DiffProsody: Diffusion-Based Latent Prosody Generation for Expressive Speech Synthesis With Prosody Conditional Adversarial Training

Hyung-Seok Oh, *Student Member, IEEE*, Sang-Hoon Lee, *Member, IEEE*, and Seong-Whan Lee, *Fellow, IEEE*

***Abstract*—Expressive text-to-speech systems have undergone significant advancements owing to prosody modeling, but conventional methods can still be improved. Traditional approaches have relied on the autoregressive method to predict the quantized prosody vector; however, it suffers from the issues of long-term dependency and slow inference. This study proposes a novel approach called DiffProsody in which expressive speech is synthesized using a diffusion-based latent prosody generator and prosody conditional adversarial training. Our findings confirm the effectiveness of our prosody generator in generating a prosody vector. Furthermore, our prosody conditional discriminator significantly improves the quality of the generated speech by accurately emulating prosody. We use denoising diffusion generative adversarial networks to improve the prosody generation speed. Consequently, DiffProsody is capable of generating prosody 16 times faster than the conventional diffusion model. The superior performance of our proposed method has been demonstrated via experiments.**

***Index Terms*—Text-to-speech, speech synthesis, denoising diffusion model, prosody modeling, generative adversarial networks.**

I. Introduction

RECENT advancements in neural text-to-speech (TTS) models have significantly enhanced the naturalness of synthetic speech. In several studies [1], [2], [3], [4], [5], [6], [7], prosody modeling has been leveraged to synthesize speech that closely resembles human expression. Prosody, which encompasses various speech properties, such as pitch, energy, and duration, plays a crucial role in the synthesis of expressive speech.

Manuscript received 29 July 2023; revised 7 January 2024 and 1 April 2024; accepted 27 April 2024. Date of publication 1 May 2024; date of current version 16 May 2024. This work was supported in part by the Institute of Information and communications Technology Planning & Evaluation (IITP) funded by the Korea government (MSIT) under Grant 2019-0-00079, in part by Artificial Intelligence Graduate School Program (Korea University) under Grant 2021-0-02068, in part by Artificial Intelligence Innovation Hub, and in part by Artificial Intelligence Convergence Innovation Human Resources Development and Voice&Avatar, Naver Cloud, Seongnam, Korea under Grant IITP-2024-RS-2023-00255968. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Zhizheng Wu. *(Corresponding author: Seong-Whan Lee.)*

Hyung-Seok Oh and Seong-Whan Lee are with the Department of Artificial Intelligence, Korea University, Seoul 02841, South Korea (e-mail: hs_oh@korea.ac.kr; sw.lee@korea.ac.kr).

Sang-Hoon Lee is with the Department of Software and Computer Engineering, Ajou University, Suwon-si, South Korea, and also with the Department of Artificial Intelligence, Ajou University, Suwon-si, South Korea (e-mail: sanghoonlee@ajou.ac.kr).

Digital Object Identifier 10.1109/TASLP.2024.3395994

In some studies [8], [9], reference encoders have been used to extract prosody vectors for prosody modeling. A global style token (GST) [10] is an unsupervised style modeling method that uses learnable tokens to model and control various styles. Meta-StyleSpeech [11] proposes the application of style vectors extracted using a reference encoder through a style-adaptive layer norm. Progressive variational autoencoder TTS [12] presents a method for gradual style adaptation. A zero-shot method for speech synthesis that comprises the use of a normalization architecture, speaker encoder, and feedforward transformer-based architecture [13] was proposed. This is a common way to transfer styles if there exists reference audio of the desired style.

Recently, methods for inferring prosody from text in the absence of reference audio have been developed [14], [15]. FastPitch [16], for instance, synthesizes speech under text and fundamental frequency conditions. FastSpeech 2 [3] aims to generate natural, human-like speech by using extracted prosody features, such as pitch, energy, and duration, through an external tool and introduce a variance adaptor module that predicts these features. To focus on generating high-quality prosody, [17] proposed a diffusion-based prosody predictor. Some studies [18], [19] have proposed hierarchical models through the design of prosody features at both coarse and fine-grained levels. However, the separate modeling of prosodic features may yield unnatural results owing to their inherent correlation [20].

Some studies have predicted a unified prosody vector, thus enhancing the representation of prosody, given the interdependence of prosody features. Text-predicted GST [21] is a method for modeling prosody without reference audio by predicting the weight of the style token from the input text. [22] proposed a method for paragraph-based prosody modeling by introducing a paragraph encoder. Gaussian-mixture-model-based phone-level prosody modelling [23] is a method for sampling reference prosody from Gaussian components. [24] proposed a method for modeling prosody using style, perception, and frame-level reconstruction loss. There are also studies in which prosody is modeled using pre-trained language models [25], [26], [27], [28]. ProsoSpeech [20] models prosody with vector quantization (VQ) using large amounts of data and predicts the index of the codebook using an autoregressive (AR) prosody predictor.

Using an AR model is a common way to model prosody [20], [29]. However, these models face challenges such as repetition, skipping, or error accumulation. To overcome these difficulties, we explore diffusion models for prosody modeling. Diffusion models have gained remarkable recognition for their ability to handle complex distributions and produce high-quality, diverse results. They have produced impressive results in areas such as text-to-image generation [30], speech synthesis [31], and text generation [32], demonstrating their versatility and effectiveness. A notable advancement in the field is the method of diffusion in latent space proposed in [33], which opens up new possibilities for generative models.

Building on this momentum, we present DiffProsody, a novel approach that exploits the generative capabilities of a diffusion model to model latent prosody. DiffProsody is designed to produce more expressive and natural-sounding speech by using a diffusion-based latent prosody generator (DLPG). This method not only exploits the sophisticated mechanics of diffusion models, but also integrates prosody conditional adversarial training. This combination aims to improve the quality of speech synthesis by addressing the need for more dynamic and expressive speech.

The primary contributions of this work are as follows:

- We propose a diffusion-based latent prosody modeling method that can generate high-quality latent prosody representations, thereby enhancing the expressiveness of synthetic speech. Furthermore, we adopted denoising diffusion generative adversarial networks (DDGANs) to reduce the number of timesteps, resulting in speeds that were 2.48$\times$ and 16$\times$ faster than those of the AR model and denoising diffusion probabilistic model (DDPM) [34], respectively.
- We propose prosody conditional adversarial training to ensure an accurate reflection of prosody using the TTS module. A significant improvement in smoothness, attributable to vector quantization, was observed in the generated speech.
- Objective and subjective evaluations demonstrated that the proposed method outperforms comparative models.

The implementation[1] of proposed method and audio samples[2] for various datasets, such as VCTK[3] and LibriTTS [35], are available online.

II. Related Works

A. Non-Autoregressive Text-to-Speech

Traditional TTS models function autoregressively. This implies that the spectrogram generates one frame at a time, with each frame conditioned on the preceding frames. Despite the high-quality speech that this approach can produce, it has a drawback in terms of speed owing to the sequential nature of the generation process. To address this issue, non-autoregressive TTS (NAR-TTS) models have been proposed as an alternative for parallel generation. These models have the advantage of simultaneously generating the entire spectrogram, thus resulting in a significant acceleration of the speech synthesis. FastSpeech [2] and FastSpeech 2 [3] serve as examples of NAR-TTS models that can synthesize speech at a much faster rate than their AR counterparts while maintaining a comparable level of quality. For parallel generation, these models require phoneme-level durations. FastSpeech uses an AR teacher model to obtain durations through knowledge distillation. The phoneme-level input is scaled to the frame-level using a length regulator, and a transformer-based network [36] is used to generate the entire utterance at once. FastSpeech 2 addresses some of the disadvantages of FastSpeech by extracting the phoneme duration from forced alignment as the training target instead of relying on the attention map of the AR teacher model and introducing more variation information in speech as conditional inputs. In contrast to these models that use an external aligner, [31], [37], [38], [39] are parallel TTS models that use an internal aligner to model duration. These parallel-generation models exhibit faster and more robust generation than the AR models. In this study, we adopted a transformer-based NAR model with a simple structure to focus on prosody modeling.

B. Generative Adversarial Networks

Generative adversarial networks (GANs) [40] are generative models in which generative and discriminative networks compete against each other. The objective of the generative network is to create samples that closely resemble the true data distribution, whereas the discriminative network strives to differentiate between the data sampled from the true and generated distributions.

These two networks play a minimax game with the following value function $V(D,G)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1-D(G(\mathbf{z})))], \tag{1}$$

where G represents the generative network, and D represents the discriminative network. The training process involves minimizing $D(1-G(z))$ to recognize D as real for G, and maximizing $log(D(x))$ for D to learn the likelihood of real data. The conditional generation model was obtained by introducing condition c into both G and D. In this study, we incorporated adversarial training conditions into prosody features to generate expressive speech.

C. Denoising Diffusion Models

The denoising diffusion model is a generative model that gradually collapses data into noise and generates data from noise. The processes of collapsing and denoising data are called the forward and reverse processes, respectively. The forward process gradually collapses data $\mathbf{x}_0$ into noise over the T-step, with a predefined variance schedule β_t.

