Interspeech 2025
17-21 August 2025, Rotterdam, The Netherlands

APTTS: Adversarial Post-training in Latent Flow Matching for Fast and High-fidelity Text-to-Speech

Hyungchan Yoon[1], *Chanwoo Lee*[2], *Hoodong Lee*[1], *Stanley Jungkyu Choi*[1]

[1]LG AI Research, Seoul, Republic of Korea
[2]Yonsei University, Seoul, Republic of Korea
hc71.yoon@lgresearch.ai

Abstract

Although recent text-to-speech (TTS) models based on flow matching have achieved remarkable generation quality, their reliance on numerous sampling steps hinders practical deployment. In this work, we introduce an adversarial post-training strategy for flow matching TTS that significantly reduces the required sampling steps. Our approach treats a pre-trained flow matching model as a few-step generator, optimizing it with reconstruction and adversarial objectives. We integrate this technique into APTTS, our novel latent flow matching framework for zero-shot TTS, and demonstrate its superiority over state-of-the-art baselines with real-time applicability. Furthermore, we validate the scalability of our adversarial post-training approach by applying it to Matcha-TTS, a publicly available flow matching model. Evaluations on a multi-speaker dataset show that our method enhances audio quality while reducing inference time, underscoring its potential as a scalable solution for real-time TTS.

Index Terms: speech synthesis, flow matching, adversarial learning, zero-shot

1. Introduction

Recent advances in deep generative models have significantly improved text-to-speech (TTS) synthesis quality. In particular, zero-shot TTS has emerged as a promising approach, aiming to replicate distinctive vocal characteristics and prosodic nuances using only a single speech prompt. Current state-of-the-art zero-shot TTS systems often leverage latent diffusion models [1–3], applying a diffusion process to well-regularized latent speech representations—encoded as discrete tokens or continuous vectors. In contrast, flow matching [4] has also been applied to zero-shot TTS, achieving performance on par with latent diffusion models [5–7]. However, unlike latent diffusion approaches, flow matching models typically rely on fixed intermediate representations (e.g., mel-spectrograms), which limits the potential for learning more optimal, task-specific representations. Although latent flow matching has been successfully adopted in other domains [8, 9], its use in TTS remains relatively underexplored. One of the key objectives of our work is to bridge this gap by incorporating latent flow matching into zero-shot TTS.

A common limitation of both diffusion and flow matching models is the need for numerous iterative sampling steps to achieve high-quality generation, which increases inference latency. To address this issue, several acceleration techniques have been proposed in recent TTS studies [10–13]. For example, FlashSpeech [10] introduces adversarial consistency learning, ReFlow-TTS [11] employs a Reflow [14] process, E1 TTS [12] incorporates distribution matching distillation [15], and PeriodWave-Turbo [13] uses adversarial flow matching optimization. Overall, one key emerging trend is the use of adversarial learning to construct efficient few-step generators.

In this paper, we introduce APTTS, a fast latent flow matching framework for zero-shot TTS that leverages an adversarial post-training (AP) strategy to accelerate inference. Our method first learns continuous latent speech representations via a variational autoencoder (VAE) [16] bottleneck, then trains a flow matching model to approximate these normalized latents from both text and speech prompts. We subsequently refine the trained flow matching model by treating it as a few-step generator and optimizing its output latents with a combination of reconstruction and adversarial objectives. This AP procedure reduces the training-inference mismatch and achieves a superior trade-off between synthesis speed and audio quality. Although similar acceleration techniques have been explored in the waveform domain [13], our work is the first to integrate the AP strategy within the intermediate representation domain of TTS through a purpose-designed discriminator, effectively mitigating the primary computational bottleneck of current zero-shot systems. Moreover, we introduce a hybrid classifier-free guidance (CFG) technique to naturally incorporate CFG [17] into our few-step generator, further enhancing context and prompt fidelity.

