresponding dMel tokens, which are converted into waveforms in tandem, using a streaming Mel-to-wave vocoder;
2. Uses both the generated speech features and the next segment of text words as context for generating the next speech segment;
3. Repeats this process until the entire text is synthesized.

The simplicity of dMel tokenization allows our model to generate high-quality speech in a streaming fashion without the complexity of managing multiple token types or separate acoustic and semantic representations. The primary latency in our system comes from accumulating the first segment of text words, making the granularity parameter m a direct control for the latency-quality trade-off.

4. Experiments

4.1. Setup

We conduct experiments using the LJSpeech [21] dataset, which consists of single-speaker English audio recordings at 22kHz with read speech from LibriVox. Following RichTTS [4], our model comprises 36 layers of transformer decoder with 258M parameters. The dMel feature is exactly the same as [4] with 25ms hop length, 16 bins, and ParallelWaveGAN vocoder [22]. The baseline models are RichTTS [4] and XTTS [20], which are trained on complete text and audio pairs. We also train multiple n-gram versions of RichTTS, where the models are trained with short segments. For TTS evaluation, we utilize WhisperX ("base.en") [23, 24] to transcribe our generated speech into text and calculate the Word Error Rate (WER).

4.2. Main Results

Our main results are presented in Table 1. The experiments show that directly applying RichTTS to streaming segments significantly degrades generation quality, with WER exceeding 68% for unigram word synthesis.

XTTS performs even worse, with WER exceeding 222% for unigram word synthesis. Upon investigating its generation, we find that XTTS hallucinates words and phonemes, leading to high insertion errors. Although these two models perform well in full text synthesis, with WER below 4%, they are not suitable for streaming synthesis. For RichTTS trained with short segments, the WER is significantly reduced, but it still remains above 60% for unigram word synthesis. In contrast, SpeakStream achieves WER around 7% for unigram word synthesis, and below 5% when more context is provided. Importantly, SpeakStream's performance is comparable to RichTTS's full text synthesis when $m = 5$ and $n = 1$. This result demonstrates that SpeakStream can achieve high-quality streaming synthesis with interleaved text and speech inputs.

For SpeakStream, Scheme 1 (S1) interleaving consistently outperforms Scheme 2 (S2). This performance difference stems from the varying speech durations of each A_i, which impacts S2 more substantially than S1. With S1, the model can easily locate the n corresponding words in T_i adjacent to A_i, while the remaining $(m-n)$ words provide supplementary pronunciation context. In contrast, S2 requires more complex attention patterns as A_i's corresponding words are separated by variable-length gaps determined by A_{i-1}'s duration. Based on these findings, we adopt S1 as SpeakStream's interleaving scheme for subsequent analysis.

The results reveal that configurations where $m = n$ yield higher WER, indicating that additional text tokens enhance segment synthesis quality. Performance improves notably when $(m-n) > 1$. As expected, increasing m generally improves performance by providing richer context. The optimal configuration occurs at $m = 5, n = 1$, achieving 3.38 WER, comparable to RichTTS's non-streaming performance. However, performance deteriorates beyond $m = 5$ due to the $(m-n)$ word repetition creating excessively long text sequences, suggesting that overly large context windows don't necessarily improve accuracy for SpeakStream.

4.3. Human Evaluation

We conducted a human evaluation to assess the quality of synthesized speech from different streaming models. We randomly sampled 100 segments from the LJSpeech dev set and asked human evaluators to rate the naturalness and coherence

Table 1: *WER of SpeakStream with Scheme 1 (S1) and Scheme 2 (S2) evaluated by WhisperX ASR (base.en). The WER of groudtruth audio is 2.09.*

	Hop n=1		Hop n=2		Hop n=3		Hop n=4		Hop n=5		Hop n=6		∞
XTTS-V2 [20]	222.28		45.01		30.92		28.97		17.13		15.28		**3.67**
RichTTS [4]	68.18		30.20		20.30		16.81		13.07		10.55		**3.28**
ngram-RichTTS	60.4		26.58		21.04		12.06		6.64		7.76		**3.28**
SpeakStream	S1	S2	S1	S2	S1	S2	S1	S2	S1	S2	S1	S2	
Window m=1	7.47		-		-		-		-		-		-
Window m=2	**4.50**	5.26	7.18		-		-		-		-		-
Window m=3	**3.99**	6.88	**4.19**	5.11	4.78		-		-		-		-
Window m=4	**3.88**	6.16	**3.80**	5.09	**4.36**	5.35	5.21		-		-		-
Window m=5	**3.38**	5.59	**3.61**	6.09	**3.65**	4.82	4.73	**4.59**	4.52		-		-
Window m=6	**4.26**	6.20	**3.61**	4.36	**3.93**	4.52	**4.36**	6.14	**4.86**	4.95	4.30		-

Table 2: *Human evaluation results for synthesized speech naturalness and coherence (Mean and standard deviation).*

