ZipL-Dialog: Memory-Efficient Long-Form Spoken Dialog Synthesis via Latent Flow Matching

Jihwan Kim[1,2], *Nam Soo Kim*[1]

[1] Department of Electrical and Computer Engineering and INMC, Seoul National University, Seoul, South Korea
[2] KT Corporation, Seoul, South Korea

jhkim17@hi.snu.ac.kr, nkim@snu.ac.kr

Abstract

Zero-shot dialog TTS benefits from flow-matching, but minute-scale generation on dense mel-spectrograms causes severe memory bottlenecks, often forcing unnatural chunked synthesis. We propose ZipL-Dialog, which shifts conditional flow-matching into a 4x time-compressed (25 Hz) latent space. To preserve acoustic fidelity under compression, we employ a deterministic mel autoencoder with auxiliary mel-domain supervision and optimize the ZipFormer's hierarchical downsampling schedule. Experiments show that ZipL-Dialog reduces maximum peak GPU memory by 11.22x and accelerates inference by 2.23x over the baseline, substantially lowering the memory footprint of single-pass multi-minute dialog synthesis while maintaining perceptual naturalness. Audio samples are available at `https://speechdemos.github.io/`.

Index Terms: Text-to-Speech, Flow Matching, Dialog Synthesis, Efficient TTS, Generative Audio

1. Introduction

Recent advances in neural speech synthesis have extended text-to-speech (TTS) beyond short, single-utterance synthesis toward *long-form* and *multi-turn* conversational audio, including podcasts, role-play dialogs, and interactive agents. In these settings, models must preserve naturalness, speaker and turn consistency, and contextual coherence over horizons approaching minutes, while remaining efficient enough for practical deployment.

A prominent approach to long-form conversational synthesis is *autoregressive* (AR) generation, in which acoustic tokens or frames are produced sequentially (e.g., [1, 2, 3]). AR models often provide strong quality and robust conditioning, but their inference latency typically increases linearly with output duration because each generation step depends on previously generated outputs. As a result, synthesizing multi-minute dialog can incur substantial wall-clock latency, which limits responsiveness in interactive applications and scalability in large-scale deployment.

To alleviate AR latency, *non-autoregressive* (NAR) diffusion and flow-based models generate acoustic sequences in parallel through a fixed number of refinement steps or ODE integration. Recent dialog-capable systems such as CoVoMix, CoVoMix2, and ZipVoice-Dialog (e.g., [4, 5, 6]) follow this direction and substantially improve throughput over AR generation. However, when such models operate directly on mel-spectrogram, long-form dialog introduces a different bottleneck: memory grows rapidly with sequence duration. Under conditional flow matching (CFM) [7], the model processes the full acoustic sequence at once, so longer audio increases activation storage, intermediate states, and, for transformer-based backbones, attention cost during training and inference. Consequently, minute-scale dialog often requires aggressive truncation, short-segment training, or chunked generation, which can weaken long-range conversational modeling.

A natural way to reduce this burden is to move generation into a temporally compressed latent space. Latent diffusion has proven effective in high-resolution image synthesis by learning an autoencoder that compresses the signal before diffusion is applied in the latent domain [8]. Likewise, flow matching can be performed in latent space to improve efficiency while retaining the flexibility of continuous generative modeling [9]. In speech synthesis, M3-TTS shows that latent acoustic modeling can improve the efficiency of NAR synthesis [10]. These results motivate latent-space flow/diffusion for speech; however, extending such methods to multi-minute, multi-turn dialog remains challenging because the latent representation must preserve phonetic and speaker-related detail while supporting much longer conversational context than standard single-utterance TTS.

In this work, we ask: *Can latent-space CFM substantially reduce the memory and runtime cost of long-form dialog synthesis while preserving competitive speech quality?* To answer this question, we propose **ZipL-Dialog**, a latent conditional flow-matching framework for efficient zero-shot spoken dialog synthesis. Instead of modeling frame-level acoustic features directly, ZipL-Dialog performs generation in a time-compressed continuous latent space and decodes the predicted latents back to the acoustic domain. This design shortens the sequence processed by the flow model and is particularly beneficial for minute-scale dialog, where memory becomes a dominant constraint.

