Diffsound: Discrete Diffusion Model for Text-to-Sound Generation

Dongchao Yang, Jianwei Yu, Helin Wang, Wen Wang, Chao Weng, Yuexian Zou, *Senior Member, IEEE*, and Dong Yu, *Fellow, IEEE*

***Abstract*—Generating sound effects that people want is an important topic. However, there are limited studies in this area for sound generation. In this study, we investigate generating sound conditioned on a text prompt and propose a novel text-to-sound generation framework that consists of a text encoder, a Vector Quantized Variational Autoencoder (VQ-VAE), a token-decoder, and a vocoder. The framework first uses the token-decoder to transfer the text features extracted from the text encoder to a mel-spectrogram with the help of VQ-VAE, and then the vocoder is used to transform the generated mel-spectrogram into a waveform. We found that the token-decoder significantly influences the generation performance. Thus, we focus on designing a good token-decoder in this study. We begin with the traditional autoregressive (AR) token-decoder. However, the AR token-decoder always predicts the mel-spectrogram tokens one by one in order, which may introduce the unidirectional bias and accumulation of errors problems. Moreover, with the AR token-decoder, the sound generation time increases linearly with the sound duration. To overcome the shortcomings introduced by AR token-decoders, we propose a non-autoregressive token-decoder based on the discrete diffusion model, named Diffsound. Specifically, the Diffsound model predicts all of the mel-spectrogram tokens in one step and then refines the predicted tokens in the next step, so the best-predicted results can be obtained by iteration. Our experiments show that our proposed Diffsound model not only produces better generation results when compared with the AR token-decoder but also has a faster generation speed, *i.e.*, MOS: 3.56 *v.s* 2.786.**

***Index Terms*—Autoregressive model, diffusion model, text-to-sound generation, vocoder.**

I. Introduction

USER controlled sound generation has a lot of potential applications, such as movie and music productions, game scene sound effects, and so on. With the development of virtual reality (VR) technology, it is very important to generate the sound effects that users want. Research on sound generation is very limited. Chen et al. [1], Zhou et al. [2] and Iashin and Rahtu [3] proposed to generate sound related to a video. Liu et al. [4] and Kong et al. [5] attempted to generate environmental sound conditioned on a one-hot label. However, at the time of this work, there are very limited published works on generating sound from text descriptions. To the best of our knowledge, this paper is among the first work in this direction. Text-to-sound generation has a wide range of applications, *e.g.*, adding background sound for speech synthesis systems. Nowadays, speech synthesis systems have been applied to poetry or novel reading. The user experience could be improved by adding background sound to scenarios represented in text. Furthermore, many music or movie designers are required to find a suitable sound for a scene. A simple approach is that they describe the scene with a sentence, and then use the text-to-sound model to generate the corresponding sound. In this work, we focus on directly generating audio based on human-written descriptions, such as "An audience cheers and applauds while a man talks". The state-of-the-art methods [3], [4] in the sound generation both employ a two-stage generation strategy, which first uses autoregressive (AR) decoder to generate a mel-spectrogram conditioned on a one-hot label or a video and then employs a vocoder (*e.g.* MelGAN [6]) to transform the generated mel-spectrogram into waveform. To improve the generation efficiency, they propose to learn a prior in the form of the Vector Quantized Variational Autoencoder (VQ-VAE) codebook [7], which aims to compress the mel-spectrogram into a token sequence. With VQ-VAE, the mel-spectrogram generation problem can be formulated as predicting a sequence of discrete tokens from the text inputs. Inspired by [3], [4], we propose a text-to-sound generation framework, which consists of a text encoder, a VQ-VAE, a token-decoder, and a vocoder. The diagram of the text-to-sound framework is shown in Fig. 1. We found that the token-decoder significantly influences the generation performance. Thus, we focus on designing a good token-decoder in this paper.

