DiffSinger: Diffusion Acoustic Model for Singing Voice Synthesis

Preprint · May 2021

CITATIONS
0

READS
162

6 authors, including:

Chengxi Li
Zhejiang University
8 PUBLICATIONS **14** CITATIONS
SEE PROFILE

Yi Ren
Zhejiang University
55 PUBLICATIONS **288** CITATIONS
SEE PROFILE

Feiyang Chen
The Education University of Hong Kong
19 PUBLICATIONS **130** CITATIONS
SEE PROFILE

DiffSinger: Singing Voice Synthesis via Shallow Diffusion Mechanism

Jinglin Liu[†*], **Chengxi Li**[†*], **Yi Ren**[†*], **Feiyang Chen**[†], **Peng Liu**[‡], **Zhou Zhao**[†]
[†]Zhejiang University [‡]Tencent AI LAB
{jinglinliu, rayeren}@zju.edu.cn

Abstract

Singing voice synthesis (SVS) system is built to synthesize high-quality and expressive singing voice, in which the acoustic model generates the acoustic features (*e.g.*, mel-spectrogram) given a music score. Previous singing acoustic models adopt simple loss (*e.g.*, L1 and L2) or generative adversarial network (GAN) to reconstruct the acoustic features, while they suffer from over-smoothing and unstable training issues respectively, which hinder the naturalness of synthesized singing. In this work, we propose DiffSinger, an acoustic model for SVS based on the diffusion probabilistic model. DiffSinger is a parameterized Markov chain which iteratively converts the noise into mel-spectrogram conditioned on the music score. By implicitly optimizing variational bound, DiffSinger can be stably trained and generates realistic outputs. To further improve the voice quality and speed up inference, we introduce a shallow diffusion mechanism to make better use of the prior knowledge learned by the simple loss. Specifically, DiffSinger starts generation at a shallow step smaller than the total number of diffusion steps, according to the intersection of the diffusion trajectories of the ground-truth mel-spectrogram and the one predicted by a simple mel-spectrogram decoder. Besides, we train a boundary prediction network to locate the intersection and determine the shallow step adaptively. The evaluations conducted on the Chinese singing dataset demonstrate that DiffSinger outperforms state-of-the-art SVS work. Our extensional experiments also prove the generalization of DiffSinger on text-to-speech task.

1 Introduction

Singing voice synthesis (SVS) which aims to synthesize natural and expressive singing voice from musical score [39], increasingly draws attention from the research community and entertainment industries [41]. The pipeline of SVS usually consists of an acoustic model to generate the acoustic features (*e.g.*, mel-spectrogram) conditioned on a music score[2], and a vocoder to convert the acoustic features to waveform [2, 3, 14, 20, 27].

Previous singing acoustic models mainly utilize simple loss (*e.g.*, L1 or L2) to reconstruct the acoustic features. However, this optimization is based on the incorrect uni-modal distribution assumptions, leading to blurry and over-smoothing outputs. Although existing methods endeavor to solve this problem by generative adversarial network (GAN) [3, 14], training an effective GAN may occasionally fail due to the unstable discriminator. These issues hinder the naturalness of synthesized singing.

Recently, a highly flexible and tractable generative model, diffusion probabilistic model (a.k.a. diffusion model) [7, 31, 32] emerges. Diffusion model consists of two processes: diffusion process and reverse process (also called denoising process). The diffusion process is a Markov chain with

[*] Equal contribution.
[2] A music score consists of lyrics, pitch and duration.

Preprint. Under review.

fixed parameters (when using the certain parameterization in [7]), which converts the complicated data into isotropic Gaussian distribution by adding the Gaussian noise gradually; while the reverse process is a Markov chain implemented by a neural network, which learns to restore the origin data from Gaussian white noise iteratively. Diffusion model can be stably trained by implicitly optimizing variational lower bound (ELBO) on the data likelihood. It has been demonstrated that diffusion model can produce promising results in image generation [7, 32] and neural vocoder [4, 13] fields.

In this work, we propose DiffSinger, an acoustic model for SVS based on diffusion model, which converts the noise into mel-spectrogram conditioned on the music score. DiffSinger can be efficiently trained by optimizing ELBO, without adversarial feedback, and generates realistic mel-spectrograms strongly matching the ground truth distribution.

To further improve the voice quality and speed up inference, we introduce a shallow diffusion mechanism to make better use of the prior knowledge learned by the simple loss. Specifically, we find that there is an intersection of the diffusion trajectories of the ground-truth mel-spectrogram M and the one predicted by a simple mel-spectrogram decoder $\widetilde{M}$[3]: sending M and $\widetilde{M}$ into the diffusion process could result in similar distorted mel-spectrograms, when the diffusion step is big enough (but not reaches the deep step where the distorted mel-spectrograms become Gaussian white noise). Thus, in the inference stage we 1) leverage the simple mel-spectrogram decoder to generate $\widetilde{M}$; 2) calculate the sample at a shallow step k through the diffusion process: $\widetilde{M}_k$[4]; and 3) start reverse process from $\widetilde{M}_k$ rather than Gaussian white noise, and complete the process by k iteration denoising steps [7, 33, 37]. Besides, we train a boundary prediction network to locate this intersection and determine the k adaptively. The shallow diffusion mechanism provides a better start point than Gaussian white noise and alleviates the burden of the reverse process, which improves the quality of synthesized audio and accelerates the inference.

Finally, since the pipeline of SVS resembles that of text-to-speech (TTS) task, we make adjustments to DiffSinger for generalization. The contributions of this work can be summarized as follows[5]:

- We propose DiffSinger, which is the first acoustic model for SVS based on diffusion probabilistic model. DiffSinger addresses the over-smoothing and unstable training issues in previous works.
- We propose a shallow diffusion mechanism to further improve the voice quality, and accelerate the inference.
- The evaluations conducted on the Chinese singing dataset demonstrate the superiority of DiffSinger (0.11 MOS gains compared with a state-of-the-art acoustic model for SVS [39]), and the effectiveness of our novel mechanism (0.14 MOS gains, 0.5 CMOS gains and 45.1% speedup with shallow diffusion mechanism).
- The extensional experiments on text-to-speech (TTS) task proves the generalization of DiffSinger (0.24/0.23 MOS gains compared with FastSpeech 2 [25] and Glow-TTS [10] respectively).