A key challenge is that aggressive temporal compression can remove phonetic and speaker-relevant detail. We address this with two design choices. First, we adopt a deterministic mel autoencoder with auxiliary mel-domain supervision, which better preserves acoustic detail under strong compression in our setting than a variational latent formulation. Second, because standard hierarchical downsampling is not necessarily optimal once the input is already temporally compressed, we redesign the ZipFormer downsampling schedule to better balance contextual aggregation and local resolution.

The contributions of this work are threefold:

- We propose **ZipL-Dialog**, a latent CFM framework for long-form spoken dialog synthesis that reduces sequence-length-dependent memory and runtime by operating in a 25 Hz time-compressed latent space.
- We show that a **deterministic latent representation** combined with an **auxiliary mel-domain reconstruction loss** preserves acoustic detail and intelligibility more effectively than a variational alternative under strong temporal compression in our setting.

- We analyze hierarchical downsampling for compressed latent sequences and identify a **ZipFormer downsampling schedule** that provides a better efficiency–quality trade-off for long-form dialog synthesis.

2. Related Work

2.1. Flow Matching for Zero-Shot Dialog TTS

Recent zero-shot TTS systems have increasingly adopted diffusion and flow-matching frameworks to enable non-autoregressive speech synthesis with strong naturalness and conditioning fidelity. For conversational speech, models such as CoVoMix and CoVoMix2 [4, 5] have shown that conditional flow matching (CFM) can model multi-turn structure, speaker alternation, and diverse dialog prosody within a unified generation framework. ZipVoice-Dialog [6] further demonstrates that a ZipFormer-based CFM backbone can produce highly natural zero-shot dialog speech while maintaining robust contextual modeling across turns. Among these prior systems, ZipVoice-Dialog is most closely related to our work. It performs dialog generation directly in a dense frame-level acoustic domain using a Vocos-compatible mel-spectrogram [11]. While this design provides strong synthesis quality, operating on full-length frame sequences remains increasingly expensive as dialog duration grows. Under minute-scale generation, activation storage, intermediate states, and sequence-processing cost all increase substantially with output length, making memory and runtime practical bottlenecks even in non-autoregressive settings.

2.2. Latent Acoustic Generation and Compression–Fidelity Trade-offs

Compressing speech into a lower-rate latent representation is a promising strategy for reducing the sequence-length burden of generative modeling. In speech synthesis, M3-TTS [10] combines a mel-latent codec with a diffusion-transformer backbone and shows that latent acoustic modeling can improve the efficiency of non-autoregressive synthesis.

However, maintaining speech quality under strong temporal compression remains challenging. Many latent generative systems rely on variational autoencoders (VAEs), whose probabilistic regularization can over-smooth local spectral and phonetic detail when the temporal resolution is aggressively reduced, especially under limited model capacity. This issue becomes more critical in long-form dialog, where the representation must preserve not only intelligibility but also speaker-related cues and turn-level continuity over much longer horizons than standard single-utterance TTS.

Prior latent acoustic generation studies have largely focused on single-utterance or general speech synthesis settings, leaving the efficiency–quality trade-off for multi-minute, multi-turn dialog relatively underexplored. These observations motivate exploring latent representations that remain temporally compact while better preserving fine acoustic detail, as well as reconsidering architectural choices once the input sequence has already been compressed.

3. Method

ZipL-Dialog performs dialog generation in a temporally compressed latent acoustic space rather than directly in the frame-level mel domain. Given a frame-level mel-spectrogram, we first encode it into a lower-rate continuous latent sequence, apply masked conditional flow matching in the latent domain, and finally decode the predicted latents back to the mel domain for waveform synthesis.

3.1. Deterministic Mel Autoencoder

To reduce the sequence length handled by the flow model, we learn an acoustic autoencoder that maps a frame-level mel-spectrogram to a time-compressed latent sequence. Instead of adopting a variational latent formulation, we use a deterministic bottleneck. In our setting, strong temporal compression combined with variational regularization tended to over-smooth local phonetic detail, whereas a deterministic latent representation allowed the model capacity to focus more directly on reconstruction fidelity.