We start by looking into the AR token-decoder. However, we discovered that the AR token-decoder is unable to produce high-fidelity and high-relevance sound with text input. Though the AR token-decoder has been widely adopted in sound generation tasks in previous research [3], [4], it has two flaws: (1) Mel-spectrogram tokens are always predicted in order (*e.g.*, from left to right) by the AR token-decoder. Such unidirectional predictions may restrict the sound generation performance to

Manuscript received 18 June 2022; revised 17 February 2023; accepted 11 April 2023. Date of publication 28 April 2023; date of current version 5 May 2023. This work was supported in part by Shenzhen S&T Research Program under Grant GXWD20201231165807007-20200814115301001. This work was done when Dongchao Yang was an Intern at Tencent AI Lab. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Stefan Goetze. *(Corresponding authors: Yuexian Zou; Jianwei Yu.)*

Dongchao Yang, Helin Wang, Wen Wang, and Yuexian Zou are with the Advanced Data and Signal Processing Laboratory and School of Electronic and Computer Engineering, Peking University, Beijing 100871, China (e-mail: 2001212832@stu.pku.edu.cn; hwang258@jh.edu; 2101212827@stu.pku.edu.cn; zouyx@pku.edu.cn).

Jianwei Yu, Chao Weng, and Dong Yu are with the Tencent AI Lab, Bellevue, WA 98004 USA (e-mail: tomasyu@tencent.com; cweng@tencent.com; dyu@tencent.com).

Code, pre-trained models, and generated samples are released. http://dongchaoyang.top/text-to-sound-synthesis-demo/.

Digital Object Identifier 10.1109/TASLP.2023.3268730

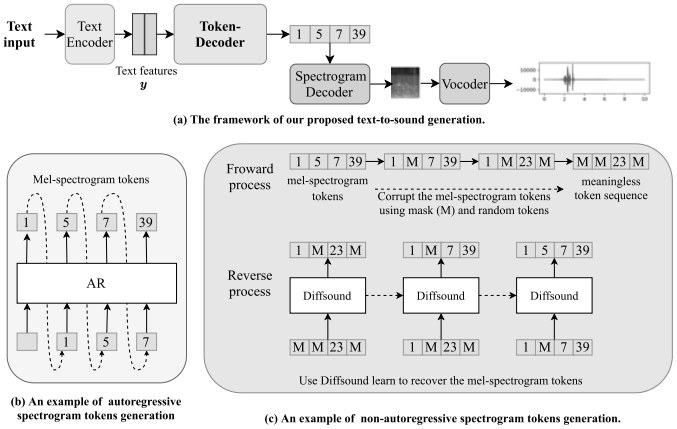

Fig. 1. (a) shows the diagram of the text-to-sound generation framework includes four parts: a text encoder that extracts text features from the text input, a token-decoder that generates mel-spectrogram tokens, a pre-trained spectrorgam decoder that transforms the tokens into mel-spectrogram, and a vocoder that transforms the generated mel-spectrogram into waveform. We explore two kinds of token-decoders, an autoregressive (AR) token-decoder and a non-autoregressive token-decoder (Diffsound). (b) and (c) show the examples of AR token-decoder and non-AR token-decoder.

be sub-optimal since the information of a specific sound event location may come from both the left and the right context; (2) During the inference phase, incorrectly predicted tokens from previous steps propagate to subsequent tokens, resulting in accumulated prediction errors. Another issue in the text-to-sound generation is lacking text-audio pairs. The largest public available text-audio dataset is AudioCaps [8], which only includes about 49 K text-audio samples. In contrast, Iashin and Rahtu [3] trains their model using VGGSound dataset [9], which has over 200 K audio-video pairs.

To address the weaknesses of the AR token-decoder, we propose a non-autoregressive token-decoder based on diffusion probabilistic models (diffusion models for short) [10], [11], [12], [13], named Diffsound. Instead of predicting the mel-spectrogram tokens one by one in order, Diffsound model predicts all of the mel-spectrogram tokens simultaneously, then it revises the previous predicted results in the following steps, so that the best results can be obtained by iterations. In each step, the Diffsound model leverages the contextual information of all tokens predicted in the previous step to estimate a new probability density distribution and uses this distribution to sample the tokens in the current step. Due to the fact that Diffsound model can make use of the contextual information of all tokens and revise any token in each step, we speculate that it can effectively alleviate the unidirectional bias and the accumulated prediction error problems. We adopt the idea from diffusion models, which use a forward process to corrupt the original mel-spectrogram tokens in T steps, and then let the model learn to recover the original tokens in a reverse process. Specifically, in the forward process, we define a transition matrix that denotes probability of each token transfer to a random token or a pre-defined MASK token. By using the transition matrix, the original tokens $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$ transfer into a stationary distribution $p(\boldsymbol{x}_T)$. In the reverse process, we let the network learn to recover the original tokens from $\boldsymbol{x}_T \sim p(\boldsymbol{x}_T)$ conditioned on the text features. Fig. 1(c) shows an example of non-autoregressive mel-spectrogram tokens generation.