2 Diffusion Model

In this section, we introduce the theory of diffusion probabilistic model [7, 31]. The full proof can be found in previous works [7, 13, 32].

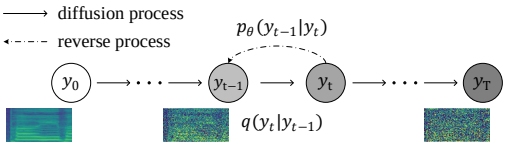


Figure 1: The directed graph for diffusion model.

Diffusion Process Define the data distribution as $q(\mathbf{y}_0)$, and sample $\mathbf{y}_0 \sim q(\mathbf{y}_0)$. The diffusion process is a Markov chain with fixed parameters [7], which converts $\mathbf{y}_0$ into the latent $\mathbf{y}_T$ in T steps:

$$q(\mathbf{y}_{1:T}|\mathbf{y}_0) \coloneqq \prod_{t=1}^{T} q(\mathbf{y}_T|\mathbf{y}_{t-1}).$$

[3]Here we use a traditional acoustic model based on Feedforward Transformer [2, 25], which is trained by L1 loss to reconstruct mel-spectrogram.

[4]$K < T$, where T is the total number of diffusion steps. $\widetilde{M}_k$ can be calculated in closed form time [7].

[5]Audio samples are available via `https://diffsinger.github.io`.

At each diffusion step $t \in [1, T]$, a tiny Gaussian noise is added to $\mathbf{y}_{t-1}$ to obtain $\mathbf{y}_t$, according to a variance schedule $\beta = \{\beta_1, \dots, \beta_T\}$:

$$q(\mathbf{y}_t|\mathbf{y}_{t-1}) \coloneqq \mathcal{N}(\mathbf{y}_t; \sqrt{1-\beta_t}\mathbf{y}_{t-1}, \beta_t\mathbf{I}).$$

If β is well designed and T is sufficiently large, then $q(y_T)$ is nearly an isotropic Gaussian distribution [7, 21]. Besides, there is a special property of diffusion process that $q(\mathbf{y}_t|\mathbf{y}_0)$ can be calculated in closed form in $O(1)$ time [7]:

$$q(\mathbf{y}_t|\mathbf{y}_0) = \mathcal{N}(\mathbf{y}_t; \sqrt{\bar{\alpha}_t}\mathbf{y}_0, (1-\bar{\alpha}_t)\mathbf{I}), \tag{1}$$

where $\bar{\alpha}_t \coloneqq \prod_{s=1}^{t} \alpha_s$, $\alpha_t \coloneqq 1 - \beta_t$.

Reverse Process The reverse process is a Markov chain with learnable parameters θ from $\mathbf{y}_T$ to $\mathbf{y}_0$. Since the exact reverse transition distribution $q(\mathbf{y}_{t-1}|\mathbf{y}_t)$ is intractable, we approximate it by a neural network with parameters θ (θ is shared at every t-th step):

$$p_\theta(\mathbf{y}_{t-1}|\mathbf{y}_t) \coloneqq \mathcal{N}(\mathbf{y}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{y}_t, t), \sigma_t^2\mathbf{I}). \tag{2}$$

Thus the whole reverse process can be defined as:

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

Training To learn the parameters θ, we minimize a variational bound of the negative log likelihood:

$$\mathbb{E}_{q(\mathbf{y}_0)}[-\log p_\theta(\mathbf{y}_0)] \geq \mathbb{E}_{q(\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_T)}[\log q(\mathbf{y}_{1:T}|\mathbf{y}_0) - \log p_\theta(\mathbf{y}_{0:T})] =: \mathbb{L}.$$

Efficient training is optimizing a random term of $\mathbb{L}$ with stochastic gradient descent [7]:

$$\mathbb{L}_{t-1} = D_{\mathrm{KL}}(q(\mathbf{y}_{t-1}|\mathbf{y}_t, \mathbf{y}_0) \parallel p_\theta(\mathbf{y}_{t-1}|\mathbf{y}_t)), \tag{3}$$

where

$$q(\mathbf{y}_{t-1}|\mathbf{y}_t, \mathbf{y}_0) = \mathcal{N}(\mathbf{y}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{y}_t, \mathbf{y}_0), \tilde{\beta}_t\mathbf{I}), \tilde{\boldsymbol{\mu}}_t(\mathbf{y}_t, \mathbf{y}_0) \coloneqq \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{y}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{y}_t,$$

where $\tilde{\beta}_t \coloneqq \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$. Eq. (3) is equivalent to:

$$\mathbb{L}_{t-1} - \mathcal{C} = \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}\|\tilde{\boldsymbol{\mu}}_t(\mathbf{y}_t, \mathbf{y}_0) - \boldsymbol{\mu}_\theta(\mathbf{y}_t, t)\|^2\right], \tag{4}$$

where $\mathcal{C}$ is a constant. And by reparameterizing Eq. (1) as $\mathbf{y}_t(\mathbf{y}_0, \boldsymbol{\epsilon}) = \sqrt{\bar{\alpha}_t}\mathbf{y}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$, and choosing the parameterization:

$$\boldsymbol{\mu}_\theta(\mathbf{y}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{y}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{y}_t, t)\right), \tag{5}$$

Eq. (4) can be simplified to:

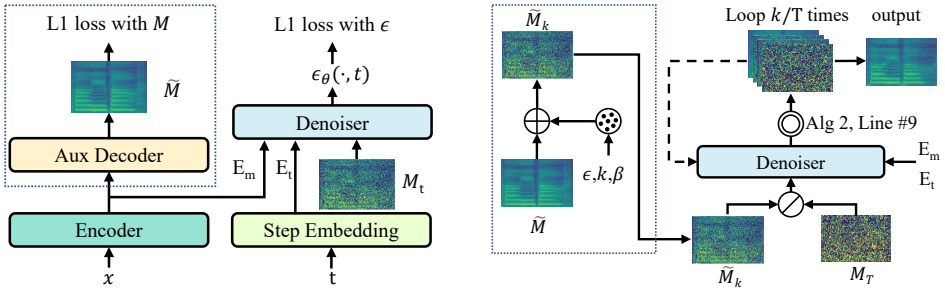
$$\mathbb{E}_{\mathbf{y}_0, \boldsymbol{\epsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{y}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right], \tag{6}$$

Finally we set σ_t^2 to $\tilde{\beta}_t$, sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\boldsymbol{\epsilon}_\theta(\cdot)$ is the outputs of the neural network.