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \tag{2}$$

where $q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t I)$. The reverse process is defined as follows:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t). \tag{3}$$

[1] [Online]. Available: https://github.com/hsoh0306/DiffProsody

[2] [Online]. Available: https://prml-lab-speech-team.github.io/demo/DiffProsody/

[3] [Online]. Available: https://datashare.ed.ac.uk/handle/10283/2651

The reverse process is driven by a denoising model parameterized by θ. The denoising model was optimized for a variational bound on the negative log-likelihood.

$$\mathbb{E}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|x_0)}\right]$$
$$= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] := L. \quad (4)$$

In the DDPM [34], the denoising distribution $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is assumed to comprise a Gaussian distribution. Moreover, it has been demonstrated that the diffusion model can generate a diverse range of complex distributions, provided that a sufficient number of iterations are performed. Proceeding with only a small step at a time is possible by setting the denoising distribution to a Gaussian distribution, which implies that a considerable number of timesteps are required. DDGAN [41] was proposed by modeling the denoising distribution with a non-Gaussian multimodal distribution to reduce the sampling step. It predicts $\mathbf{x}_0$ with the generator G_θ, which models the implicit distribution, in contrast to the denoising network of the DDPM, which predicts noise. In the DDGAN, the conditional probability $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is defined as follows:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \int p_\theta(\mathbf{x}_0|\mathbf{x}_t) q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) d\mathbf{x}_0$$
$$= \int p(\mathbf{z}) q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0 = G_\theta(\mathbf{x}_t, \mathbf{z}, t)) d\mathbf{z}, \quad (5)$$

where $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ represents the implicit distribution imposed by the generator $G_\theta(\mathbf{x}, \mathbf{z}, t)$, which outputs $\mathbf{x}_0$ given $\mathbf{x}_t$ and the latent variable $\mathbf{z} \sim p(\mathbf{z}) := \mathcal{N}(\mathbf{z}; 0, \mathbf{I})$. In the DDGAN, the denoising distribution $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is modeled as a complex multimodal distribution, in contrast to the unimodal distribution in the DDPM. Sampling $\mathbf{x}_0$ with small timesteps is possible by leveraging the complicated $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$. We introduced this model to reduce the sampling timesteps while maintaining its ability to generate diffusion models.

D. *Prosody Modeling*

Although the pronunciation capabilities of TTS models have seen significant advancements, they still fail to replicate the naturalness inherent in human speech. Various studies have proposed methods for prosody modeling to address this limitation. One such method is the reference-based approach [8], [9], [10], [11], [42], which is a type of expressive speech synthesis that extracts styles from the reference audio. This approach is particularly beneficial for style transfers, but can result in unnatural results when the text and reference audio do not align well. However, there are methods that directly model prosody properties such as pitch, energy, and duration [3], [16], [43], [44]. These methods offer the advantages of explainability and controllability because they directly use and model prosody features. In the context of ProsoSpeech [20], the authors argue that prosodic features are interdependent and that modeling them separately can result in unnatural outcomes. To address this, they proposed the modeling of prosody as a latent prosody vector (LPV) and introduced an AR prosody predictor to obtain the LPV. In this study, we adopt a similar latent vector approach to model prosody. The use of pre-trained language models was also suggested [25], [26], [27], [28]. These models comprise the use of models that have been pre-trained on large datasets, such as BERT [45] or GPT-3 [46].

III. DiffProsody

The proposed method, called DiffProsody, aims to enhance speech synthesis by incorporating a diffusion-based latent prosody generator (DLPG) and prosody conditional adversarial training. The overall structure and process of DiffProsody are presented in Fig. 1. In the first stage, we trained a TTS module and a prosody encoder using a text sequence and a reference Mel-spectrogram as inputs. The prosody conditional discriminator evaluates the prosody vector from the prosody encoder and the Mel-spectrogram from the TTS module to provide feedback on their quality. In the second stage, we train a DLPG to sample a prosody vector that corresponds to the input text and speaker. During inference, the TTS module synthesizes speech without relying on a reference Mel-spectrogram. Instead, it uses the output of a DLPG. This facilitates the generation of expressive speech that accurately reflects the desired prosody.

A. *Text-to-Speech Module*

The TTS module is designed to transform text into Mel-spectrograms using speaker and prosody vectors as conditions. The overall structure of the model is presented in Fig. 1(a). The TTS module comprises a text encoder and a decoder. The text encoder processes the text at both the phoneme and word levels, as illustrated in Fig. 1(c). The input text, denoted as $\mathbf{x}_{txt}$, is converted into a text hidden representation $\mathbf{h}_{txt}$, by the phoneme encoder E_p and word encoder E_w. The E_p takes the phoneme-level text $\mathbf{x}_{ph}$ and the E_w takes as input the word-level text $\mathbf{x}_{wd}$. The $\mathbf{h}_{txt}$ is then obtained as the element-wise sum of the outputs of $E_p(\mathbf{x}_{ph})$ and $E_w(\mathbf{x}_{wd})$ expanded to the phoneme-level.

$$\mathbf{h}_{txt} = E_p(\mathbf{x}_{ph}) + expand(E_w(\mathbf{x}_{wd})), \quad (6)$$

where $expand$ is an operation that repeats the word-level features to the phoneme-level. Obtaining the quantized prosody vector $\mathbf{z}_{pros}$ involves using $\mathbf{h}_{txt}$ and speaker hidden representation $\mathbf{h}_{spk}$ as inputs for the prosody module. In addition, $\mathbf{h}_{spk}$ is acquired using a pre-trained speaker encoder. We use Resemblyzer,[4] an open-source model trained with generalized end-to-end loss (GE2E) [47], to extract $\mathbf{h}_{spk}$.

During the first stage of training, a prosody encoder is employed, which receives the target Mel-spectrogram. In the inference, $\mathbf{z}'_{pros}$ is obtained by inputting $\mathbf{h}_{txt}$ and $\mathbf{h}_{spk}$ into a DLPG, and this is performed without a reference Mel-spectrogram. Finally, the information related to the text, speaker, and prosody is combined by expanding the latent vectors $\mathbf{h}_{txt}$, $\mathbf{h}_{spk}$, and $\mathbf{z}_{pros}$ to the phoneme-level and then performing an element-wise summation.

$$\mathbf{h}_{total} = \mathbf{h}_{txt} + \mathbf{h}_{spk} + \mathbf{z}_{pros}. \quad (7)$$

[4][Online]. Available: https://github.com/resemble-ai/Resemblyzer

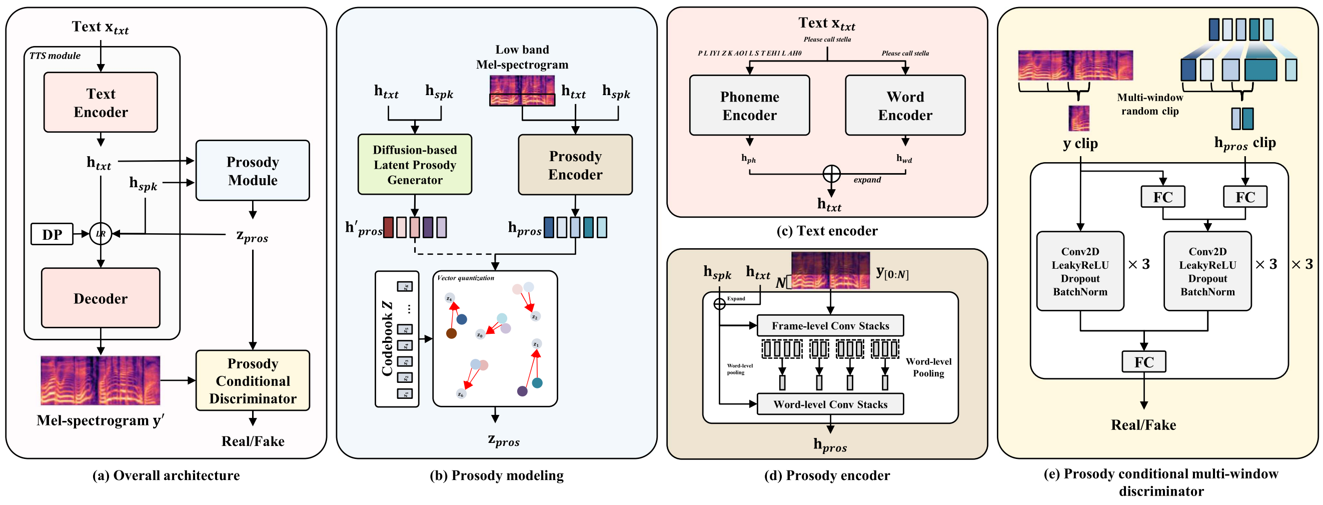


Fig. 1. Framework of DiffProsody. (a) Overall architecture including TTS and prosody modeling with prosody conditional adversarial training; (b) Prosody modeling by vector quantization with prosody encoder and diffusion-based latent prosody generator; (c) Text encoder that models text at the phoneme-level and word-level; (d) Prosody encoder that models the word-level target prosody; (e) Prosody conditional discriminator for adversarial training. DP represents a duration predictor, and LR represents a length regulator. In the first stage, the TTS and prosody encoder are trained jointly, and in the second stage, a diffusion-based latent prosody generator (DLPG) is trained using the output of the pre-trained prosody encoder as a target. In inference, the TTS module synthesizes speech conditioned on the prosody vector generated by DLPG.