To address the problem of lacking text-audio pairs, we propose to let the Diffsound model learn knowledge from the AudioSet dataset [14] and then fine-tune the pre-trained Diffsound model on a small-scale text-audio dataset (*e.g.*, AudioCaps). AudioSet is the largest available dataset in the audio field, but it only provides the event labels for each audio clip. To utilize the AudioSet dataset, we propose a mask-based text generation strategy (MBTG) that can generate a text description according to the event labels so that a new text-audio dataset is built. Furthermore, we observe a phenomenon: it is easier to generate audio that only includes one single event than audio that includes multiple events. To help the Diffsound model learn better, we mimic the human learning process by letting the Diffsound model learn from easy clips and gradually advance to complex clips and knowledge. Specifically, we propose a curriculum learning strategy in our pre-training stage, that is, we first select the audio clips that only include one event (easy sample) to the training set, and gradually add the audio clips that include multiple events (hard sample) to the training set.

Human evaluation of sound generation models is an expensive and tedious procedure. Thus, objective evaluation metrics are necessary for sound generation tasks. We explore three objective evaluation metrics: Frechet Inception Distance (FID) [15], KL-divergence [3] and audio caption loss. We demonstrate that these metrics can effectively evaluate the fidelity and relevance of the generated sound. Furthermore, we also use the Mean Opinion Score (MOS) to assess our methods.

Experiments show that our text-to-sound generation framework can generate high-quality sound, *e.g.*, MOS: 3.56 (ours) *v.s* 4.11 (ground truth), and our proposed Diffsound model has better generation performance and speed compared to the AR token-decoder, *e.g.*, MOS: 3.56 *v.s* 2.786, and the generation speed is five times faster than the AR token-decoder. Our main contributions are listed as follows:

1) For the first time, we investigate how to generate sound based on text descriptions and offer a text-to-sound generation framework. Furthermore, we propose a novel token-decoder (Diffsound) based on a discrete diffusion model that outperforms the AR token-decoder in terms of generation performance and speed.
2) To solve the problem of lacking text-audio pairs in the text-to-sound generation task, we propose a mask-based text generation strategy (MBTG), which helps build a large-scale text-audio dataset based on the AudioSet dataset. We demonstrate the effectiveness of pre-training the Diffsound on the AudioSet, which gives the insight to improve the performance of the generation model under a data deficiency situation.
3) We initially explore three objective evaluation metrics for the text-to-sound generation task. We show that these metrics can comprehensively evaluate the generated sound's relevance (*i.e.*, how well the audio content relates to the text descriptions), as well as its fidelity (*i.e.*, how

closely the generated audio clip resembles actual environmental sound).

II. Related Work

A. GAN-Based Content Generation

In the past few years, Generative Adversarial Networks (GANs) have shown promising results in image generation [16], [17], [18], speech synthesis [6], [19], [20] and music generation [21]. GAN-based models are capable of synthesizing high-fidelity images/sounds. However, they suffer from well-known training instability and mode collapse issues, which lead to a lack of sample diversity. Most related to our work is RegNet [1], which aims to generate sound conditioned by visual information. It is noted that RegNet only generates sounds on a single domain dataset (*e.g.*, dog bark or drum), which means that it struggles on complex scenes with multiple sound events. Furthermore, Iashin and Rahtu [3] have demonstrated that using an autoregressive generation model can produce better results than RegNet.

B. Autoregressive Models

AR models [22], [23] have shown powerful generation capability and have been applied for image generation [7], [24], [25], [26], [27], [28], [29], speech synthesis [30] and sound generation [3], [4]. To generate high-resolution images, VQ-VAE [7], [27], VQGAN [24] and ImageBART [31] train an encoder to compress the image into a low-dimensional discrete latent space. After that, the AR models learn from low-dimensional discrete latent space directly, which greatly reduces the time complexity and improves the performance. Liu et al. [4] and Iashin and Rahtu [3] also apply the similar idea to generate sound, and achieve good generation performance.

C. Diffusion Probabilistic Models

Diffusion generative models were first proposed in [11] and achieved strong results on image generation [13], [31], [32], [33] and speech synthesis [34], [35], [36], [37]. Diffusion models with discrete state spaces were first introduced by Sohl-Dickstein et al. [11], who considered a diffusion process over binary random variables. Hoogeboom et al. [38] extended the model to categorical random variables with transition matrices characterized by uniform transition probabilities. Jacob et al. propose Discrete Denoising Diffusion Probabilistic Models (D3PMs) [12], which further improve and extend discrete diffusion models by using a more structured categorical corruption process to corrupt the forward process. D3PMs [12] and VQ-Diffusion [13] have applied discrete diffusion models to image generation.