Let $\mathbf{Y} \in \mathbb{R}^{T \times F}$ denote the input mel-spectrogram, where T is the number of time frames and F is the number of mel bins. The encoder $\mathcal{E}$ first applies a 2D convolutional patch embedding with kernel and stride (F, r), spanning the full frequency axis and r consecutive frames, to produce a sequence of temporal tokens. After a depthwise 1D convolution for local positional mixing, the token sequence is processed by a stack of Transformer encoder layers and projected through a 1D convolutional bottleneck to a latent sequence $\mathbf{Z} \in \mathbb{R}^{T' \times D}$, where $T' = T/r$.

The decoder $\mathcal{D}$ maps $\mathbf{Z}$ back to the hidden dimension, refines the features using ConvNeXt-style residual blocks, and restores the original frame rate through transposed-convolution upsampling layers. In all experiments, we use a temporal compression factor of $r = 4$ and a latent dimension of $D = 100$, which yields a 25 Hz latent sequence when the frame-level acoustic representation is extracted at 100 Hz.

3.2. Masked Latent Conditional Flow Matching

Fig. 1 illustrates the training procedure of ZipL-Dialog. Given a ground-truth mel-spectrogram $\mathbf{Y}$, we first obtain its compressed latent sequence $\mathbf{Z} = \mathcal{E}(\mathbf{Y}) \in \mathbb{R}^{T' \times D}$ using the acoustic autoencoder. We then divide $\mathbf{Z}$ into an observed prefix context region and a target region to be generated. This partition is represented by a binary mask $\mathbf{m} \in \{0, 1\}^{T'}$ (broadcast over channels), where $\mathbf{m}_t = 1$ denotes the target region.

To construct the conditioning inputs, we first form a masked latent sequence

$$\mathbf{Z}^{\text{mask}} = (1 - \mathbf{m}) \odot \mathbf{Z} + \mathbf{m} \odot \mathbf{z}_{\text{mask}}, \tag{1}$$

where $\mathbf{z}_{\text{mask}}$ is the mask embedding used for the target region. Following conditional flow matching, we sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $t \sim \mathcal{U}(0, 1)$, and define the noisy latent state as

$$\mathbf{Z}_t = (1 - \mathbf{m}) \odot \mathbf{Z} + \mathbf{m} \odot ((1 - t)\boldsymbol{\epsilon} + t\mathbf{Z}). \tag{2}$$

Thus, the context region remains clean, while only the target region follows the linear interpolation path between Gaussian noise and the clean latent.

The input text, augmented with explicit speaker and turn tokens (e.g., `[S1]` and `[S2]`), is encoded by a text encoder. Its token-level representations are then average-upsampled to latent-frame-level embeddings $\mathbf{E} \in \mathbb{R}^{T' \times d}$ aligned to the latent sequence length. The ZipFormer-based flow decoder takes the concatenation of $\mathbf{Z}_t$, $\mathbf{Z}^{\text{mask}}$, and $\mathbf{E}$ as input and predicts the target velocity $\hat{\mathbf{v}}_\theta$.

We optimize the model with a masked velocity-matching objective on the target region:

$$\mathcal{L}_{\text{vel}} = \|\mathbf{m} \odot (\hat{\mathbf{v}}_\theta - (\mathbf{Z} - \boldsymbol{\epsilon}))\|_2^2. \tag{3}$$

To further preserve acoustic detail, we recover a denoised

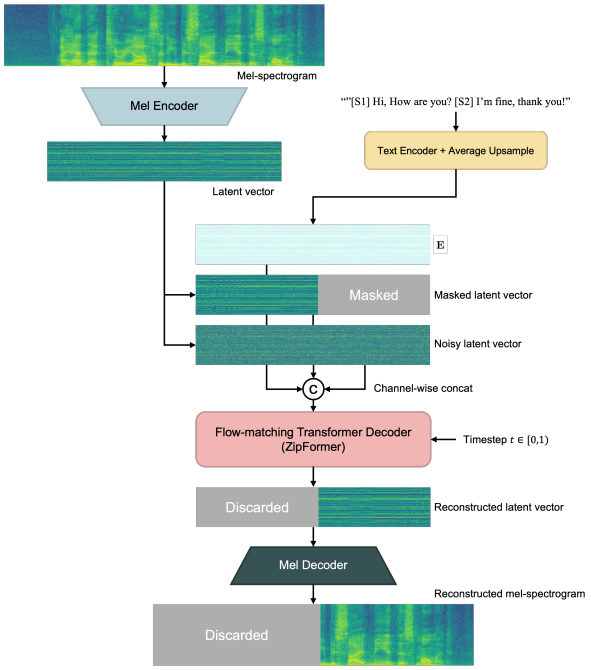

Figure 1: *Training overview of ZipL-Dialog. A ground-truth mel-spectrogram is encoded into a $4\times$ time-compressed latent sequence (25 Hz). The ZipFormer decoder predicts the CFM velocity on the masked target region, conditioned on text embeddings and noisy latents. The model is jointly optimized with a latent velocity loss and an auxiliary mel-domain reconstruction loss.*