Sampling Sample $\mathbf{y}_T$ from $p(\mathbf{y}_T) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and run the reverse process to obtain a data sample.

3 DiffSinger

As illustrated in Figure 2, DiffSinger is built on the diffusion model. Since SVS task models the conditional distribution $p_\theta(M_0|x)$, where M is the mel-spectrogram and x is the music score corresponding to M, we add x to the diffusion denoiser as the condition in the reverse process. In this section, we first describe a naive version of DiffSinger (Section 3.1); then we introduce a novel shallow diffusion mechanism to improve the model performance and efficiency (Section 3.2); finally, we describe the boundary prediction network which can adaptively find the intersection boundary required in shallow diffusion mechanism (Section 3.3).



(a) The training procedure of DiffSinger. (b) The inference procedure of DiffSinger.

Figure 2: The overview of DiffSinger (with shallow diffusion mechanism in the dotted line boxes). In subfigure (a), x is the music score; t is the step number; M means the ground truth mel-spectrogram; $\widetilde{M}$ means the blurry mel-spectrogram generated by the auxiliary decoder trained with L1 loss; M_t is M at the t-th step in the diffusion process. In subfigure (b), M_T means the M at T-th diffusion step (Gaussian white noise); k is the predicted intersection boundary; there is a switch to select M_T (naive version) or $\widetilde{M}_k$ (with shallow diffusion) as the start point of the inference procedure.

3.1 Naive Version of DiffSinger

In the naive version of DiffSinger (without dotted line boxes in Figure 2): In the training procedure (shown in Figure 2a), DiffSinger takes in the mel-spectrogram at t-th step M_t in the diffusion process and predicts the random noise $\boldsymbol{\epsilon}_\theta(\cdot)$ in Eq. (6), conditioned on t and the music score x. The inference procedure (shown in Figure 2b) starts at the Gaussian white noise sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$, as the previous diffusion models do [7, 13]. Then the procedure iterates for T times to repeatedly denoise the intermediate samples with two steps: 1) predict the $\boldsymbol{\epsilon}_\theta(\cdot)$ using the denoiser; 2) obtain M_{t-1} from M_t using the predicted $\boldsymbol{\epsilon}_\theta(\cdot)$, according to Eq. (2) and Eq. (5):

$$M_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(M_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(M_t, x, t)\right) + \sigma_t \mathbf{z},$$

where $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ when $t > 1$, and $z = 0$ when $t = 1$. Finally, a mel-spectrogram $\mathcal{M}$ corresponding to x could be generated.

3.2 Shallow Diffusion Mechanism

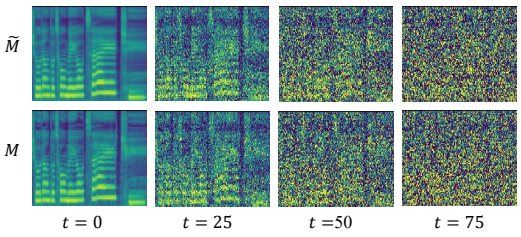


Figure 3: The mel-spectrograms at different steps in the diffusion process. The first line shows the diffusion process of mel-spectorgrams $\widetilde{M}$ generated by a simple decoder trained with L1 loss; the second line shows that of ground truth mel-spectrograms.

Although the previous acoustic model trained by the simple loss has intractable drawbacks, it still generates samples showing strong connection[6] to the ground-truth data distribution, which could provide plenty of prior knowledge to DiffSinger. To explore this connection and find a way to make better use of the prior knowledge, we conduct the empirical observation leveraging the diffusion process (shown in Figure 3): 1) when $t = 0$, M has rich details between the neighboring harmonics, which can influence the naturalness of the synthesized singing voice, but $\widetilde{M}$ is over-smoothing as we introduced in Section 1; 2) as t increases, samples of two process become indistinguishable. We illustrate this observation in Figure 4: the trajectory from $\widetilde{M}$ manifold to Gaussian noise manifold and the trajectory from M to Gaussian noise manifold intersect when the diffusion step is big enough.

Inspired by this observation, here comes the shallow diffusion mechanism: instead of starting with the Gaussian white noise, the reverse process starts at the intersection of two trajectories shown in

[6]The samples fail to maintain the variable aperiodic parameters, but they usually have a clear "skeleton" (harmonics) matching the ground truth.

Figure 4. Thus the burden of the reverse process could be alleviated distinctly[7]. Specifically, in the inference stage we 1) leverage an auxiliary decoder to generate $\widetilde{M}$, which is trained with L1 conditioned on the music score encoder outputs, as shown in the dotted line box in Figure 2a; 2) generate the intermediate sample at a shallow step k through the diffusion process, as shown in the dotted line box in Figure 2b according to Eq. (1):

$$\widetilde{M}_k(\widetilde{M}, \boldsymbol{\epsilon}) = \sqrt{\bar{\alpha}_k}\widetilde{M} + \sqrt{1-\bar{\alpha}_k}\boldsymbol{\epsilon},$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\bar{\alpha}_k := \prod_{s=1}^{k} \alpha_s$, $\alpha_k := 1 - \beta_k$. If the intersection boundary k is properly chosen, $\widetilde{M}_k$ could be regarded as M_k approximately; 3) start reverse process from $\widetilde{M}_k$, and complete the process by k iteration denoising.

The training and inference procedures with shallow diffusion mechanism are described in Algorithm 1 and 2 respectively. The theoretical proof of the intersection of two trajectories can be found in Appendix A.

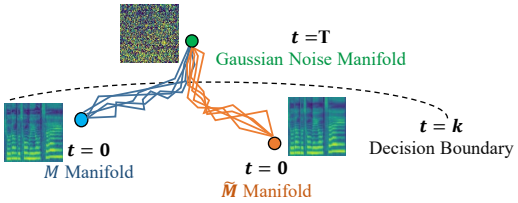


Figure 4: The diffusion trajectories of the ground-truth mel-spectrogram M and $\widetilde{M}$ predicted by the simple mel-spectrogram decoder.

