FlashSpeech: Efficient Zero-Shot Speech Synthesis

Anonymous Authors

ABSTRACT

Recent progress in large-scale zero-shot speech synthesis has been significantly advanced by language models and diffusion models. However, the generation process of both methods is slow and computationally intensive. Efficient speech synthesis using a lower computing budget to achieve quality on par with previous work remains a significant challenge. In this paper, we present FlashSpeech, a large-scale zero-shot speech synthesis system with approximately 5% of the inference time compared with previous work. FlashSpeech is built on the latent consistency model and applies a novel adversarial consistency training approach that can train from scratch without the need for a pre-trained diffusion model as the teacher. Furthermore, a new prosody generator module enhances the diversity of prosody, making the rhythm of the speech sound more natural. The generation processes of FlashSpeech can be achieved efficiently with one or two sampling steps while maintaining high audio quality and high similarity to the audio prompt for zero-shot speech generation. Our experimental results demonstrate the superior performance of FlashSpeech. Notably, FlashSpeech can be about 20 times faster than other zero-shot speech synthesis systems while maintaining comparable performance in terms of voice quality and similarity. Furthermore, FlashSpeech demonstrates its versatility by efficiently performing tasks like voice conversion, speech editing, and diverse speech sampling. Audio samples can be found in https://flashspeech.github.io/.

CCS CONCEPTS

• **Applied computing** → **Sound and music computing**; • **Computing methodologies** → **Natural language generation**.

KEYWORDS

Zero-Shot Speech Synthesis, Latent Consistency Model, Adversarial Training

1 INTRODUCTION

In recent years, the landscape of speech synthesis has been transformed by the advent of large-scale generative models. Consequently, the latest research efforts have achieved notable advancements in zero-shot speech synthesis systems by significantly increasing the size of both datasets and models. Zero-shot speech synthesis, such as text-to-speech (TTS), voice conversion (VC) and Editing, aims to generate speech that incorporates unseen speaker characteristics from a reference audio segment during inference, without the need for additional training. Current advanced zero-shot speech synthesis systems typically leverage language models (LMs) [20, 22, 39, 59, 61, 63, 66] and diffusion-style models [18, 23, 26, 52] for in-context speech generation on the large-scale dataset. However, the generation process of these methods needs a long-time iteration. For example, VALL-E [59] builds on the language model to predict 75 audio token sequences for a 1-second speech, in its first-stage autoregressive (AR) token sequence generation. When using a non-autoregressive (NAR) latent diffusion model [47] based framework, NaturalSpeech 2 [52] still requires 150 sampling steps. As a result, although these methods can produce human-like speech, they require significant computational time and cost. Some efforts have been made to accelerate the generation process. Voicebox [26] adopts flow-matching [29] so that fewer sampling steps (NFE[1]: 64) can be achieved because of the optimal transport path. ClaM-TTS [22] proposes a mel-codec with a superior compression rate and a latent language model that generates a stack of tokens at once. Although the slow generation speed issue has been somewhat alleviated, the inference speed is still far from satisfactory for practical applications. Moreover, the substantial computational time of these approaches leads to significant computational cost overheads, presenting another challenge.

The fundamental limitation of speech generation stems from the intrinsic mechanisms of language models and diffusion models, which require considerable time either auto-regressively or through a large number of denoising steps. Hence, the primary objective of this work is to accelerate inference speed and reduce computational costs while preserving generation quality at levels comparable to

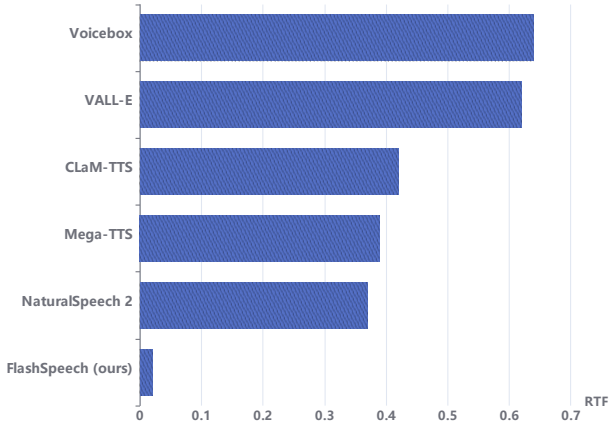

Figure 1: The inference time comparisons of different zero-shot speech synthesis systems using the real-time factor (RTF).

[1]NFE: number of function evaluations.

Unpublished working draft. Not for distribution.

ACM MM, 2024, Melbourne, Australia

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

the prior research. In this paper, we propose FlashSpeech as the next step towards efficient zero-shot speech synthesis. To address the challenge of slow generation speed, we leverage the latent consistency model (LCM) [32], a recent advancement in generative models. Building upon the previous non-autoregressive TTS system [52], we adopt the encoder of a neural audio codec to convert speech waveforms into latent vectors as the training target for our LCM. To train this model, we propose a novel technique called adversarial consistency training, which utilizes the capabilities of pre-trained speech language models [1, 6, 11] as discriminators. This facilitates the transfer of knowledge from large pre-trained speech language models to speech generation tasks, efficiently integrating adversarial and consistency training to improve performance. The LCM is conditioned on prior vectors obtained from a phoneme encoder, a prompt encoder, and a prosody generator. Furthermore, we demonstrate that our proposed prosody generator leads to more diverse expressions and prosody while preserving stability.

Our contributions can be summarized as follows:

- We propose FlashSpeech, an efficient zero-shot speech synthesis system that generates voice with high audio quality and speaker similarity in zero-shot scenarios.
- We introduce adversarial consistency training, a novel combination of consistency and adversarial training leveraging pre-trained speech language models, for training the latent consistency model from scratch, achieving speech generation in one or two steps.
- We propose a prosody generator module that enhances the diversity of prosody while maintaining stability.
- FlashSpeech significantly outperforms strong baselines in audio quality and matches them in speaker similarity. Remarkably, it achieves this at a speed approximately 20 times faster than comparable systems, demonstrating unprecedented efficiency.

2 RELATED WORK

2.1 Large Scale Speech Synthesis

Motivated by the success of the large language model, the speech research community has recently shown increasing interest in scaling the sizes of model and training data to bolster generalization capabilities, producing natural speech with diverse speaker identities and prosody under zero-shot settings. The pioneering work is VALL-E [59], which adopts the Encodec [8] to discretize the audio waveform into tokens. Therefore, a language model can be trained via in-context learning that can generate the target utterance where the style is consistent with prompt utterance. However, generating audio in such an autoregressive manner [39, 61]can lead to unstable prosody, word skipping, and repeating issues [44, 52, 57]. To ensure the robustness of the system, non-autoregressive methods such as NaturalSpeech2 [52] and Voicebox [26] utilize diffusion-style model (VP-diffusion [55] or flow-matching [29]) to learn the distribution of a continuous intermediate vector such as mel-spectrogram or latent vector of codec. Both LM-based methods [67] and diffusion-based methods show superior performance in speech generation tasks. However, their generation is slow due to the iterative computation. Considering that many speech generation scenarios require real-time inference and low computational costs, we employ the latent consistency model for large-scale speech generation that inference with one or two steps while maintaining high audio quality.