Experimental evaluations on zero-shot TTS demonstrate that APTTS achieves state-of-the-art performance with real-time applicability. In addition, we validate the scalability of our AP approach by integrating it into Matcha-TTS [18], which not only accelerates inference but also improves generation quality in a multi-speaker TTS task.[1]

2. Preliminary: Flow Matching

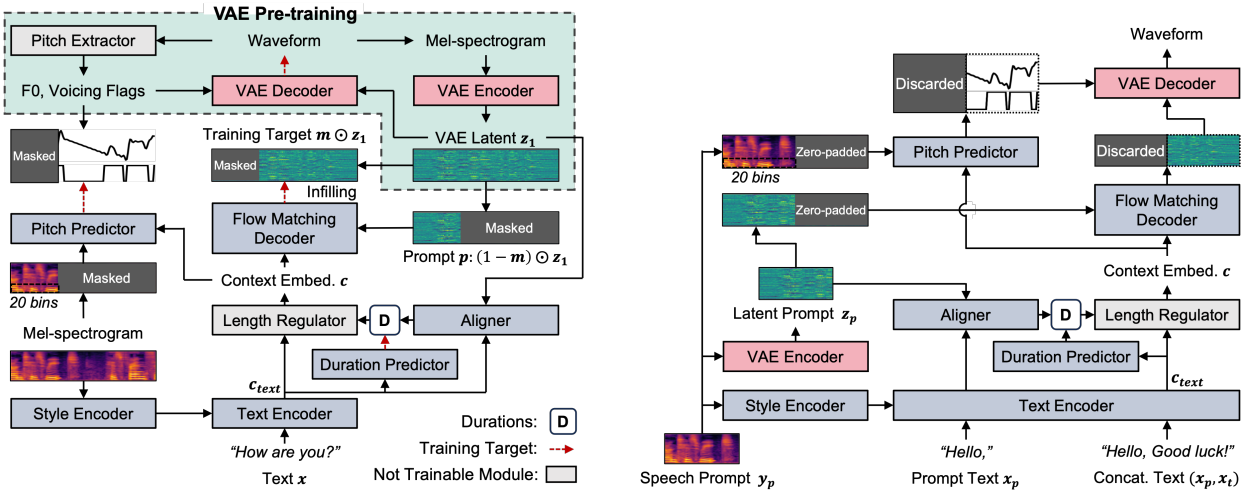
Flow matching (FM) [4] is a generative modeling framework that defines a probability path $p_t(x)$ transforming a known initial distribution $p_0(x)$ into a target distribution $p_1(x)$, which approximates a true data distribution $q(x)$. The FM model, parameterized by a neural network θ, learns a time-dependent vector field $u_t(x) : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ that constructs this path. Specifically, conditional flow matching (CFM) trains the FM model to estimate a conditional vector field $u_t(x|x_1)$ derived from the conditional probability path $p_t(x|x_1)$:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,q(x_1),p_t(x|x_1)}\left[||u_t(x|x_1) - v_t(x;\theta)||^2\right], \quad (1)$$

where $t \sim \mathcal{U}[0, 1]$, x_1 is drawn from $q(x)$, and $v_t(x;\theta)$ is the predicted vector field. During inference, an ordinary differential equation (ODE) solver is used to generate x_1 from $x_0 \sim p_0(x)$ by querying the learned FM model for the current vector field at each time step.

[1]Demo samples are available at https://lfmaptts.github.io/aptts

10.21437/Interspeech.2025-455



(a) *Training* (b) *Inference*

Figure 1: *Overall framework of the (a) training and (b) inference stages of the proposed model.*

A particularly convenient choice of the conditional path is an optimal transport path [4] that transfers each x_0, drawn from standard Gaussian distribution $p_0(x)$, toward the target x_1 at a constant velocity. In this setting, the vector field is time-invariant with constant direction and speed, defined as $u_t(x|x_1) = x_1 - (1 - \sigma_{\min})x_0$ and its corresponding conditional probability path is $p_t(x|x_1) = \mathcal{N}(x|tx_1, (1-(1-\sigma_{\min})t)^2 I)$, where $\sigma_{\min}$ is typically a small constant. This simple formulation yields practical benefits, including faster training convergence and reduced error accumulation during inference.

3. Method

Fig. 1a illustrates the overall framework of APTTS. Our training procedure is divided into three stages. First, we pre-train a VAE to learn a continuous latent space from speech, with pitch information extracted separately and fed into the VAE decoder. Second, we train the remaining modules: a FM decoder approximates the learned latent and a pitch predictor estimates pitch information, conditioned on text and speech prompt. In our framework, the mel-spectrogram serves as the speech prompt. Finally, we apply the AP strategy exclusively to the FM decoder.