The phoneme duration is modeled using the duration predictor DP. The goal of the DP is to predict the phoneme duration at the frame-level based on the input variable $\mathbf{h}_{total}$.

$$dur' = DP(\mathbf{h}_{total}), \tag{8}$$

where dur' represents the predicted phoneme duration. In addition, there is a length regulator LR that expands the input variable to the frame-level using the phoneme duration. The ground-truth duration dur is used for training, and the predicted duration dur' is used for inference. The expanded $\mathbf{h}_{total}$ is then transformed to Mel-spectrogram $\mathbf{y}'$ by Mel-spectrogram decoder D_{mel}.

$$\mathbf{y}' = D_{mel}(LR(\mathbf{h}_{total}, dur)). \tag{9}$$

For TTS modeling, we use two types of losses: the mean square error (MSE) and structural similarity index (SSIM) loss [48]. These losses aid in accurately modeling the TTS. For the duration modeling, we use the MSE loss.

$$\mathcal{L}_{rec} = \mathcal{L}_{MSE}(\mathbf{y}, \mathbf{y}') + \mathcal{L}_{SSIM}(\mathbf{y}, \mathbf{y}'). \tag{10}$$

$$\mathcal{L}_{dur} = \mathcal{L}_{MSE}(dur, dur'). \tag{11}$$

B. Prosody Module

Fig. 1(b) presents the prosody module, which includes a prosody encoder E_{pros} that derives a prosody vector from a reference Mel-spectrogram, a DLPG that produces a prosody vector using text and speaker hidden states, and a codebook $Z = \{\mathbf{z}_k\}_{k=1}^{K} \in \mathbb{R}^{K\times d_z}$, where K represents the size of the codebook and d_z is the dimension of the codes. During the training of E_{pros}, instead of a full-band Mel-spectrogram, we used a low-frequency band Mel-spectrogram to alleviate disentanglement, as in the case of ProsoSpeech [20]. Fig. 1(d) presents the structure of E_{pros}, which comprises two 1D convolutional stacks and a word-level pooling layer. h_{txt} is scaled to frame-level according to the ground-truth phoneme-duration, and h_{spk} is also scaled to frame-level. The first convolution stack processes the data at the frame-level. It then performs word-level pooling to obtain semantic features. The features are fed into the second convolution stack to get h_{pros}. The frame-level expanded h_{txt} and h_{spk} are used as conditions in the convolution stack, and the second convolution stack is conditioned after word-level pooling. To extract the target prosody, E_{pros} uses the lowest N bins of the target Mel-spectrogram $\mathbf{y}_{[0:N]}$, along with the $\mathbf{h}_{txt}$ and $\mathbf{h}_{spk}$, as its inputs. The output of this process is a prosody vector, $\mathbf{h}_{pros} \in \mathbb{R}^{L\times d_z}$, where L is the word-level length of the input text.

$$\mathbf{h}_{pros} = E_{pros}(\mathbf{y}_{[0:N]}, \mathbf{h}_{txt}, \mathbf{h}_{spk}). \tag{12}$$

During the inference stage, the prosody vector $\mathbf{h}'_{pros}$ is obtained using the prosody generator trained in the second stage.

$$\mathbf{h}'_{pros} = DLPG(\mathbf{h}_{txt}, \mathbf{h}_{spk}). \tag{13}$$

The $DLPG$ process is described in Section III-D. To obtain the discrete prosody token sequence $\mathbf{z}_{pros} \in \mathbb{R}^{L\times d_z}$, the vector quantization layer Z maps each prosody vector $\mathbf{h}^i_{pros} \in \mathbb{R}^{d_z}$ to the nearest element of the codebook entry $\mathbf{z}_k \in \mathbb{R}^{d_z}$.

$$\mathbf{z}^i_{pros} = \underset{\mathbf{z}_k\in Z}{\arg\min}\, ||\mathbf{h}^i_{pros} - \mathbf{z}_k||_2 \text{ for } i = 1 \text{ to } L, \tag{14}$$

where $\mathbf{z}^i_{pros}$ is i-th element of $\mathbf{z}_{pros}$. In the first stage, the TTS module is trained jointly with the codebook Z and prosody encoder E_{pros}.

$$\mathcal{L}_{vq} = ||sg[\mathbf{h}_{pros}] - \mathbf{z}_{pros}||_2^2 + \beta||\mathbf{h}_{pros} - sg[\mathbf{z}_{pros}]||_2^2, \tag{15}$$

where $sg[\cdot]$ denotes the stop-gradient operation. Moreover, we employ an exponential moving average (EMA) [49] to enhance the learning efficiency by applying it to codebook updates.

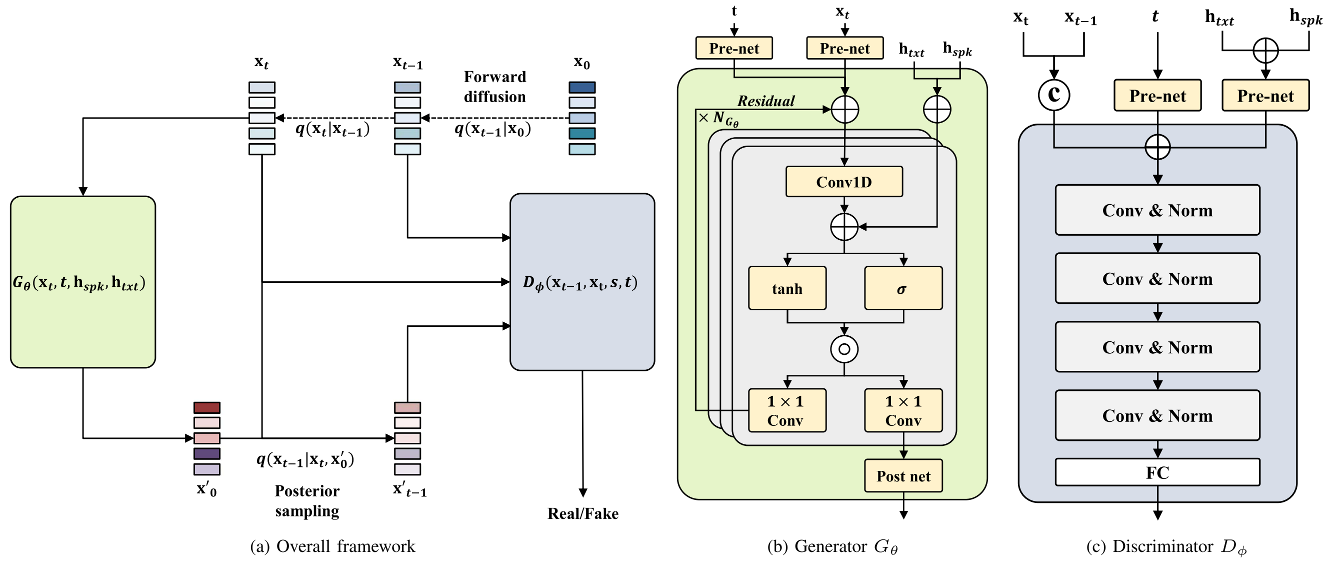


Fig. 2. (a) Training a diffusion-based latent prosody generator. We adopt the design of DDGANs [41] to shorten the diffusion timestep. The generator G_θ takes speaker hidden representation $\mathbf{h}_{spk}$ and text hidden representation $\mathbf{h}_{txt}$, timestep t, and noisy data $\mathbf{x}_t$ as input to generate $\mathbf{x}'_0$, and the disriminator D_ϕ determines which of $\mathbf{x}'_{t-1}$ obtained by posterior sampling on $\mathbf{x}'_0$ and $\mathbf{x}_{t-1}$ obtained by forward process on $\mathbf{x}_0$ is compatible with $\mathbf{x}_t$ at t timestep; (b) The architecture of diffusion-based prosody generator G_θ; (c) The architecture of discriminator D_ϕ.

C. Prosody Conditional Adversarial Training

For our prosody adversarial training, we develop prosody conditional discriminators (PCDs) that handle inputs of varying lengths. This design is inspired by multi-length window discriminators [50]. The PCD structure is presented in Fig. 1 e. The PCD is designed to accept a Mel-spectrogram y and a quantized prosody vector $\mathbf{z}_{pros}$ as inputs, and its role is to determine whether these input features are original or generated. The PCD comprises two lightweight convolutional neural networks (CNNs) and fully connected layers. One of the CNNs is designed to receive only the Mel-spectrogram, whereas the other is designed to receive a combination of $\mathbf{z}_{pros}$ and y. To match the corresponding PCD, the length of each Mel-spectrogram and the extended $\mathbf{z}_{pros}$ are randomly clipped. For our objective function, we adopt the least square GAN loss [51]:

$$\mathcal{L}_D = \sum_i [\mathbb{E}[(PCD^i(\mathbf{y}', \mathbf{z}_{pros}))^2] + \mathbb{E}[(PCD^i(\mathbf{y}, \mathbf{z}_{pros}) - 1)^2]], \tag{16}$$

$$\mathcal{L}_G = \sum_i \mathbb{E}[(PCD^i(\mathbf{y}', \mathbf{z}_{pros}) - 1)^2], \tag{17}$$

where $\mathcal{L}_D$ denotes the training goal of the discriminators and $\mathcal{L}_G$ represents the feedback on the TTS module. The final object $\mathcal{L}_{TTS}$ of the TTS module is as follows:

$$\mathcal{L}_{TTS} = \mathcal{L}_{rec} + \mathcal{L}_{dur} + \mathcal{L}_{vq} + \lambda_1 \mathcal{L}_G, \tag{18}$$

where λ_1 corresponds to the weight of the adversarial loss.

