
DEX-TTS: Diffusion-based EXpressive Text-to-Speech with Style Modeling on Time Variability

Hyun Joon Park, Jin Sob Kim, Wooseok Shin, Sung Won Han*

Korea University

{winddori2002, jinsob, wsshin95, swhan}@korea.ac.kr

Abstract

Expressive Text-to-Speech (TTS) using reference speech has been studied extensively to synthesize natural speech, but there are limitations to obtaining well-represented styles and improving model generalization ability. In this study, we present Diffusion-based EXpressive TTS (DEX-TTS), an acoustic model designed for reference-based speech synthesis with enhanced style representations. Based on a general diffusion TTS framework, DEX-TTS includes encoders and adapters to handle styles extracted from reference speech. Key innovations contain the differentiation of styles into time-invariant and time-variant categories for effective style extraction, as well as the design of encoders and adapters with high generalization ability. In addition, we introduce overlapping patchify and convolution-frequency patch embedding strategies to improve DiT-based diffusion networks for TTS. DEX-TTS yields outstanding performance in terms of objective and subjective evaluation in English multi-speaker and emotional multi-speaker datasets, without relying on pre-training strategies. Lastly, the comparison results for the general TTS on a single-speaker dataset verify the effectiveness of our enhanced diffusion backbone. Demos are available here.²

1 Introduction

Text-to-Speech (TTS) [1, 2, 3, 4] is the task of synthesizing natural speech from a given text, which is applied to various applications such as voice assistant services. To generate diverse and high-fidelity speech, researchers have studied Transformer [5]-, GAN [6, 7]-, and normalizing flow [8, 9]-based TTS as deep generative models. Recently, with the success of diffusion models in various generative tasks [10, 11, 12], researchers have shifted their focus to diffusion-based TTS [13, 14, 15, 16] and proved the outstanding performance of diffusion models in TTS as well.

Despite the improvement of the above general TTS studies, synthesizing human-like speech remains challenging due to the lack of expressiveness of synthesized speech such as the limited styles of reading, prosody, and emotion [17]. Since expressiveness can be reflected during synthesizing acoustic features such as mel-spectrograms, acoustic models have been investigated for expressive TTS. Although some studies [18, 19, 20] generate expressive speech using emotion labels or style tags, the necessity of label information constrains the applicability of the methods.

Considering the previous limitation, researchers have adopted reference-based TTS, which can operate without explicit labels [21, 22, 23, 24, 25, 26, 27], for expressive TTS. These methods extract styles (e.g., emotion, timbre, and prosody) from reference speech and reflect these styles in the speech. For real-world applications, the reference-based TTS is designed to enable handling unseen reference speech, like reference speech from unseen speakers during training.

*Corresponding author.

²Audio samples are available at <https://dextts.github.io/demo.github.io/>

As aforementioned, expressive TTS utilizing a reference involves two steps, extracting the reference information (extractor) and incorporating the information into the synthesis process (adapter). For outstanding expressive TTS, the extractor and adapter should be designed based on the following two aspects: a well-represented style and generalization. That is, expressive TTS should extract rich styles from references and incorporate these styles into the synthesis process. Furthermore, it should have a high generalization ability to operate even in zero-shot scenarios. However, previous studies lacked considerations for network design from the above perspectives, resulting in lower performance in zero-shot or insufficient style reflection. It can be exacerbated when expressive speech is used as a reference because expressive speech contains diverse style information. It suggests the necessity of the network design under the above perspectives for expressive TTS. Some studies [25, 26, 27] attempted to address this problem through pre-training stages or networks. However, problems such as complicated pipelines, additional label requirements, and dependencies on other models remain.

Another focus of this study is designing a strong TTS backbone, which is a component of expressive TTS, to obtain superior expressive TTS. We investigate diffusion-based TTS since it can synthesize high-quality speech through iterative denoising processes. Furthermore, we expect that style information can be effectively reflected by iteratively incorporating style information during the denoising process. A few studies [14, 15, 28] on diffusion-based TTS have improved TTS performance by adapting diffusion formulations to suit TTS. However, the network of these studies was confined to simple U-Net, leading to limited latent representations. Although U-DiT-TTS [29] used DiT [30] for the diffusion network, DiT is yet to be effectively leveraged for TTS.

To address the discussion, we propose a novel acoustic model, Diffusion-based EXpressive TTS (DEX-TTS). Based on a general diffusion TTS, DEX-TTS contains encoders and adapters to handle the styles of reference speech. First, we adopt overlapping patchify and convolution-frequency patch embedding to enhance the DiT-based diffusion TTS backbone. Furthermore, we separate styles into time-invariant and time-variant styles to extract diverse styles even from expressive reference speech. We design each time-invariant and time-variant encoder which utilizes multi-level feature maps and vector quantization, making well-refined styles. Lastly, we propose time-invariant and time-variant adapters that incorporate each extracted style into the speech synthesis process. For effective style reflection and high generalization capability, each method is based on Adaptive Instance Normalization (AdaIN) [31] and cross-attention [32] methods. To effectively leverage the iterative denoising process of diffusion TTS, we design adapters that adaptively reflect styles over time. Through the proposed methods, we can synthesize high-quality and reference-style speech.

We conduct experiments on multi-speaker and emotional multi-speaker datasets to verify the proposed methods. The results reveal that, including zero-shot scenarios, DEX-TTS achieves more outstanding performance than the previous expressive TTS methods in terms of speech quality and similarity. Unlike some existing methods that rely on pre-training strategies, DEX-TTS achieves superior performance as an independent model. Furthermore, to investigate the effect of our strategies to improve the diffusion TTS backbone, we conduct experiments on general TTS using our diffusion backbone. The results on the single-speaker dataset demonstrate the superior performance of the proposed method compared with previous diffusion TTS methods.

2 Related Works

2.1 Diffusion-based Text-to-Speech

As diffusion models in image synthesis have proven their outstanding performance [10, 33, 34], researchers have studied diffusion-based TTS. Diff-TTS [14], Grad-TTS [15], and CoMoSpeech [28] properly utilized diffusion methods for TTS. Although they effectively applied diffusion formulation for TTS, diffusion networks in previous studies were limited to U-Net architectures. It led to limited latent representations, indicating the necessity of improvement in the diffusion network design. U-DiT-TTS [29] improved the design of diffusion networks in TTS by adopting DiT [30] blocks while retaining some aspects of U-Net down- and up-sampling. DiT can extract detailed representations using attention operations between small patches. However, U-DiT-TTS applied large patch strategies and sinusoidal position embedding to synthesize speech of variable time lengths, preventing it from fully leveraging the advantages of DiT. In our work, we adopt an overlapping patchify and convolution-frequency patch embedding to exploit the advantage of DiT structure fully and to design an improved diffusion-based TTS model.

2.2 Expressive Text-to-Speech

Reference-based expressive TTS has attracted considerable interest due to the limitations of previous studies that require additional label information [18, 19, 20]. To condition reference information, some studies [21, 22, 25] used summation or concatenation, but these methods exhibited limited performance in zero-shot. On the other hand, MetaStyleSpeech [24] and StyleTTS [26] utilized adaptive normalization as a style conditioning method for robust performance in zero-shot. However, in these methods, pooling was applied to reference representations to obtain only a single-style vector, which did not effectively extract diverse styles from references. Although GenerSpeech [27] proposed a multi-level style adapter to obtain diverse styles, their conditioning method during the synthesis process was confined to summation or concatenation. Previous studies lacked in designing methods to effectively process styles and improve generalization ability. Furthermore, previous studies [25, 26, 27] have limitations requiring pre-training strategies for feature extraction. In our work, we introduce a novel standalone diffusion-based TTS that handles well-represented styles with dedicated extractors and adapters and exhibits strong generalization performance.

3 DEX-TTS

3.1 Preliminaries

Diffusion Formulation Before introducing our methods, we review the diffusion used in the study. The diffusion model consists of two processes: the diffusion process, which adds Gaussian noise to the original data, and the reverse process, which removes Gaussian noise to generate samples. Diffusion process yields noisy data $\{x\}_{t=0}^T$, where $p_0(x)$ is a data distribution $p_{data}(x)$ and $p_T(x)$ is indistinguishable from pure Gaussian noise. Song et al. [33] addressed the diffusion process as a stochastic process over time t and defined diffusion process with Stochastic Differential Equations (SDE) as $dx = f(x, t)dt + g(t)dw$ where w is Brownian motion, and $f(\cdot, t)$ and $g(\cdot)$ are drift and diffusion coefficients. Song et al. [33] also presented that a probability flow Ordinary Differential Equations (ODE) corresponds to the deterministic process of SDE, and it is defined as below:

$$dx = [f(x, t) - \frac{1}{2}g(t)^2\nabla_x \log p_t(x)]dt \quad (1)$$

The deterministic process is determined from the SDE when the score predicted by the score function $\nabla_x \log p_t(x)$ is known. For the reverse process, a numerical ODE solver such as Euler can be used.