III. Proposed Text-to-Sound Framework

The overall architecture of the proposed text-to-sound framework is demonstrated in Fig. 1(a), which consists of four parts including a text encoder, a VQ-VAE, a token-decoder, and a vocoder. The detailed design of each part will be introduced in this section.

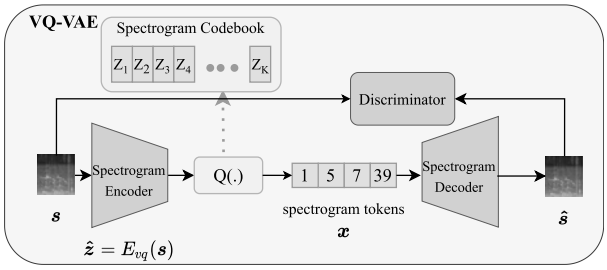

Fig. 2. The overall architecture of VQ-VAE, which consists of four parts: an spectrogram encoder that extracts the representation $\hat{\boldsymbol{z}}$ from the mel-spectrogram, a codebook that contains a finite number of embedding vectors, a spectrogram decoder that reconstructs the mel-spectrogram based on mel-spectrogram tokens, and a discriminator that distinguishes the mel-spectrogram is original or reconstructed. $Q(.)$ denotes a quantizer that maps each features $\hat{\boldsymbol{z}}_{ij}$ into its closest codebook entry $\boldsymbol{z}_k$ to obtain the mel-spectrogram tokens.

A. Text Encoder

The first step in the text-to-sound generation task is designing a good text encoder to extract the sound event information from the context while other information should be excluded. In this study, we employed the pretrained BERT [39] and the text encoder of a pretrained Contrastive Language-Image Pre-Training (CLIP) model [40] to extract the text features (a vector to represent the contextual information). Our experiments indicated that using CLIP model can bring better generation performance. One possible explanation is that CLIP is trained by contrastive learning between the representations of images and text, the use of multi-modality information may make the text representations computed from CLIP contain more semantics related to sound events *e.g.*, "dog barks and birds sing". Note that the text encoder is fixed in our training process.

B. Learning Discrete Latent Space of Mel-Spectrograms via VQ-VAE

In this part, we introduce the vector quantized variational autoencoder (VQ-VAE) [7] to simplify the process of decoder generates the mel-spectrograms.

Most of the text-to-speech (TTS) methods [34], [35], [41] directly learn the mapping from text to wave samples or raw spectrogram pixels for the reason that the synthesized speech's content relies on the words of text. Unlike TTS, there is no direct correspondence between text and sound in the sound generation task. To this end, the text-to-sound task needs to extract the event information from the text input and then generate the corresponding events. Considering a sound may consist of multiple events and each event has its own unique characteristics. We propose to use the VQ-VAE to learn a codebook to encode the characteristic of events, and then generate the mel-spectrogram based on the codebook following Liu et al. [4] and Iashin and Rahtu [3].

As Fig. 2 shows, a mel-spectrogram can be approximated by a sequence of mel-spectrogram tokens. Thus, the mel-spectrogram generation problem transfers to predicting a sequence of tokens. In the following, we will introduce the details of VQ-VAE.

VQ-VAE is trained to approximate an input using a compressed intermediate representation, retrieved from a discrete codebook. VQ-VAE consists of a spectrogram encoder E_{vq}, a spectrogram decoder D_s and a codebook $\boldsymbol{Z} = \{\boldsymbol{z}_k\}_{k=1}^{K} \in \mathbb{R}^{K \times n_z}$ containing a finite number of embedding vectors, where K is the size of the codebook and n_z is the dimension of codes. Given a mel-spectrogram $\boldsymbol{s} \in \mathbb{R}^{F \times L}$, the input $\boldsymbol{s}$ is firstly encoded into a small-scale representation (encoder E_{vq} consists of multiple convolution and pooling layers) $\hat{\boldsymbol{z}} = E_{vq}(\boldsymbol{s}) \in \mathbb{R}^{F' \times L' \times n_z}$, where F', L' and n_z represent the down-sampled frequency, the time dimension and the feature dimension. Then we use a vector quantizer $Q(\cdot)$ which maps each time-frequency vector $\hat{\boldsymbol{z}}_{ij} \in \mathbb{R}^{n_z}$ into its closest codebook entry $\boldsymbol{z}_k$ to obtain a discrete spectrogram token sequence $\boldsymbol{x} \in \mathbb{Z}^{F' \times L'}$ as follows:

$$\boldsymbol{x} = Q\left(\hat{\boldsymbol{z}}\right) := \left(\underset{\boldsymbol{z}_k \in \boldsymbol{Z}}{\arg\min} ||\hat{\boldsymbol{z}}_{ij} - \boldsymbol{z}_k||_2^2 \text{ for all } (i,j) \text{ in } (F', L') \right) \tag{1}$$

The mel-spectrogram can be approximately reconstructed via the codebook $\boldsymbol{Z}$ and the spectrogram decoder i.e., $\hat{\boldsymbol{s}} = D_s(Q(\hat{\boldsymbol{z}}))$. Note that the spectrogram tokens are quantized latent variables in the sense that they take discrete values. The encoder E_{vq}, the spectrogram decoder D_s, and the codebook $\boldsymbol{Z}$ can be trained in an end-to-end manner via the following loss function:

$$\mathcal{L}_{VQVAE} = ||\boldsymbol{s} - \hat{\boldsymbol{s}}||_1 + ||\mathcal{SG}[E_{vq}(\boldsymbol{s})] - \boldsymbol{x}||_2^2 \tag{2}$$

$$+ ||\mathcal{SG}[\boldsymbol{x}] - E_{vq}(\boldsymbol{s})||_2^2 \tag{3}$$

where $\mathcal{SG}[\cdot]$ is the stop-gradient operation that acts as an identity during the forward pass but has zero gradients at the backward pass. To preserve the reconstruction quality when upsampled from a smaller-scale representation, we follow the setting of VQGAN [24], which adds a patch-based adversarial loss [42] to the final training loss

$$\mathcal{L}_f = \mathcal{L}_{VQVAE} + \lambda_d(\log(D(\boldsymbol{s})) + \log(1 - D(\hat{\boldsymbol{s}}))) \tag{4}$$

where D is a discriminator (it consists of several convolution layers), which aims to distinguish the mel-spectrogram is original or reconstructed. λ_d is a hyper-parameter to control the weight of adversarial loss.

C. Token-Decoder

The token-decoder in our framework is used to transfer the text features into the discrete mel-spectrogram token sequence. An autoregressive token-decoder is first investigated in this paper.

Specifically, given the text-audio pairs, the inputs of the token encoder are extracted from the text description with the text encoder. Following [3], the discrete mel-spectrogram tokens $\boldsymbol{x} \in \mathbb{Z}^{F' \times L'}$ obtained from a pretrained VQ-VAE are first reshaped to a token sequence $\hat{\boldsymbol{x}} \in \mathbb{Z}^{1 \times F'L'}$ and then used as the training target. By using the AR token-decoder, the decoding process can be viewed as an autoregressive next-token prediction: Given tokens $\hat{x}_{<i}$, the decoder learns to predict the distribution of possible next tokens, *i.e.*, $p(\hat{x}_i|\hat{\boldsymbol{x}}_{<i}, \boldsymbol{y})$ to compute the likelihood of the full representation as $p(\hat{\boldsymbol{x}}|\boldsymbol{y}) = \prod_i p(\hat{x}_i|\hat{\boldsymbol{x}}_{<i}, \boldsymbol{y})$. The decoder is trained with a cross-entropy (CE) loss, comparing the probabilities of the predicted mel-spectrogram tokens to those obtained from the ground truth. Due to the wrongly predicted results of previous steps influencing the current step, "teacher-forcing" strategy [31] is used to guarantee the stability of training. The "teacher-forcing" strategy uses ground truth tokens as previous step prediction results. After the token decoding process, the mel-spectrogram can be computed by the pretrained VQ-VAE spectrogram decoder. In the inference stage, we can set L' to determine the duration of the generated sound.