latent estimate

$$\hat{\mathbf{Z}} = (1-\mathbf{m}) \odot \mathbf{Z} + \mathbf{m} \odot (\mathbf{Z}_t + (1-t)\hat{\mathbf{v}}_\theta), \qquad (4)$$

decode it back to the mel domain as $\hat{\mathbf{Y}} = \mathcal{D}(\hat{\mathbf{Z}})$, and apply an auxiliary masked reconstruction term

$$\mathcal{L}_{\text{mel}} = \left\| \mathbf{m}^{\uparrow} \odot \left(\hat{\mathbf{Y}} - \mathbf{Y} \right) \right\|_2^2, \qquad (5)$$

where $\mathbf{m}^{\uparrow}$ denotes the mask upsampled to the mel frame rate. The final training objective is

$$\mathcal{L} = \mathcal{L}_{\text{vel}} + \lambda \mathcal{L}_{\text{mel}}. \qquad (6)$$

The observed prefix context is used only as conditioning and is excluded from the training loss.

At inference time, the observed dialog prefix is encoded into the latent domain, the masked target region is initialized from Gaussian noise, and the learned velocity field is integrated from $t = 0$ to $t = 1$ to generate the target latent sequence. The generated latent sequence is then decoded to a mel-spectrogram and converted to waveform using the neural vocoder.

3.3. ZipFormer Downsampling Design

The default ZipFormer hierarchy used in ZipVoice-Dialog was designed for dense frame-level acoustic inputs. When the same downsampling schedule is applied after $4\times$ temporal compression, each higher-level token covers a substantially longer time span, which can degrade short-phoneme resolution and harm local acoustic fidelity. Therefore, the default hierarchy is not directly optimal in the latent setting.

Our goal is to balance three factors: (i) preserving sufficient local acoustic resolution, (ii) expanding the receptive field for stable text–speech alignment and long-range dialog modeling, and (iii) retaining the efficiency benefits of hierarchical processing. To this end, we compare representative downsampling schedules spanning no, moderate, and aggressive temporal reduction, and select the one that provides the best efficiency–quality trade-off. As shown in Section 4.7, the default schedule is suboptimal for compressed latent sequences, and a milder hierarchy yields better overall behavior in our setting.

4. Experiments

We conduct all experiments on English data, matching the language coverage of the large-scale training mixture and the evaluation protocols of prior dialog TTS benchmarks.

4.1. Training Datasets

Backbone pretraining. Following the ZipVoice training recipe, we pretrain the backbone on a mixture of large-scale English speech corpora: HiFiTTS2 (~31.7k hours) [12], Emilia-English (~20k hours) [13], and LibriTTS (~585 hours) [14].
Dialog fine-tuning. Following ZipVoice-Dialog, we fine-tune on the English subset (~5k hours) of the 6.8k-hour OpenDialog training corpus, which is constructed from in-the-wild spoken-dialog data.

4.2. Implementation Details

Shared configuration. All models are trained on four NVIDIA A100 (80GB) GPUs. For fair comparison, ZipL-Dialog and the ZipVoice-Dialog baseline use identical Vocos-compatible 100-bin mel-spectrograms (24 kHz, 100 Hz frame rate). Unless otherwise noted, the optimizer and learning-rate schedules follow the official ZipVoice-Dialog implementation.
Mel autoencoder. The deterministic autoencoder (∼34 M parameters) is trained on the shared English speech mixture using 3 s random audio crops. It is optimized via AdamW (learning rate 10^{-4} with warmup) and a batch size of 200 for 200k steps. Its weights are then frozen during ZipL-Dialog training.
ZipL-Dialog training. The backbone (∼123 M parameters) incorporates our optimized `[1,1,2,1,1]` downsampling schedule. Training utilizes dynamic batching (up to 1,200 s per batch) in two stages. First, the backbone is pretrained for 200k steps, filtering data to utterances under 30 s for memory control and stability. Subsequently, it is fine-tuned on the dialog data for 100k steps. The auxiliary mel-domain reconstruction loss weight is set to $\lambda = 0.5$.