2.2 Acceleration of Speech Synthesis

Since early neural speech generation models [57] use autoregressive models such as Tacotron [60] and TransformerTTS [27], causing slow inference speed, with $O(N)$ computation, where N is the sequence length. To address the slow inference speed, FastSpeech [44, 45] proposes to generate a mel-spectrogram in a non-autoregressive manner. However, these models [46] result in blurred and over-smoothed mel-spectrograms due to the regression loss they used and the capability of modeling methods. To further enhance the speech quality, diffusion models are utilized [13, 40, 41] which increase the computation to $O(T)$, where T is the diffusion steps. Therefore, distillation techniques [33] for diffusion-based methods such as CoMoSpeech [64], CoMoSVC [31] and Reflow-TTS [10] emerge to reduce the sampling steps back to $O(1)$, but require additional pre-trained diffusion as the teacher model. Unlike previous distillation techniques, which require extra training for the diffusion model as a teacher and are limited by its performance, our proposed adversarial consistency training technique can directly train from scratch, significantly reducing training costs. In addition, previous acceleration methods only validate speaker-limited recording-studio datasets with limited data diversity. To the best of our knowledge, FlashSpeech is the first work that reduces the computation of a large-scale speech generation system back to $O(1)$.

2.3 Consistency Model

The consistency model is proposed in [53, 54] to generate high-quality samples by directly mapping noise to data. Furthermore, many variants [21, 25, 30, 51] are proposed to further increase the generation quality of images. The latent consistency model is proposed by [32] which can directly predict the solution of PF-ODE in latent space. However, the original LCM employs consistency distillation on the pre-trained latent diffusion model (LDM) which leverages large-scale off-the-shelf image diffusion models [47]. Since there are no pre-trained large-scale TTS models in the speech community, and inspired by the techniques [21, 25, 30, 51, 53], we propose the novel adversarial consistency training method which can directly train the large-scale latent consistency model from scratch utilizing the large pre-trained speech language model [1, 6, 11] such as WavLM for speech generation.

3 FLASHSPEECH

3.1 Overview

Our work is dedicated to advancing the speech synthesis efficiency, achieving $O(1)$ computation cost while maintaining comparable performance to prior studies that require $O(T)$ or $O(N)$ computations. The framework of the proposed method, FlashSpeech, is illustrated in Fig. 2. FlashSpeech integrates a neural codec, an encoder for phonemes and prompts, a prosody generator, and an LCM, which are utilized during both the training and inference stages. Exclusively during training, a conditional discriminator is employed. FlashSpeech adopts the in-context learning paradigm [59], initially segmenting the latent vector $\mathbf{z}$, extracted from the

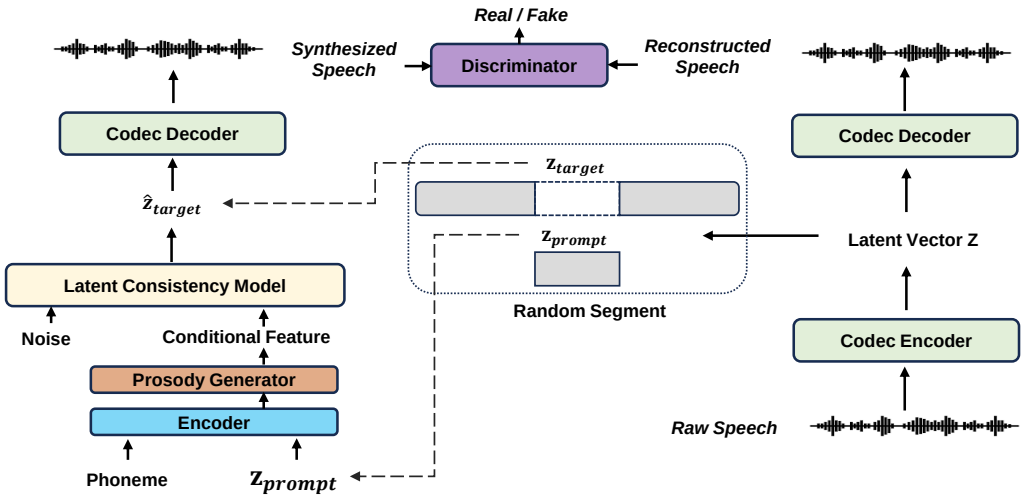

Figure 2: Overall architecture of FlashSpeech. Our FlashSpeech consists of a codec encoder/decoder and a latent consistency model conditioned on feature from a phoneme and z_{prompt} encoder and a prosody generator. A discriminator is used during training.

codec, into $\mathbf{z}_{target}$ and $\mathbf{z}_{prompt}$. Subsequently, the phoneme and $\mathbf{z}_{prompt}$ are processed through the encoder to produce the hidden feature. A prosody generator then predicts pitch and duration based on the hidden feature. The pitch and duration embeddings are combined with the hidden feature and inputted into the LCM as the conditional feature. The LCM model is trained from scratch using adversarial consistency training. After training, FlashSpeech can achieve efficient generation within one or two sampling steps.

3.2 Latent Consistency Model

The consistency model [54] is a new family of generative models that enables one-step or few-step generation. Let us denote the data distribution by $p_{\text{data}}(\mathbf{x})$. The core idea of the consistency model is to learn the function that maps any points on a trajectory of the PF-ODE to that trajectory's origin, which can be formulated as:

$$f(\mathbf{x}_\sigma, \sigma) = \mathbf{x}_{\sigma_{min}} \tag{1}$$

where $f(\cdot,\cdot)$ is the consistency function and $\mathbf{x}_\sigma$ represents the data $\mathbf{x}$ perturbed by adding zero-mean Gaussian noise with standard deviation σ. σ_{min} is a fixed small positive number. Then $\mathbf{x}_{\sigma_{min}}$ can then be viewed as an approximate sample from the data distribution $p_{\text{data}}(\mathbf{x})$. To satisfy property in equation (1), following [54], we parameterize the consistency model as

$$f_\theta(\mathbf{x}_\sigma, \sigma) = c_{\text{skip}}(\sigma)\mathbf{x} + c_{\text{out}}(\sigma)F_\theta(\mathbf{x}_\sigma, \sigma) \tag{2}$$

where f_θ is to estimate consistency function f by learning from data, F_θ is a deep neural network with parameter θ, $c_{\text{skip}}(\sigma)$ and $c_{\text{out}}(\sigma)$ are are differentiable functions with $c_{\text{skip}}(\sigma_{min}) = 1$ and $c_{\text{out}}(\sigma_{min}) = 0$ to ensure boundary condition. A valid consistency model should satisfy the self-consistency property [54]

$$f_\theta(\mathbf{x}_\sigma, \sigma) = f_\theta(\mathbf{x}_{\sigma'}, \sigma'), \quad \forall \sigma, \sigma' \in [\sigma_{min}, \sigma_{max}]. \tag{3}$$

where $\sigma_{max} = 80$ and $\sigma_{min} = 0.002$ following [19, 53, 54]. Then the model can generate samples in one step by evaluating

$$\mathbf{x}_{\sigma_{min}} = f_\theta(\mathbf{x}_{\sigma_{max}}, \sigma_{max}) \tag{4}$$

from distribution $\mathbf{x}_{\sigma_{max}} \sim \mathcal{N}(0, \sigma^2_{max}\mathbf{I})$.