In the AR token-decoder, the adoption of the "teacher-forcing" training strategy can cause a mismatch between model training and inference. During training, we use the previous ground truth tokens to predict the current token, while during inference, we use the predicted tokens. The accuracy of the predicted tokens can be affected by the "accumulated errors" in previous predicted tokens in inference [43]. Such mismatches can cause the model's performance to be sub-optimal. In addition, the prediction of the current token only depends on the previous tokens in the AR decoder, which ignores the future context information. Such "unidirectional bias" can also lead to sub-optimal model performance. To this end, a non-autoregressive token-decoder based on a discrete diffusion model is proposed. (Details will be given in Section IV.)

D. Vocoder

The vocoder aims at transforming the generated mel-spectrogram into waveform $\hat{w}$. This type of vocoder is a hot research topic. Griffin-Lim [44], WaveNet [45], MelGAN [6], and HiFi-GAN [46] are very popular vocoders for speech synthesis task. The Griffin-Lim method is a classic signal processing method that is very fast and easy to implement. However, Griffin-Lim produces low-fidelity results when operating on mel-spectrograms [3]. WaveNet provides high-quality results but remains relatively slow in generation time. In this study, considering its generation efficiency and quality, we employ MelGAN which is a non-autoregressive approach to reconstructing the waveform. MelGAN has been widely used in speech synthesis fields. However, many pre-trained MelGAN models are trained on speech or music data, so they are not suitable for environmental sound generation. We train a MelGAN on a large-scale audio event dataset (AudioSet) [14], which contains 527 unique sound events.

IV. Diffusion-Based Decoder

In this section, we introduce our proposed non-autoregressive token-decoder based on discrete diffusion model, named Diffsound. As discussed in Section III-C, Diffsound model is proposed to address the unnatural bias and accumulation of errors issues in AR decoders. In the following, we first introduce the diffusion models. Then we discuss the details of the training and inference of the Diffsound model. Lastly, we discuss how to use the pre-training strategy to further improve the performance of the Diffsound model.

A. Diffusion Probabilistic Models

Diffusion probabilistic models (diffusion models for short) [11] have proved to be a powerful generation model in the image and speech fields [13], [32]. In this section, we briefly introduce some basic principles of the diffusion models.

1) Vanilla Diffusion Model: A diffusion model consists of two processes: the *forward* process with steps $t \in \{0, 1, 2, \ldots, T\}$ and the *reverse* process $t \in \{T, T-1, \ldots, 1, 0\}$. The forward process corrupts the original data $\boldsymbol{x}_0$ into a noisy latent variable $\boldsymbol{x}_T$ which belongs to a stationary distribution (*e.g.*, Gaussian distribution), and the reverse process learns to recover the original data $\boldsymbol{x}_0$ from $\boldsymbol{x}_T$.

a) Forward Process: Given the audio data $\boldsymbol{x}_0$, the forward process aims to corrupt the data $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$ into a sequence of increasingly noisy latent variables $\boldsymbol{x}_{1:T} = \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T$. Each of noisy latent variables $\boldsymbol{x}_t$ has the same dimensionality as $\boldsymbol{x}_0$. The forward process from data $\boldsymbol{x}_0$ to the variable $\boldsymbol{x}_T$ can be formulated as a fixed Markov chain

$$q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \tag{5}$$

Following [11], Gaussian noise is selected in each step, so that the conditional probability distribution can be $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \boldsymbol{I})$, where β_t is a small positive constant. According to the pre-defined schedule $\beta_1, \beta_2, \ldots, \beta_T$ (detials are given in Section VII), the forward process gradually converts original $\boldsymbol{x}_0$ to a latent variable with an isotropic Gaussian distribution of $p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ when T is enough large [11]. Based on the properties of Markov chain [10], the probability distribution $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ can be written as

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\overline{\alpha}_t}\boldsymbol{x}_0, (1-\overline{\alpha}_t)\boldsymbol{I}) \tag{6}$$

where $\alpha_t = 1 - \beta_t$ and $\overline{\alpha}_t = \prod_{s=1}^{t} \alpha_s$.

b) Reverse Process: The reverse process converts the latent variable $\boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ into $\boldsymbol{x}_0$, whose joint probability follows:

$$p_\theta(\boldsymbol{x}_{0:T}) = p(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) \tag{7}$$

where $p_\theta(\cdot)$ is the distribution of the reverse process with learnable parameters θ. The posterior $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ can be derived according to Bayes formula as follows:

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0) q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \tag{8}$$