EDM [34] defines the score function as $\nabla_x \log p_t(x) = (D_\theta(x, t) - x_t)/\sigma_t^2$ given σ_t^2 is $\int g(t)^2 dt$, and D_θ is a denoiser network trained by denoising error $\|D_\theta(x_t, t) - x\|_2^2$. To train a denoiser while avoiding gradient variation, EDM introduces pre-conditioning and t -dependent skip connection which are also investigated in CoMoSpeech [28] where the schedule σ_t is t .

$$D_\theta(x_t, t) = c_{skip}(t)x_t + c_{out}(t)F_\theta(c_{in}(t)x, c_{noise}(t)) \quad (2)$$

F_θ is the network before conditioning. We follow Equation 2 with the parameter settings in [34] to build diffusion. In practice, we can forward text and style representations into D_θ and F_θ to condition the denoising process, as described in [28].

3.2 Overall Architecture

Figure 1 depicts the architecture of DEX-TTS. In our TTS system, a vocoder is applied to the synthesized mel-spectrogram to convert it into a signal. DEX-TTS contains encoders and adapters to extract and incorporate style information based on a basic TTS architecture. To effectively extract styles from the reference speech, we define style information as two features, namely time-invariant (T-IV) and time-variant (T-V) styles. T-IV styles contain global information that rarely varies within speech, whereas T-V styles contain information that varies within speech, such as intonation. Based on this approach, the T-IV encoder passes extracted representation h_{inv} to the diffusion decoder, and the T-IV adapter reflects it regardless of the time axis. On the other hand, to preserve the temporal information of the representation h_v^d from the T-V encoder, the T-V adapter in the diffusion decoder reflects the representation by considering the time axis. In addition, the T-V encoder forwards the representation h_v^e to the text encoder since the text representation varies over time.

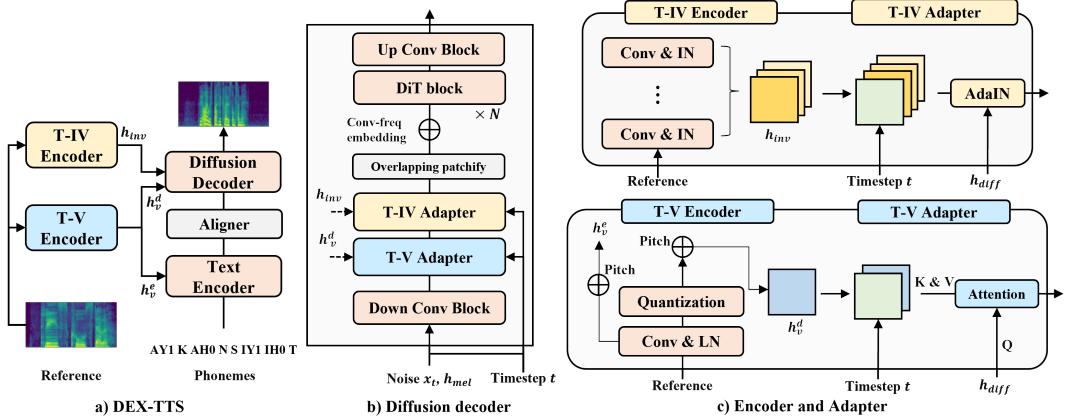


Figure 1: Architecture of DEX-TTS, diffusion decoder, and style encoders and adapters.

Text Encoder Given the input phonemes, the text encoder extracts the text representation h_{text} . The text encoder consists of 8 layers, each composed of Transformer encoder structure [32] that includes Multi-Head Self-Attention (MHSA) and Feed-Forward Network (FFN). To enhance the encoder, we incorporate relative position embedding, RoPE [35], into the attention mechanism and apply the swish gate used in RetNet [36] after the attention operation. Since text varies over time, h_v^e , defined as T-V styles, can be effectively utilized to condition styles for h_{text} . We utilize Adaptive Layer Normalization (AdaLN) [37] after each MHSA and FFN to inject h_v^e , extracted by the T-V encoder given a reference input, into h_{text} . See Section A.1 for more details.

Aligner We adopt a convolution-based Duration Predictor (DP) [9] in which h_{text} extracted by the text encoder is used. DP predicts the duration \hat{d} which maps h_{text} to frames of the mel-spectrogram for the initial mel-spectrogram representation h_{mel} used as the condition input in the diffusion decoder. Aligner is trained by the Monotonic Alignment Search (MAS) algorithm.

Diffusion Decoder Given time t and corresponding noise x_t generated by the diffusion process, the diffusion decoder synthesizes a denoised mel-spectrogram \hat{x} . Here, the initial mel-spectrogram representation h_{mel} and styles $h_{inv,v}$ are utilized as conditioning information. For diffusion representation h_{diff} , we concatenate x_t , h_{mel} , and t , and pass it to the diffusion decoder, where t is projected by sinusoidal encoding and linear layers.

The diffusion decoder comprises convolution blocks, adapters, and DiT blocks [30]. To leverage powerful denoising in the latent space, we utilize up and down convolution blocks, as in [15], to decrease and increase the resolution of h_{diff} . In the bottleneck, each adapter incorporates $h_{inv,v}$ into h_{diff} to reflect each style information. Furthermore, we forward t as an additional condition into adapters to effectively reflect styles during the iterative denoising process.

After adapters, we utilize DiT blocks to enhance latent representations. To effectively exploit DiT blocks, we introduce overlapping patchify and convolution-frequency (conv-freq) patch embedding. Unlike previous methods, we allow overlapping between patches, mitigating boundary artifacts between patches and enabling natural speech synthesis. For patchify, a convolution layer with a kernel size of $2 \times P - 1$ and stride of P is used given patch size P . Let $h \in \mathbb{R}^{C \times F \times T}$ be the representations after adapters, where C , F , and T are the hidden, frequency, and time sizes respectively. Given $F_2 = F/P$ and $T_2 = T/P$, the convolution layer patchifies h into $h_p \in \mathbb{R}^{C \times F_2 \times T_2}$.

Before converting the spatial dimensions of h_p into a sequence for DiT inputs, embeddings are added to each frequency and time dimension. Since the speech length is variable, patch embedding should be able to handle unseen lengths during training. For the time axis, we apply a convolution layer to h_p and take a time-wise average to obtain the relative positional embeddings $PE_T \in \mathbb{R}^{C \times 1 \times T_2}$. On the other hand, we use fixed-size learnable parameters as frequency embedding $PE_F \in \mathbb{R}^{C \times F_2 \times 1}$ since the frequency size is not variable in speech synthesis. PE_T and PE_F are added to h_p , and the spatial dimension is converted into a sequence to be used as input for DiT. By the above embedding approach, we can get robust embedding for variable lengths compared to conventional embeddings

obtained by fixed-size parameters or sinusoidal encoding. After the DiT block, the up-convolution block predicts a denoised mel-spectrogram from latent features.

3.3 Time-Invariant Style Modeling

We model T-IV and T-V styles to extract well-represented styles from the reference speech. For T-IV styles, we design the encoder and adapter to process global information within the speech.

T-IV Encoder As depicted in Figure 1.c, the T-IV encoder consists of a few residual convolution blocks to extract the representation from the reference speech. To maintain individual characteristics regardless of the temporal information within a batch, we utilize Instance Normalization (IN) after each block. Inspired by [38], we use multi-level feature maps as T-IV styles $h_{inv} \in \mathbb{R}^{L \times C \times T}$, where L is the number of layers in the T-IV encoder. Since h_{inv} comprises all stacked feature maps from the convolution block, it contains T-IV information across the convolution blocks.

T-IV adapter For T-IV Adatpor, we apply AdaIN, which can reflect styles regardless of temporal information, to inject h_{inv} into h_{diff} during the denoising process in the diffusion decoder. Given mean μ and standard deviation σ as bias and scale for AdaIN, the process is defined as follows:

$$AdaIN(h_{diff}, \mu, \sigma) = IN(h_{diff}) \times \sigma + \mu \quad (3)$$

where μ and σ are obtained using h_{inv} . Since h_{inv} contains the feature maps of all layers, we compute the channel-wise mean and standard deviation for each layer and utilize attention pooling (AP) [38] to obtain representative statistics (μ and σ) for AdaIN. Furthermore, we include time t in the pooling process to ensure adaptive operation at each time step during the denoising process. The process for extracting μ and σ is represented as follows:

$$\begin{aligned} AP(x) &= \text{sum}(\text{softmax}(xW_{ap}) \times x) \\ \tilde{\mu} &= [t; avg(h_{inv}^1); \dots; avg(h_{inv}^L)], \mu = AP(\tilde{\mu}) \\ \tilde{\sigma} &= [t; std(h_{inv}^1); \dots; std(h_{inv}^L)], \sigma = AP(\tilde{\sigma}) \end{aligned} \quad (4)$$

where W_{ap} is the linear weight for AP. Through AP, we can extract common features across multi-level feature maps to utilize as a general T-IV style. Furthermore, time conditioning enables adaptive style incorporation at each timestep.

3.4 Time-Variant Style Modeling

We define T-V styles as features that emerge with temporal variation within speech. Based on this, we design the encoder and adapter to preserve or reflect the temporal information of the reference.

T-V Encoder Similar to the T-IV encoder, the T-V encoder contains a few residual convolution blocks, but we employ Layer Normalization (LN) instead of IN to preserve temporal relationships in each instance. The T-V encoder extracts two styles $h_v^{e,d}$ for each text encoder and diffusion decoder. We obtain h_v^e by applying convolution blocks to the reference and adding the pitch information h_{f0} . We use h_{f0} to reflect changes in speech over time, and it is extracted by applying GRU layers to the log fundamental frequency of the reference speech. After channel-wise pooling on h_v^e to obtain the overall T-V style information, h_v^e is forwarded to the text encoder.