D. Diffusion-Based Latent Prosody Generator

We propose a new module called the DLPG, which leverages the powerful generative capabilities of diffusion models. In addition, we introduce a DDGAN framework [41], which enables faster sampling by reducing the number of required timesteps. Fig. 2(a) presents the training process for the DLPG. During training, the DLPG aims to generate target $\mathbf{h}_{pros}$, which is extracted from the prosody encoder trained in the first stage. The DLPG is trained to produce $\mathbf{h}'_{pros}$ based on the $\mathbf{h}_{spk}$ and $\mathbf{h}_{txt}$. In the diffusion model, we set $\mathbf{x}_0$ as the target $\mathbf{h}_{pros}$. The DLPG generator G_θ directly generates $\mathbf{x}'_0$.

$$\mathbf{x}'_0 = G_\theta(\mathbf{x}_t, t, \mathbf{h}_{spk}, \mathbf{h}_{txt}), \tag{19}$$

where t is timestep of diffusion process. To ensure adversarial training, $\mathbf{x}'_{t-1}$ is derived from $\mathbf{x}_t$ and $\mathbf{x}'_0$ using posterior sampling $q(\mathbf{x}'_{t-1}|\mathbf{x}_t, \mathbf{x}'_0)$. Subsequently, a time-dependent discriminator D_ϕ determines the compatibility of $\mathbf{x}_{t-1}$ (obtained from the forward processing of $\mathbf{x}_0$) and $\mathbf{x}'_{t-1}$ (generated through posterior sampling of $\mathbf{x}'_0$) with respect to t and $\mathbf{x}_t$, conditioned on $\mathbf{h}_{spk}$ and $\mathbf{h}_{txt}$. The objective function of G_θ is then defined as follows:

$$\mathcal{L}^{adv}_{G_\theta} = \sum_{t\geq 1} \mathbb{E}[(D_\phi(\mathbf{x}'_{t-1}, \mathbf{x}_t, t, \mathbf{h}_{txt}, \mathbf{h}_{spk}) - 1)^2], \tag{20}$$

$$\mathcal{L}^{rec}_{G_\theta} = L_{MAE}(\mathbf{x}_0, \hat{\mathbf{x}}_0), \tag{21}$$

where $\mathcal{L}^{adv}_{G_\theta}$ corresponds to the adversarial loss, and $\mathcal{L}^{rec}_{G_\theta}$ denotes the reconstruction loss of G_θ. The total generator loss $\mathcal{L}_{G_\theta}$ is expressed as follows:

$$\mathcal{L}_{G_\theta} = \mathcal{L}^{rec}_{G_\theta} + \lambda_2 \mathcal{L}^{adv}_{G_\theta}, \tag{22}$$

where λ_2 is the weight of adversarial loss $\mathcal{L}_{G_\theta adv}$. The objective function of the D_ϕ is as follows:

$$\mathcal{L}_{D_\phi} = \sum_{t\geq 1} [\mathbb{E}[D_\phi(\mathbf{x}'_{t-1}, \mathbf{x}_t, t, \mathbf{h}_{txt}, \mathbf{h}_{spk})^2] + \mathbb{E}[(D_\phi(\mathbf{x}_{t-1}, \mathbf{x}_t, t, \mathbf{h}_{txt}, \mathbf{h}_{spk}) - 1)^2]]. \tag{23}$$

The DLPG leverages the DDGAN framework to achieve stable and high-quality results in only a few timesteps. This process involves the G_θ, which iteratively generates $\mathbf{x}'_0$ T times during the inference. We set $\mathbf{x}_T$ to follow a normal distribution. The $\mathbf{h}'_{pros}$ is obtained as the final $\mathbf{x}'_0$ of the reverse process. The final prosody vector $\mathbf{z}_{pros}$ is then derived through the vector quantization of $\mathbf{h}'_{pros}$. Fig. 2(b) shows the architecture of G_θ, which consists of N_{G_θ} non-causal Wavenet blocks, as described in [52]. G_θ receives t, $\mathbf{x}_t$, $\mathbf{h}_{txt}$, and $\mathbf{h}_{spk}$ as input and predicts $\mathbf{x}_0$. Fig. 2(c) represents $D\phi$. $D\phi$ takes $\mathbf{x}_t$, $\mathbf{x}_{t-1}$, t, $\mathbf{h}_{txt}$, and $\mathbf{h}_{spk}$ as input and outputs a value between 0 and 1, where a value closer to 1 means that $\mathbf{x}_{t-1}$ is more likely to have been sampled from the original $\mathbf{x}_0$.

E. *Inference*

Here is the step-by-step process of generating a Mel-spectrogram using the trained TTS module and DLPG.

1) The $\mathbf{h}_{txt}$ is extracted from the text encoder, and $\mathbf{h}_{spk}$ is extracted from the pre-trained speaker encoder.
2) The DLPG generates a $\mathbf{h}'_{pros}$ with $\mathbf{h}_{txt}$ and $\mathbf{h}_{spk}$ as inputs.
3) $\mathbf{h}'_{pros}$ is mapped to a codebook, denoted as Z, in the VQ layer to obtain the prosody vector, which is denoted as $\mathbf{z}_{pros}$.
4) The decoder D_{mel} generates a Mel-spectrogram $\mathbf{y}'$ using $\mathbf{h}_{txt}$, $\mathbf{h}_{spk}$ and $\mathbf{z}_{pros}$. This process involves the expansion $\mathbf{h}_{txt}$, $\mathbf{h}_{spk}$, and $\mathbf{z}_{pros}$ to the frame-level. The phoneme-duration is predicted by the duration predictor.
5) The Mel-spectrogram y' was converted to a raw waveform using a pre-trained vocoder.

IV. Experimental Result and Discussion

A. *Experimental Setup*

We conducted experiments using the VCTK dataset and LibriTTS [35]. VCTK dataset is a multispeaker English dataset consisting of audio clips of 44,200 sentences recorded by 109 speakers for approximately 400 sentences. A total of 2,180 audio clips were randomly selected with 20 sentences per speaker, which constituted the test set. Furthermore, 545 audio clips randomly selected from five sentences per speaker were used as the verification sets, and the remainder were used as the training sets. For the LibriTTS dataset, which includes 110 hours of audio from 1,151 speakers, we utilized the *train-clean-100* and *train-clean-360* subsets. We constructed our test and validation sets using 10 sentences each from 20 randomly selected speakers, while the remainder of the dataset was allocated for training. We sampled audio at 22,050 Hz and then transformed it to an 80-bin Mel-spectrogram using an STFT with a window length of 1,024 and hop size of 256. The text was converted into phoneme sequences for text input using a grapheme-to-phoneme tool.[5] We extracted the phoneme duration using the Montreal Forced Aligner [53] tool. Furthermore, we used the AdamW optimizer [54] with $\beta_1 = 0.9$ and $\beta_2 = 0.98$. The learning rates for the TTS and latent prosody generator training were set as 5×10^{-4} and 2×10^{-4}, respectively. Throughout the training process, the batch size used was 48. The TTS and prosody encoder were updated in 160 k steps, and the latent prosody generator was updated in 320 k steps. In the experiment, all the audio was synthesized using the official implementation of the HiFi-GAN[6] [55] and a pre-trained model. We trained the TTS module and prosody encoder for approximately 16 h and the DLPG for 7 h using a single NVIDIA RTX A6000 GPU.

B. *Implementation Details*

The number of layers, hidden size, filter size, and kernel size of the feed-forward transformer blocks of the phoneme encoder, word encoder, and decoder were set as 4, 192, 384, and 5, respectively. The extracted speaker embedding was projected onto 192 dimensions, and the number of dimensions of the prosody vector was 192. The structure of the prosody encoder follows that of ProsoSpeech [20]. For VQ, we set the size of codebook and dimension of code as 128 and 192, respectively. and updated it using EMA with a decay rate of 0.998. The codebook was initialized as the center of the k-means clustering after 20 k steps of TTS training. The number of Mel-spectrogram bins used in the prosody encoder N was set as 20. We set up the PCD to receive different input sizes. Inspired by methods [55], [56], [57] that take in multiple inputs. The PCD comprises two 2D convolution stacks and three fully connected layers. The convolution stacks in the PCD consist of three 2D convolutions: LeakyReLU, BatchNorm, and a linear layer with a multilength window size of [32, 64, 128]. The latent prosody generator consists of 20 residual blocks with a hidden size of 384. The prosody discriminator is configured with four convolution layers and hidden dimensions of 384. We investigated λ_1 and λ_2 between 0.001 and 1.0 and set λ_1 and λ_2 as 0.01 and 0.05, respectively. The number of timesteps used in the training and inference was set as four.

C. *Comparative Studies*

We developed various prosody models for comparative experiments, including our proposed model, models from previous works, and an ablation study model. All the models produced an 80-bin Mel-spectrogram and synthesized speech using the same vocoder.

1) Gt: Human-recorded audio.

2) GT (vocoded): Audio from ground truth Mel-spectrogram converted to HiFi-GAN v1.

3) FastSpeech 2: This model synthesizes speech using explicit prosody features (pitch and energy).

4) Prosospeech: This method models a vector quantized latent prosody vector and predicts the prosody vector using an autoregressive predictor.

5) Diffprosody: The proposed method improves the TTS module and prosody vector through a prosody conditional discriminator, and then generates the prosody vector through DLPG.

6) DiffProsody (Pitch & Energy): It is a modified version of the proposed method to use explicit prosody attributes, where DLPG directly predicts pitch and energy.

[5][Online]. Available: https://github.com/Kyubyong/g2p

[6][Online]. Available: https://github.com/jik876/hifi-gan

7) DiffProsody (AR): It utilizes the same TTS module as the proposed model, but instead of DLPG, it uses an autoregressive predictor to predict the index of the codebook.

8) DiffProsody (DDPM): It utilizes the same TTS module as the proposed model, but uses the DDPM framework instead of the DDGAN framework.