As we apply a consistency model on the latent space of audio, we use the latent features z which are extracted prior to the residual quantization layer of the codec,

$$\mathbf{z} = CodecEncoder(\mathbf{y}) \tag{5}$$

where $\mathbf{y}$ is the speech waveform. Furthermore, we add the feature from the prosody generator and encoder as the conditional feature c, our objective has changed to achieve

$$f_\theta(\mathbf{z}_\sigma, \sigma, c) = f_\theta(\mathbf{z}_{\sigma'}, \sigma', c) \quad \forall \sigma, \sigma' \in [\sigma_{min}, \sigma_{max}]. \tag{6}$$

During inference, the synthesized waveform $\hat{y}$ is transformed from $\hat{z}$ via the codec decoder. The predicted $\hat{z}$ is obtained by one sampling step

$$\hat{\mathbf{z}} = f_\theta(\epsilon * \sigma_{max}, \sigma_{max}) \tag{7}$$

or two sampling steps

$$\hat{\mathbf{z}}_{\text{inter}} = f_\theta(\epsilon * \sigma_{max}, \sigma_{max}) \tag{8}$$

$$\hat{\mathbf{z}} = f_\theta(\hat{\mathbf{z}}_{\text{inter}} + \epsilon * \sigma_{\text{inter}}, \sigma_{\text{inter}}) \tag{9}$$

where $\hat{\mathbf{z}}_{\text{inter}}$ means the intermediate step, σ_{inter} is set to 2 empirically. ϵ is sampled from a standard Gaussian distribution.

3.3 Adversarial Consistency Training

A major drawback of the LCM[32] is that it needs to pre-train a diffusion-based teacher model in the first stage, and then perform distillation to produce the final model. This would make the training process complicated, and the performance would be limited as a result of the distillation. To eliminate the reliance on the teacher

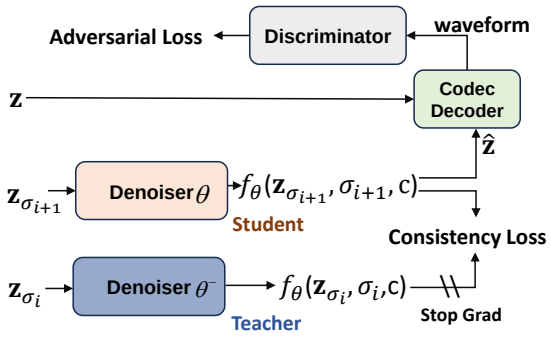

Figure 3: An illustration of adversarial consistency training.

model training, in this paper, we propose a novel adversarial consistency training method to train LCM from scratch. Our training procedure is outlined in Fig. 3, which has three parts:

3.3.1 Consistency Training. To achieve the property in equation (3), we adopt following consistency loss

$$\mathcal{L}_{ct}^{N}(\theta, \theta^{-}) = \mathbb{E}[\lambda(\sigma_i) d(f_\theta(\mathbf{z}_{i+1}, \sigma_{i+1}, c), f_{\theta^-}(\mathbf{z}_i, \sigma_i, c))]. \quad (10)$$

where σ_i represents the noise level at discrete time step i, $d(\cdot,\cdot)$ is the distance function, $f_\theta(\mathbf{z}_{i+1}, \sigma_{i+1}, c)$ and $f_{\theta^-}(\mathbf{z}_i, \sigma_i, c)$ are the student with the higher noise level and the teacher with the lower noise level, respectively. The discrete time steps denoted as $\sigma_{min} = \sigma_0 < \sigma_1 < \cdots < \sigma_N = \sigma_{max}$ are divided from the time interval $[\sigma_{min}, \sigma_{max}]$, where the discretization curriculum N increases correspondingly as the number of training steps grows

$$N(k) = \min(s_0 2^{\lfloor \frac{k}{K'} \rfloor}, s_1) + 1 \quad (11)$$

where $K' = \left\lfloor \frac{K}{\log_2 \lfloor s_1/s_0 \rfloor + 1} \right\rfloor$, k is the current training step and K is the total training steps. s_1 and s_0 are hyperparameters to control the size of $N(k)$. The distance function $d(\cdot,\cdot)$ uses the Pseudo-Huber metric [3]

$$d(x, y) = \sqrt{\|x - y\|^2 + a^2} - a, \quad (12)$$

where a is an adjustable constant, making the training more robust to outliers as it imposes a smaller penalty for large errors than ℓ_2 loss. The parameters θ^- of teacher model are

$$\theta^- \longleftarrow stopgrad(\theta), \quad (13)$$

which are identical to the student parameters θ. This approach [53] has been demonstrated to improve sample quality of previous strategies that employ varying decay rates [54]. The weighting function refers to

$$\lambda(\sigma_i) = \frac{1}{\sigma_{i+1} - \sigma_i} \quad (14)$$

which emphasizes the loss of smaller noise levels. LCM through consistency training can generate speech with acceptable quality in a few steps, but it still falls short of previous methods. Therefore, to further enhance the quality of the generated samples, we integrate adversarial training.

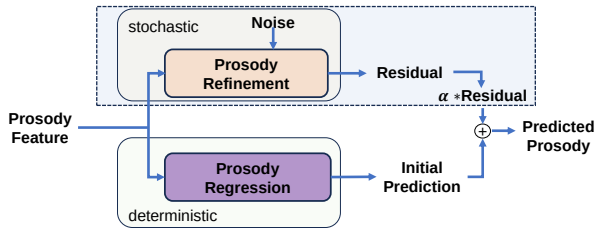

Figure 4: An illustration of prosody generator.

3.3.2 Adversarial Training. For the adversarial objective, the generated samples $\hat{\mathbf{z}} \leftarrow f_\theta(\mathbf{z}_\sigma, \sigma, c)$ and real samples $\mathbf{z}$ are passed to the discriminator D_η which aims to distinguish between them, where η refers to the trainable parameters. Thus, we employ adversarial training loss

$$\mathcal{L}_{\text{adv}}(\theta, \eta) = \mathbb{E}_{\mathbf{z}}[\log D_\eta(\mathbf{z})] + \mathbb{E}_\sigma \mathbb{E}_{z_\sigma}[\log(1 - D_\eta(f_\theta(\mathbf{z}_\sigma, \sigma, c)))]. \quad (15)$$

In this way, the error signal from the discriminator guides f_θ to produce more realistic outputs. For details, we use a frozen pre-trained speech language model *SLM* and a trainable lightweight discriminator head D_{head} to build the discriminator. Since the current *SLM* is trained on the speech waveform, we covert both $\mathbf{z}$ and $\hat{\mathbf{z}}$ to ground truth waveform and predicted waveform using the codec decoder. To further increase the similarity between prompt audio and generated audio, our discriminator is conditioned on the prompt audio feature. This prompt feature F_{prompt} is extracted using *SLM* on prompt audio and applies average pooling on the time axis. Therefore,

$$D_\eta = D_{\text{head}}(F_{\text{prompt}} \odot F_{\text{gt}}, F_{\text{prompt}} \odot F_{\text{pred}}) \quad (16)$$

where F_{gt} and F_{pred} refer to feature extracted through *SLM* for ground truth waveform and predicted waveform. The discriminator head consists of several 1D convolution layers. The input feature of the discriminator is conditioned on F_{prompt} via projection [36].