For h_v^d , we additionally apply Vector Quantization (VQ) to the output of convolution blocks. For VQ, we utilize a latent discrete codebook $e \in \mathbb{R}^{K \times D}$, where K is the codebook size and D is the dimension size. The VQ layer maps the outputs to a discrete space based on the distance to the codebook. This process removes noise from a continuous space, obtaining well-refined style information that can be used as generalized features. After adding the pitch information, h_v^d is passed to the decoder without pooling to preserve temporal information.

AdaLN In the text encoder, we apply AdaLN to reflect the overall style h_v^e while preserving the temporal aspects of the text representation. AdaLN with h_{text} in the encoder is defined as follows:

$$AdaLN(h_{text}, h_v^e) = g(h_v^e) \times LN(h_{text}) + b(h_v^e) \quad (5)$$

where $g(\cdot)$ and $b(\cdot)$ are linear layers for scaling and bias, respectively.

T-V adapter To reflect T-V styles h_v^d to h_{diff} while preserving temporal information, we design the T-V adapter with cross-attention. We use h_{diff} as the query and h_v^d as the key and values for cross-attention (CA), and it is defined as below.

$$\begin{aligned} Q &= IN(h_{diff})W_q, \quad K = h_v^d W_k, \quad V = h_v^d W_v \\ CA(Q, K, V) &= softmax(QK^\top)V \end{aligned} \quad (6)$$

where $W_{q,k,v}$ denotes the linear weight. As presented in Equation 6, IN is applied to h_{diff} for the query. It maintains instance-level features for computing attention scores and enables reflecting suitable T-V style for each instance.

3.5 Loss Function

To train DEX-TTS, we follow the loss formulation of previous diffusion-based TTS studies [15, 28], in which duration loss \mathcal{L}_{dur} , prior loss \mathcal{L}_{prior} , and diffusion loss \mathcal{L}_{diff} are used. \mathcal{L}_{dur} is utilized to train DP that predicts the duration mapping h_{text} to mel frames, and it is defined as $\|\log(d) - \log(\hat{d})\|_2^2$. The duration label d is obtained by MAS algorithm. Given x is the ground-truth mel-spectrogram, \mathcal{L}_{prior} calculates the loss between the initial mel-spectrogram h_{mel} from the aligner and x for stable learning, defined as $\|h_{mel} - x\|_2^2$. To train the diffusion decoder (Denoiser D_θ), we use a denoising error for each timestep t , defined as follows:

$$\mathcal{L}_{diff} = \lambda(t)\|D_\theta(x_t, t, h_{\{mel, inv, v\}}) - x\|_2^2 \quad (7)$$

$\lambda(t)$ is the weight for noise levels determined by t used in [34]. For VQ loss, we adopt a commitment loss [39] \mathcal{L}_{vq} defined as $\|h - sg(e)\|_2^2$, where h is the representation before quantization and sg is the stop-gradient operation. We get the total loss \mathcal{L} by the summation of \mathcal{L}_{dur} , \mathcal{L}_{prior} , \mathcal{L}_{diff} , and \mathcal{L}_{vq} .

4 Experiments

4.1 Experiment Setup

Dataset To evaluate the proposed method, we use the VCTK dataset [40], an English multi-speaker dataset, consisting of approximately 400 utterances per 109 speakers. We split the dataset into about 70%, 15%, and 15% for the train, validation, and test sets, respectively, based on speakers to consider both the seen and unseen (zero-shot) scenarios. For the zero-shot scenario, 10 unseen speakers are used. In addition, we conduct experiments on the Emotional Speech Dataset (ESD) [41] to verify whether the models can reflect styles using expressive reference speech. The ESD contains 10 English and 10 Chinese speakers with 400 sentences per speaker for five emotions (happy, sad, neutral, surprise, and angry). We only use English speakers and keep the same split ratio as that in the VCTK dataset. Two unseen speakers are used for the zero-shot scenario. Considering real-world applications, we design both parallel and non-parallel test scenarios based on whether the input text is the same as the text of the reference speech. For the experimental results, we record the average performances of the parallel and non-parallel scenarios. Finally, all datasets are resampled to 22 kHz.

Baselines For comparison, we set the following systems as baselines: **1) Ref**, the reference audio. **2) MetaStyleSpeech** [24], multi-speaker adaptive TTS with meta-learning. **3) YourTTS** [25], VITS-based zero-shot multi-speaker TTS with the pre-trained speaker encoder. **4) GenerSpeech** [27], style transfer method for out-of-domain TTS. **5) StyleTTS** [26], style-based TTS with transferable aligner and AdaIN. Except for YourTTS (end-to-end TTS), the generated mel-spectrograms are transformed into waveforms by the pre-trained HiFi-GAN [42]. We record the performance of the baselines after training with their codes.

Implementation Details For training, we take 1000 and 1500 epochs for the VCTK and ESD datasets, respectively. An Adam optimizer with a learning rate of 10^{-4} and batch size of 32 are used. Regarding the model hyperparameters of the diffusion decoder, we take a patch size P of 2, number of DiT blocks N of 4, and hidden size C of 64. The T-IV and T-V encoders use the 6 layers L , and their dimension sizes are matched with the diffusion decoder. We set a codebook size K of 512 and dimension size D of 192 for the VQ layer in the T-V encoder. We extract mel-spectrograms with 80 mel bins based on the FFT size of 1024, hop size of 256, and window size of 1024, which is

Table 1: Comparison results for expressive TTS on the VCTK dataset.

Model	Seen scenarios				Unseen (zero-shot) scenarios			
	WER	COS	MOS-N	MOS-S	WER	COS	MOS-N	MOS-S
Ref	6.23	-	3.97	-	6.23	-	3.97	-
MetaStyleSpeech [24]	16.58	78.10	3.43	3.63	16.50	73.53	3.38	3.30
YourTTS [25]	21.27	78.78	3.20	3.11	18.34	75.00	3.33	3.08
GenerSpeech [27]	13.87	77.46	3.40	3.25	11.37	73.23	3.46	3.06
StyleTTS [26]	7.72	82.93	3.57	3.70	6.58	77.90	3.53	3.65
DEX-TTS (ours)	7.85	85.31	3.75	3.88	5.84	80.45	3.76	3.81

Table 2: Comparison results for expressive TTS on the ESD dataset.

Model	Seen scenarios				Unseen (zero-shot) scenarios			
	WER	COS	MOS-N	MOS-S	WER	COS	MOS-N	MOS-S
Ref	7.12	-	3.90	-	7.12	-	3.90	-
MetaStyleSpeech [24]	24.56	79.41	3.09	3.53	25.84	73.01	3.19	3.34
YourTTS [25]	16.57	77.61	3.33	3.40	16.35	69.38	3.28	2.96
GenerSpeech [27]	12.75	75.09	3.23	3.28	11.78	70.54	3.06	2.78
StyleTTS [26]	12.59	79.65	3.41	3.50	12.11	72.27	3.23	3.06
DEX-TTS (ours)	8.34	82.71	3.73	3.84	8.35	75.58	3.57	3.52

compatible with the HiFi-GAN vocoder used in our TTS system. For the diffusion denoising steps, we use 50 Number of Function Evaluations (NFE) with the Euler solver (See Section B.3 to find results depending on NFE). All experiments are conducted on a single NVIDIA 3090 GPU. Codes are available here.³

Evaluation Metrics We consider objective and subjective evaluation metrics. As objective metrics, we utilize Word Error Rate (WER %) and Cosine Similarity (COS). WER represents how accurately the model synthesizes the given text, and it is calculated as the error between the predicted text, obtained by applying the pre-trained Wav2Vec 2.0 [43] to the synthesized speech, and the given text. On the other hand, COS indicates the similarity in the feature space between the synthesized and reference speech, and it is calculated using a pre-trained speaker verification model.⁴ For convenience, we show COS multiplied by 100 in the experimental results. For the subjective metrics, we adopt Mean Opinion Score for naturalness and similarity (MOS-N and S). We use Amazon Mechanical Turk (AMT) and ask participants to score on a scale from 1 to 5. They assess the synthesized speech for its naturalness by listening to it, or they compare the synthesized speech with reference speech to evaluate similarity. For every MOS evaluation, we randomly select 30 utterances for each model and guarantee at least 27 participants.

4.2 Experimental Results

We conduct experiments including seen and unseen scenarios on multi-speaker datasets. Since the Ref is used as the reference to calculate the cosine similarity with the synthesized speech, we record only WER and MOS-N for Ref. As depicted in Table 1, results on the VCTK dataset, DEX-TTS outperforms the previous methods in terms of objective and subjective evaluations. Although StyleTTS shows slightly better WER in seen scenarios, the difference is marginal compared to other metrics. DEX-TTS consistently achieves high COS and MOS-S across all scenarios, indicating its ability to effectively capture and reflect rich styles from reference speech. In particular, we observe the high generalization ability of our style modeling since DEX-TTS also shows superior COS and MOS-S in zero-shot. Furthermore, the improved WER performance demonstrates that DEX-TTS can obtain enriched text representations and reflect styles without compromising text information. The outstanding MOS results suggest that DEX-TTS can synthesize reference-style speech with high fidelity. However, pooling-based single-style utilization (MetaStyleSpeech and StyleTTS) and summation- or concatenation-based style reflection methods (YourTTS and GenerSpeech) are not effective for synthesizing reference-style speech.