9) DiffProsody (W/o PCD): This is a DiffProsody trained on the TTS module without the assistance of prosody conditional discriminator.

10) DiffProsody (W/o VQ): This is a DiffProsody trained without a VQ layer on the prosody encoder.

11) DiffProsody N: This is a DiffProsody that uses the lowest N Mel-bins as input to the prosody encoder.

We utilized an open-source implementation[7] to train FastSpeech 2. For ProsoSpeech, we followed closely the hyperparameters described in the original paper and chose the model variant that showed the best performance. All models used in the comparison share the same text encoder and decoder architecture. The primary distinction between ProsoSpeech and DiffProsody (w/o PCD) lies in their respective methods of generating the prosody vector. The main difference between ProsoSpeech and DiffProsody (w/o PCD) is in the way they generate prosody vectors. The former predicts the index of the codebook with an autoregressive predictor, and the latter predicts the prosody vector before the codebook and maps it to the codebook.

D. Subjective Metrics

A subjective assessment is conducted to confirm the effectiveness of the proposed method. To measure the level of naturalness, we used the mean opinion score (MOS). For this evaluation, we employed Amazon Mechanical Turk (MTurk), a crowdsourcing service, to gather feedback from 20 native Americans. MOS was assessed using a 5-point scale, and confidence intervals were calculated at a 95% level. For the evaluation, 100 samples were randomly selected from the test set.

E. Objective Metrics

For realizing an objective evaluation, we calculated the equal error rate (EER) using a pre-trained speaker verification model[8] [56]. We used the pre-trained wav2vec 2.0 [57] to the compute character error rate (CER) and word error rate (WER).

For the prosodic evaluation, we computed the average differences in utterance duration (DDUR) [58], [59], pitch error (RMSE$_{f_0}$) (in cents), periodicity error (RMSE$_{period}$) and F1 score of the voiced/unvoiced classification (F1$_{v/uv}$). We used torchcrepe[9] to extract the pitch and periodicity features for evaluation. In addition, we measured the Kullback–Leibler (KL) divergence of log f0 and log energy to compare the distributions of the prosody features in the generated audio. Finally, we calculated the real-time factor (RTF) to compare the generation speeds.

[7] [Online]. Available: https://github.com/NATSpeech/NATSpeech
[8] [Online]. Available: https://github.com/clovaai/voxceleb_trainer
[9] [Online]. Available: https://github.com/maxrmorrison/torchcrepe

1) Average Differences in the Utterance Duration (DDUR): We obtained $DDUR$ by calculating the mean absolute error of the difference in the duration of each utterance.

$$DDUR = \frac{1}{N}\sum_{i=1}^{N}|dur_i - dur'_i|, \tag{24}$$

where dur_i denotes the duration of the i-th GT utterance and dur'_i denotes the duration of the i-th generated utterance.

2) Pitch Error: To measure the pitch error RMSE$_{f_0}$, we aligned the pitches extracted in hertz with dynamic time warping (DTW) [60] and calculated the root mean square (RMSE) in cents, defined as $1200\log_2(y/\hat{y})$ for pitch of the GT speech y and pitch of the generated speech $\hat{y}$. We measured the portion wherein both the GT speech and generated speech were voiced.

$$RMSE_{f_0} = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(1200\log_2(y_i/y'_i))^2}. \tag{25}$$

3) Periodicity Error: We measured the periodicity error RMSE$_{period}$ by root mean square between the periodicities ψ aligned with the DTW.

$$RMSE_{period} = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(\psi_i - \psi'_i)^2}, \tag{26}$$

where ψ_i means the i-th periodicity value.

4) F1 Score of voiced/unvoiced: We obtained voiced/unvoiced flags from the aligned pitches using DTW and calculated the F1 score (F1$_{v/uv}$) between them. We defined a match between the GT voiced flag u and generated voiced flag u' as a true positive (TP), a match between the GT unvoiced flag uv and generated voiced flag v' as a false positive (FP), and a match between the GT voiced flag v and generated unvoiced flag uv' as a false negative (FN).

$$TP = \sum_{i}^{n}[v_i = v'_i], \tag{27}$$

$$FP = \sum_{i}^{n}[uv_i = v'_i], \tag{28}$$

$$FN = \sum_{i}^{n}[v_i = uv'_i], \tag{29}$$

where n is the length of the sequence, and $[a_i = b_i]$ is a function that returns 1 if the i-th element has the same value, and 0 otherwise. The precision and recall are defined as follows:

$$precision = \frac{TP}{TP+FP}, recall = \frac{TP}{TP+FN}. \tag{30}$$

We then calculated the F1 score as follows:

$$F1_{v/uv} = \frac{2}{recall^{-1} + precision^{-1}}. \tag{31}$$

5) KL Divergence of Log f0 / Log Energy: We also measured the KL divergence to analyze the pitch and energy distribution. We first extracted the pitch and energy in log-scale. We then binned the entire range into 100 bins and applied a kernel density

TABLE I
Objective and Subjective Evaluation of Previous Methods on the VCTK Dataset

Method	**MOS (CI)** (↑)	**CER (%)** (↓)	**WER (%)** (↓)	**DDUR** (↓)	**EER (%)** (↓)	**RMSE**$_{f0}$ (↓)	**RMSE**$_{period}$ (↓)	**F1**$_{v/uv}$ (↑)	**RTF** (↓)	**Params**
GT	4.24 (± 0.02)	0.63	1.53	-	2.184	-	-	-	-	-
GT (Vocoded)	4.17 (± 0.03)	0.76	1.74	-	2.539	22.80	0.336	0.7923	-	-
FastSpeech2	3.84 (± 0.03)	1.81	4.89	0.305	8.685	88.83	0.486	0.6848	**0.015**	27M
ProsoSpeech	3.97 (± 0.04)	1.68	3.82	0.296	7.200	59.66	0.483	0.6874	0.149	59M
DiffProsody	**4.03** (± 0.03)	**0.90**	**2.55**	**0.295**	**5.404**	**54.60**	**0.475**	**0.6952**	0.054	53M

TABLE II
Objective and Subjective Evaluation of Previous Methods on the LibriTTS Dataset

Method	**MOS (CI)** (↑)	**CER (%)** (↓)	**WER (%)** (↓)	**DDUR** (↓)	**EER (%)** (↓)	**RMSE**$_{f0}$ (↓)	**RMSE**$_{period}$ (↓)	**F1**$_{v/uv}$ (↑)	**RTF** (↓)	**Params**
GT	3.96 (± 0.06)	0.55	3.40	-	2.835	-	-	-	-	-
GT (Vocoded)	3.93 (± 0.07)	0.91	4.17	-	3.030	38.27	0.311	0.7947	-	-
FastSpeech2	3.76 (± 0.07)	2.11	6.20	0.389	14.584	152.87	**0.471**	**0.7072**	-	27M
ProsoSpeech	3.60 (± 0.07)	11.53	19.25	0.443	21.549	220.89	0.482	0.6051	-	59M
DiffProsody	**3.87 (± 0.07)**	**1.85**	**4.91**	**0.356**	**14.141**	**138.33**	0.472	0.6768	-	53M

estimation [57] to each bin to calculate the KL divergence for a smoothed distribution. The KL divergence for feature x is defined as follows:

$$KLD_x = \frac{1}{N}\sum_{i=1}^{N}(KLD(KDE(x_i), KDE(x'_i)), \tag{32}$$

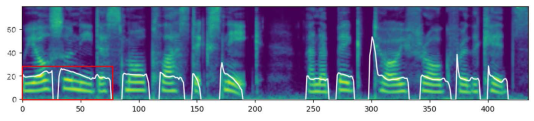
where N is the number of bins, x_i is the probability in the i-th bin of the distribution of x, KDE is the kernel density estimator function, and KLD is the KL divergence calculation.

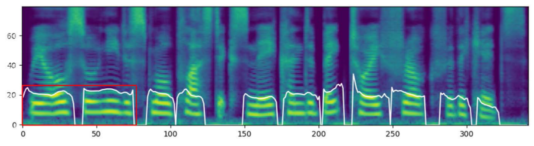
F. Evaluation Results

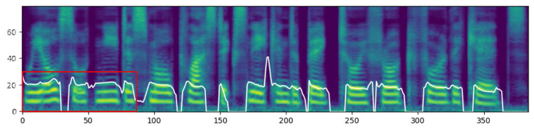
Table I lists the MOS results and objective evaluations. The results demonstrate that the proposed model DiffProsody surpasses the other models in terms of both subjective and objective metrics. The p-values for paired-t tests between DiffProsody and FastSpeech2 and DiffProsody and ProsoSpeech on the VCTK dataset are 1.2×10^{-15} and 0.03, respectively, and on the LibriTTS dataset they are 0.01 and 3.6×10^{-9}, respectively. Thus, we can observe a significant difference (p-value < 0.05) between the proposed method and the comparison methods. DiffProsodys outperformed the other models in DDUR, EER, RMSE$_{f_0}$, RMSE$_{period}$, and F1$_{v/uv}$ on the VCTK dataset. On the LibriTTS dataset, DiffProsody performed best for DDUR, EER, and RMSE$_{f_0}$, while FastSpeech 2 performed best for RMSE$_{period}$ and F1$_{v/uv}$. We found that ProsoSpeech performed poorly on the LibriTTS dataset. We hypothesize that this is due to the fact that LibriTTS has longer sentences than VCTK, which leads to more failure cases in the AR model. We conducted all of our experiments on a single GPU and it seems that more resources are needed to optimize the model. Our model also showed the ability to synthesize speech with more accurate pronunciation, as it had lower WER and CER across all datasets.