3.3.3 Combined Together. Since there is a large gap on the loss scale between consistency loss and adversarial loss, it can lead to instability and failure in training. Therefore, we follow [9] to compute the adaptive weight with

$$\lambda_{adv} = \frac{\|\nabla_{\theta_L} \mathcal{L}_{\text{ct}}^{N}(\theta, \theta^-)\|}{\|\nabla_{\theta_L} \mathcal{L}_{\text{adv}}(\theta, \eta)\|} \quad (17)$$

where θ_L is the last layer of the neural network in LCM. The final loss of training LCM is defined as $\mathcal{L}_{\text{ct}}^{N}(\theta, \theta^-) + \lambda_{adv} \mathcal{L}_{\text{adv}}(\theta, \eta)$. This adaptive weighting significantly stabilizes the training by balancing the gradient scale of each term.

3.4 Prosody Generator

3.4.1 Analysis of the prediction of prosody. Previous regression methods for prosody prediction [44, 52], due to their deterministic mappings and assumptions of unimodal distribution, often fail to capture the inherent diversity and expressiveness of human speech prosody. This leads to predictions that lack variation and can appear over-smoothed. On the other hand, diffusion methods [26, 28] for prosody prediction offer a promising alternative by providing greater prosody diversity. However, they come with challenges

regarding stability, and the potential for unnatural prosody. Additionally, the iterative inference process in DMs requires a significant number of sampling steps that may also hinder real-time application. Meanwhile, LM-based methods [15, 59] also need a long time for inference. To alleviate these issues, our prosody generator consists of a prosody regression module and a prosody refinement module to enhance the diversity of prosody regression results with efficient one-step consistency model sampling.

3.4.2 Prosody refinement via consistency model. As shown in 4, our prosody generator consists of two parts which are prosody regression and prosody refinement. We first train the prosody regression module to get a deterministic output. Next, we freeze the parameters of the prosody regression module and use the residual of ground truth prosody and deterministic predicted prosody as the training target for prosody refinement. We adopt a consistency model as a prosody refinement module. The conditional feature of the consistency model is the feature from prosody regression before the final projection layer. Thus, the residual from a stochastic sampler refines the output of a deterministic prosody regression and produces a diverse set of plausible prosody under the same transcription and audio prompt. One option for the final prosody output p_{final} can be represented as:

$$p_{\text{final}} = p_{\text{res}} + p_{\text{init}}, \tag{18}$$

where p_{final} denotes the final prosody output, p_{res} represents the residual output from the prosody refinement module, capturing the variations between the ground truth prosody and the deterministic prediction, p_{init} is the initial deterministic prosody prediction from the prosody regression module. However, this formulation may negatively affect prosody stability, a similar observation is found in [26, 58]. More specifically, higher diversity may cause less stability and sometimes produce unnatural prosody. To address this, we introduce a control factor α that finely tunes the balance between stability and diversity in the prosodic output:

$$p_{\text{final}} = \alpha p_{\text{res}} + p_{\text{init}} \tag{19}$$

where α is a scalar value ranging between 0 and 1. This adjustment allows for controlled incorporation of variability into the prosody, mitigating issues related to stability while still benefiting from the diversity offered by the prosody refinement module.

3.5 Applications

This section elaborates on the practical applications of FlashSpeech. We delve into its deployment across various tasks such as zero-shot TTS, speech editing, voice conversion, and diverse speech sampling. All the sample audios of applications are available on the demo page.

3.5.1 Zero-Shot TTS. Given a target text and reference audio, we first convert the text to phoneme using g2p (grapheme-to-phoneme conversion). Then we use the codec encoder to convert the reference audio into $\mathbf{z}_{prompt}$. Speech can be synthesized efficiently through FlashSpeech with the phoneme input and $\mathbf{z}_{prompt}$, achieving high-quality text-to-speech results without requiring pre-training on the specific voice.

3.5.2 Voice Conversion. Voice conversion aims to convert the source audio into the target audio using the speaker's voice of the reference audio. Following [43, 52], we first apply the reverse of ODE to diffuse the source audio into a starting point that still maintains some information in the source audio. After that, we run the sampling process from this starting point with the reference audio as $\mathbf{z}_{prompt}$ and condition c. The condition c uses the phoneme and duration from the source audio and the pitch is predicted by the prosody generator. This method allows for zero-shot voice conversion while preserving the linguistic content of the source audio, and achieving the same timbre as the reference audio.

3.5.3 Speech Editing. Given the speech, the original transcription, and the new transcription, we first use MFA (Montreal Forced Aligner) to align the speech and the original transcription to get the duration of each word. Then we remove the part that needs to be edited to construct the reference audio. Next, we use the new transcription and reference to synthesize new speech. Since this task is consistent with the in-context learning, we can concatenate the remaining part of the raw speech and the synthesized part as the final speech, thus enabling precise and seamless speech editing.

3.5.4 Diverse Speech Sampling. FlashSpeech leverages its inherent stochasticity to generate a variety of speech outputs under the same conditions. By employing stochastic sampling in its prosody generation and LCM, FlashSpeech can produce diverse variations in pitch, duration, and overall audio characteristics from the same phoneme input and audio prompt. This feature is particularly useful for generating a wide range of speech expressions and styles from a single input, enhancing applications like voice acting, synthetic voice variation for virtual assistants, and more personalized speech synthesis. In addition, the synthetic data via speech sampling can also benefit other tasks such as ASR [48].

4 EXPERIMENT

In the experimental section, we begin by introducing the datasets and the configurations for training in our experiments. Following this, we show the evaluation metrics and demonstrate the comparative results against various zero-shot TTS models. Subsequently, ablation studies are conducted to test the effectiveness of several design choices. Finally, we also validate the effectiveness of other tasks such as voice conversion. We show our speech editing and diverse speech sampling results on our demo page.

4.1 Experimental Settings

4.1.1 Data and Preprocessing. We use the English subset of Multilingual LibriSpeech (MLS) [42], including 44.5k hours of transcribed audiobook data and it contains 5490 distinct speakers. The audio data is resampled at a frequency of 16kHz. The input text is transformed into a sequence of phonemes through grapheme-to-phoneme conversion [56] and then we use our internal alignment tool aligned with speech to obtain the phoneme-level duration. We adopt a hop size of 200 for all frame-level features. The pitch sequence is extracted using PyWorld[2]. we adopt Encodec [8] as our audio codec. We use a modified version [3] and train it on MLS. We

[2]https://github.com/JeremyCCHsu/Python-Wrapper-for-World-Vocoder
[3]https://github.com/yangdongchao/UniAudio/tree/main/codec

Table 1: The evaluation results for FlashSpeech and the baseline methods on LibriSpeech testclean. ⋆ means the evaluation is conducted with 1 NVIDIA V100 GPU. ♢ means the device is not available. Abbreviation: MLS (Multilingual LibriSpeech [42]),G (GigaSpeech [4]), L(LibriTTS-R [24]), V(VCTK [62]), LJ(LJSpeech [12]), W(WenetSpeech [65]).