³Codes are available at <https://github.com/winddori2002/DEX-TTS/>

⁴<https://github.com/resemble-ai/Resemblyzer>

Table 3: Ablation studies on the ESD dataset.

Model	Seen		Unseen	
	WER	COS	WER	COS
DEX-TTS	8.34	82.71	8.35	75.58
a) T-IV adapter → AdaIN	11.91	82.75	10.28	75.29
b) w/o T-IV styles (h_{inv})	10.84	82.62	11.05	74.81
c) w/o T-V styles (h_v^e)	12.19	78.26	11.72	71.91
d) w/o T-V styles (h_v^d)	9.51	82.41	9.18	74.90
e) w/o pitch (h_{f0})	12.7	81.48	10.95	74.72
f) w/o VQ	15.70	82.53	16.84	76.38
g) w/o t for adapters	9.08	82.13	9.15	74.65

To verify the ability of the model to handle the styles of expressive speech, we conduct experiments on the ESD dataset in Table 2. Similar to the results on the VCTK dataset, DEX-TTS outperforms previous TTS methods. It suggests that our style modeling, which handles styles based on time variability, is also effective for expressive reference speech. The outstanding performance of COS and MOS-S in the unseen scenarios indicates a strong generalization ability of DEX-TTS even in the emotional dataset. Furthermore, we observe that DEX-TTS can reflect styles without compromising speech quality compared to previous methods. Finally, unlike previous methods (YourTTS, Gener-Speech, and StyleTTS) that rely on pre-training strategies, DEX-TTS achieves excellent performance without dependence on pre-trained models. It suggests that DEX-TTS can be easily extended to various applications as a standalone model. In Section B.1, we provide results with error bars for the above experiments.

4.3 Ablation Studies

To investigate the effect of the components of DEX-TTS, we conduct ablation studies in Table 3. First, we analyze the effect of the T-IV adapter by replacing it with a simple AdaIN in experiment a). While the T-IV adapter utilizes all the feature maps extracted from the encoder, AdaIN only employs the last feature map. The results show considerable degradation in WER. This suggests that utilizing common features appearing in multi-level feature maps as T-IV styles is more effective. In addition, it enables to obtain well-refined styles that do not affect other speech qualities such as text content.

Experiments from b) to d) show the performance when each style, separated according to time variability, is removed. We observe that T-V and T-IV styles significantly impact WER and COS. It indicates the effectiveness of our approach which distinguishes and processes styles based on their time variability. Moreover, the most significant performance degradation is observed when removing h_v^e , suggesting the importance of incorporating style in the text encoder. Since the output of the text encoder is used as the initial mel representation for the prior loss calculation, style reflection in the text encoder has a considerable effect. Furthermore, the results of experiment e) show the necessity of injecting the pitch information of the reference into the T-V styles. It suggests that pitch information contains additional time-variant styles that cannot be extracted solely from the reference mel-spectrograms.

We observe interesting results for experiment f) in which the VQ layer for T-V styles h_v^d is removed. Although an improvement in the COS in unseen scenarios is observed, there is a significant overall decrease in WER. To preserve the temporal information while incorporating h_v^d , we designed a cross-attention-based T-V adapter. However, when the VQ layer is not applied, it includes excessively detailed style information, improving similarity but significantly degrading other aspects of speech quality. Thus, the VQ layer contributes to obtaining a well-refined time-variant style, enabling an effective reflection of style information while preserving temporal details. The experiment g) demonstrates the results of removing our time step conditioning from the adapters in the diffusion decoder. Overall performance decrease is observed, highlighting the necessity of time step conditioning in adaptively incorporating styles during the iterative denoising process of the diffusion network.

4.4 Further Experiments

As discussed in Section 1, another focus of this study is designing a strong TTS backbone. We improved diffusion-based TTS via overlapping patch strategy and conv-freq embedding, which

Table 4: Results on the LJSpeech test set (left) and ablation studies for encoding types on the LJSpeech test set (right). † indicates the overlapped patch strategy is applied.

Model	WER	COS	MOS-N	Model	Encoding Type	WER	COS
GT	6.56	-	4.60		sin-cos	16.37	69.41
FastSpeech2 [44]	7.70	91.31	3.16		time-freq	8.01	88.15
Grad-TTS [15]	7.70	91.37	4.16		pos-freq	11.83	81.14
CoMoSpeech [28]	8.21	91.58	4.13		conv-freq	7.31	91.66
GeDEX-TTS [†] (ours)	6.55	91.75	4.26	GeDEX-TTS [†]	conv-freq	6.55	91.75

enables the comprehensive utilization of DiT. To investigate the improvements in our diffusion network, we conduct experiments for general TTS which does not use reference speech. We eliminate the modules dependent on reference (See Section A.2 for details), thus the model can operate as general TTS and we call this version General DEX-TTS (GeDEX-TTS). For comparison, we select previous diffusion-based TTS models and train models on a single-speaker dataset, LJSpeech [45], following the set split of [15]. FastSpeech2 is adopted for comparison since it is a popular baseline model in general TTS. We consider 2000 epochs and P of 4 for GeDEX-TTS and other training settings are the same as DEX-TTS. For inference, we use the NFE of 50 with Euler solver for all diffusion models. MOS-N is recorded by evaluations of 16 participants.

As shown in Table 4 (left), GeDEX-TTS achieves the best performance compared to the previous methods in both objective and subjective evaluations. By leveraging patchify and embedding strategies, GeDEX-TTS effectively utilizes the structural advantages of DiT, resulting in superior performance compared to simple U-Net-based diffusion models (i.e., Grad-TTS and CoMoSpeech). Notably, the WER performance of GeDEX-TTS is on par with that of the Ground Truth (GT), showing the validity of network improvement in diffusion TTS. The results reveal that improvements in the diffusion network are consistently effective beyond expressive TTS to general TTS as well, indicating that the proposed method also exhibits considerable significance as a general TTS network.

To analyze the effect of the network improvement strategies, we conduct ablation studies on the LJSpeech dataset. The first block of Table 4 (right) shows the results depending on different ways of encoding for patch embedding. Instead of conv-freq embedding used in our model, we apply other popular methods: **1) sin-cos**, frequency-based positional embeddings [46]. **2) time-freq**, fixed size learnable parameters for time and frequency axis [47]. **3) pos-freq**, positional encoding for the time axis and fixed size learnable parameters for the frequency axis (we added it to compare with conv-freq). The comparison results show lower performance of the conventional encoding types despite their stable performance when using fixed image sizes in image synthesis. This suggests that relative patch embedding using convolution is more suitable for tasks with significant variations in the temporal axis length, such as speech synthesis. Lastly, the results of the second block suggest that the overlapping patchify strategy contributes to synthesizing more natural speech by mitigating the boundary artifacts between patches.

5 Conclusion

In this study, we proposed DEX-TTS, a reference-based TTS, which can synthesize high-quality and reference-style speech. First, we improved the diffusion-based TTS backbone by overlapping patchify and conv-freq embedding strategies, which enable the effective utilization of DiT architecture. To extract well-represented styles from the reference, we categorized the styles into time-invariant and time-variant styles, with T-IV and T-V encoders using multi-level feature maps and vector quantization for obtaining well-refined styles in each manner. We designed adapters with adaptive normalization and cross-attention methods for effective style reflection with high generalization ability. The experimental results on the VCTK and ESD datasets suggest that DEX-TTS, even without using pre-trained strategies, outperformed the previous expressive TTS models. In addition, DEX-TTS consistently exhibited superior performance across all metrics, indicating its effective style reflection ability which did not compromise speech quality, unlike other models. Lastly, to validate our strategies for improving the diffusion network, we conducted experiments using our diffusion TTS backbone in a general TTS task. The results on the LJSpeech dataset demonstrated that our diffusion backbone also achieved outstanding performance in the general TTS task.

Acknowledgments and Disclosure of Funding

This research was supported by a Korea TechnoComplex Foundation Grant (R2112653) and Korea University Grant (K2403371). This research was also supported by Brain Korea 21 FOUR.