3.3 Boundary Prediction

From Section 3.2, we can see that choosing a proper intersection boundary k is crucial for the shallow diffusion mechanism. Hence, we propose a boundary predictor (BP) to locate the intersection in Figure 4 and determine k adaptively. Concretely, as shown in Figure 5, BP consists of a classifier and a module for adding noise to mel-spectrograms according to Eq. (1). Given the step number $t \in [0, T]$, we label the M_t as 1 and $\widetilde{M}_t$ as 0, and use cross entropy loss to train the boundary predictor to judge whether the input mel-spectrogram at t step in diffusion process comes from M or $\widetilde{M}$. The training loss $\mathbb{L}_{BP}$ can be written as:

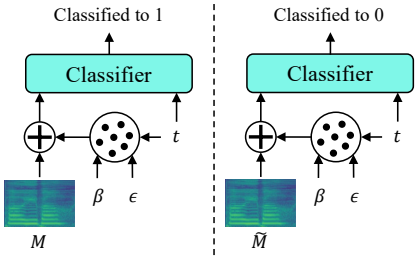


Figure 5: The boundary predictor in two cases: input M or $\widetilde{M}$.

$$\mathbb{L}_{BP} = -\mathbb{E}_{M\in\mathcal{Y}, t\in[0,T]}[\log BP(M_t, t) + \log(1 - BP(\widetilde{M}_t, t))],$$

where $\mathcal{Y}$ is the training set of mel-spectrograms. When BP have been trained, we use the predicted score of BP, which indicates the probability of a sample classified to be 1, to determine k. As shown in each subfigure of Figure 6, the curves of BP(M_t, t) and BP($\widetilde{M}_t$, t) become closer and finally converge together as the diffusion step increases, which means the classifier cannot distinguish M_t from $\widetilde{M}_t$. For every $M \in \mathcal{Y}$, we find the earliest step k' where the 95% steps t in $[k', T]$ satisfies: the margin between BP(M_t, t) and BP($\widetilde{M}_t$, t) is under the threshold. Then we choose the average of k' as the intersection boundary k.

3.4 Model Structures

Encoder The encoder encodes the music score into the condition sequence, which consists of 1) a lyrics encoder to map the phoneme ID into embedding sequence, and a series of Transformer blocks [36] to convert this sequence into linguistic hidden sequence; 2) a length regulator to expand the linguistic hidden sequence to the length of mel-spectrograms according to the duration information; and 3) a pitch encoder to map the pitch ID into pitch embedding sequence. Finally, the encoder adds linguistic sequence and pitch sequence together as the music condition sequence E_m following [27].

Step Embedding The diffusion step t is another conditional input for denoiser $\boldsymbol{\epsilon}_\theta$, as shown in Eq. (6). To convert the discrete step t to continuous hidden, we use the sinusoidal position embedding [36] followed by two linear layers to obtain step embedding E_t with C channels.

[7]Converting M_k into M_0 is easier than converting M_T (Gaussion white noise) into M_0 ($k < T$). Thus the former could improve the quality of synthesized audio and accelerates the inference.

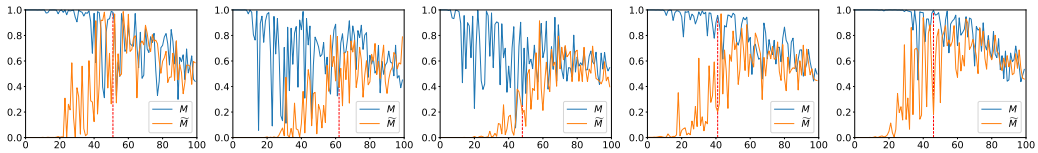


Figure 6: The visualization of the score predicted by BP of five pairs M and $\widetilde{M}$. The horizontal axis means diffusion step t, and the vertical axis exhibits the score BP($\cdot$, t).

Auxiliary Decoder We introduce a simple mel-spectrogram decoder called the auxiliary decoder, which is composed of stacked FeedForward Transformer (FFT) blocks and generates $\widetilde{M}$ as the final outputs, the same as the mel-spectrogram decoder in FastSpeech 2 [25],

Denoiser Denoiser $\boldsymbol{\epsilon}_\theta$ takes in M_t as input to predict $\boldsymbol{\epsilon}$ added in diffusion process conditioned on the step embedding E_t and music condition sequence E_m. Since diffusion model imposes no architectural constraints [13, 31], the design of denoiser has multiple choices. We adopt a non-causal WaveNet [35] architecture proposed by [13, 28] as our denoiser[8]. The denoiser is composed of a 1×1 convolution layer to project M_t with H_m channels to the input hidden sequence $\mathcal{H}$ with C channels and N convolution blocks with residual connections. Each convolution block consists of 1) an element-wise adding operation which adds E_t to $\mathcal{H}$; 2) a non-causal convolution network which converts $\mathcal{H}$ from C to $2C$ channels; 3) a 1×1 convolution layer which converts the E_m to $2C$ channels; 4) a gate unit to merge the information of input and conditions; and 5) a residual block to split the merged hidden into two branches with C channels (the residual as the following $\mathcal{H}$ and the "skip hidden" to be collected as the final results), which enables the denoiser to incorporate features at several hierarchical levels for final prediction.

Algorithm 1: Training procedure.

Input : The denoiser $\boldsymbol{\epsilon}_\theta$; the intersection boundary k; the training set $(\mathcal{X}, \mathcal{Y})$.

repeat
  Sample (x, M) from $(\mathcal{X}, \mathcal{Y})$;
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
  $t \sim \text{Uniform}(\{1, \dots, k\})$;
  Take gradient descent step on
    $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t} M + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, x, t) \right\|^2$
until *convergence*;

Algorithm 2: Inference procedure.

Input : The denoiser $\boldsymbol{\epsilon}_\theta$; the auxiliary decoder; the intersection boundary k; the source testing set $\mathcal{X}$.