	NonStreaming	Streaming (m=4)	Streaming (m=6)
		Naturalness	
GroundTruth	**4.4** $\pm$ 0.1	-	-
RichTTS	3.8 $\pm$ 0.1	2.2 $\pm$ 0.1	2.5 $\pm$ 0.1
XTTS	3.9 $\pm$ 0.1	2.1 $\pm$ 0.1	2.5 $\pm$ 0.1
SpeakStream	-	**3.7** $\pm$ 0.1	**3.5** $\pm$ 0.1
		Coherence	
GroundTruth	**4.2** $\pm$ 0.0	-	-
RichTTS	3.9 $\pm$ 0.1	2.3 $\pm$ 0.1	2.2 $\pm$ 0.1
XTTS	4.1 $\pm$ 0.1	1.8 $\pm$ 0.1	2.7 $\pm$ 0.1
SpeakStream	-	**3.9** $\pm$ 0.1	**3.8** $\pm$ 0.1

Table 3: *Latency (ms) of SpeakStream with S1, tested with Apple Silicon (15-inch, M3, 24GB, 2024).*

SpeakStream	TTS	Vocoder	Total
Window m=1	$51_{\pm 14}$	$326_{\pm 61}$	$402_{\pm 55}$
Window m=2	$51_{\pm 17}$	$353_{\pm 101}$	$430_{\pm 104}$
Window m=3	$64_{\pm 19}$	$395_{\pm 140}$	$494_{\pm 166}$
Window m=4	$50_{\pm 13}$	$426_{\pm 192}$	$501_{\pm 192}$
Window m=5	$63_{\pm 13}$	$513_{\pm 235}$	$600_{\pm 230}$

of each segment on a scale of 1 to 5. Each segment was evaluated by 7 random annotators. We report the average scores and standard deviations for each model. We evaluated 4-gram and 6-gram versions of RichTTS and XTTS, corresponding to $(m = 4, n = 2)$ and $(m = 6, n = 2)$ configurations of SpeakStream. The results are shown in Table 2.

We observed that non-streaming TTS results are similar between RichTTS and XTTS, with XTTS slightly outperforming RichTTS in both naturalness and coherence. However, when applied in streaming settings, XTTS's performance drops significantly, especially in coherence. For streaming models, surprisingly, human evaluators rated SpeakStream as coherent as non-streaming RichTTS, suggesting that SpeakStream successfully maintains coherence despite the model's streaming nature.

4.4. Latency Analysis

We conducted latency analysis of SpeakStream with streaming text input and streaming audio output. Our implementation features sequential word input to the TTS model, while the streaming output pipeline consists of streaming frame generation from TTS, streaming waveform synthesis from the vocoder, and real-time audio player. We implemented the complete system using MLX [25] and deployed it on a 15-inch MacBook Air 2024 with M3 Apple Silicon (24GB RAM). For evaluation, we randomly sampled 25 sentences from the LibriSpeech [26] dev-clean set and measured the performance of SpeakStream models trained with configuration S1, n=1. We report three latency metrics:

1. **Total latency**: Time elapsed between the TTS model receiving its first word and the audio player outputting the first waveform chunk.
2. **Vocoder latency**: Time elapsed between the vocoder receiving its first frame input and generating the first waveform.
3. **TTS latency**: Time elapsed between the TTS model receiving its first word and generating its first frame.

Note, we are interested not in time when the first waveform outputted but rather in time of the first spoken phoneme. *We observe* < 50*ms additional latency for the first spoken phoneme to be produced by SpeakStream.* Also, it should be noted that this paper primarily focuses on reducing TTS latency. In real conversational agents, the token generation time should also be considered; however, since this depends on the LLM's size and inference infrastructure, we use word count rather than milliseconds to measure this component.

Our results in Table 3 demonstrate that SpeakStream achieves low latency, requiring only around 50ms to generate the first frame. Also, SpeakStream only waiting m words before it starts the generation process. This efficiency remains consistent across all configurations. The total response time ranges from 0.4 to 0.6 seconds, which is also favorable for interactive applications. The primary bottleneck is the vocoder latency, as our implementation uses ParallelWaveGAN [22], which requires 10 frames before generating audio output. Recent streaming vocoder [27] should improve latency in this scenario – we leave this for future exploration.

5. Conclusion

We presented SpeakStream, a decoder-only streaming TTS system that enables real-time speech synthesis through interleaved text-speech modeling. Our extensive experiments demonstrate that SpeakStream successfully bridges the quality gap between streaming and non-streaming TTS systems, achieving WER comparable to full-text synthesis while operating in a streaming fashion. The system's ability to maintain coherence across segments, as confirmed by human evaluations, makes it a promising solution for interactive applications where both responsiveness and naturalness are critical. Future work could explore extending this approach to multi-speaker settings, larger datasets, and cross-lingual applications.

6. References

[1] Z. Borsos, R. Marinier, D. Vincent, E. Kharitonov, O. Pietquin, M. Sharifi, D. Roblek, O. Teboul, D. Grangier, M. Tagliasacchi *et al.*, "Audiolm: a language modeling approach to audio generation," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 31, pp. 2523–2533, 2023.