References

- [1] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.
- [2] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*, 2017.
- [3] Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4779–4783. IEEE, 2018.
- [4] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech: Fast, robust and controllable text to speech. *Advances in neural information processing systems*, 32, 2019.
- [5] Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, and Ming Liu. Neural speech synthesis with transformer network. In *Proceedings of the AAAI conference on artificial intelligence*, 2019.
- [6] Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel wavenet: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6199–6203. IEEE, 2020.
- [7] Jinhyeok Yang, Jae-Sung Bae, Taejun Bak, Youngik Kim, and Hoon-Young Cho. Ganspeech: Adversarial training for high-fidelity multi-speaker speech synthesis. *arXiv preprint arXiv:2106.15153*, 2021.
- [8] Chenfeng Miao, Shuang Liang, Minchuan Chen, Jun Ma, Shaojun Wang, and Jing Xiao. Flow-tts: A non-autoregressive network for text to speech based on flow. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7209–7213. IEEE, 2020.
- [9] Jaehyeon Kim, Sungwon Kim, Jungil Kong, and Sungroh Yoon. Glow-tts: A generative flow for text-to-speech via monotonic alignment search. *Advances in Neural Information Processing Systems*, 33:8067–8077, 2020.
- [10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- [11] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.
- [12] Flavio Schneider. Archisound: Audio generation with diffusion. *arXiv preprint arXiv:2301.13267*, 2023.
- [13] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.
- [14] Myeonghun Jeong, Hyeongju Kim, Sung Jun Cheon, Byoung Jin Choi, and Nam Soo Kim. Diff-tts: A denoising diffusion model for text-to-speech. *arXiv preprint arXiv:2104.01409*, 2021.
- [15] Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail Kudinov. Grad-tts: A diffusion probabilistic model for text-to-speech. In *International Conference on Machine Learning*, pages 8599–8608. PMLR, 2021.
- [16] Songxiang Liu, Dan Su, and Dong Yu. Diffgan-tts: High-fidelity and efficient text-to-speech with denoising diffusion gans. *arXiv preprint arXiv:2201.11972*, 2022.
- [17] Xu Tan, Tao Qin, Frank Soong, and Tie-Yan Liu. A survey on neural speech synthesis. *arXiv preprint arXiv:2106.15561*, 2021.

- [18] Younggun Lee, Azam Rabiee, and Soo-Young Lee. Emotional end-to-end neural speech synthesizer. *arXiv preprint arXiv:1711.05447*, 2017.
- [19] Minchan Kim, Sung Jun Cheon, Byoung Jin Choi, Jong Jin Kim, and Nam Soo Kim. Expressive text-to-speech using style tag. *arXiv preprint arXiv:2104.00436*, 2021.
- [20] Tao Li, Shan Yang, Liumeng Xue, and Lei Xie. Controllable emotion transfer for end-to-end speech synthesis. In *2021 12th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pages 1–5. IEEE, 2021.
- [21] Yuxuan Wang, Daisy Stanton, Yu Zhang, RJ-Skerry Ryan, Eric Battenberg, Joel Shor, Ying Xiao, Ye Jia, Fei Ren, and Rif A Saurous. Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis. In *International conference on machine learning*, pages 5180–5189. PMLR, 2018.
- [22] Rafael Valle, Jason Li, Ryan Prenger, and Bryan Catanzaro. Mellotron: Multispeaker expressive voice synthesis by conditioning on rhythm, pitch and global style tokens. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6189–6193. IEEE, 2020.
- [23] Keon Lee, Kyumin Park, and Daeyoung Kim. Styler: Style factor modeling with rapidity and robustness via speech decomposition for expressive and controllable neural text to speech. *arXiv preprint arXiv:2103.09474*, 2021.
- [24] Dongchan Min, Dong Bok Lee, Eunho Yang, and Sung Ju Hwang. Meta-stylespeech: Multi-speaker adaptive text-to-speech generation. In *International Conference on Machine Learning*, pages 7748–7759. PMLR, 2021.
- [25] Edresson Casanova, Julian Weber, Christopher D Shulby, Arnaldo Candido Junior, Eren Gölge, and Moacir A Ponti. Yourtts: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone. In *International Conference on Machine Learning*, pages 2709–2720. PMLR, 2022.
- [26] Yinghao Aaron Li, Cong Han, and Nima Mesgarani. Styletts: A style-based generative model for natural and diverse text-to-speech synthesis. *arXiv preprint arXiv:2205.15439*, 2022.
- [27] Rongjie Huang, Yi Ren, Jinglin Liu, Chenye Cui, and Zhou Zhao. Generspeech: Towards style transfer for generalizable out-of-domain text-to-speech synthesis. *arXiv preprint arXiv:2205.07211*, 2022.
- [28] Zhen Ye, Wei Xue, Xu Tan, Jie Chen, Qifeng Liu, and Yike Guo. Comospeech: One-step speech and singing voice synthesis via consistency model. *arXiv preprint arXiv:2305.06908*, 2023.
- [29] Xin Jing, Yi Chang, Zijiang Yang, Jiangjian Xie, Andreas Triantafyllopoulos, and Bjoern W Schuller. U-dit tts: U-diffusion vision transformer for text-to-speech. *arXiv preprint arXiv:2305.13195*, 2023.
- [30] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.
- [31] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017.
- [32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [33] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.
- [34] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022.
- [35] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.
- [36] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.
- [37] Mingjian Chen, Xu Tan, Bohan Li, Yanqing Liu, Tao Qin, Sheng Zhao, and Tie-Yan Liu. Adaspeech: Adaptive text to speech for custom voice. *arXiv preprint arXiv:2103.00993*, 2021.

- [38] Hyun Joon Park, Seok Woo Yang, Jin Sob Kim, Wooseok Shin, and Sung Won Han. Triaan-vc: Triple adaptive attention normalization for any-to-any voice conversion. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.
- [39] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.
- [40] Junichi Yamagishi, Christophe Veaux, and Kirsten MacDonald. Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit (version 0.92). In *University of Edinburgh. The Centre for Speech Technology Research (CSTR)*, 2019.
- [41] Kun Zhou, Berrak Sisman, Rui Liu, and Haizhou Li. Seen and unseen emotional style transfer for voice conversion with a new emotional speech dataset. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 920–924. IEEE, 2021.
- [42] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033, 2020.
- [43] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460, 2020.
- [44] Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech 2: Fast and high-quality end-to-end text to speech. *arXiv preprint arXiv:2006.04558*, 2020.
- [45] Keith Ito and Linda Johnson. The lj speech dataset. <https://keithito.com/LJ-Speech-Dataset/>, 2017.
- [46] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [47] Khaled Koutini, Jan Schlüter, Hamid Eghbal-Zadeh, and Gerhard Widmer. Efficient training of audio transformers with patchout. *arXiv preprint arXiv:2110.05069*, 2021.
- [48] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.
- [49] Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. *arXiv preprint arXiv:2310.02279*, 2023.

Appendix / supplemental material

A Details of DEX-TTS

In this section, we provide further information about DEX-TTS. Specifically, we describe in detail the text encoder process and GeDEX-TTS.

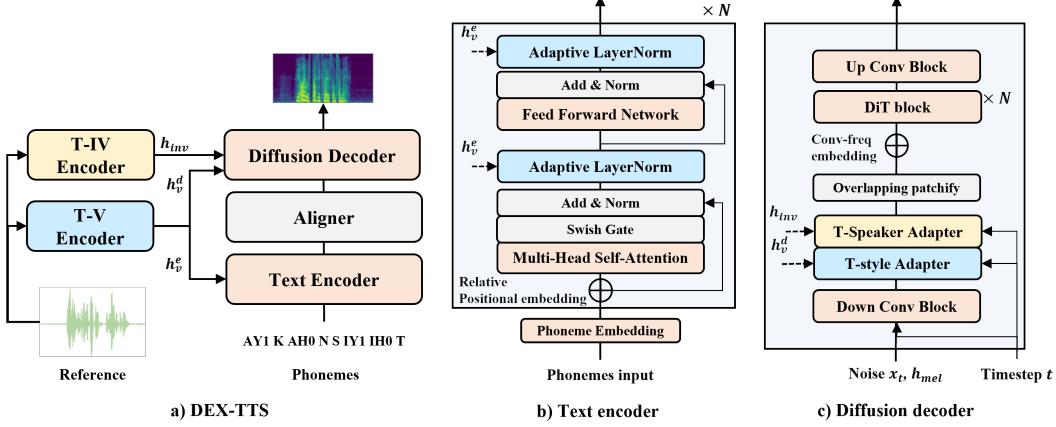


Figure 2: Overall architecture of DEX-TTS, Text encoder, and Diffusion decoder.

A.1 Text Encoder

As depicted in Figure 2, the text encoder of DEX-TTS consists of N Transformer encoder layers. We incorporate relative position embedding, RoPE [35], into the attention mechanism, and apply the swish gate used in RetNet [36] after the attention operation to improve the text encoder. In addition, AdaLN [37] is used to condition time-variant (T-V) style h_v^e .

Let $X \in \mathbb{R}^{L_p \times C}$ be the initial text representation from phonemes embedding, where L_p is the phonemes lengths and C is the hidden size. Before describing Multi-Head Self-Attention (MHSA), our self-attention mechanism with RoPE, which injects the absolute position encoding by rotations and keeps relative position by the inner product, is defined as follows:

$$Q = XW_q\Theta, \quad K = XW_k\bar{\Theta}, \quad V = XW_v, \quad \Theta_n = e^{in\theta} \quad (8)$$

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^\top / \sqrt{d_k})V$$

where $W_{q,k,v} \in \mathbb{R}^{C \times C}$ is the linear weight for the projection, and $\sqrt{d_k}$ is the scaling factor. n is the absolute position number in lengths L_p , and $\bar{\Theta}$ is the complex conjugate of Θ . Given h is the number of attention heads, we extend Equation 8 to MHSA with a swish gate as follows:

$$\begin{aligned} \text{head}_i &= \text{Attention}(Q_i, K_i, V_i) \\ Y &= GN_h([\text{head}_1; \dots; \text{head}_h]) \\ \text{MHSA}(X) &= \text{swish}(XW_g)YW_o \end{aligned} \quad (9)$$

Q_i , K_i , and V_i are used to compute each attention head, each having dimensions divided from the original dimension by the number of heads. Then, Group Normalization (GN) is applied to the concatenated heads. Lastly, we utilize a swish gate, where $W_{g,o} \in \mathbb{R}^{C \times C}$ is the linear weight for projection. Based on the MHSA, the text encoder layer processes with residual connection, layer normalization (LN), and AdaLN are defined in Equation 10.

$$\begin{aligned} Y &= \text{MHSA}(LN(X)) + X \\ Y &= \text{AdaLN}(Y, h_v^e) \\ X' &= \text{FFN}(LN(Y)) + Y \\ X' &= \text{AdaLN}(X', h_v^e) \end{aligned} \quad (10)$$

where h_v^e is T-V style from the T-V encoder, and FFN is the feed-forward network which consists of two linear weights with GeLU activation. In the experiments, we use N of 8, C of 192, and h of 2.