Sample x from $\mathcal{X}$ as condition;
Generate $\widetilde{M}$ by the auxiliary decoder;
$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
$\widetilde{M}_k(\widetilde{M}, \boldsymbol{\epsilon}) = \sqrt{\bar{\alpha}_k}\widetilde{M} + \sqrt{1-\bar{\alpha}_k}\boldsymbol{\epsilon}$;
$M_k = \widetilde{M}_k$;
for $t = k, k-1, ..., 1$ **do**
  if $t = 1$ **then** $\mathbf{z} = \mathbf{0}$;
  else Sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
  $M_{t-1} =$
    $\frac{1}{\sqrt{\alpha_t}} \left(M_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(M_t, x, t) \right) +$
    $\sigma_t \mathbf{z}$; `// Line # 9`
end

Boundary Predictor The classifier in the boundary predictor is composed of 1) a step embedding with the same structure in Para.3.4 to provide E_t; 2) a ResNet [6] with stacked convolutional layers and a linear layer, which takes in the mel-spectrograms at t-th step and E_t to classify M_t and $\widetilde{M}_t$.

More details of model structure and configurations are shown in Appendix B.

4 Experiments

In this section, we first describe the experimental setup, and then provide the main results on SVS with analysis. Finally, we conduct the extensional experiments on TTS.

4.1 Experimental Setup

Dataset Since there is no publicly available high-quality unaccompanied singing dataset, we collect and annotate a Chinese Mandarin pop songs dataset: PopCS, to evaluate our methods. PopCS contains 127 Chinese pop songs (total ~5.95 hours with lyrics) collected from a qualified female

[8] We also tried other architectures such as Transformer or convolutional blocks and could get similar results.

vocalist. All the audio files are recorded in a recording studio. Every song is sampled at 24kHz with 16-bit quantization. To obtain more accurate music scores corresponding to the songs [14], we 1) split each whole song into sentence pieces following DeepSinger [27] and train an MFA [18] model on those sentence-level pairs to obtain the phoneme-level alignments between song piece and its corresponding lyrics; 2) extract F_0 (fundamental frequency) as pitch information from the raw waveform using Parselmouth, following [2, 27, 39]. After annotation, there are 5,498 song pieces with phoneme-level aligned music scores (lyrics, duration and pitch). These song pieces last from 0.5 seconds to 15 seconds (3.89 seconds on average). We randomly choose 275 pieces for validation and 275 pieces for testing. We will release PopCS and corresponding annotations once the paper is published.

Implementation Details We convert Chinese lyrics into phonemes by pypinyin following [27]; and extract the mel-spectrogram [30] from the raw waveform; and set the hop size and frame size to 128 and 512 in respect of the sample rate 24kHz. The size of phoneme vocabulary is 61. The number of mel bins H_m is 80. The mel-spectrograms are linearly scaled to the range [-1, 1], and F0 is normalized to have zero mean and unit variance. In the lyrics encoder, the dimension of phoneme embeddings is 256 and the Transformer blocks have the same setting as that in FastSpeech 2 [25]. In the pitch encoder, the size of the lookup table and encoded pitch embedding are set to 300 and 256. The channel size C mentioned before is set to 256. In the denoiser, the number of convolution layers N is 20 with the kernel size 3, and we set the dilation to 1 (without dilation) at each layer[9]. We set T to 100 and β to constants increasing linearly from $\beta_1 = 10^{-4}$ to $\beta_T = 0.06$. The auxiliary decoder has the same setting as the mel-spectrogram decoder in FastSpeech 2. In the boundary predictor, the number of convolutional layers is 5, and the threshold is set to 0.4 empirically.

Training and Inference The training has two stages: 1) warmup stage: separately train the auxiliary decoder for 160k steps with the music score encoder, and then leverage the auxiliary decoder to train the boundary predictor for 30k steps to obtain k; 2) main stage: training DiffSinger as Algorithm 1 describes for 160k steps until convergence. In the inference stage, for all the experiments, we uniformly use a pretrained Parallel WaveGAN (PWG) [40][10] as vocoder to transform the generated mel-spectrograms into waveforms (audio samples).

4.2 Main Results and Analysis

4.2.1 Audio Performance

To evaluate the perceptual audio quality, we conduct the MOS (mean opinion score) evaluation on the test set. Eighteen qualified listeners are asked to make judgments about the synthesized song samples in terms of three aspects: pronunciation accuracy, sound quality and naturalness [16]. We compare the MOS of the song samples generated by DiffSinger with the following systems: 1) *GT*, the ground truth singing audio; 2) *GT (Mel + PWG)*, where we first convert the ground truth singing audio to the ground truth mel-spectrograms, and then convert these mel-spectrograms back to audio using PWG vocoder described in Section 4.1; 3) *FFT-NPSS [2] (WORLD)*, the SVS system which generates WORLD vocoder features [19] through feed-forward Transformer (FFT) and uses WORLD vocoder to synthesize audio; 4) *FFT-Singer (Mel + PWG)* the SVS system which generates mel-spectrograms through FFT network and uses PWG vocoder to synthesize audio[11]; 5)

Table 1: The Mean Opinion Score (MOS) of song samples with 95% confidence intervals. DiffSinger Naive means the naive version of DiffSinger without shallow diffusion mechanism.

Method	MOS
GT	4.30 ± 0.09
GT (Mel + PWG)	4.04 ± 0.11
FFT-NPSS [2] (WORLD)	1.75 ± 0.17
FFT-Singer (Mel + PWG)	3.67 ± 0.11
GAN-Singer [39] (Mel + PWG)	3.74 ± 0.12
DiffSinger Naive (Mel + PWG)	3.71 ± 0.10
DiffSinger (Mel + PWG)	**3.85** ± 0.11

[9] Unlike the waveform generation, we do not require very long receptive field for mel-spectrogram generation.

[10] We adjust PWG to take in F0 driven source excitation [38] as additional condition, similar to that in [3], and fit the 24kHz sample rate. We train our modified PWG on PopCS.

[11] Since we found that the experiments with PWG vocoder outperformed those with WORLD vocoder or Griffin-Lim algorithm by a large margin, here we provide the results of all the systems with "*Mel + PWG*" setting for a fair comparison. Also, all the systems use the same music score condition introduced before.

GAN-Singer [39] (Mel + PWG), the SVS system with adversarial training using multiple random window discriminators.

The results are shown in Table 1. The quality of *GT (MEL + PWG)* (4.04 ± 0.11) is the upper limit of the acoustic model for SVS. *DiffSinger* outperforms the baseline system with simple training loss (*FFT-Singer*) by a large margin, and shows the superiority compared with the state-of-the-art GAN-based method (*GAN-Singer* [39]), which demonstrate the effectiveness of our method.