3.1. Pre-training VAE with Explicit Pitch Modeling

We adopt a VAE architecture similar to PeriodVITS [19], modifying the encoder to receive a mel-spectrogram instead of a linear spectrogram. Explicit fundamental frequencies (F0) and binary voicing flags are separately extracted to generate a sinusoidal source signal, which is fed to the VAE decoder. This separated design improves pitch reconstruction, thereby enhancing the naturalness of synthesized speech. Unlike PeriodVITS, which performs TTS directly with a conditional VAE, our approach focuses on learning a well-regularized latent space. To this end, we use a latent dimension of 64 and down-weight the regularization loss (defined as the KL divergence between the latent distribution and a standard Gaussian) to 1×10^{-4}. To further boost audio fidelity, we employ a multi-period discriminator (MPD) [20] and a multi-resolution discriminator (MRD) [21] with a truncated pointwise relativistic loss [22]. Other loss terms, including a reconstruction loss, adversarial loss, and feature matching loss, follow [19].

3.2. FM Decoder and Pitch Predictor

After pre-training the VAE, we train the remaining modules to estimate the learned latent and pitch information from text and speech prompt. First, we encode a text sequence using a transformer encoder [23] that is conditioned on a global style embedding extracted from the speech prompt. Specifically, an SC-CNN [24] mechanism injects the style embedding into the text encoder, while a NeXt-TDNN [25]-based style encoder produces the embedding from a mel-spectrogram. A self-supervised aligner [26] is jointly trained to get the alignment between the text encoder output c_{text} and the frame-level latent z_1, and a duration predictor [23] estimates the text durations from c_{text}. A length regulator then upsamples c_{text} into frame-level c, a style-aware context embedding that is used for both the FM decoder and the pitch predictor.

For the FM decoder, we adopt a Diffusion Transformer (DiT) [6, 27] as the backbone. RoPE [28] is integrated into the self-attention layers, and adaLN-single [29] is employed to reduce model parameters. Following prior work [3, 5–7], we frame zero-shot TTS as a speech infilling task, where the model learns to reconstruct masked segments using both text and surrounding speech segments. Let z_1 denote the learned latent, z_0 a sample from a standard Gaussian, and m a binary temporal mask. Under an optimal transport path setting, the decoder receives: a flow step $t \in [0, 1]$, a noisy latent $z_t = (1-(1-\sigma_{\min})t)z_0 + tz_1$, a latent prompt $p = (1-m) \odot z_1$, and the context embedding c. The flow step t is embedded via sinusoidal positional encoding, and the sequences (z_t, p, c) are concatenated and projected into the hidden dimensions of DiT. The decoder is trained with the $\mathcal{L}_{\text{CFM}}$ objective to estimate the target vector field $v_t = z_1 - (1-\sigma_{\min})z_0$, with loss computed only on the masked frames. Here, the mask m covers a random contiguous region whose length is uniformly sampled between 70% and 100% of the full frame length.

To enhance context and prompt fidelity, we adopt a dual CFG strategy [30, 31] to the FM decoder. During training, the context embedding c is dropped with a 5% probability, the prompt p with 10%, and both together (c, p) with 10%. These dropout schemes enable the model to learn both conditional and unconditional outputs. At inference, the latent is sampled via ODE solver by guiding the vector field at each time step t as:

$$\begin{aligned}\tilde{v}_t\left(z_t, p, c\right) &= v_t\left(z_t, p, c\right) \\ &+ \alpha_p\left(v_t\left(z_t, p, \varnothing\right) - v_t\left(z_t, \varnothing, \varnothing\right)\right) \\ &+ \alpha_c\left(v_t\left(z_t, \varnothing, c\right) - v_t\left(z_t, \varnothing, \varnothing\right)\right), \end{aligned} \tag{2}$$

where $\varnothing$ denotes a null condition (i.e., zero tensor), and α_p and α_c denote the guidance scales for each fidelity.