In order to optimize the generative model $p_\theta(\boldsymbol{x}_0)$ to fit the data distribution $q(\boldsymbol{x}_0)$, one typically optimizes a variational upper bound on the negative log-likelihood [10]:

$$\mathcal{L}_{vb} = \mathbb{E}_{q(\boldsymbol{x}_0)}\Big[D_{KL}[q(\boldsymbol{x}_T|\boldsymbol{x}_0)||p(\boldsymbol{x}_T)] + \sum_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\big[D_{KL}[q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)||p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)]\big]\Big]. \tag{9}$$

2) Discrete Diffusion Model: One limitation of vanilla diffusion model is that, for original data $\boldsymbol{x}_0$ in discrete space, *e.g.*, for any element x_0^r in $\boldsymbol{x}_0$, $x_0^r \in \{1, 2, \ldots, P\}$, we cannot corrupt the $\boldsymbol{x}_0$ by adding Gaussian noise in the forward process since the range of x_0^r belongs to P different discrete values. To solve this issue, discrete diffusion model [11], [12] is proposed. In discrete diffusion model, a transition probability matrix is defined to indicate how $\boldsymbol{x}_0$ transits to $\boldsymbol{x}_t$ for each step of forward process. Assuming that $\boldsymbol{x}_0 \in \mathbb{Z}^N$ and $x_0^k \in \{1, 2, \ldots, P\}$. It is worth noting that $\boldsymbol{x}_0$ denotes a vector with N scalar elements. To better illustrate the transition probability matrix, we will use x_0 to denote any one scalar element in $\boldsymbol{x}_0$ in the following. The matrices $[\boldsymbol{Q}_t]_{ij} = q(x_t = i|x_{t-1} = j) \in \mathbb{R}^{P \times P}$ defines the probabilities that x_{t-1} transits to x_t. Then the forward process for the whole token sequence can be written as:

$$q(x_t|x_{t-1}) = \boldsymbol{c}^\top(x_t)\boldsymbol{Q}_t\boldsymbol{c}(x_{t-1}) \tag{10}$$

where $\boldsymbol{c}(\dot{})$ denotes a function that can transfer a scalar element into a one-hot column vector. The categorical distribution over x_t is given by the vector $\boldsymbol{Q}_t\boldsymbol{c}(x_{t-1})$. Due to the property of Markov chain, one can marginalize out the intermediate steps and derive the probability of x_t at arbitrary timestep directly from x_0 as follows:

$$q(x_t|x_0) = \boldsymbol{c}^\top(x_t)\overline{\boldsymbol{Q}}_t\boldsymbol{c}(x_0), \textit{with } \overline{\boldsymbol{Q}}_t = \boldsymbol{Q}_t \ldots \boldsymbol{Q}_1 \tag{11}$$

Therefore, $q(x_{t-1}|x_t, x_0)$ can be computed based on (11):

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1}, x_0)q(x_{t-1}|x_0)}{q(x_t|x_0)} = \frac{\big(\boldsymbol{c}^\top(x_t)\boldsymbol{Q}_t\boldsymbol{c}(x_{t-1})\big)\big(\boldsymbol{c}^\top(x_{t-1})\overline{\boldsymbol{Q}}_{t-1}\boldsymbol{c}(x_0)\big)}{\boldsymbol{c}^\top(x_t)\overline{\boldsymbol{Q}}_t\boldsymbol{c}(x_0)} \tag{12}$$

B. Non-Autoregressive Mel-Spectrograms Generation via Diffsound

In contrast to the AR token-decoder, which predicts the mel-spectrogram tokens one by one, we expect the Diffsound model to predict all of the tokens in a non-AR manner. Specifically, the Diffsound model can predict all of the tokens simultaneously, then refine the predicted results in the following steps so that the best results can be obtained through iterations. In other words, we expect the predicted results can be improved through T-step iterations. In contrast, AR decoder needs N steps to get results, where N denotes the number of tokens (in practice, $N \gg T$). The Diffsound model can make use of the contextual information of all tokens and revise any token in each step. We speculate that it effectively diminishes the unnatural bias and the accumulated prediction error problems. To realize the process, we adapt the idea from discrete diffusion model [11], [12], [13], which designs a strategy to corrupt the original mel-spectrogram token sequence $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$ into a totally meaningless sequence $\boldsymbol{x}_T \sim p(\boldsymbol{x}_T)$ in T steps, and then let network learn to recover the original sequence $\boldsymbol{x}_0$ based on the text information in T steps. Fig. 1(c) shows an example of the forward and reverse processes. In the inference stage, the reverse process is used to help generate mel-spectrogram tokens. We randomly sample a token sequence $\boldsymbol{x}_T$ from $p(\boldsymbol{x}_T)$, and then let the network predict a new mel-spectrogram token sequence based on $\boldsymbol{x}_T$ and the text features. According to the previous description in Section IV-A,

the key point of training a discrete diffusion model is to design a suitable strategy to pre-define Markov transition matrices $\boldsymbol{Q}_t$.