As shown in Figure 7, we compare the ground truth, the generated mel-spectrograms from Diffsinger, GAN-singer and FFT-Singer with the same music score. It can be seen that both Figure 7c and Figure 7b contain more delicate details between harmonics than Figure 7d does. Moreover, the performance of Diffsinger in the region of mid or low frequency is more competitive than that of GAN-singer while maintaining similar quality of the high-frequency region.

In the meanwhile, the shallow diffusion mechanism accelerates the inference of naive diffusion model by 45.1% (RTF 0.191 vs. 0.348, RTF is the real-time factor, that is the seconds it takes to generate one second of audio).

4.2.2 Ablation Studies

We conduct ablation studies to demonstrate the effectiveness of our proposed methods and some hyper-parameters studies to seek the best model configurations. We conduct CMOS evaluation for these experiments. The results of variations on DiffSinger are listed in Table 2. It can be seen that: 1) removing the shallow diffusion mechanism results in quality drop (-0.500 CMOS), which is consistent with the MOS test results and verifies the effectiveness of our shallow diffusion mechanism (row 1 vs. row 2); 2) adopting other k (row 1 vs. row 3) rather than the one predicted by our boundary predictor causes quality drop, which verifies that our boundary prediction network can predict an accurate intersection for shallow diffusion mechanism; and 3) the model with configurations $C = 256$ and $L = 20$ produces the best results (row 1 vs. row 4,5,6,7), indicating that our model capacity is sufficient.

Table 2: Variations on the *DiffSinger*. T in all the experiments is set to 100. Unlisted values are identical to those of the model in line No.1. C is channel size; L is the number of layers in denoiser; $k = 54$ is our predicted intersection boundary.

No.	C	L	k	CMOS
1	256	20	54	0.000
2				-0.500
3			25	-0.053
4	128			-0.071
5	512			-0.044
6		10		-0.293
7		30		-0.445

4.3 Extensional Experiments on TTS

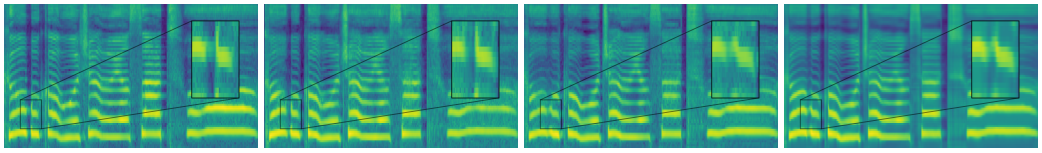
To verify the generalization of *DiffSinger* on TTS task, we conduct the extensional experiments on LJSpeech dataset [8], which contains 13,100 English audio clips (total ∼24hours) with corresponding transcripts. We follow the train-val-test dataset splits, the pre-processing of mel-spectrograms, and the grapheme-to-phoneme tool[12] in FastSpeech 2. To fit TTS task, we 1) add a pitch predictor and a duration predictor to *DiffSinger* as those in FastSpeech 2; 2) adopt $k = 70$ for shallow diffusion mechanism.

Table 3: The Mean Opinion Score (MOS) of speech samples with 95% confidence intervals.

Method	MOS
GT	4.22 ± 0.07
GT (Mel + HiFi-GAN)	4.15 ± 0.07
Tacotron 2 [30] (Mel + HiFi-GAN)	3.54 ± 0.05
BVAE-TTS [15] (Mel + HiFi-GAN)	3.48 ± 0.06
FastSpeech 2 [26] (Mel + HiFi-GAN)	3.68 ± 0.06
Glow-TTS [10] (Mel + HiFi-GAN)	3.69 ± 0.07
DiffSinger Naive (Mel + HiFi-GAN)	3.69 ± 0.05
DiffSinger (Mel + HiFi-GAN)	**3.92** ± 0.06

We used the Amazon Mechanical Turk (ten testers) to make subjective evaluation and the results are shown in Table 3. All the systems adopt HiFi-GAN [12] as vocoder. *DiffSinger* outperforms FastSpeech 2 and Glow-TTS, which demonstrates the generalization of *DiffSinger*. Besides, the last two rows in Table 3 also show the effectiveness of shallow diffusion mechanism (with 29.2% speedup, RTF 0.121 vs. 0.171).

[12]https://github.com/Kyubyong/g2p



(a) *GT* (b) *Diffsinger* (c) *GAN-singer* (d) *FFT-Singer*

Figure 7: Visualizations of mel-spectrograms in four systems: GT, DiffSinger, GAN-Singer and FFT-Singer.

5 Related Work

5.1 Singing Voice Synthesis

Initial works of singing voice synthesis generate the sounds using concatenated [9, 17] or HMM-based parametric [23, 29] methods, which are kind of cumbersome and lack flexibility and harmony. Thanks to the rapid evolution of deep learning, several SVS systems based on deep neural networks have been proposed in the past few years. [1, 5, 11, 20, 22] utilize neural networks to map the contextual features to acoustic features. [27] build the SVS system from scratch using singing data mined from music websites. [2] propose a feed-forward Transformer SVS model for fast inference and avoiding exposure bias issues caused by autoregressive models. Besides, with the help of adversarial training, [14] propose an end-to-end framework which directly generates linear-spectrograms. [39] present a multi-singer SVS system with limited available recordings and improve the voice quality by adding multiple random window discriminators. [3] introduce multi-scale adversarial training to synthesize singing with a high sampling rate (48kHz). The voice naturalness and diversity of SVS system have been continuously improved in recent years.

5.2 Denoising Diffusion Probabilistic Models

A diffusion probabilistic model is a parameterized Markov chain trained by optimizing variational lower bound, which generates samples matching the data distribution in constant steps [7]. Diffusion model is first proposed by [31]. [7] make progress of diffusion model to generate high-quality images using a certain parameterization and reveal an equivalence between diffusion model and denoising score matching [33, 34]. Most recently, [13] and [4] apply the diffusion model to neural vocoders, which generate high-fidelity waveform conditioned on mel-spectrogram. [4] also propose a continuous noise schedule to reduce the inference iterations while maintaining synthesis quality. [32] extend diffusion model by providing a faster sampling mechanism, and a way to interpolate between samples meaningfully. Diffusion model is a fresh and developing technique, which has been applied in the fields of unconditional image generation, conditional spectrogram-to-waveform generation (neural vocoder). And in our work, we propose a diffusion model for the acoustic model which generates mel-spectrogram given music scores (or text). There is also a concurrent work [24] which adjusts diffusion model as the acoustic model for TTS task.