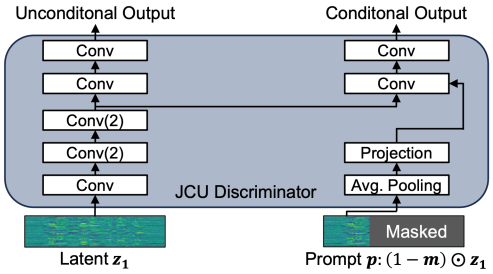


Figure 2: *Overall architecture of our discriminator. Conv(d) denotes a 1D convolutional layer with a dilation factor of d.*

The pitch predictor estimates pitch information from c and is also conditioned on the speech prompt similarly to the FM decoder. Specifically, the same mask m is applied to the pitch prediction loss [19] and surrounding segments $(1-m)$ of first 20 mel-spectrogram bins are used as the prompt, which is concatenated with c at the input of the pitch predictor.

3.3. Adversarial Post-training

To accelerate inference, we apply the AP strategy to the trained FM decoder. In this stage, the FM decoder is repurposed as a few-step generator that employs an ODE solver with fixed time steps, inherently reducing the training-inference mismatch. We directly align its output latents with the target using a reconstruction loss, and further incorporate an adversarial loss via a discriminator to better align the generated latent distribution with the target under limited sampling steps. As shown in Fig. 2, we adopt a joint conditional and unconditional (JCU) discriminator based on the architecture proposed in [32], with modifications tailored to our framework. Specifically, we condition the discriminator on a time-averaged embedding of the prompt p to enhance the similarity between the generated speech and the prompt, and replace strided convolutions with dilated convolutions to facilitate masked loss computation. The overall AP loss is defined as:

$$L_{AP} = m \odot (L_{recon} + L_{JCU} + \lambda_{fm} L_{fm}), \tag{3}$$

where L_{recon} is the L1 reconstruction loss, L_{JCU} and L_{fm} the least-squares adversarial loss and the feature matching loss (dynamically scaled by λ_{fm}) from [32]. The mask m is applied to all loss terms.

Hybrid CFG. Recent few-step acceleration methods [14, 15] often adopt a fixed CFG scale during training; however, this approach presents several issues. In particular, it increases training time because both conditional and unconditional outputs must be computed in a single forward process, and limits user flexibility in controlling fidelity at inference. Moreover, using a large guidance scale leads to a substantial discrepancy between the generated and target latent distributions at the initial stage of AP, due to the error accumulation inherent in CFG. To overcome these drawbacks while retaining the benefits of CFG, we introduce a hybrid CFG technique. During AP, the decoder is always conditioned on both the prompt p and the context embedding c, in contrast to the original $\mathcal{L}_{\text{CFM}}$ training phase where these conditions are occasionally dropped. At inference, we refine the vector field of the few-step generator by leveraging the outputs of the *base* FM decoder, which uses the checkpoint prior to AP. Specifically, all terms in Eq. 2 except $v_t\,(z_t, p, c)$ are derived from the base model at each time step. This hybrid CFG strategy offers better prompt and context fidelity than the few-step generator alone, in a controllable manner.

3.4. Inference

Fig. 1b outlines the inference pipeline. Given a speech prompt y_p and its corresponding text x_p, we aim to synthesize speech for a target text x_t that emulates the characteristics of y_p. First, we encode x_p and obtain its alignment with z_p using the aligner, where z_p is extracted from y_p through the VAE encoder. Next, we process the concatenated sequence (x_p, x_t) through the text encoder, and upsample the resulting output to obtain the context embedding c: the initial segment for x_p follows the pre-obtained alignment, while the subsequent segment for x_t is upsampled according to durations predicted by the duration predictor. Conditioned on c and the prompt, the FM decoder and the pitch predictor generate the latent and pitch information, respectively. Here, the prompt is defined as z_p for the FM decoder and as the first 20 bins of y_p for the pitch predictor. The initial segments of these outputs—corresponding to the length of y_p—are discarded, and the remaining outputs are converted into waveform via the VAE decoder.

4. Experiment

4.1. Settings

4.1.1. Datasets and feature preprocessing

For training APTTS, we utilized four datasets—LibriTTS [33] (*train* and *dev* subsets), VCTK [34], Hi-Fi TTS [35], and Expresso [36] (*read* subset)—comprising approximately 918 hours of speech from 2507 speakers. For evaluation, we used the test-clean subset of LibriSpeech [37], selecting samples with durations between 4 and 10 seconds. All audio samples were downsampled to 22050 Hz, and transcriptions were converted to IPA phoneme sequences using open-source software[2]. We extracted 80-band mel-spectrograms with a 1024-point FFT, a hop size of 256, and a window size of 1024. Continuous F0 [38] and voicing flags were extracted using WORLD [39].