4.3. Evaluation Datasets

We evaluate ZipL-Dialog on two dialog-oriented benchmarks: (i) the OpenDialog test set used in ZipVoice-Dialog, and (ii) the CoVoMix2 dialog test set proposed in [5]. The CoVoMix2 dialog test set contains 1k dialog transcripts sampled from DailyDialog [15] and uses acoustic prompts from LibriSpeech test-clean [16].

4.4. Baseline Models

We compare ZipL-Dialog against two strong baselines from complementary generative paradigms:

- **ZipVoice-Dialog (NAR baseline) [6]:** the most closely related frame-level non-autoregressive dialog TTS baseline built on the same architectural family.
- **VibeVoice 1.5B (AR baseline) [1]:** a large-scale autoregressive speech synthesis model included to contrast the efficiency–quality trade-off at multi-minute dialog duration.

Table 1: *Efficiency results measured on a single NVIDIA A100 40GB GPU (lower is better).*

Metric	Ours	ZipVoice-Dialog	VibeVoice 1.5B
CoVoMix2 test set (avg=32.878 s, max=178.891 s)			
Inference time (s)	**1.075**	2.396	50.160
RTF	**0.056**	0.089	1.455
Peak memory (GB)	**1.10**	4.01	5.49
Max peak memory (GB)	**3.23**	36.21	6.20
OpenDialog test set (avg=26.029 s, max=65.141 s)			
Inference time (s)	**1.045**	1.477	45.069
RTF	**0.052**	0.069	1.558
Peak memory (GB)	**0.97**	2.03	5.34
Max peak memory (GB)	**1.30**	5.94	5.41

Table 2: *Quality results (WER lower is better; cpSIM/UTMOS higher is better).*

Metric	Ours	ZipVoice-Dialog	VibeVoice 1.5B
CoVoMix2 test set			
WER (%)	5.203	**4.229**	4.959
cpSIM	0.444	**0.493**	0.485
UTMOS	**3.523**	3.477	**3.523**
OpenDialog test set			
WER (%)	5.362	**3.550**	12.979
cpSIM	0.304	0.346	**0.350**
UTMOS	**3.198**	3.089	2.312

4.5. Evaluation Metrics

Efficiency metrics. All runtime and memory measurements are obtained end-to-end with a batch size of 1 on a single NVIDIA A100 40GB GPU, encompassing prompt processing, latent generation, mel decoding, and waveform synthesis. We report inference time, average RTF, average peak GPU memory, and the maximum peak memory observed across the test set.

Quality metrics. Following ZipVoice-Dialog [6], WER is computed using WhisperD [17] after standard text normalization and speaker-tag removal. For cpSIM, we utilize Pyannote diarization [18] and WavLM-ECAPA embeddings [19, 20] to find the maximum average cosine similarity under possible speaker permutations. We also report perceptual quality via UTMOS [21].

Inference settings. Both ZipL-Dialog and the ZipVoice-Dialog baseline employ a 16-step Euler ODE solver with classifier-free guidance. For VibeVoice, we evaluate using its official open-source inference implementation.

4.6. Main Results

Efficiency and Scalability. Table 1 shows that ZipL-Dialog is substantially more efficient than both baselines. It is 43–47$\times$ faster than VibeVoice and reduces maximum peak GPU memory by up to 11.22$\times$ relative to ZipVoice-Dialog, indicating that 25 Hz latent-space flow matching markedly lowers the runtime and memory cost of long-form dialog synthesis.

Quality and Efficiency–Quality Trade-off. Table 2 shows that ZipL-Dialog does not match the uncompressed baseline on all objective metrics: ZipVoice-Dialog yields lower WER on both sets, and ZipVoice-Dialog/VibeVoice obtain higher cpSIM on CoVoMix2/OpenDialog. Still, ZipL-Dialog achieves the best or tied-best UTMOS on both benchmarks, suggesting that overall naturalness is preserved despite modest losses in local phonetic and speaker-similarity detail.