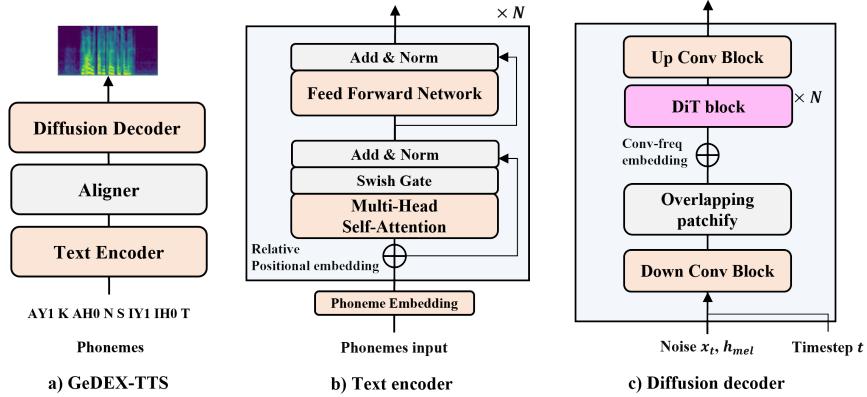


Figure 3: Overall architecture of GeDEX-TTS.

A.2 GeDEX-TTS

To verify improvements in diffusion-based TTS backbone, we designed General DEX-TTS (GeDEX-TTS) which synthesizes speech without references, and we conducted experiments in Section 4.4. To enable GeDEX-TTS to operate without a reference, we removed modules dependent on the reference. As shown in Figure 3, we removed T-IV and T-V encoders. In addition, AdaLN in the text encoder and each adapter in the diffusion decoder are removed. With the exception of the removed modules, the other components are the same as DEX-TTS.

B Additional Analysis of Experimental Results

Table 5: Comparison results on the VCTK dataset (Table 1) with error ranges.

Model	Seen scenarios				Unseen (zero-shot) scenarios			
	WER	COS	MOS-N	MOS-S	WER	COS	MOS-N	MOS-S
Ref	6.23 ± 0.58	-	3.97 ± 0.04	-	6.23 ± 0.58	-	3.97 ± 0.04	-
MetaStyleSpeech [24]	16.58 ± 1.20	78.10 ± 0.39	3.43 ± 0.05	3.63 ± 0.06	16.5 ± 1.18	73.53 ± 0.40	3.38 ± 0.06	3.30 ± 0.06
YourTTS [25]	21.27 ± 1.26	78.78 ± 0.28	3.20 ± 0.05	3.11 ± 0.06	18.34 ± 1.28	75.00 ± 0.31	3.33 ± 0.05	3.08 ± 0.06
GenerSpeech [27]	13.87 ± 1.21	77.46 ± 0.41	3.40 ± 0.05	3.25 ± 0.05	11.37 ± 1.09	73.23 ± 0.55	3.46 ± 0.05	3.06 ± 0.06
StyleTTS [26]	7.72 ± 0.72	82.93 ± 0.26	3.57 ± 0.05	3.70 ± 0.06	6.58 ± 0.72	77.90 ± 0.31	3.53 ± 0.05	3.65 ± 0.06
DEX-TTS	7.85 ± 0.73	85.31 ± 0.22	3.75 ± 0.05	3.88 ± 0.05	5.84 ± 0.61	80.45 ± 0.26	3.76 ± 0.04	3.81 ± 0.06

Table 6: Comparison results on the ESD dataset (Table 2) with error ranges.

Model	Seen scenarios				Unseen (zero-shot) scenarios			
	WER	COS	MOS-N	MOS-S	WER	COS	MOS-N	MOS-S
Ref	7.12 ± 0.68	-	3.90 ± 0.05	-	7.12 ± 0.68	-	3.90 ± 0.05	-
MetaStyleSpeech [24]	24.56 ± 1.71	79.41 ± 0.39	3.09 ± 0.05	3.53 ± 0.05	25.84 ± 1.73	73.01 ± 0.38	3.19 ± 0.05	3.34 ± 0.05
YourTTS [25]	16.57 ± 1.32	77.61 ± 0.36	3.33 ± 0.05	3.40 ± 0.05	16.35 ± 1.25	69.38 ± 0.37	3.28 ± 0.05	2.96 ± 0.05
GenerSpeech [27]	12.75 ± 1.26	75.09 ± 0.57	3.23 ± 0.05	3.28 ± 0.05	11.78 ± 1.1	70.54 ± 0.47	3.06 ± 0.05	2.78 ± 0.06
StyleTTS [26]	12.59 ± 1.08	79.65 ± 0.34	3.41 ± 0.05	3.50 ± 0.05	12.11 ± 1.17	72.27 ± 0.36	3.23 ± 0.05	3.06 ± 0.05
DEX-TTS	8.34 ± 0.95	82.71 ± 0.30	3.73 ± 0.05	3.84 ± 0.05	8.35 ± 0.93	75.58 ± 0.31	3.57 ± 0.05	3.52 ± 0.05

Table 7: Results on the LJSpeech test set (Table 4) with error ranges.

Model	WER	COS	MOS-N
GT	6.56 ± 0.55	-	4.60 ± 0.04
FastSpeech2	7.70 ± 0.57	91.31 ± 0.15	3.16 ± 0.06
Grad-TTS	7.70 ± 0.59	91.37 ± 0.18	4.16 ± 0.05
ComoSpeech	8.21 ± 0.60	91.58 ± 0.17	4.13 ± 0.05
GeDEX-TTS	6.55 ± 0.55	91.75 ± 0.16	4.26 ± 0.05

B.1 More Information about Experimental Results

To analyze the experimental results in the main text beyond the single-dimensional summaries of performance, we further present the sample errors of each evaluation metric. We provide the information for the main experimental results of expressive TTS (on the VCTK and ESD dataset - Table 1 and Table 2) and general TTS (on the LJSpeech dataset - Table 4) conducted in the main text.

As evident from the information in Tables 5, 6, and 7, the sample errors of the proposed models are generally lower, indicating higher stability than the previous methods. Furthermore, considering the magnitudes of the values, the performance improvements in the main text are sufficiently significant.

Table 8: Model complexities. We record the RTFs for each model including the vocoder (HiFi-GAN) process except for YourTTS which does not require the vocoder. The models requiring the vocoder additionally need the number of parameters for the vocoder (about 14M).

Task	Type	Model	RTF	# Params	Task	Type	Model	RTF	# Params
Expressive TTS	Non-diffusion	MetaStyleSpeech	0.034	27.67M	General TTS	Non-diffusion	FastSpeech2	0.021	34.65M
	Non-diffusion	YourTTS	0.062	94.60M		Diffusion	Grad-TTS	0.171	14.84M
	Non-diffusion	GenerSpeech	0.138	51.64M		Diffusion	ComoSpeech	0.178	14.84M
	Non-diffusion	StyleTTS	0.038	68.34M		Diffusion	GeDEX-TTS	0.163	15.04M
	Diffusion	DEX-TTS	0.297	18.36M					

Table 9: Experiment results on the VCTK and LJSpeech datasets depending on NFE. For the performance of the DEX-TTS, we take the average of seen and unseen scenarios.

Model	NFE	WER	COS	CMOS-N	RTF
DEX-TTS	10	6.72 ± 0.68	82.77 ± 0.25	-0.07 ± 0.04	0.087
	25	7.04 ± 0.70	82.84 ± 0.24	-0.06 ± 0.04	0.167
	50	6.84 ± 0.67	82.88 ± 0.24	0	0.297
GeDEX-TTS	10	6.50 ± 0.56	91.68 ± 0.15	-0.1 ± 0.06	0.044
	25	6.45 ± 0.54	91.71 ± 0.16	-0.08 ± 0.06	0.089
	50	6.55 ± 0.55	91.75 ± 0.16	0	0.164

B.2 Analysis on Model Complexities

To investigate model complexities, we record the number of model parameters and the real-time factor (RTF—the ratio between the model synthesizing time and the duration of the synthesized speech) in Table 8. RTFs are measured on a single NVIDIA 3090 GPU. As shown in Table 8, DEX-TTS requires the smallest number of parameters among the expressive TTS methods, showing superior efficiency in the parameter size. However, DEX-TTS has a higher RTF compared to the previous expressive TTS methods. This is a challenge confronted by diffusion-based TTS models, which require multiple denoising steps during speech synthesis (we discuss this further in Sections B.3 and E). On the other hand, GeDEX-TTS achieves more satisfactory results when comparing it with other diffusion-based TTS models in general TTS. GeDEX-TTS achieves the lowest RTF among diffusion-based TTS models with similar parameter sizes. The results demonstrate that our diffusion network design is effective not only for improving performance but also for enhancing inference speed.