4.1.2. Model details

The text encoder comprised 9 transformer layers [23], with SC-CNN [24] applied to the last three layers. The FM decoder consisted of 24 layers with 16 attention heads, and its embedding/feed-forward network dimensions were set to 1024/2048. For the VAE decoder, upsample rates of [8, 8, 2, 2] were used. The pitch predictor comprised three stacked layers, each containing a transformer layer and a residual 1D convolutional layer with a kernel size of 3. During the AP phase, a 4-step Euler method with fixed time steps of [0, 0.08, 0.29, 0.62] was employed, emphasizing earlier time steps as inspired by [6]. We set α_p to 3 and α_c to 2.

The VAE was pre-trained for 1.4M steps on 4 A100 GPUs with a batch size of 16 and an initial learning rate of 2×10^{-4}, with other configurations following [20]. Subsequently, all remaining modules were trained for 300K steps on 8 A100 GPUs using a dynamic batch size averaging 17. We used the AdamW optimizer with $\beta_1 = 0.9, \beta_2 = 0.999$, and zero weight decay, along with a linear learning rate scheduler with a peak learning rate of 1×10^{-4} warmed up over 10K steps. For the AP phase, training was conducted for 200K steps with the average batch size reduced to 5 and the peak learning rate to 1×10^{-5}.

4.1.3. Evaluation metrics

Following previous works [3, 5, 6], we evaluated zero-shot TTS performance using a *cross-sentence* task in which the model received a 3-second speech prompt along with its transcription

[2] https://github.com/bootphon/phonemizer

Table 1: *Zero-shot TTS results comparison with public models.*

Model	SMOS ↑	MOS ↑	WER ↓
Ground Truth	-	3.70±0.06	2.66
YourTTS	2.70±0.08	2.41±0.08	7.23
HierSpeech++	3.43±0.07	3.43±0.08	7.07
F5-TTS (32 steps)	3.60±0.06	3.66±0.08	2.52
APTTS (4 steps)	**3.65±0.05**	3.58±0.07	**2.08**
w/o AP (32 steps)	3.56±0.06	**3.76±0.08**	2.42

Table 2: *Objective results for zero-shot TTS. * denotes the score reported in baseline papers.*

Model	Data (hours)	WER ↓	CER ↓	SIM-o ↑	RTF ↓
YourTTS	0.5K+	7.55	2.83	0.45	0.054
HierSpeech++	2.8K Multi.	4.19	1.33	0.50	0.061
VALL-E	60K	5.9*	-	-	0.62*
Voicebox (64 steps)	60K	1.9*	-	**0.66***	0.64*
CLaM-TTS	55K	5.11*	2.87*	0.50*	0.415*
DiTTo-TTS	55K	2.56*	0.89*	0.63*	0.162*
FlashSpeech	45K	2.7*	-	0.52*	**0.02***
F5-TTS (32 steps)	100K Multi.	2.61	0.79	<u>0.64</u>	0.217
APTTS (4 steps)	0.9K	**1.73**	**0.40**	0.55	<u>0.052</u>
w/o AP (32 steps)	0.9K	<u>1.82</u>	<u>0.44</u>	0.57	0.274

and a target text to generate. For subjective evaluation, mean opinion score (MOS) assessed the naturalness of the generated speech on a 1-to-5 scale, while similarity MOS (SMOS) evaluated speaker similarity between the generated speech and the prompt on a 1-to-4 scale. Both metrics were reported with 95% confidence intervals and evaluated by 20 raters recruited via Amazon MTurk. For objective evaluation, word and character error rates (WER/CER) were measured using a CTC-based HuBERT-Large[3] to assess intelligibility, and speaker similarity (SIM-o) was computed following the procedure described in [5]. Additionally, the real-time factor (RTF) was used to assess inference speed, defined as the average generation time for 1 second of speech on a single A100 GPU.