Fig. 3 presents the Mel-spectrogram and pitch contour of speech from each model. The red box in the figure indicates that the DiffProsody model was more similar to the GT. To further evaluate the prosody of the generated speech, we examined pitch and energy distributions.

Fig. 4 presents the histogram distribution for log f0 (pitch), and Fig. 5 presents the histogram distribution for log energy.


(a) Ground truth


(b) FastSpeech2


(c) ProsoSpeech

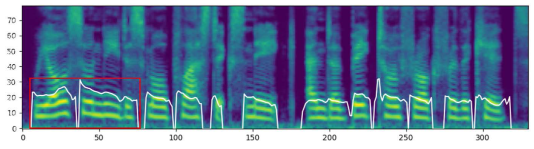
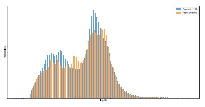
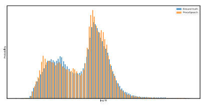

(d) DiffProsody

Fig. 3. Comparison of the visualized spectrogram and pitch contour. The red box indicates that the proposed model is more similar to the GT.


(a) FastSpeech2


(b) ProsoSpeech

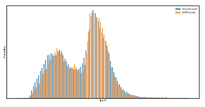
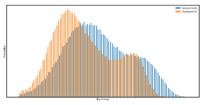
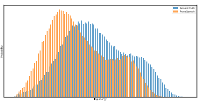

(c) DiffProsody

Fig. 4. Histogram visualization of log f0, where the blue bars represent the GT distribution and orange bars represent the generated distribution. The distribution of the proposed model overlaps to a greater extent with the GT distribution than the other comparison models.

In both the figures, the blue bars represent the distribution of the GT features, and the orange bars represent the distribution of the generated features. The results indicate that the proposed model aligns more closely with the GT distribution than the other


(a) FastSpeech2


(b) ProsoSpeech

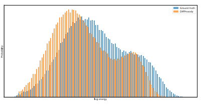

(c) DiffProsody

Fig. 5. Histogram visualization of log energy, where the blue bars represent the GT distribution and orange bars represent the generated distribution. The proposed model's distribution has a greater overlap with the GT distribution compared to the other models being compared.

TABLE III
KL Divergence of Log F0 and Energy

Number of Mel-bins	KL divergence $\log f0$	KL divergence $\log energy$
FastSpeech 2	0.00574	0.02375
ProsoSpeech	0.00542	0.02657
DiffProsody	**0.00343**	**0.01547**

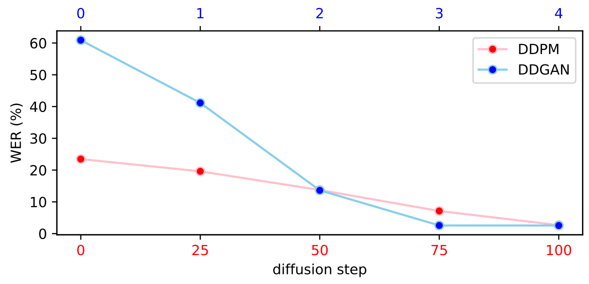
models. Table III lists the KL divergence values for comparison. ProsoSpeech exhibited a better performance in f0 than FastSpeech 2, but not in the case of energy. However, DiffProsody outperformed the comparison model in terms of f0 and energy.

G. Latent Prosody Generation Method

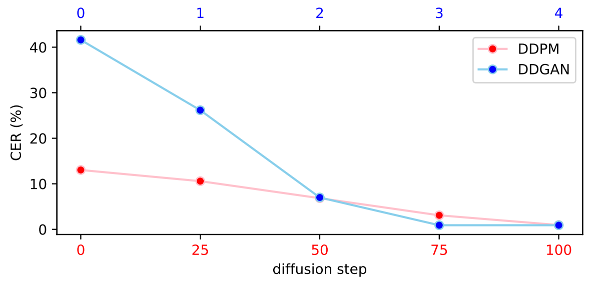
In Table IV, DiffProsody (AR) is a model comprising the use of an AR prosody predictor. DiffProsody (DDPM) is a model comprising the use of DDPM (100 timesteps). DiffProsody is a model comprising the use of DDGAN (four timesteps). The AR predictor has the same structure as the prosody predictor in ProsoSpeech, but it does not use context encoders, and the denoising network in the DDPM has the same structure as generator g_θ in the DDGAN. We also conducted a 7-point comparative mean opinion score (CMOS) evaluation to compare the latent prosody generation methods and measured the RTF to compare the generation speeds. DDGAN achieved +0.172 and −0.015 CMOS compared with AR and DDPM, respectively. The objective metrics showed that the DDPM and DDGAN outperformed the AR. The DDGAN achieved results nearly identical to those of the DDPM for all the metrics. The experimental results showed that the diffusion models performed better than the AR models, as reported in [61]. Furthermore, DDGAN can generate high-quality prosody vectors such as DDPM in only four timesteps. According to the RTF results, the DDGAN produces a 2.7× and 16× faster prosody than the AR and DDPM, respectively.

Fig. 6 presents the traces of the WER, CER, and EER results for the speech generated by the DLPG with the prosody vector generated by each denoising iteration. The red and blue lines represent the results obtained using the DDPM and DDGAN, respectively. We observed that the DDGAN has a larger error rate than the DDPM in the early stages and that the midpoint and final stages of each model have almost the same value. This implies that the DDGAN compresses the timesteps of the DDPM by modeling the denoising process as a multimodal non-Gaussian distribution.

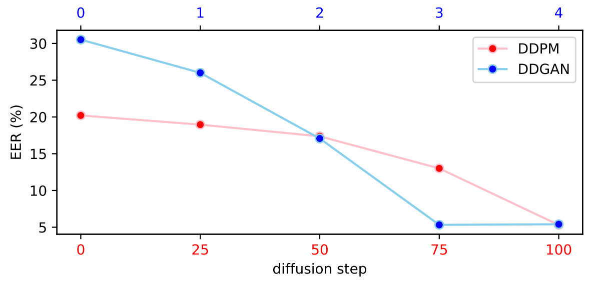
Fig. 7 shows the objective evaluation results for DDPM and DDGAN trained over different time steps: [4, 8, 16, 32, 64, 100]. It was observed that DDPM showed suboptimal performance



(a) WER



(b) CER



(c) EER

Fig. 6. Comparison of objective evaluation results based on diffusion timesteps when using the DDPM and DDGAN framework in DLPG. The blue line is the result for the DDGAN and the red line is the result for the DDPM.

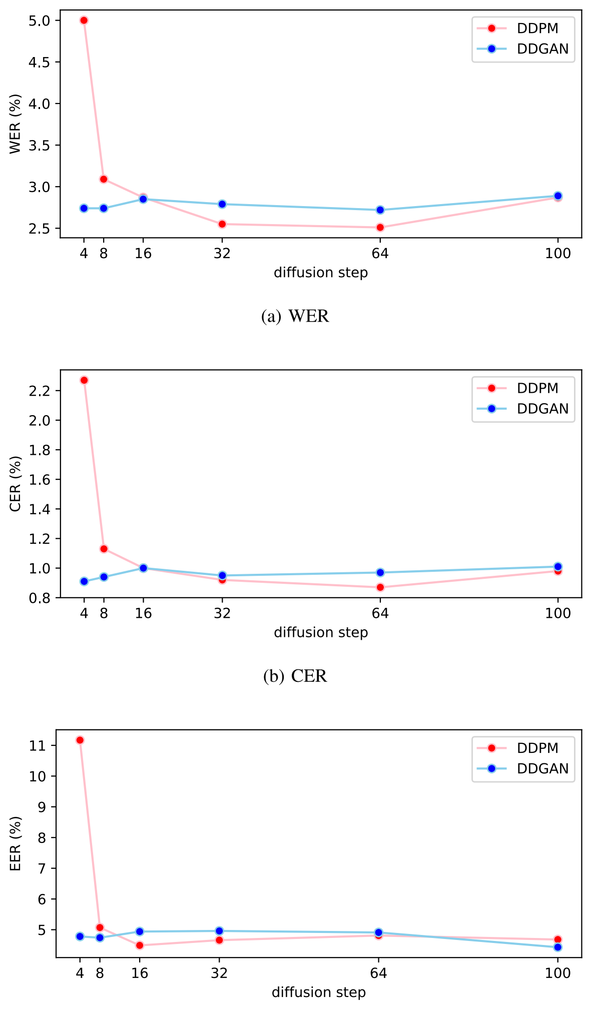
at lower timesteps. In contrast, DDGAN showed consistent robustness across the range of timesteps.