B.3 Analysis on NFE and RTF

In this subsection, we analyze the proposed models depending on various NFEs. In Table 9, we perform evaluations on the VCTK and LJSpeech datasets for DEX-TTS and GeDEX-TTS, using NFE of 10, 25, and 50. We also conduct a comparative MOS-N (CMOS-N) test to investigate the

Table 10: Experiment results on the VCTK and ESD dataset depending on the patch size.

Dataset	Model	Seen scenarios		Unseen scenarios	
		WER	COS	WER	COS
VCTK	DEX-TTS-P2	7.85 ± 0.73	85.31 ± 0.22	5.84 ± 0.61	80.45 ± 0.26
	DEX-TTS-P4	6.82 ± 0.70	84.19 ± 0.23	6.05 ± 0.66	79.27 ± 0.27
	DEX-TTS-P8	6.93 ± 0.69	83.94 ± 0.23	5.95 ± 0.64	78.47 ± 0.28
ESD	DEX-TTS-P2	8.34 ± 0.95	82.71 ± 0.30	8.35 ± 0.93	75.58 ± 0.31
	DEX-TTS-P4	9.30 ± 1.01	82.60 ± 0.30	9.00 ± 1.01	73.02 ± 0.40
	DEX-TTS-P8	10.09 ± 1.10	81.92 ± 0.29	9.60 ± 1.03	74.04 ± 0.31

performance differences in speech naturalness depending on NFEs. The test is performed on 10 participants. The model versions with an NFE of 50 are used as references for comparison. As depicted in Table 9, the naturalness of the synthesized speech slightly decreases as NFE decreases. However, there are no significant performance differences depending on NFEs in objective measures. This indicates that, despite being a diffusion-based TTS, the proposed method can achieve excellent performance even with a small NFE. Specifically, DEX-TTS with an NFE of 10 achieves competitive performance for both TTS performance and efficiency (RTF and parameter size) compared to other expressive TTS methods.

C Further Experiments

In this section, we conduct additional experiments not covered in the main text. These experiments include adjusting the patch size and setting up zero-shot scenarios using unseen emotions to analyze the proposed methods.

C.1 Experiments depending on Patch Size

To analyze the model performance depending on the patch size P of our networks, we conduct the experiments in Table 10. We utilize P of 2, 4, and 8 for the experiments. As shown in Table 10, we observe an improvement in WER in the seen scenarios of the VCTK dataset when we use patch sizes larger than 2. However, performance degrades for the other metrics and an overall performance decrease is observed for the ESD dataset. In particular, when comparing the COS performance between patch sizes 2 and 8, performance degradation is evident with a patch size of 8 in both datasets. It indicates that DiT blocks can extract detailed representations of patches when smaller patches are used.

C.2 Experiments on the Unseen Emotion

Whereas we set zero-shot scenarios with unseen speakers for the ESD dataset in the main text, we design three zero-shot scenarios based on a few combinations of emotions in this subsection. As depicted in Table 11, the emotion columns indicate the emotion lists for each scenario. That is, the emotion list in the seen scenarios is used for training DEX-TTS. We observe that WER is similar between the emotion zero-shot and speaker zero-shot experiments (Table 2—average WER for seen and unseen scenarios is 8.34). However, in the speaker zero-shot experiment, a notable difference of 7.13 is observed in the COS performance between the seen and unseen scenarios, whereas in the emotion zero-shot experiment, the COS performance difference is only approximately 3. This result indicates that adapting to unseen speakers is more difficult than adapting to unseen emotions. In addition, the consistent performance across various emotion zero-shot scenarios suggests that DEX-TTS can adapt to diverse unseen emotions.

C.3 Experiments on the VCTK dataset for GeDEX-TTS

In the main text, we conducted experiments on the single-speaker dataset, the LJSpeech dataset, to verify our improved diffusion-based TTS model, GeDEX-TTS. We further perform experiments on the multi-speaker dataset to investigate the results. We use the VCTK dataset as the multi-speaker dataset. To enable GeDEX-TTS to synthesize speech depending on pre-defined speakers, we utilize a common technique, speaker embeddings using a lookup table, as in [15]. Table 12 shows the

Table 11: Evaluation for unseen emotions on the ESD dataset.

Model	Seen scenarios			Unseen (zero-shot) scenarios		
	Emotion	WER	COS	Emotion	WER	COS
DEX-TTS	Happy, Neutral, Surprise	9.23 ± 0.99	82.05 ± 0.30	Angry, Sad	7.71 ± 0.87	80.00 ± 0.33
DEX-TTS	Happy, Sad, Surprise	9.28 ± 0.94	82.51 ± 0.31	Angry, Neutral	7.83 ± 0.90	78.85 ± 0.36
DEX-TTS	Angry, Neutral, Sad	7.44 ± 0.89	82.80 ± 0.30	Happy, Surprise	8.53 ± 0.87	79.43 ± 0.33

Table 12: Results on the VCTK dataset for general TTS task.

Model	WER	COS	MOS-N
GT	6.23 ± 0.58	-	4.38 ± 0.04
FastSpeech2	9.80 ± 0.67	85.38 ± 0.28	3.32 ± 0.06
Grad-TTS	7.70 ± 1.00	85.97 ± 0.31	3.91 ± 0.05
ComoSpeech	14.09 ± 0.88	82.66 ± 0.25	3.64 ± 0.06
GeDEX-TTS	6.36 ± 0.55	86.14 ± 0.19	3.98 ± 0.05

comparison results on the VCTK dataset. MOS-N is recorded by evaluations of 16 participants. We observe that GeDEX-TTS outperforms the previous methods, indicating GeDEX-TTS can also synthesize high-quality speech in multi-speaker settings.

C.4 Further Comparison Results in General TTS

In the main text, we did not compare GeDEX-TTS with U-DiT-TTS since an official code is not provided. Instead, we verified our strategies by conducting experiments using various encoding types (Table 4) and patch sizes (Table 10). To further investigate the effect of our strategies, we reproduce U-DiT-TTS based on the GeDEX-TTS system. We remove the patchify and embedding strategies and utilize the large patch sizes mentioned in their study. As presented in Table 13, GeDEX-TTS outperforms our implemented U-DiT-TTS in both WER and COS across various datasets. The experimental results validate that leveraging small patch size, overlapping patchify, and conv-freq embedding strategies enables the effective utilization of DiT block.

Table 13: Comparison results with U-DiT-TTS.

Dataset	Model	WER	COS
LJSpeech	U-DiT-TTS	8.91 ± 0.61	86.76 ± 0.20
	GeDEX-TTS	6.55 ± 0.55	91.75 ± 0.16
VCTK	U-DiT-TTS	8.91 ± 0.83	83.93 ± 0.42
	GeDEX-TTS	6.36 ± 0.55	86.14 ± 0.19

D Visualizations

D.1 Style Visualizations

To further investigate the proposed model, we visualize the extracted T-IV and T-V styles. Based on the T-IV and T-V encoders, we first extract each style (h_{inv} , h_v^e , and h_v^d) from the reference speech. Since h_v^d has the time dimension, we apply a channel-wise average to h_v^d to obtain a style vector. Then, we utilize T-SNE to visualize each style. We include $h_{inv} + h_v^e$ to analyze the synergy between h_{inv} and h_v^e . We obtain $h_{inv} + h_v^e$ by concatenating h_{inv} and h_v^e . In addition, we record the distance within the cluster (DWC) and the distance between clusters (DBC) to analyze deeper.