4.2. Evaluation of Zero-shot TTS

We first compared the performance of APTTS against several high-performing public models [6, 40, 41] with randomly selected 39 samples from the LibriSpeech test-clean subset, ensuring no speaker overlap. As shown in Table 1, our 32-step model without AP achieves superior naturalness and intelligibility—albeit with slightly lower speaker similarity—compared to F5-TTS, a state-of-the-art FM model with a mel-spectrogram bottleneck, and outperforms other baselines across all metrics. These results underscore the advantages of our latent flow matching framework over mel-spectrogram-based approaches. Notably, applying AP reduces the number of inference steps by $8\times$; although this reduction incurs a degradation in naturalness, it improves both speaker similarity and intelligibility, ultimately surpassing F5-TTS with far fewer sampling steps. We attribute this gain in SMOS to our discriminator, which effectively incorporates the prompt as a condition.

Next, we compared APTTS with state-of-the-art models trained on large-scale datasets [3, 5, 6, 10, 42, 43] by evaluating objective metrics over the entire LibriSpeech test-clean subset (samples between 4 and 10 seconds). As shown in Table 2, our model achieves the lowest WER and CER, significantly outperforming all baselines in intelligibility, which is a key indicator of speech robustness. Although the SIM-o score of APTTS is lower than that of some models, we attribute this discrepancy to significant differences in dataset scale; indeed, APTTS attains better subjective similarity than F5-TTS despite its lower SIM-o, highlighting its stregnth in practical listening conditions. Furthermore, APTTS achieves a synthesis speed of 0.052 RTF, making it suitable for real-time applications. While only FlashSpeech exhibits lower latency, APTTS outperforms it in both intelligibility and speaker similarity. Overall, APTTS offers an improved trade-off between audio quality and synthesis speed, with AP reducing inference time by more than $5\times$ relative to the 32-step model while incurring only minimal performance differences.

[3] https://huggingface.co/facebook/hubert-large-ls960-ft

Table 3: *Experimental results for VCTK multi-speaker TTS. MAT-(n) denotes the Matcha-TTS with n sampling steps.*

Model	WER ↓	MOS ↑	RTF ↓
Ground Truth	4.93	4.17±0.10	-
VITS	13.63	3.90±0.11	0.061
MAT-2	5.38	3.53±0.12	**0.027**
MAT-10	6.81	3.71±0.12	0.048
MAT-2 + AP	**4.80**	**3.96±0.10**	**0.027**

4.3. Evaluation of AP Scalability

To evaluate the scalability of our AP strategy, we applied it to the multi-speaker variant of Matcha-TTS, a publicly available FM model that employs a mel-spectrogram bottleneck. Unlike zero-shot TTS models that rely on speech prompts, Matcha-TTS conditions on speaker embeddings derived from speaker IDs. Accordingly, during AP, we conditioned the discriminator on speaker embeddings rather than time-averaged prompts and omitted masked loss computation. We employed a 2-step Euler method with fixed time steps of [0, 0.5]. Specifically, we started from the official checkpoint[4] of the VCTK model and further trained it for 250K steps using the same configurations.

For evaluation, we randomly selected 20 samples from distinct speakers. As shown in Table 3, VITS—a prominent TTS model based on normalizing flows—exhibits better naturalness than the original Mathca-TTS; however, it significantly underperforms in intelligibility. When comparing MAT-10 and MAT-2, reducing the number of sampling steps from 10 to 2 nearly doubles inference speed, albeit with some loss in naturalness. Notably, after applying AP to MAT-2, the model achieves the best performance across all metrics, even surpassing MAT-10. We found that the original Matcha-TTS produces relatively oversmoothed mel-spectrograms, whereas the AP-applied model preserves finer acoustic details. These results imply that our AP strategy can be effectively applied to other FM TTS models—not only for zero-shot TTS, but also across different representation domains.

5. Conclusion

We proposed APTTS, a latent flow matching TTS system designed for real-time applications. To achieve this, we introduced an adversarial post-training strategy that accelerates inference in flow matching models while preserving performance via a prompt-conditioned discriminator and hybrid classifier-free guidance. Experimental results on zero-shot TTS demonstrated that APTTS attains a state-of-the-art trade-off between audio quality and synthesis speed, showing remarkable strength in intelligibility. Furthermore, applying our strategy to Matcha-TTS confirmed that our approach is both effective and scalable across diverse TTS tasks and representation domains.