6 Conclusion

In this work, we proposed DiffSinger, an acoustic model for SVS based on diffusion probabilistic model. To further improve the voice quality and speed up inference, we proposed a shallow diffusion mechanism. Specifically, we found that the diffusion trajectories of M and $\widetilde{M}$ converge together when the diffusion step is big enough. Inspired by this, we started the reverse process at the intersection (step k) of two trajectories rather than at the very deep diffusion step T. Thus the burden of the reverse process could be alleviated distinctly. Besides, we proposed a boundary predictor to locate the intersection and determine k adaptively. However, there exists preclusion of calculation of log-likelihoods due to this short-circuiting operation. The experiments conducted on the Chinese singing dataset demonstrate the superiority of DiffSinger compared with previous works, and the effectiveness of our novel shallow diffusion mechanism. The extensional experiments conducted on LJSpeech dataset prove the generalization of DiffSinger on TTS task.

References

[1] Merlijn Blaauw and Jordi Bonada. A neural parametric singing synthesizer modeling timbre and expression from natural songs. *Applied Sciences*, 7(12):1313, 2017.

[2] Merlijn Blaauw and Jordi Bonada. Sequence-to-sequence singing synthesis using the feed-forward transformer. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7229–7233. IEEE, 2020.

[3] Jiawei Chen, Xu Tan, Jian Luan, Tao Qin, and Tie-Yan Liu. Hifisinger: Towards high-fidelity neural singing voice synthesis. *arXiv preprint arXiv:2009.01776*, 2020.

[4] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. In *International Conference on Learning Representations*, 2021.

[5] Yu Gu, Xiang Yin, Yonghui Rao, Yuan Wan, Benlai Tang, Yang Zhang, Jitong Chen, Yuxuan Wang, and Zejun Ma. Bytesing: A chinese singing voice synthesis system using duration allocated encoder-decoder acoustic models and wavernn vocoders. *arXiv preprint arXiv:2004.11012*, 2020.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6832–6843. Curran Associates, Inc., 2020.

[8] Keith Ito. The lj speech dataset. `https://keithito.com/LJ-Speech-Dataset/`, 2017.

[9] Hideki Kenmochi and Hayato Ohshita. Vocaloid-commercial singing synthesizer based on sample concatenation. In *Eighth Annual Conference of the International Speech Communication Association*, 2007.

[10] 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, 2020.

[11] Juntae Kim, Heejin Choi, Jinuk Park, Sangjin Kim, Jongjin Kim, and Minsoo Hahn. Korean singing voice synthesis system based on an lstm recurrent neural network. In *INTERSPEECH 2018*. ISCA, 2018.

[12] 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, 2020.

[13] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021.

[14] Juheon Lee, Hyeong-Seok Choi, Chang-Bin Jeon, Junghyun Koo, and Kyogu Lee. Adversarially trained end-to-end korean singing voice synthesis system. *Proc. Interspeech 2019*, pages 2588–2592, 2019.

[15] Yoonhyung Lee, Joongbo Shin, and Kyomin Jung. Bidirectional variational inference for non-autoregressive text-to-speech. In *International Conference on Learning Representations*, 2020.

[16] Peiling Lu, Jie Wu, Jian Luan, Xu Tan, and Li Zhou. Xiaoicesing: A high-quality and integrated singing voice synthesis system. *Proc. Interspeech 2020*, pages 1306–1310, 2020.

[17] Michael Macon, Leslie Jensen-Link, E Bryan George, James Oliverio, and Mark Clements. Concatenation-based midi-to-singing voice synthesis. In *Audio Engineering Society Convention 103*. Audio Engineering Society, 1997.

[18] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal forced aligner: Trainable text-speech alignment using kaldi. In *Interspeech*, pages 498–502, 2017.

[19] Masanori Morise, Fumiya Yokomori, and Kenji Ozawa. World: a vocoder-based high-quality speech synthesis system for real-time applications. *IEICE TRANSACTIONS on Information and Systems*, 99(7):1877–1884, 2016.

[20] Kazuhiro Nakamura, Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku, and Keiichi Tokuda. Singing voice synthesis based on convolutional neural networks. *arXiv preprint arXiv:1904.06868*, 2019.

[21] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models, 2021.

[22] Masanari Nishimura, Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku, and Keiichi Tokuda. Singing voice synthesis based on deep neural networks. In *Interspeech*, pages 2478–2482, 2016.

[23] Keiichiro Oura, Ayami Mase, Tomohiko Yamada, Satoru Muto, Yoshihiko Nankaku, and Keiichi Tokuda. Recent development of the hmm-based singing voice synthesis system—sinsy. In *Seventh ISCA Workshop on Speech Synthesis*, 2010.

[24] Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail Kudinov. Grad-tts: A diffusion probabilistic model for text-to-speech. *arXiv preprint arXiv:2105.06337*, 2021.

[25] 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. In *International Conference on Learning Representations*, 2021.

[26] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech: Fast, robust and controllable text to speech. In *Advances in Neural Information Processing Systems*, pages 3171–3180, 2019.

[27] Yi Ren, Xu Tan, Tao Qin, Jian Luan, Zhou Zhao, and Tie-Yan Liu. Deepsinger: Singing voice synthesis with data mined from the web. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1979–1989, 2020.

[28] Dario Rethage, Jordi Pons, and Xavier Serra. A wavenet for speech denoising. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5069–5073. IEEE, 2018.

[29] Keijiro Saino, Heiga Zen, Yoshihiko Nankaku, Akinobu Lee, and Keiichi Tokuda. An hmm-based singing voice synthesis system. In *Ninth International Conference on Spoken Language Processing*, 2006.

[30] 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 *ICASSP 2018*, pages 4779–4783. IEEE, 2018.

[31] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265, 2015.

[32] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[33] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, 2019.

[34] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[35] Aäron 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. In *9th ISCA Speech Synthesis Workshop*, pages 125–125.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[37] Pascal Vincent. A connection between score matching and denoising autoencoders. In *Neural Computation*, 2011.