[2] A. Défossez, L. Mazaré, M. Orsini, A. Royer, P. Pérez, H. Jégou, E. Grave, and N. Zeghidour, "Moshi: a speech-text foundation model for real-time dialogue," *arXiv preprint arXiv:2410.00037*, 2024.

[3] T. A. Nguyen, B. Muller, B. Yu, M. R. Costa-Jussa, M. Elbayad, S. Popuri, C. Ropers, P.-A. Duquenne, R. Algayres, R. Mavlyutov *et al.*, "Spirit-lm: Interleaved spoken and written language model," *Transactions of the Association for Computational Linguistics*, vol. 13, pp. 30–52, 2025.

[4] H. Bai, T. Likhomanenko, R. Zhang, Z. Gu, Z. Aldeneh, and N. Jaitly, "dmel: Speech tokenization made simple," *arXiv preprint arXiv:2407.15835*, 2024.

[5] T. Dang, D. Aponte, D. Tran, and K. Koishida, "Livespeech: Low-latency zero-shot text-to-speech via autoregressive modeling of audio discrete codes," *arXiv preprint arXiv:2406.02897*, 2024.

[6] 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 *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 1399–1411. [Online]. Available: https://proceedings.mlr.press/v162/bai22d.html

[7] 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.

[8] Z. Du, Q. Chen, S. Zhang, K. Hu, H. Lu, Y. Yang, H. Hu, S. Zheng, Y. Gu, Z. Ma *et al.*, "Cosyvoice: A scalable multilingual zero-shot text-to-speech synthesizer based on supervised semantic tokens," *arXiv preprint arXiv:2407.05407*, 2024.

[9] Y. Gao, N. Morioka, Y. Zhang, and N. Chen, "E3 tts: Easy end-to-end diffusion-based text to speech," in *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2023, pp. 1–8.

[10] 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," *arXiv preprint arXiv:2006.04558*, 2020.

[11] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, "Tacotron: Towards end-to-end speech synthesis," *arXiv preprint arXiv:1703.10135*, 2017.

[12] OpenAI. (2024) Text-to-speech guide. [Online]. Available: https://platform.openai.com/docs/guides/text-to-speech

[13] T. Dang, D. Aponte, D. Tran, T. Chen, and K. Koishida, "Zero-shot text-to-speech from continuous text streams," *arXiv preprint arXiv:2410.00767*, 2024.

[14] A. Dekel, S. Shechtman, R. Fernandez, D. Haws, Z. Kons, and R. Hoory, "Speak while you think: Streaming speech synthesis during text generation," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 11 931–11 935.

[15] Y. Yang, Z. Ma, S. Liu, J. Li, H. Wang, L. Meng, H. Sun, Y. Liang, R. Xu, Y. Hu *et al.*, "Interleaved speech-text language models are simple streaming text to speech synthesizers," *arXiv preprint arXiv:2412.16102*, 2024.

[16] C. Du, Y. Guo, H. Wang, Y. Yang, Z. Niu, S. Wang, H. Zhang, X. Chen, and K. Yu, "Vall-t: Decoder-only generative transducer for robust and decoding-controllable text-to-speech," *arXiv preprint arXiv:2401.14321*, 2024.

[17] M. V. Koroteev, "Bert: a review of applications in natural language processing and understanding," *arXiv preprint arXiv:2103.11943*, 2021.

[18] J. Shen, Y. Jia, M. Chrzanowski, Y. Zhang, I. Elias, H. Zen, and Y. Wu, "Non-attentive tacotron: Robust and controllable neural tts synthesis including unsupervised duration modeling," *arXiv preprint arXiv:2010.04301*, 2020.

[19] M. Kim, M. Jeong, B. J. Choi, D. Lee, and N. S. Kim, "Transduce and speak: Neural transducer for text-to-speech with semantic token prediction," in *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2023, pp. 1–7.

[20] E. Gölge and The Coqui TTS Team, "Coqui TTS: A deep learning toolkit for Text-to-Speech, battle-tested in research and production," 1 2021. [Online]. Available: https://www.coqui.ai

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

[22] R. Yamamoto, E. Song, and J.-M. Kim, "Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6199–6203.

[23] M. Bain, J. Huh, T. Han, and A. Zisserman, "Whisperx: Time-accurate speech transcription of long-form audio," *INTERSPEECH 2023*, 2023.

[24] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *International Conference on Machine Learning*. PMLR, 2023, pp. 28 492–28 518.

[25] A. Hannun, J. Digani, A. Katharopoulos, and R. Collobert, "MLX: Efficient and flexible machine learning on apple silicon," 2023. [Online]. Available: https://github.com/ml-explore

[26] 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.

[27] R. Shi, A. Bär, M. Sach, W. Tirry, and T. Fingscheidt, "Non-causal to causal ssl-supported transfer learning: Towards a high-performance low-latency speech vocoder," in *2024 18th International Workshop on Acoustic Signal Enhancement (IWAENC)*. IEEE, 2024, pp. 359–363.