H. Prosody Conditional Adversarial Training

We conducted CMOS to assess the effectiveness of PCD. The CMOS values were used to measure the degree of PCD preference in comparison with the reference model. Our evaluation involved the analysis of three different models: DiffProsody using PCD and a diffusion-based model, DiffProsody without PCD (referred to as DiffProsody (w/o PCD)) using a diffusion-based model, and ProsoSpeech using an AR prosody predictor without PCD. The results presented in Table V clearly indicate that models comprising the use of the PCD are preferred over those trained without the PCD. Furthermore, by comparing the CMOS results of DiffProsody (w/o PCD) and ProsoSpeech, we can infer that the diffusion-based method is preferable to the method employed by ProsoSpeech. To provide a more comprehensive analysis, we also incorporated the objective metrics for DiffProsody (w/o PCD) in Table IV. These objective metric

TABLE IV
OBJECTIVE AND SUBJECTIVE EVALUATIONS OF ABLATION STUDY MODELS

Method	CMOS	CER (%) (↓)	WER (%) (↓)	DDUR (↓)	EER (%) (↓)	RMSE$_{period}$ (↓)	RMSE$_{period}$ (↓)	F1$_{v/uv}$ (↑)	RTF (↓)	Params
DiffProsody	0.0	0.90	2.55	0.295	5.404	54.60	0.475	0.6952	0.054	53M
DiffProsody (pitch & energy)	−0.118	1.34	3.39	0.385	5.965	55.07	0.480	0.6822	0.054	45M
DiffProsody (AR)	−0.172	1.41	3.73	0.308	6.947	65.63	0.486	0.6797	0.134	59M
DiffProsody (DDPM)	+0.015	0.96	2.76	0.282	5.263	54.04	0.477	0.6933	0.871	53M
DiffProsody (w/o PCD)	-	1.19	3.39	0.306	5.762	56.09	0.477	0.6921	0.054	53M
DiffProsody (w/o VQ)	-	1.41	3.73	0.308	7.306	62.03	0.482	0.6783	0.054	53M



(a) WER

(b) CER

(c) EER

Fig. 7. Comparison of objective evaluation results of DDPM and DDGAN trained with [4,8,16,32,64,100] timesteps. The blue line is the result for the DDGAN and the red line is the result for the DDPM.

TABLE V
CMOS RESULTS FOR PROSODY CONDITIONAL DISCRIMINATOR

Method	reference	CMOS
DiffProsody	✓	0.0
DiffProsody (w/o PCD)		−0.171
DiffProsody	✓	0.0
ProsoSpeech		−0.183
DiffProsody (w/o PCD)	✓	0.0
ProsoSpeech		−0.065

TABLE VI
OBJECTIVE EVALUATION BASED ON THE NUMBER OF MEL-BINS

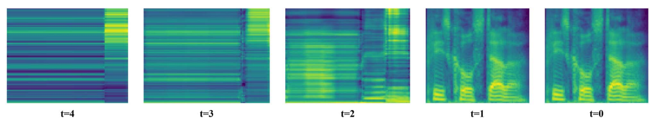
Number of Mel-bins	EER	CER	WER
DiffProsody (10)	5.71	0.95	**2.49**
DiffProsody (20)	**5.40**	**0.90**	2.55
DiffProsody (30)	6.26	1.0	2.58
DiffProsody (40)	6.56	0.98	2.59
DiffProsody (60)	6.20	0.97	2.52
DiffProsody (80)	7.11	0.98	2.67

results demonstrate that DiffProsody outperformed DiffProsody (w/o PCD) in all aspects. However, DiffProsody (w/o PCD) still received a better objective evaluation than ProsoSpeech. We also compared DiffProsody(AR) to ProsoSpeech and found that DiffProsody(AR) consistently performed better in most objective evaluations. This shows that PCD contributes to performance improvements, given that we used the same AR predictor. These findings validate that both PCD and DLPG significantly improve the model performance.

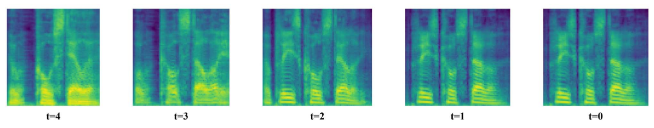
I. *Prosody Encoder Evaluation*

In this section, the effectiveness of the prosody encoder is evaluated. We focus on two main aspects: the impact of the number of Mel-bins used in the reference Mel-spectrogram and the role of the vector quantization layer. Detailed results of these evaluations are presented in the following subsections.

1) Impact of the Number of Mel-Bins: We conducted an experiment to compare the performance of DiffProsody when trained using various numbers of Mel-bins. The results, including the EER, CER, and WER scores, are presented in Table VI. We examined the performance of 10, 20, 30, 40, 60, and 80 (full-band) Mel-bins. The findings indicated that as the number of bins exceeded 20 (baseline), the EER tended to increase, while no significant difference was observed in terms of the CER and WER. It should be noted that the model with 10 bins outperformed the baseline (20 bins) in terms of the WER but yielded higher EER results. As the number of Mel-bins used for the training was increased, the results of the Mel-spectrogram progressively smoothened and eventually collapsed. This phenomenon occurs because the large amount of information in the reference forces the model to reconstruct the Mel-spectrogram by leveraging the prosody vectors. Through this experiment, we found that, as N increases, linguistic information becomes increasingly entangled. Consequently, it is reasonable to employ 20 Mel-bins as the input to the prosody encoder for realizing effective prosody modeling.



(a) DiffProsody (w/o VQ)



(b) DiffProsody

Fig. 8. Visualization of a Mel-spectrogram trace synthesized with latent generation for each diffusion timestep. (a) DiffProsody (w/o VQ) trained without vector quantization in prosody encoder; (b) DiffProsody. (a) shows that, in contrast (b), the early stages of diffusion produce a completely collapsed Mel-spectrogram. This is caused by the inclusion of linguistic information in the prosody vector.

2) Vector Quantization Layer Analysis: Fig. 8 presents a Mel-spectrogram trace synthesized using the prosody vector generated for each diffusion step of the DLPG. We compared two versions of DiffProsody: one trained without the vector quantization layer (Fig. 8(a); defined as DiffProsody (w/o VQ)) and the other trained with the vector quantization layer (Fig. 8(b); defined as DiffProsody). In the case of DiffProsody (w/o VQ), the early steps exhibited a completely distorted Mel-spectrogram, but there was a significant recovery in the middle steps. Conversely, the initial step of DiffProsody exhibits a smoothed but slightly distorted Mel-spectrogram that gradually returns to its original state over the subsequent steps. This phenomenon is also related to prosody disentangling, and we confirmed that the prosody disentangling failed in DiffProsody (w/o VQ). This experiment demonstrated that vector quantization plays a crucial role in effective prosody disentangling. The last row of Table IV presents the objective evaluation results for DiffProsody (w/o VQ). DiffProsody (w/o VQ) performed worse for all the objective measurements. These results provide an objective assessment of how the failure to properly disentangle prosody affects the overall performance of the system.

V. Conclusion

In this study, a novel technique called DiffProsody is proposed, the aims of which is to synthesize high-quality expressive speech. Through prosody conditional adversarial training, we observed significant improvements in speech quality with a more pronounced display of expressive prosody. In addition, our DLPG successfully generated expressive prosody. Our proposed method outperformed comparative models in terms of producing accurate and expressive prosody, as evidenced by the prosody evaluation metrics. Moreover, our method demonstrated superior accuracy in pronunciation, as indicated by the CER and WER evaluations. The KL divergence and histogram analysis further support the claim that DiffProsody yields a more accurate prosody distribution than the other models. Furthermore, we successfully reduced the sampling speed while maintaining the expected performance by introducing DDGAN.

Despite these advances, we recognize certain challenges. Vector quantization is key for separating different elements in our model, but it can sometimes reduce the model's effectiveness. To tackle this, we are considering the use of advanced techniques like residual vector quantizers [62], [63], [64]. We also realize that there are some challenges in simulating natural speech patterns, or prosody, using only TTS data. A possible approach could be to use a language model pre-trained on a larger dataset, such as HuBERT [65], which could significantly improve the naturalness and expressiveness of the speech we synthesize. In the future, we improve our method by adding the ability to more precisely control emotional tones in speech [66].

References

[1] J. Shen et al., "Natural TTS synthesis by conditioning wavenet on Mel spectrogram predictions," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2018, pp. 4779–4783.
[2] Y. Ren et al., "FastSpeech: Fast, robust and controllable text to speech," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, vol. 32.
[3] Y. Ren et al., "FastSpeech 2: Fast and high-quality end-to-end text to speech," in *Proc. Int. Conf. Learn. Representations*, 2021.
[4] Y. Wang et al., "Tacotron: Towards end-to-end speech synthesis," in *Proc. Interspeech*, 2017, pp. 4006–4010.
[5] Y. Ren, J. Liu, and Z. Zhao, "PortaSpeech: Portable and high-quality generative text-to-speech," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 13963–13974.
[6] J. Liu, C. Li, Y. Ren, F. Chen, and Z. Zhao, "DiffSinger: Singing voice synthesis via shallow diffusion mechanism," in *Proc. AAAI Conf. Artif. Intell.*, 2022, vol. 36, no. 10, pp. 11020–11028.
[7] S.-H. Lee, H.-R. Noh, W.-J. Nam, and S.-W. Lee, "Duration controllable voice conversion via phoneme-based information bottleneck," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 30, pp. 1173–1183, 2022.
[8] R. Skerry-Ryan et al., "Towards end-to-end prosody transfer for expressive speech synthesis with Tacotron," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, vol. 80, pp. 4693–4702.
[9] G. Xu, W. Song, Z. Zhang, C. Zhang, X. He, and B. Zhou, "Improving prosody modelling with cross-utterance bert embeddings for end-to-end speech synthesis," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2021, pp. 6079–6083.
[10] Y. Wang et al., "Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, vol. 80, pp. 5180–5189.
[11] D. Min, D. B. Lee, E. Yang, and S. J. Hwang, "Meta-StyleSpeech : Multi-speaker adaptive text-to-speech generation," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, vol. 139, pp. 7748–7759.
[12] J.-H. Lee, S.-H. Lee, J.-H. Kim, and S.-W. Lee, "PVAE-TTS: Adaptive text-to-speech via progressive style adaptation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2022, pp. 6312–6316.
[13] N. Kumar, A. Narang, and B. Lall, "Zero-shot normalization driven multi-speaker text to speech synthesis," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 30, pp. 1679–1693, 2022.
[14] S. Seshadri, T. Raitio, D. Castellani, and J. Li, "Emphasis control for parallel neural TTS," in *Proc. Interspeech* 2022, pp. 3378–3382.
[15] H. Bae and Y.-S. Joo, "Enhancement of pitch controllability using timbre-preserving pitch augmentation in FastPitch," in *Proc. Interspeech*, 2022, pp. 6–10.
[16] A. Łańcucki, "Fastpitch: Parallel text-to-speech with pitch prediction," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2021, pp. 6588–6592.
[17] X. Li, S. Liu, M. W. Y. Lam, Z. Wu, C. Weng, and H. Meng, "Diverse and expressive speech prosody prediction with denoising diffusion probabilistic model," in *Proc. InterSpeech*, 2023, pp. 4858–4862.
[18] T. Raitio, J. Li, and S. Seshadri, "Hierarchical prosody modeling and control in non-autoregressive parallel neural TTS," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2022, pp. 7587–7591.
[19] M. Chen et al., "AdaSpeech: Adaptive text to speech for custom voice," in *Proc. Int. Conf. Learn. Representations*, 2021.
[20] Y. Ren et al., "Prosospeech: Enhancing prosody with quantized vector pre-training in text-to-speech," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2022, pp. 7577–7581.