Table 3: *Controlled ablation in a single-speaker zero-shot TTS setting trained on LibriTTS and evaluated on LibriSpeech-PC. SIM-o denotes speaker similarity in the single-speaker setting; its absolute values are not directly comparable to the multi-speaker cpSIM scores in Tables 1 and 2.*

Setting	WER↓	SIM-o↑	UTMOS↑
(1) Latent Representation			
VAE	6.535	**0.518**	3.877
AE (Ours)	**3.634**	0.489	**3.982**
(2) Auxiliary Loss (with AE)			
w/o $\mathcal{L}_{mel}$	4.024	0.485	3.896
w/ $\mathcal{L}_{mel}$ (Ours)	**3.634**	**0.489**	**3.982**
(3) Downsampling (with AE, w/ $\mathcal{L}_{mel}$)			
A (`[1,1,1,1,1]`)	51.964	0.418	3.676
B (`[1,1,2,1,1]`, Ours)	**3.634**	**0.489**	**3.982**
C (`[1,2,4,2,1]`, Def.)	27.432	0.357	3.671

4.7. Ablation Study: Latent Types, Auxiliary Loss, and Downsampling

For a single-speaker zero-shot TTS evaluation, we follow the F5-TTS protocol [22] and report WER, SIM-o, and UTMOS on the LibriSpeech-PC test-clean set [23]. WER is computed using a HuBERT-large ASR model [24]. Note that these absolute scores are not directly comparable to our dialog benchmarks due to differences in the evaluation setup.

Results and Analysis. Table 3 details our ablation results. Although the VAE latent slightly favors speaker similarity (SIM-o 0.518 vs. 0.489), our deterministic AE substantially improves intelligibility (WER 3.634%) and naturalness (UTMOS 3.982), making it the optimal choice to prevent phonetic blurring under 4$\times$ compression. Furthermore, incorporating the auxiliary loss ($\mathcal{L}_{\text{mel}}$) consistently improves all metrics over the baseline AE. Finally, our moderate downsampling schedule (`[1,1,2,1,1]`) drastically outperforms both the no-downsampling and the overly aggressive default schedules, proving that compressed latents still require carefully balanced hierarchical processing.

5. Conclusion

We proposed ZipL-Dialog, an efficient latent conditional flow-matching framework for long-form spoken dialog synthesis. By moving generation to a 25 Hz deterministic latent space and redesigning the ZipFormer downsampling schedule, ZipL-Dialog substantially reduces the memory and runtime cost of minute-scale synthesis. On the CoVoMix2 and OpenDialog test sets, it achieves up to 11.22$\times$ lower maximum peak GPU memory and up to 2.23$\times$ faster inference than ZipVoice-Dialog, while maintaining competitive perceptual quality. Although temporal compression introduces modest trade-offs in objective intelligibility and speaker-similarity metrics, ZipL-Dialog attains the best or tied-best UTMOS on both benchmarks, demonstrating a strong efficiency–quality trade-off for long-form dialog generation. Future work will investigate improved latent modeling and reconstruction objectives to better recover local phonetic detail without sacrificing efficiency.

6. Generative AI Use Disclosure

The authors utilized OpenAI ChatGPT and Google Gemini to improve the English phrasing, readability, and grammar of this manuscript. All content and ideas remain the sole original work of the authors.

7. References

[1] Z. Peng, J. Yu, W. Wang, Y. Chang, Y. Sun, L. Dong, Y. Zhu, W. Xu, H. Bao, Z. Wang *et al.*, "Vibevoice technical report," *arXiv preprint arXiv:2508.19205*, 2025.

[2] K. Xie, F. Shen, J. Li, F. Xie, X. Tang, and Y. Hu, "Fireredtts-2: Towards long conversational speech generation for podcast and chatbot," *arXiv preprint arXiv:2509.02020*, 2025.

[3] H. Xie, H. Lin, W. Cao, D. Guo, W. Tian, J. Wu, H. Wen, R. Shang, H. Liu, Z. Jiang *et al.*, "Soulx-podcast: Towards realistic long-form podcasts with dialectal and paralinguistic diversity," *arXiv preprint arXiv:2510.23541*, 2025.