Model	Training Data	RTF ↓	Sim-O ↑	Sim-R ↑	WER ↓	CMOS ↑	SMOS (↑)
GroundTruth	-	-	0.68	-	1.9	0.11	4.39
VALL-E reproduce [59]	Librilight	0.62 ♢	0.47	0.51	6.1	-0.48	4.11
NaturalSpeech 2 [52]	MLS	0.37 (NFE:150) ⋆	**0.53**	**0.60**	**1.9**	-0.31	4.20
Voicebox reproduce [26]	Librilight	0.66 (NFE:64) ♢	0.48	0.50	2.1	-0.58	3.95
Mega-TTS [17]	G+W	0.39 ♢	-	-	3.0	-	-
CLaM-TTS [22]	MLS+G+L+V+LJ	0.42 ♢	0.50	0.54	5.1	-	-
FlashSpeech (ours)	MLS	**0.02** (NFE: 2) ⋆	0.52	0.57	2.7	**0.00**	**4.29**

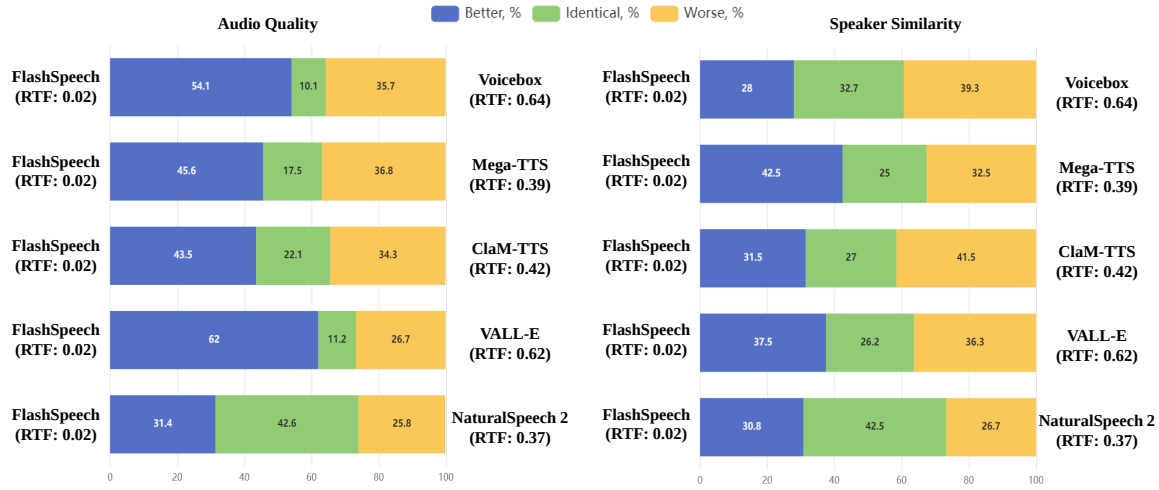

Figure 5: User preference study. We compare the audio quality and speaker similarity of FlashSpeech against baselines with their official demo.

use the dense features extracted before the residual quantization layer as our latent vector z.

4.1.2 Training Details . Our training consists of two stages, in the first stage we train LCM and the prosody regression part. We use 8 H800 80GB GPUs with a batch size of 20k frames of latent vectors per GPU for 650k steps. We use the AdamW optimizer with a learning rate of 3e-4, warm up the learning rate for the first 30k updates and then linear decay it. We deactivate adversarial training with $\lambda_{adv} = 0$ before 600K training iterations. For hyper-parameters, we set a in Equation (12) to 0.03. In equation (10), $\sigma_i = \left(\sigma_{\min}^{1/\rho} + \frac{i-1}{N(k)-1}\left(\sigma_{\max}^{1/\rho} - \sigma_{\min}^{1/\rho}\right)\right)^{\rho}$, where $i \in [1, N(k)]$, $\rho = 7$, $\sigma_{\min} = 0.002$, $\sigma_{\max} = 80$. For N(k) in Equation (11), we set $s_0 = 10, s_1 = 1280, K = 600k$. After 600k steps, we activate adversarial loss, and N(k) can be considered as fixed to 1280. We crop the waveform length fed into the discriminator into minimum waveform length in a minibatch. In addition, the weight of the feature extractor WavLM and the codec decoder are frozen.

In the second stage, we train 150k steps for the prosody refinement module with consistency training in Equation (10). Different from the above setting, we empirically set $s_1 = 160$, $K = 150k$. During training, only the weight of the prosody refinement part is updated.

4.1.3 Model Details . The model structures of the prompt encoder and phoneme encoder are follow[52]. The neural function part in LCM is almost the same as the [52]. We rescale the sinusoidal position embedding in the neural function part by a factor of 1000. As for the prosody generator, we adopt 30 non-casual wavenet [37] layers for the neural function part in the prosody refinement module and the same configurations for prosody regression parts in [52]. And we set $\alpha = 0.2$ for the prosody refinement module empirically. For the discriminator's head, we stack 5 convolutional layers with weight normalization [50] for binary classification.

4.2 Evaluation Metrics

We use both objective and subjective evaluation metrics, including

- **RTF**: Real-time-factor (RTF) measures the time taken for the system to generate one second of speech. This metric is crucial for evaluating the efficiency of our system, particularly for applications requiring real-time processing. We measure

the time of our system end-to-end on an NVIDIA V100 GPU following [52].

- **Sim-O and Sim-R**: These metrics assess the speaker similarity. Sim-R measures the objective similarity between the synthesized speech and the reconstruction reference speech through the audio codec, using features embedding extracted from the pre-trained speaker verification model [22, 59][4]. Sim-O is calculated with the original reference speech. Higher scores in Sim-O and Sim-R indicate a higher speaker similarity.
- **WER (Word Error Rate)**: To evaluate the accuracy and clarity of synthesized speech from the TTS system, we employ the Automatic Speech Recognition (ASR) model [59] [5] to transcribe generated audio. The discrepancies between these transcriptions and original texts are quantified using the Word Error Rate (WER), a crucial metric indicating intelligibility and robustness.
- **CMOS, SMOS, UTMOS**: we rank the comparative mean option score (CMOS) and similarity mean option score (SMOS) using mturk. The prompt for CMOS refers to 'Please focus on the audio quality and naturalness and ignore other factors.'. The prompt for SMOS refers to 'Please focus on the similarity of the speaker to the reference, and ignore the differences of content, grammar or audio quality.' Each audio has been listened to by at least 10 listeners. UTMOS [49] is a Speech MOS predictor[6] to measure the naturalness of speech. We use it in ablation studies which reduced the cost for evaluation.
- **Prosody JS Divergence**: To evaluate the diversity and accuracy of the prosody prediction in our TTS system, we include the Prosody JS Divergence metric. This metric employs the Jensen-Shannon (JS) divergence [35] to quantify the divergence between the predicted and ground truth prosody feature distributions. Prosody features, including pitch, and duration, are quantized and their distributions in both synthesized and natural speech are compared. Lower JS divergence values indicate closer similarity between the predicted prosody features and those of the ground truth, suggesting a higher diversity of the synthesized speech.

4.3 Experimental Results on Zero-shot TTS

Following [59], We employ LibriSpeech [38] test-clean for zero-shot TTS evaluation. We adopt the cross-sentence setting in [59] that we randomly select 3-second clips as prompts from the same speaker's speech. The results are summarized in table 1 and figure 5.