[21] D. Stanton, Y. Wang, and R. Skerry-Ryan, "Predicting expressive speaking style from text in end-to-end speech synthesis," in *Proc. IEEE Spoken Lang. Technol. Workshop*, 2018, pp. 595–602.

[22] L. Xue, F. K. Soong, S. Zhang, and L. Xie, "ParaTTS: Learning linguistic and prosodic cross-sentence information in paragraph-based TTS," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 30, pp. 2854–2864, 2022.

[23] C. Du and K. Yu, "Phone-level prosody modelling with GMM-Based MDN for diverse and controllable speech synthesis," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 30, pp. 190–201, 2022.

[24] R. Liu, B. Sisman, G. Gao, and H. Li, "Expressive TTS training with frame and style reconstruction loss," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 29, pp. 1806–1818, 2021.

[25] B. Stephenson, L. Besacier, L. Girin, and T. Hueber, "BERT, can HE predict contrastive focus? Predicting and controlling prominence in neural TTS using a language model," in *Proc. Interspeech*, 2022, pp. 3383–3387.

[26] H.-W. Yoon et al., "Language model-based emotion prediction methods for emotional speech synthesis systems," in *Proc. Interspeech*, 2022, pp. 4596–4600.

[27] P. Makarov et al., "Simple and effective multi-sentence TTS with expressive and coherent prosody," in *Proc. Interspeech*, 2022, pp. 3368–3372.

[28] J.-S. Hwang, S.-H. Lee, and S.-W. Lee, "PauseSpeech: Natural speech synthesis via pre-trained language model and pause-based prosody modeling," 2023, *arXiv:2306.07489*.

[29] G. Sun et al., "Generating diverse and natural text-to-speech samples using a quantized fine-grained vae and autoregressive prosody prior," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 6699–6703.

[30] C. Saharia et al., "Photorealistic text-to-image diffusion models with deep language understanding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 36479–36494.

[31] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, and M. Kudinov, "Grad-TTS: A diffusion probabilistic model for text-to-speech," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, vol. 139, pp. 8599–8608.

[32] X. Li, J. Thickstun, I. Gulrajani, P. S. Liang, and T. B. Hashimoto, "Diffusion-LM improves controllable text generation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 4328–4343.

[33] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10684–10695.

[34] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 6840–6851.

[35] H. Zen et al., "LibriTTS: A corpus derived from LibriSpeech for text-to-speech," in *Proc. Interspeech*, 2019, pp. 1526–1530.

[36] A. Vaswani et al., "Attention is All you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017.

[37] J. Kim, S. Kim, J. Kong, and S. Yoon, "Glow-TTS: A generative flow for text-to-speech via monotonic alignment search," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 8067–8077.

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

[39] S.-H. Lee, S.-B. Kim, J.-H. Lee, E. Song, M.-J. Hwang, and S.-W. Lee, "HierSpeech: Bridging the gap between text and speech by hierarchical variational inference using self-supervised representations for speech synthesis," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 16624–16636.

[40] I. Goodfellow et al., "Generative adversarial Nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, vol. 27.

[41] Z. Xiao, K. Kreis, and A. Vahdat, "Tackling the generative learning trilemma with denoising diffusion GANs," in *Proc. Int. Conf. Learn. Representations*, 2022.

[42] X. An, F. K. Soong, and L. Xie, "Disentangling style and speaker attributes for TTS style transfer," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 30, pp. 646–658, 2022.

[43] J. Lee, W. Jung, H. Cho, and J. Kim, "PITS: Variational pitch inference without fundamental frequency for end-to-end pitch-controllable TTS," 2023, *arXiv:2302.12391*.

[44] Y. Ju et al., "TriniTTS: Pitch-controllable end-to-end TTS without external aligner," in *Proc. Interspeech*, 2022, pp. 16–20.

[45] J. D. M.-W. C. Kenton and L. K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chap. Assoc. Comput. Linguistics - Hum. Lang. Technol.*, 2019, pp. 4171–4186.

[46] T. Brown et al., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 1877–1901.

[47] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, "Generalized end-to-end loss for speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2018, pp. 4879–4883.

[48] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[49] A. van den Oord, O. Vinyals, and K. kavukcuoglu, "Neural Discrete Representation Learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30.

[50] Y. Ren, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, "Revisiting over-smoothness in text to speech," in *Proc. 60th Annu. Meeting Assoc. Comput. Linguistics*, 2022, pp. 8197–8213.

[51] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2794–2802.

[52] A. van den Oord et al., "WaveNet: A generative model for raw audio," in *Proc. 9th ISCA Speech Synth. Workshop*, 2016, p. 125.

[53] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal forced aligner: Trainable text-speech alignment using kaldi," in *Proc. InterSpeech*, 2017, pp. 498–502.

[54] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. Int. Conf. Learn. Representations*, 2019.

[55] J. Kong, J. Kim, and J. Bae, "HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 17022–17033.

[56] J. W. Jung, Y. Kim, H.-S. Heo, B.-J. Lee, Y. Kwon, and J. S. Chung, "Pushing the limits of raw waveform speaker recognition," in *Proc. Interspeech*, 2022, pp. 2228–2232.

[57] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 12449–12460.

[58] J.-X. Zhang, Z.-H. Ling, L.-J. Liu, Y. Jiang, and L.-R. Dai, "Sequence-to-sequence acoustic modeling for voice conversion," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 3, pp. 631–644, Mar. 2019.

[59] J.-X. Zhang, Z.-H. Ling, and L.-R. Dai, "Non-parallel sequence-to-sequence voice conversion with disentangled linguistic and speaker representations," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 28, pp. 540–552, 2020.

[60] M. Müller, "Dynamic time warping," *Inf. Retrieval Music Motion*, pp. 69–84, 2007.

[61] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," 2022, *arXiv:2204.06125*.

[62] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "SoundStream: An end-to-end neural audio codec," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 30, pp. 495–507, 2022.

[63] Z. Borsos et al., "AudioLM: A language modeling approach to audio generation," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 31, pp. 2523–2533, 2023.

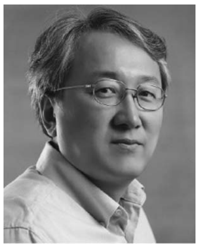
[64] J.-S. Hwang, S.-H. Lee, and S.-W. Lee, "HiddenSinger: High-quality singing voice synthesis via neural audio codec and latent diffusion models," 2023, *arXiv:2306.06814*.

[65] 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 Trans. Audio, Speech, Lang. Process.*, vol. 29, pp. 3451–3460, 2021.

[66] C.-B. Im, S.-H. Lee, S.-B. Kim, and S.-W. Lee, "EMOQ-TTS: Emotion intensity quantization for fine-grained controllable emotional text-to-speech," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2022, pp. 6317–6321.

Hyung-Seok Oh (Student Member, IEEE) received the B.S. degree in computer science and engineering from Konkuk University, Seoul, South Korea, in 2021. He served his internships with Voice&Avatar, Naver Cloud, Seongnam, South Korea, in 2023. He is currently working toward the integrated master's and Ph.D. degrees with the Department of Artificial Intelligence, Korea University, Seoul. His research interests include artificial intelligence and audio signal processing.

Sang-Hoon Lee (Member, IEEE) received the B.S. degree in life science from Dongguk University, Seoul, South Korea, in 2016, and the Ph.D. degree in brain and cognitive engineering from Korea University, Seoul, in 2023. He was a Postdoctoral Researcher with AI Research Center, Korea University, in 2024. He is currently an Assistant Professor with the Department of Software and Computer Engineering from Ajou University, Suwon, South Korea. His research interests include artificial intelligence and audio signal processing.



Seong-Whan Lee (Fellow, IEEE) received the B.S. degree in computer science and statistics from Seoul National University, Seoul, South Korea, in 1984, and the M.S. and Ph.D. degrees in computer science from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 1986 and 1989, respectively. He is currently the Head of the Department of Artificial Intelligence, Korea University, Seoul. His research interests include artificial intelligence, pattern recognition, and brain engineering. He is a Fellow of the International Association of Pattern Recognition (IAPR) and the Korea Academy of Science and Technology.