[4] https://github.com/shivammehta25/Matcha-TTS

6. References

[1] K. Shen, Z. Ju, X. Tan, Y. Liu *et al.*, "Naturalspeech 2: Latent diffusion models are natural and zero-shot speech and singing synthesizers," in *Proc. ICLR*, 2023.

[2] Z. Ju, Y. Wang, K. Shen, X. Tan, D. Xin, D. Yang, Y. Liu, Y. Leng, K. Song, S. Tang *et al.*, "Naturalspeech 3: Zero-shot speech synthesis with factorized codec and diffusion models," in *Proc. ICML*, 2024.

[3] K. Lee, D. W. Kim, J. Kim, and J. Cho, "DiTTo-TTS: Efficient and scalable zero-shot text-to-speech with diffusion transformer," in *Proc. ICLR*, 2025.

[4] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *Proc. ICLR*, 2023.

[5] M. Le, A. Vyas, B. Shi, B. Karrer, L. Sari, R. Moritz, M. Williamson, V. Manohar, Y. Adi, J. Mahadeokar, and W.-N. Hsu, "Voicebox: Text-guided multilingual universal speech generation at scale," in *Proc. NIPS*, vol. 36, 2023, pp. 14 005–14 034.

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

[7] S. E. Eskimez, X. Wang, M. Thakker, C. Li *et al.*, "E2 TTS: Embarrassingly easy fully non-autoregressive zero-shot TTS," in *arXiv:2406.18009*, 2024.

[8] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel *et al.*, "Scaling rectified flow transformers for high-resolution image synthesis," in *Proc. ICML*, 2025.

[9] W. Guan, K. Wang, W. Zhou, Y. Wang, F. Deng, H. Wang, L. Li, Q. Hong, and Y. Qin, "LAFMA: A latent flow matching model for text-to-audio generation," in *Proc. INTERSPEECH*, 2024.

[10] Z. Ye, Z. Ju, H. Liu, X. Tan, J. Chen, Y. Lu, P. Sun, J. Pan, W. Bian, S. He *et al.*, "FlashSpeech: Efficient zero-shot speech synthesis," in *Proc. ACM Multimedia*, 2024, pp. 6998–7007.

[11] W. Guan, Q. Su, H. Zhou, S. Miao, X. Xie, L. Li, and Q. Hong, "ReFlow-TTS: A rectified flow model for high-fidelity text-to-speech," in *Proc. ICASSP*, 2024, pp. 10 501–10 505.

[12] Z. Liu, S. Wang, P. Zhu, M. Bi, and H. Li, "E1 TTS: Simple and fast non-autoregressive TTS," *arXiv:2409.09351*, 2024.

[13] S.-H. Lee, H.-Y. Choi, and S.-W. Lee, "Accelerating high-fidelity waveform generation via adversarial flow matching optimization," in *arXiv:2408.08019*, 2024.

[14] X. Liu, X. Zhang, J. Ma, J. Peng *et al.*, "InstaFlow: One step is enough for high-quality diffusion-based text-to-image generation," in *Proc. ICLR*, 2024.

[15] T. Yin, M. Gharbi, T. Park, R. Zhang, E. Shechtman, F. Durand, and W. T. Freeman, "Improved distribution matching distillation for fast image synthesis," in *Proc. NIPS*, 2024.

[16] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. ICLR*, 2014.

[17] J. Ho and T. Salimans, "Classifier-free diffusion guidance," in *Proc. NIPS*, 2021.

[18] S. Mehta, R. Tu, J. Beskow, É. Székely, and G. E. Henter, "Matcha-TTS: A fast tts architecture with conditional flow matching," in *Proc. ICASSP*, 2024, pp. 11 341–11 345.

[19] Y. Shirahata, R. Yamamoto, E. Song, R. Terashima, J.-M. Kim, and K. Tachibana, "Period VITS: Variational inference with explicit pitch modeling for end-to-end emotional speech synthesis," in *Proc. ICASSP*, 2023, pp. 1–5.

[20] J. Kong, J. Kim, and J. Bae, "HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Proc. NIPS*, 2020, pp. 17 022–17 033.

[21] W. Jang, D. Lim, J. Yoon, B. Kim, and J. Kim, "UnivNet: A neural vocoder with multi-resolution spectrogram discriminators for high-fidelity waveform generation," in *Proc. INTERSPEECH*, 2021.