4.3.1 Evaluation Baselines.

- VALL-E [59]: VALL-E predicts codec tokens using both AR and NAR models. RTF[7] is obtained from [22, 26]. We use our reproduced results for MOS, Sim, and WER. Additionally, we do a preference test with their official demo.
- Voicebox [26]: Voicebox uses flow-matching to predict maksed mel-spectrogram. RTF is from the original paper. We use our reproduced results for MOS, Sim, and WER. We also implement a preference test with their official demo.
- NaturalSpeech2 [52]: NaturalSpeech2 uses a latent diffusion model to predict latent features of codec. The RTF is from the original paper. the Sim, WER and samples for MOS are obtained through communication with the authors. We also do a preference test with their official demo.
- Mega-TTS [17][8]: Mega-TTS uses both language model and GAN to predict mel-spectrogram. We obtain RTF from mobilespeech [14] and WER from the original paper. We do a preference test with their official demo.
- ClaM-TTS [22]: ClaM-TTS uses the AR model to predict mel codec tokens. We obtain the objective evaluation results from the original paper and do a preference test with their official demo.

4.3.2 Generation Quality. FlashSpeech stands out significantly in terms of speaker quality, surpassing other baselines in both CMOS and audio quality preference tests. Notably, our method closely approaches ground truth recordings, underscoring its effectiveness. These results affirm the superior quality of FlashSpeech in speech synthesis. our method.

4.3.3 Generation Similarity. Our evaluation of speaker similarity utilizes Sim, SMOS, and speaker similarity preference tests, where our methods achieve 1st, 2nd, and 3rd place rankings, respectively. These findings validate our methods' ability to achieve comparable speaker similarity to other methods. Despite our training data (MLS) containing approximately 5k speakers, fewer than most other methods (e.g., Librilight with about 7k speakers or self-collected data), we believe that increasing the number of speakers in our methods can further enhance speaker similarity.

4.3.4 Robustness. Our methods achieve a WER of 2.7, placing them in the first echelon. This is due to the non-autoregressive nature of our methods, which ensures robustness.

4.3.5 Generation Speed. FlashSpeech achieves a remarkable approximately 20x faster inference speed compared to previous work. Considering its excellent audio quality, robustness, and comparable speaker similarity, our method stands out as an efficient and effective solution in the field of large-scale speech synthesis.

4.4 Ablation Studies

4.4.1 Ablation studies of LCM. We explored the impact of different pre-trained models in adversarial training on UTMOS and Sim-O. As shown in the table 2, the baseline, which employs consistency training alone, achieved a UTMOS of 3.62 and a Sim-O of 0.45. Incorporating adversarial training using wav2vec2-large[9], hubert-large[10], and wavlm-large[11] as discriminators significantly improved both UTMOS and Sim-O scores. Notably, the application of adversarial training with Wavlm-large achieved the highest scores

[4] https://github.com/microsoft/UniSpeech/tree/main/downstreams/speaker_verification
[5] https://huggingface.co/facebook/hubert-large-ls960-ft
[6] https://github.com/tarepan/SpeechMOS
[7] In CLaM-TTS and Voicebox, they report the inference time for generating 10 seconds of speech. Therefore, we divide by 10 to obtain the time for generating 1 second of speech (RTF).
[8] Since we do not find any audio samples for Mega-TTS2 [16] under the 3-second cross-sentence setting, we are not able to compare with them.
[9] https://huggingface.co/facebook/wav2vec2-large
[10] https://huggingface.co/facebook/hubert-large-ll60k
[11] https://huggingface.co/microsoft/wavlm-large

Table 2: The ablation study of discriminator design.

Method	UTMOS ↑	Sim-O ↑
Consistency training baseline	3.62	0.45
+ Adversarial training (Wav2Vec2-large)	3.92	0.50
+ Adversarial training (Hubert-large)	3.83	0.47
+ Adversarial training (Wavlm-large)	**4.00**	**0.52**
- prompt projection	3.97	0.51

Table 3: The ablation study of sampling steps for LCM

NFE	UTMOS ↑	Sim-O ↑
1	3.99	0.51
2	**4.00**	**0.52**
4	3.91	0.51

(UTMOS: 4.00, Sim-O: 0.52), underscoring the efficacy of this pre-trained model in enhancing the quality and speaker similarity of synthesized speech. Additionally, without using the audio prompt's feature as a condition the discriminator shows a slight decrease in performance (UTMOS: 3.97, Sim-O: 0.51), highlighting the importance of conditional features in guiding the adversarial training process.

As shown in table 3, the effect of sampling steps (NFE) on UTMOS and Sim-O revealed that increasing NFE from 1 to 2 marginally improves UTMOS (3.99 to 4.00) and Sim-O (0.51 to 0.52). However, further increasing to 4 sampling steps slightly reduced UTMOS to 3.91 due to the accumulation of score estimation errors [5, 34]. Therefore, we use 2 steps as the default setting for LCM.

4.4.2 Ablation studies of Prosody Generator. In this part, we investigated the effects of a control factor, denoted as α, on the prosodic features of pitch and duration in speech synthesis, by setting another influencing factor to zero. Our study specifically conducted an ablation analysis to assess how α influences these features, emphasizing its critical role in balancing stability and diversity within our framework's prosodic outputs.

Table 4 elucidates the effects of varying α on the pitch component. With α set to 0, indicating no inclusion of the residual output from prosody refinement, we observed a Pitch JSD of 0.072 and a WER of 2.8. A slight modification to $\alpha = 0.2$ resulted in a reduced Pitch JSD of 0.067, maintaining the same WER. Notably, setting α to 1, fully incorporating the prosody refinement's residual output, further decreased the Pitch JSD to 0.063, albeit at the cost of increased WER to 3.7, suggesting a trade-off between prosody diversity and speech intelligibility.

Similar trends in table 5 are observed in the duration component analysis. With $\alpha = 0$, the Duration JSD was 0.0175 with a WER of 2.8. Adjusting α to 0.2 slightly improved the Duration JSD to 0.0168, without affecting WER. However, fully embracing the refinement module's output by setting $\alpha = 1$ yielded the most significant improvement in Duration JSD to 0.0153, which, similar to pitch analysis, came with an increased WER of 3.9. The results underline the delicate balance required in tuning α to optimize between diversity and stability of prosody without compromising speech intelligibility.

Table 4: The ablation study of control factor for pitch

α	Pitch JSD ↓	WER↓
0	0.072	2.8
0.2	0.067	2.8
1	0.063	3.7

Table 5: The ablation study of control factor for duration

α	Duration JSD ↓	WER ↓
0	0.0175	2.8
0.2	0.0168	2.8
1	0.0153	3.9

Table 6: Voice Conversion

Method	CMOS ↑	SMOS ↑	Sim-O ↑
YourTTS [2]	-0.16	3.26	0.23
DDDM-VC [7]	-0.28	3.43	0.28
Ours	**0.00**	**3.50**	**0.35**

4.5 Evaluation Results for Voice Conversion

In this section, we present the evaluation results of our voice conversion system, FlashSpeech, in comparison with state-of-the-art methods, including YourTTS [12] [2] and DDDM-VC [13] [7]. We conduct the experiments with their official checkpoints in our internal test set.

Our system outperforms both YourTTS and DDDM-VC in terms of CMOS, SMOS and Sim-O, demonstrating its capability to produce converted voices with high quality and similarity to the target speaker. These results confirm the effectiveness of our FlashSpeech approach in voice conversion tasks.