[4] L. Zhang, Y. Qian, L. Zhou, S. Liu, D. Wang, X. Wang, M. Yousefi, Y. Qian, J. Li, L. He *et al.*, "Covomix: Advancing zero-shot speech generation for human-like multi-talker conversations," *Advances in Neural Information Processing Systems*, vol. 37, pp. 100 291–100 317, 2024.

[5] L. Zhang, Y. Qian, X. Wang, M. Thakker, D. Wang, J. Yu, H. Wu, Y. Hu, J. Li, Y. Qian *et al.*, "Covomix2: Advancing zero-shot dialogue generation with fully non-autoregressive flow matching," *arXiv preprint arXiv:2506.00885*, 2025.

[6] H. Zhu, W. Kang, L. Guo, Z. Yao, F. Kuang, W. Zhuang, Z. Li, Z. Han, D. Zhang, X. Zhang *et al.*, "Zipvoice-dialog: Non-autoregressive spoken dialogue generation with flow matching," *arXiv preprint arXiv:2507.09318*, 2025.

[7] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," *arXiv preprint arXiv:2210.02747*, 2022.

[8] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[9] Q. Dao, H. Phung, B. Nguyen, and A. Tran, "Flow matching in latent space," *arXiv preprint arXiv:2307.08698*, 2023.

[10] X. Wang, C. Qiang, R. Fu, Z. Wen, X. Liu, Y. Liu, Y. Liang, K. Yin, Y. Xie, H. Xie *et al.*, "M3-tts: Multi-modal dit alignment & mel-latent for zero-shot high-fidelity speech synthesis," *arXiv preprint arXiv:2512.04720*, 2025.

[11] H. Siuzdak, "Vocos: Closing the gap between time-domain and fourier-based neural vocoders for high-quality audio synthesis," *arXiv preprint arXiv:2306.00814*, 2023.

[12] R. Langman, X. Yang, P. Neekhara, S. Hussain, E. Casanova, E. Bakhturina, and J. Li, "Hifitts-2: A large-scale high bandwidth speech dataset," *arXiv preprint arXiv:2506.04152*, 2025.

[13] H. He, Z. Shang, C. Wang, X. Li, Y. Gu, H. Hua, L. Liu, C. Yang, J. Li, P. Shi *et al.*, "Emilia: An extensive, multilingual, and diverse speech dataset for large-scale speech generation," in *2024 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2024, pp. 885–890.

[14] 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.

[15] Y. Li, H. Su, X. Shen, W. Li, Z. Cao, and S. Niu, "Dailydialog: A manually labelled multi-turn dialogue dataset," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2017, pp. 986–995.

[16] 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.

[17] J. Darefsky, G. Zhu, and Z. Duan, "Parakeet: A natural sounding, conversational text-to-speech model," 2024, blog post. [Online]. Available: https://jordandarefsky.com/blog/2024/parakeet/

[18] H. Bredin, R. Yin, J. M. Coria, G. Gelly, P. Korshunov, M. Lavechin, D. Fustes, H. Titeux, W. Bouaziz, and M.-P. Gill, "Pyannote. audio: neural building blocks for speaker diarization," in *ICASSP 2020-2020 IEEE International conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2020, pp. 7124–7128.

[19] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao *et al.*, "Wavlm: Large-scale self-supervised pre-training for full stack speech processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 6, pp. 1505–1518, 2022.

[20] B. Desplanques, J. Thienpondt, and K. Demuynck, "Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification," *arXiv preprint arXiv:2005.07143*, 2020.

[21] T. Saeki, D. Xin, W. Nakata, T. Koriyama, S. Takamichi, and H. Saruwatari, "Utmos: Utokyo-sarulab system for voicemos challenge 2022," *arXiv preprint arXiv:2204.02152*, 2022.

[22] Y. Chen, Z. Niu, Z. Ma, K. Deng, C. Wang, J. JianZhao, K. Yu, and X. Chen, "F5-tts: A fairytaler that fakes fluent and faithful speech with flow matching," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025, pp. 6255–6271.

[23] A. Meister, M. Novikov, N. Karpov, E. Bakhturina, V. Lavrukhin, and B. Ginsburg, "Librispeech-pc: Benchmark for evaluation of punctuation and capitalization capabilities of end-to-end asr models," in *2023 IEEE automatic speech recognition and understanding workshop (ASRU)*. IEEE, 2023, pp. 1–7.

[24] 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.