[38] Xin Wang and Junichi Yamagishi. Using cyclic noise as the source signal for neural source-filter-based speech waveform model. *Proc. Interspeech 2020*, pages 1992–1996, 2020.

[39] Jie Wu and Jian Luan. Adversarially trained multi-singer sequence-to-sequence singing synthesizer. *Proc. Interspeech 2020*, pages 1296–1300, 2020.

[40] Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel wavegan: 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.

[41] Liqiang Zhang, Chengzhu Yu, Heng Lu, Chao Weng, Chunlei Zhang, Yusong Wu, Xiang Xie, Zijin Li, and Dong Yu. Durian-sc: Duration informed attention network based singing voice conversion system. *Proc. Interspeech 2020*, pages 1231–1235, 2020.

A Theoretical Proof of Intersection

Given a data sample M_0 and its corresponding $\widetilde{M}_0$, the conditional distributions of M_t and $\widetilde{M}_t$ are:

$$q(M_t|M_0) = \mathcal{N}(M_t; \sqrt{\bar{\alpha}_t}M_0, (1-\bar{\alpha}_t)\mathbf{I})$$
$$q(\widetilde{M}_t|\widetilde{M}_0) = \mathcal{N}(\widetilde{M}_t; \sqrt{\bar{\alpha}_t}\widetilde{M}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

respectively. The KL-divergence between two Gaussian distributions is:

$$D_{\mathrm{KL}}\left(\mathcal{N}_0 \| \mathcal{N}_1\right) = \frac{1}{2}[\mathrm{tr}\left(\Sigma_1^{-1}\Sigma_0\right) + \left(\mu_1-\mu_0\right)^{\top}\Sigma_1^{-1}\left(\mu_1-\mu_0\right) - k + \ln(\frac{\det\Sigma_1}{\det\Sigma_0})],$$

where k is the dimension; μ_0, μ_1 are means; Σ_0, Σ_1 are covariance matrices. Thus, in our case:

$$D_{KL}(\mathcal{N}(M_t)||\mathcal{N}(\widetilde{M}_t)) = \frac{\bar{\alpha}_t}{2(1-\bar{\alpha}_t)}\|\widetilde{M}_0 - M_0\|_2^2$$

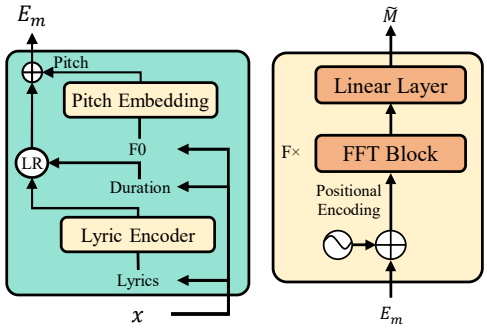
Since $\frac{\bar{\alpha}_t}{2(1-\bar{\alpha}_t)}$ decreases towards 0 rapidly as t increases, this KL-divergence also decreases towards 0 rapidly as t increases. This guarantees the intersection of trajectories of the diffusion process.

Moreover, since the auxiliary decoder has been optimized by simple reconstruction loss (L1/L2 mentioned in the main paper) on the training set, $\|\widetilde{M}_0 - M_0\|_2^2$ is usually smaller than 1 (average 0.06 on the validation set), which facilitates this intersection. In addition, $\widetilde{M}_k$ does not need to be exactly the same as M_k, but just needs to come from vicinity of the mode of $q(M_k|M_0)$ (according to the theories of score matching and Langevin dynamics).

B Details of Model Structure and Supplementary Configurations

B.1 Details of model structure

The detailed model structure of encoder, auxiliary decoder and denoiser are shown in Figure 8a, Figure 8b and Figure 9 respectively.



(a) Encoder. (b) Auxiliary Decoder.

Figure 8: The detailed model structure of Encoder and Auxiliary Decoder. x is the music score. E_m is the music condition sequence. $\widetilde{M}$ means the blurry mel-spectrogram generated by the auxiliary decoder trained with L1 loss.

B.2 Supplementary configurations

In each FFT block: the number of FFT layers (F in Figure 8b) is set to 4; the hidden size of self-attention layer is 256; the number of attention heads is 2; the kernel sizes of 1D-convolution in the 2-layer convolutional layers are set to 9 and 1.

C Model Size

The model footprints of main systems for comparison in our paper are shown in Table 4. It can be seen that DiffSinger has the similar learnable parameters as other state-of-the-art models.

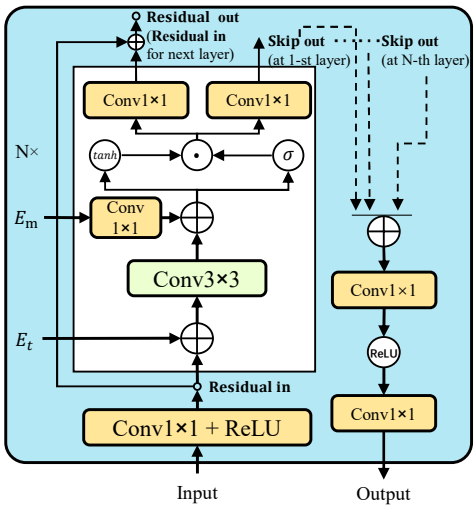


Figure 9: The detailed model structure of Denoiser. E_t is step embedding and E_m is music condition sequence. N is the number of residual layers. The model structure is derived from non-causal WaveNet, but simplified by replacing dilation layer to naive convolution layer.

Model	**Param(M)**
SVS Models	
DiffSinger	26.744
FFT-Singer	24.254
GAN-Singer	24.254 (Generator) 0.963 (Discriminator)
TTS Models	
DiffSinger	27.722
Tacotron 2	28.193
BVAE-TTS	15.991
FastSpeech 2	24.179
Glow-TTS	28.589

Table 4: The model footprints. Param means the learnable parameters.

D Details of Training and Inference

We train DiffSinger on 1 NVIDIA V100 GPU with 48 batch size. We adopt the Adam optimizer with learning rate $lr = 10^{-3}$. During training, the warmup stage costs about 16 hours and the main stage costs about 12 hours; During inference, the RTF of acoustic model for SVS and TTS are 0.191 and 0.121 respectively.