4.6 Conclusions and Future Work

In this paper, we presented FlashSpeech, a novel speech generation system that significantly reduces computational costs while maintaining high-quality speech output. Utilizing a novel adversarial consistency training method and an LCM, FlashSpeech outperforms existing zero-shot TTS systems in efficiency, achieving speeds about 20 times faster without compromising on voice quality, similarity, and robustness. In the future, we aim to further refine the model to improve the inference speed and reduce computational demands. In addition, we will expand the data scale and enhance the system's ability to convey a broader range of emotions and more nuanced prosody. For future applications, FlashSpeech can be integrated for real-time interactions in applications such as virtual assistants and educational tools.

[12] https://github.com/coqui-ai/TTS
[13] https://github.com/hayeong0/DDDM-VC

REFERENCES

[1] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems* 33 (2020), 12449–12460.

[2] Edresson Casanova, Julian Weber, Christopher D Shulby, Arnaldo Candido Junior, Eren Gölge, and Moacir A Ponti. 2022. Yourtts: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone. In *International Conference on Machine Learning*. PMLR, 2709–2720.

[3] Pierre Charbonnier, Laure Blanc-Féraud, Gilles Aubert, and Michel Barlaud. 1997. Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on image processing* 6, 2 (1997), 298–311.

[4] Guoguo Chen, Shuzhou Chai, Guanbo Wang, Jiayu Du, Wei-Qiang Zhang, Chao Weng, Dan Su, Daniel Povey, Jan Trmal, Junbo Zhang, et al. 2021. Gigaspeech: An evolving, multi-domain asr corpus with 10,000 hours of transcribed audio. *arXiv preprint arXiv:2106.06909* (2021).

[5] Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru R Zhang. 2022. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. *arXiv preprint arXiv:2209.11215* (2022).

[6] Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, et al. 2022. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing* 16, 6 (2022), 1505–1518.

[7] Ha-Yeong Choi, Sang-Hoon Lee, and Seong-Whan Lee. 2024. Dddm-vc: Decoupled denoising diffusion models with disentangled representation and prior mixup for verified robust voice conversion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 17862–17870.

[8] Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. 2022. High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438* (2022).

[9] Patrick Esser, Robin Rombach, and Bjorn Ommer. 2021. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 12873–12883.

[10] Wenhao Guan, Qi Su, Haodong Zhou, Shiyu Miao, Xingjia Xie, Lin Li, and Qingyang Hong. 2023. Reflow-tts: A rectified flow model for high-fidelity text-to-speech. *arXiv preprint arXiv:2309.17056* (2023).

[11] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), 3451–3460.

[12] Keith Ito and Linda Johnson. 2017. The LJ Speech Dataset. https://keithito.com/LJ-Speech-Dataset/.

[13] Myeonghun Jeong, Hyeongju Kim, Sung Jun Cheon, Byoung Jin Choi, and Nam Soo Kim. 2021. Diff-tts: A denoising diffusion model for text-to-speech. *arXiv preprint arXiv:2104.01409* (2021).

[14] Shengpeng Ji, Ziyue Jiang, Hanting Wang, Jialong Zuo, and Zhou Zhao. 2024. MobileSpeech: A Fast and High-Fidelity Framework for Mobile Zero-Shot Text-to-Speech. arXiv:2402.09378 [eess.AS]

[15] Ziyue Jiang, Jinglin Liu, Yi Ren, Jinzheng He, Zhenhui Ye, Shengpeng Ji, Qian Yang, Chen Zhang, Pengfei Wei, Chunfeng Wang, Xiang Yin, Zejun MA, and Zhou Zhao. 2024. Boosting Prompting Mechanisms for Zero-Shot Speech Synthesis. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=mvMI3N4AvD

[16] Ziyue Jiang, Jinglin Liu, Yi Ren, Jinzheng He, Zhenhui Ye, Shengpeng Ji, Qian Yang, Chen Zhang, Pengfei Wei, Chunfeng Wang, Xiang Yin, Zejun Ma, and Zhou Zhao. 2024. Mega-TTS 2: Boosting Prompting Mechanisms for Zero-Shot Speech Synthesis. arXiv:2307.07218 [eess.AS]

[17] Ziyue Jiang, Yi Ren, Zhenhui Ye, Jinglin Liu, Chen Zhang, Qian Yang, Shengpeng Ji, Rongjie Huang, Chunfeng Wang, Xiang Yin, et al. 2023. Mega-tts: Zero-shot text-to-speech at scale with intrinsic inductive bias. *arXiv preprint arXiv:2306.03509* (2023).

[18] Ziyue Jiang, Qian Yang, Jialong Zuo, Zhenhui Ye, Rongjie Huang, Yi Ren, and Zhou Zhao. 2023. FluentSpeech: Stutter-Oriented Automatic Speech Editing with Context-Aware Diffusion Models. In *Findings of the Association for Computational Linguistics: ACL 2023*. 11655–11671.

[19] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. 2022. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems* 35 (2022), 26565–26577.

[20] Eugene Kharitonov, Damien Vincent, Zalán Borsos, Raphaël Marinier, Sertan Girgin, Olivier Pietquin, Matt Sharifi, Marco Tagliasacchi, and Neil Zeghidour. 2023. Speak, read and prompt: High-fidelity text-to-speech with minimal supervision. *arXiv preprint arXiv:2302.03540* (2023).

[21] Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. 2023. Consistency Trajectory Models: Learning Probability Flow ODE Trajectory of Diffusion. In *The Twelfth International Conference on Learning Representations*.

[22] Jaehyeon Kim, Keon Lee, Seungjun Chung, and Jaewoong Cho. 2024. CLaM-TTS: Improving Neural Codec Language Model for Zero-Shot Text-to-Speech. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=ofzeypWosV

[23] Sungwon Kim, Kevin J Shih, Rohan Badlani, Joao Felipe Santos, Evelina Bakhturina, Mikyas T Desta, Rafael Valle, Sungroh Yoon, and Bryan Catanzaro. 2023. P-Flow: A Fast and Data-Efficient Zero-Shot TTS through Speech Prompting. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[24] Yuma Koizumi, Heiga Zen, Shigeki Karita, Yifan Ding, Kohei Yatabe, Nobuyuki Morioka, Michiel Bacchiani, Yu Zhang, Wei Han, and Ankur Bapna. 2023. Libritts-r: A restored multi-speaker text-to-speech corpus. *arXiv preprint arXiv:2305.18802* (2023).

[25] Fei Kong, Jinhao Duan, Lichao Sun, Hao Cheng, Renjing Xu, Hengtao Shen, Xiaofeng Zhu, Xiaoshuang Shi, and Kaidi Xu. 2023. ACT: Adversarial Consistency Models. *arXiv preprint arXiv:2311.14097* (2023).

[26] Matthew Le, Apoorv Vyas, Bowen Shi, Brian Karrer, Leda Sari, Rashel Moritz, Mary Williamson, Vimal Manohar, Yossi Adi, Jay Mahadeokar, et al. 2024. Voicebox: Text-guided multilingual universal speech generation at scale. *Advances in neural information processing systems* 36 (2024).

[27] Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, and Ming Liu. 2019. Neural speech synthesis with transformer network. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 6706–6713.