As shown in Figure 4.a), we visualize each style depending on the emotions in the ESD dataset. h_{inv} and h_v^e show dense clusters based on emotions, indicating that both time-invariant and time-variant

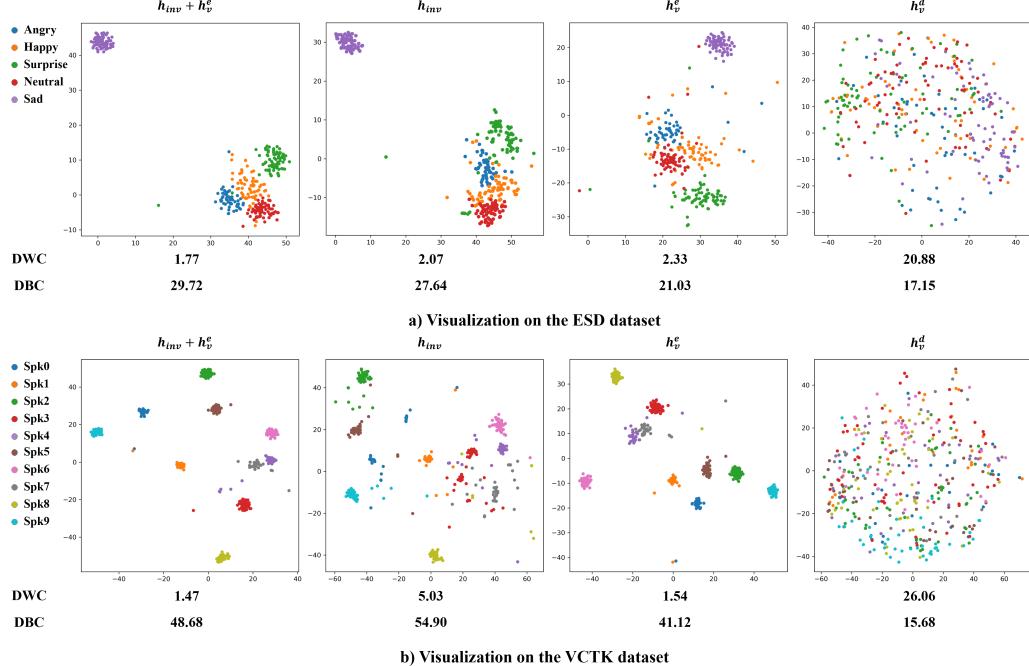


Figure 4: Style visualizations using T-SNE on the ESD and VCTK datasets. DWC and DBC indicate distance within clusters and distance between clusters. For the ESD dataset, T-SNE is used based on the five emotions of speaker 0016. For the VCTK dataset, T-SNE is applied based on unseen speakers of the dataset. DEX-TTS trained with each dataset is used for style extraction.

styles encompass emotion-related information. Furthermore, $h_{inv} + h_v^e$ provides superior clustering results with a decrease in DWC and an increase in DBC compared to h_{inv} and h_v^e . It suggests that h_{inv} and h_v^e represent distinct information and that the synergy arises when both styles are utilized. These results align with the findings of the ablation studies (Table 3), proving the necessity of both h_{inv} and h_v^e . The results for the VCTK dataset are shown in Figure 4.b). We observe similar results to those of the ESD dataset. h_{inv} and h_v^e form clusters based on speakers, while $h_{inv} + h_v^e$ yields better clustering results than h_{inv} and h_v^e through synergy. The consistent visualization results across various datasets verify that the proposed style modeling can capture emotion or speaker information without explicit labels.

Lastly, we find interesting results for h_v^d in both datasets. Unlike h_{inv} and h_v^e , h_v^d does not form clusters based on emotions or speakers. However, considering the effect on the performance of h_v^d in ablation studies (Table 3), h_v^d contains significant time-variant styles besides, speaker or emotions. It suggests that even the same types of time-variant styles (h_v^e and h_v^d) can contain different information depending on the extraction and reflection methods. In summary, the proposed style modeling method can extract diverse styles including speaker and emotional information, achieving well-represented styles for expressive TTS.

D.2 Mel-Spectrograms Visualizations

In this subsection, we plot mel-spectrograms and pitch for non-parallel samples of the ESD dataset. As shown in Figure 5, synthesized speech represents diverse styles of reference speech. In specific, DEX-TTS can follow the prosodic styles of the reference and properly represent it for a given text (see the orange lines in Figure 5). The red boxes in the plots indicate that DEX-TTS can make detailed frequency bins, similar to those of reference speech. In addition, we observe that DEX-TTS can resemble other reference styles, beyond the prosodic or detailed frequency styles. The blue boxes in Figure 5.b) demonstrate that the synthesized speech contains the intermediate pause point like reference speech. To understand the results, we provide demos of these samples at our demo site.

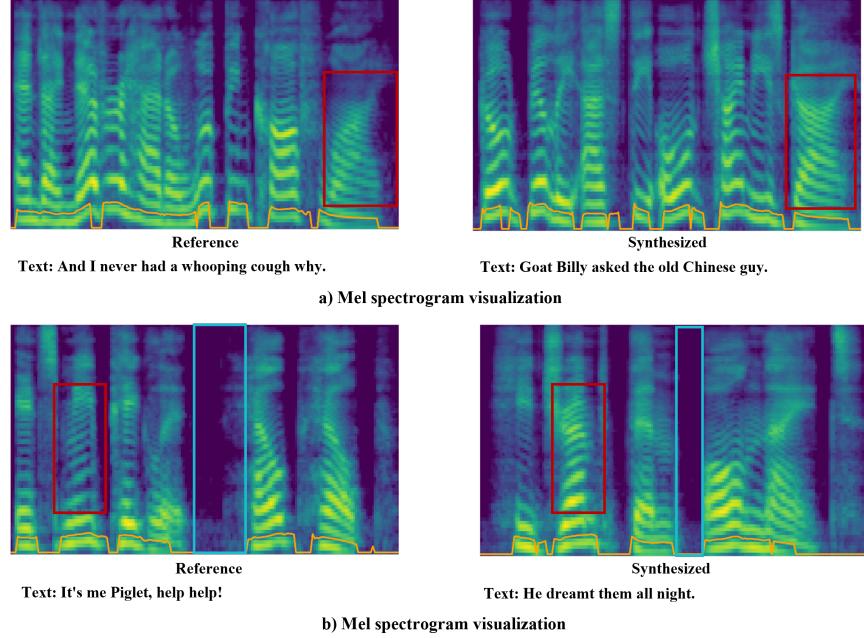


Figure 5: Visualization of mel-spectrograms for reference and synthesized speech on the ESD dataset. The orange lines indicate pitch information. Red boxes are used for comparing frequency bins and blue boxes are used for comparing pause points style.

E Limitation and Future Work

As discussed in Sections B.2 and B.3, diffusion-based TTS models generally exhibit higher RTF than non-diffusion-based TTS models since they require an iterative denoising process. Nevertheless, the proposed methods are efficient in terms of parameter size and exhibit faster inference speed compared to other diffusion-based TTS models. In addition, the proposed methods can achieve competitive performance even with fewer NFEs. Despite inspiring results with competitive RTFs, the iterative denoising process inherent in diffusion-based TTS remains a challenge.

Recent studies [48, 49] have provided suggestions to address the limitations of this study. Song et al. [48] introduced a consistency model (CM) that can map any point x_t on the ODE trajectory to its origin x_0 for generative modeling. Since the model is trained to be consistent for points on the same trajectory, these models are referred to as consistency models. By adopting distillation methods, they obtained CM and generated high-quality data with only one sampling process. CoMoSpeech [28] also adopted CM to accelerate the sampling process in diffusion-based TTS. However, as the consistency trajectory model (CTM) [49] mentioned, CM does not exhibit a speed-quality trade-off (i.e., the generation quality does not improve as NFE increases). They introduced an alternative way to bridge score-based and distillation models to accelerate the sampling process while maintaining a speed-quality trade-off. Inspired by their works, we plan to extend our study to accelerate the sampling process for the proposed diffusion-based TTS models. In future work, we will address the limitations of this study and propose diffusion-based TTS models with excellent performance and fast generation capabilities while maintaining the speed-quality trade-off.

F Subjective evaluation

We provide MOS evaluation interface (screenshot) in Figure 6. It contains an overall interface and instructions for participants.

a) Interface of MOS-N

Instructions

1. Listen to the audio samples (speech) and assess the quality of the audio based on how close it is to natural speech.
2. Please focus on the audio quality and naturalness.
3. You should not evaluate the samples about linguistic contents or grammar, or styles.
4. We recommend hearing audio samples with headphones and working in a quiet environment.

1. Listen to the sample and rate a score from 1 to 5 based on how natural the speech is.
Assign a score from 1 to 5 (Higher scores are more natural audio).
Disregard linguistic content, grammar, or styles for evaluation.

▶ 0:00 / 0:01 [audio player] ⏪ ⏴

1-Completely Unnatural
 2-Unnatural
 3-Fair
 4-Natural
 5-Completely Natural

b) Interface of MOS-S

Instructions

1. Listen to the sample and rate a score from 1 to 5 based on the similarity between audio1 and audio2.
2. Please focus on comparing the speaking style and the speaker voice for evaluation rather than audio quality. **Audio1**
3. You should not evaluate the samples about linguistic contents or grammar.
4. We recommend hearing audio samples with headphones and working in a quiet environment.

1. Listen to the sample and rate a score from 1 to 5 based on similarity between audio1 and audio2.
Higher scores are more similar. Disregard linguistic content or grammar for evaluation.

▶ 0:00 / 0:00 [audio player] ⏪ ⏴

Audio1

▶ 0:00 / 0:02 [audio player] ⏪ ⏴

1-Completely Different
 2-Different
 3-Fair
 4-Similar
 5-Completely Similar

Instructions

1. Listen to the sample and select your preference based on the naturalness and quality of the two audio.
2. Please focus on comparing the naturalness and quality of the two audio.
3. You should not evaluate the samples about linguistic contents or grammar.
4. We recommend hearing audio samples with headphones and working in a quiet environment.

1. Listen to the sample and select your preference based on the naturalness and quality of the two audio.
Disregard linguistic content or grammar for evaluation.

Audio1

▶ 0:00 / 0:00 [audio player] ⏪ ⏴

Audio2

▶ 0:00 / 0:02 [audio player] ⏪ ⏴

Audio1 is much better
 Audio1 is better
 Fair
 Audio2 is better
 Audio2 is much better

c) Interface of CMOS-N

Figure 6: Interfaces of MOS evaluations.