[22] Y. Li and C. Wang, "Improve gan-based neural vocoder using truncated pointwise relativistic least square gan," in *Proceedings of the 4th International Conference on Advanced Information Science and System*, 2022, pp. 1–7.

[23] J. Kim, J. Kong, and J. Son, "Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech," in *Proc. ICML*, 2021, pp. 5530–5540.

[24] H. Yoon, C. Kim, S. Um, H.-W. Yoon, and H.-G. Kang, "SC-CNN: Effective speaker conditioning method for zero-shot multi-speaker text-to-speech systems," in *IEEE SPL.*, 2023.

[25] H.-J. Heo, U.-H. Shin, R. Lee, Y. Cheon, and H.-M. Park, "NeXt-TDNN: Modernizing multi-scale temporal convolution backbone for speaker verification," in *Proc. ICASSP*, 2024.

[26] R. Badlani, A. Łańcucki, K. J. Shih, R. Valle, W. Ping, and B. Catanzaro, "One TTS alignment to rule them all," in *Proc. ICASSP*, 2022, pp. 6092–6096.

[27] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proc. ICCV*, 2023, pp. 4195–4205.

[28] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, p. 127063, 2024.

[29] J. Chen, Y. Jincheng, G. Chongjian, L. Yao, E. Xie, Z. Wang, J. Kwok, P. Luo, H. Lu, and Z. Li, "Pixart-$alpha$: Fast training of diffusion transformer for photorealistic text-to-image synthesis," in *Proc. ICLR*, 2023.

[30] J. Yang, J. Lee, H.-S. Choi, S. Ji, H. Kim, and J. Lee, "DualSpeech: Enhancing speaker-fidelity and text-intelligibility through dual classifier-free guidance," in *Proc. INTERSPEECH*, 2024, pp. 4423–4427.

[31] Y. Lee, I. Yeon, J. Nam, and J. S. Chung, "VoiceLDM: Text-to-speech with environmental context," in *Proc. ICASSP*, 2024.

[32] J. Yang, J.-S. Bae, T. Bak, Y. Kim, and H.-Y. Cho, "GANSpeech: Adversarial training for high-fidelity multi-speaker speech synthesis," in *Proc. INTERSPEECH*, 2021.

[33] 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," in *Proc. INTERSPEECH*, 2019.

[34] J. Yamagishi, C. Veaux, K. MacDonald *et al.*, "Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit (version 0.92)," 2019.

[35] E. Bakhturina, V. Lavrukhin, B. Ginsburg, and Y. Zhang, "Hi-fi multi-speaker english tts dataset," in *Proc. INTERSPEECH*, 2021.

[36] T. A. Nguyen, W.-N. Hsu, A. d'Avirro, B. Shi, I. Gat, M. Fazel-Zarani, T. Remez, J. Copet, G. Synnaeve, M. Hassid *et al.*, "Expresso: A benchmark and analysis of discrete expressive speech resynthesis," in *Proc. INTERSPEECH*, 2023, pp. 4823–4827.

[37] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *Proc. ICASSP*, 2015, pp. 5206–5210.

[38] K. Yu and S. Young, "Continuous f0 modeling for hmm based statistical parametric speech synthesis," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 5, 2010.

[39] M. Morise, F. Yokomori, and K. Ozawa, "World: a vocoder-based high-quality speech synthesis system for real-time applications," *IEICE TRANSACTIONS on Information and Systems*, 2016.

[40] 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 *Proc. ICML*, 2022.

[41] S.-H. Lee, H.-Y. Choi, S.-B. Kim, and S.-W. Lee, "Hierspeech++: Bridging the gap between semantic and acoustic representation of speech by hierarchical variational inference for zero-shot speech synthesis," *arXiv:2311.12454*, 2023.

[42] C. Wang, S. Chen, Y. Wu, Z. Zhang *et al.*, "Neural codec language models are zero-shot text to speech synthesizers," in *arXiv:2301.02111*, 2023.

[43] J. Kim, K. Lee, S. Chung, and J. Cho, "CLaM-TTS: Improving neural codec language model for zero-shot text-to-speech," in *Proc. ICLR*, 2024.