[28] Xiang Li, Songxiang Liu, Max WY Lam, Zhiyong Wu, Chao Weng, and Helen Meng. 2023. Diverse and Expressive Speech Prosody Prediction with Denoising Diffusion Probabilistic Model. *arXiv preprint arXiv:2305.16749* (2023).

[29] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. 2022. Flow Matching for Generative Modeling. In *The Eleventh International Conference on Learning Representations*.

[30] Haoye Lu, Yiwei Lu, Dihong Jiang, Spencer Ryan Szabados, Sun Sun, and Yaoliang Yu. 2023. Cm-gan: Stabilizing gan training with consistency models. In *ICML 2023 Workshop on Structured Probabilistic Inference {\&} Generative Modeling*.

[31] Yiwen Lu, Zhen Ye, Wei Xue, Xu Tan, Qifeng Liu, and Yike Guo. 2024. Co-MoSVC: Consistency Model-based Singing Voice Conversion. *arXiv preprint arXiv:2401.01792* (2024).

[32] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. 2023. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378* (2023).

[33] Weijian Luo. 2023. A comprehensive survey on knowledge distillation of diffusion models. *arXiv preprint arXiv:2304.04262* (2023).

[34] Junlong Lyu, Zhitang Chen, and Shoubo Feng. 2024. Sampling is as easy as keeping the consistency: convergence guarantee for Consistency Models. https://openreview.net/forum?id=mWT3Ftkc3e

[35] ML Menéndez, JA Pardo, L Pardo, and MC Pardo. 1997. The jensen-shannon divergence. *Journal of the Franklin Institute* 334, 2 (1997), 307–318.

[36] Takeru Miyato and Masanori Koyama. 2018. cGANs with Projection Discriminator. In *International Conference on Learning Representations*.

[37] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499* (2016).

[38] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 5206–5210.

[39] Puyuan Peng, Po-Yao Huang, Daniel Li, Abdelrahman Mohamed, and David Harwath. 2024. VoiceCraft: Zero-Shot Speech Editing and Text-to-Speech in the Wild. *arXiv preprint arXiv:2403.16973* (2024).

[40] Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail Kudinov. 2021. Grad-TTS: A Diffusion Probabilistic Model for Text-to-Speech. In *International Conference on Machine Learning*. PMLR, 8599–8608.

[41] Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, Mikhail Sergeevich Kudinov, and Jiansheng Wei. 2021. Diffusion-Based Voice Conversion with Fast Maximum Likelihood Sampling Scheme. In *International Conference on Learning Representations*.

[42] Vineel Pratap, Qiantong Xu, Anuroop Sriram, Gabriel Synnaeve, and Ronan Collobert. 2020. Mls: A large-scale multilingual dataset for speech research. *arXiv preprint arXiv:2012.03411* (2020).

[43] Konpat Preechakul, Nattanat Chatthee, Suttisak Wizadwongsa, and Supasorn Suwajanakorn. 2022. Diffusion autoencoders: Toward a meaningful and decodable representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10619–10629.

[44] Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. 2020. FastSpeech 2: Fast and High-Quality End-to-End Text to Speech. In *International Conference on Learning Representations*.

[45] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. 2019. Fastspeech: Fast, robust and controllable text to speech. *Advances in neural information processing systems* 32 (2019).

[46] Yi Ren, Xu Tan, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2022. Revisiting Over-Smoothness in Text to Speech. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 8197–8213.

[47] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10684–10695.

[48] Nick Rossenbach, Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2020. Generating synthetic audio data for attention-based speech recognition systems. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 7069–7073.

[49] Takaaki Saeki, Detai Xin, Wataru Nakata, Tomoki Koriyama, Shinnosuke Takamichi, and Hiroshi Saruwatari. 2022. Utmos: Utokyo-sarulab system for voicemos challenge 2022. *arXiv preprint arXiv:2204.02152* (2022).

[50] Tim Salimans and Diederik P. Kingma. 2016. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. arXiv:1602.07868 [cs.LG]

[51] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. 2023. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042* (2023).

[52] Kai Shen, Zeqian Ju, Xu Tan, Eric Liu, Yichong Leng, Lei He, Tao Qin, sheng zhao, and Jiang Bian. 2024. NaturalSpeech 2: Latent Diffusion Models are Natural and Zero-Shot Speech and Singing Synthesizers. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=Rc7dAwVL3v

[53] Yang Song and Prafulla Dhariwal. 2023. Improved Techniques for Training Consistency Models. In *The Twelfth International Conference on Learning Representations*.

[54] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. 2023. Consistency models. *arXiv preprint arXiv:2303.01469* (2023).

[55] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2020. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*.

[56] Hao Sun, Xu Tan, Jun-Wei Gan, Hongzhi Liu, Sheng Zhao, Tao Qin, and Tie-Yan Liu. 2019. Token-level ensemble distillation for grapheme-to-phoneme conversion. *arXiv preprint arXiv:1904.03446* (2019).

[57] Xu Tan, Tao Qin, Frank Soong, and Tie-Yan Liu. 2021. A survey on neural speech synthesis. *arXiv preprint arXiv:2106.15561* (2021).

[58] Apoorv Vyas, Bowen Shi, Matthew Le, Andros Tjandra, Yi-Chiao Wu, Baishan Guo, Jiemin Zhang, Xinyue Zhang, Robert Adkins, William Ngan, et al. 2023. Audiobox: Unified audio generation with natural language prompts. *arXiv preprint arXiv:2312.15821* (2023).

[59] Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, et al. 2023. Neural codec language models are zero-shot text to speech synthesizers. *arXiv preprint arXiv:2301.02111* (2023).

[60] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. 2017. Tacotron: Towards End-to-End Speech Synthesis. *Interspeech 2017* (2017).

[61] Zhichao Wang, Yuanzhe Chen, Lei Xie, Qiao Tian, and Yuping Wang. 2023. Lm-vc: Zero-shot voice conversion via speech generation based on language models. *IEEE Signal Processing Letters* (2023).

[62] Junichi Yamagishi, Christophe Veaux, and Kirsten MacDonald. 2019. CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit (version 0.92). https://doi.org/10.7488/ds/2645

[63] Dongchao Yang, Jinchuan Tian, Xu Tan, Rongjie Huang, Songxiang Liu, Xuankai Chang, Jiatong Shi, Sheng Zhao, Jiang Bian, Xixin Wu, et al. 2023. Uniaudio: An audio foundation model toward universal audio generation. *arXiv preprint arXiv:2310.00704* (2023).

[64] Zhen Ye, Wei Xue, Xu Tan, Jie Chen, Qifeng Liu, and Yike Guo. 2023. CoMoSpeech: One-Step Speech and Singing Voice Synthesis via Consistency Model. *arXiv preprint arXiv:2305.06908* (2023).

[65] Binbin Zhang, Hang Lv, Pengcheng Guo, Qijie Shao, Chao Yang, Lei Xie, Xin Xu, Hui Bu, Xiaoyu Chen, Chenchen Zeng, et al. 2022. Wenetspeech: A 10000+ hours multi-domain mandarin corpus for speech recognition. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 6182–6186.

[66] Ziqiang Zhang, Long Zhou, Chengyi Wang, Sanyuan Chen, Yu Wu, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, et al. 2023. Speak foreign languages with your own voice: Cross-lingual neural codec language modeling. *arXiv preprint arXiv:2303.03926* (2023).

[67] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).