As discussed in Section III-B, the codebook of VQ-VAE encodes the time-frequency features of sound events. According to this property, we propose three strategies to corrupt the mel-spectrogram tokens. Firstly, changing the context by randomly replacing the original token. Secondly, masking the context by introducing an extra mask token. However, we find that changing the context by randomly replacing tokens results in the reverse process being hard to learn. We speculate that one of the reasons is that a mel-spectrogram token may be replaced with a completely unrelated category token, *e.g.*, a token representing the dog barking may be replaced by a token representing the man speaking. Furthermore, we conjecture that there is a context relationship between adjacent tokens; if we only use the mask token, the model may tend to focus on the mask token and ignore the context relationship. Thus, we propose to combine the changing and masking context strategies. We define three transition matrices according to the three strategies, respectively: Uniform transition matrix, mask transition matrix, and mask and uniform transition matrix.

In the following, we first introduce the three transition matrices. Then we discuss the noise schedule strategy and loss function. Lastly, we introduce the reparameterization trick and fast inference strategy.

1) Uniform Transition Matrix: Uniform transition matrix was first proposed by Sohl-Dickstein et al. [11] for binary random variables, *e.g.*, variable zero can transfer to one or zero with a uniform distribution. Hoogeboom et al. [38] later extended this to categorical variables. The core idea is that each variable can transfer to all the pre-defined categories with a uniform distribution. In practice, the transition matrix $\boldsymbol{Q}_t \in \mathbb{R}^{K\times K}$ can be defined as

$$\boldsymbol{Q}_t = \begin{bmatrix} \alpha_t+\beta_t & \beta_t & \cdots & \beta_t \\ \beta_t & \alpha_t+\beta_t & \cdots & \beta_t \\ \vdots & \vdots & \ddots & \vdots \\ \beta_t & \beta_t & \cdots & \alpha_t+\beta_t \end{bmatrix} \tag{13}$$

where $\beta_t \in [0,1]$ and $\alpha_t = 1 - K\beta_t$. This transition matrix denotes that each token has a probability of $K\beta_t$ to be resampled uniformly over all the K categories, while with a probability of α_t to remain the previous value at the present step. As Section IV-A2 described, we could calculate $q(x_t|x_0)$ according to formula (11), $q(x_t|x_0) = \boldsymbol{c}^\top(x_t)\overline{\boldsymbol{Q}}_t\boldsymbol{c}(x_0)$. However, when the number of categories K and time step T is too large, it can quickly become impractical to store all of the transition matrices $\boldsymbol{Q}_t$ in memory, as the memory usage grows like $O(K^2T)$. Inspired by [13], we find that it is unnecessary to store all of the transition matrices. Instead we only store all of $\overline{\alpha}_t$ and $\overline{\beta}_t$ in advance, because we can calculate $q(x_t|x_0)$ according to following formula:

$$\overline{\boldsymbol{Q}}_t\boldsymbol{c}(x_0) = \overline{\alpha}_t\boldsymbol{c}(x_0) + \overline{\beta}_t. \tag{14}$$

where $\overline{\alpha}_t = \prod_{t=1}^{t}\alpha_t$, $\overline{\beta}_t = (1-\overline{\alpha}_t)/K$. When T is enough large, $\overline{\alpha}_t$ is close to 0. Thus, we can derive the stationary distribution $p(\boldsymbol{x}_T)$ as:

$$p(\boldsymbol{x}_T) = [\overline{\beta}_T, \overline{\beta}_T, \ldots, \overline{\beta}_T] \tag{15}$$

where $\overline{\beta}_T = 1/K$. The Proof can be found in Appendix A.

Algorithm 1: Training of the Diffsound Model.

Require:
A transition matrix $\boldsymbol{Q}_t$, the number of timesteps T, network parameters θ, training epoch N, text-audio dataset $\boldsymbol{D}$, the encoder of VQ-VAE E_{vq}.

for $i = 1$ to N **do**
  for (text, audio) in $\boldsymbol{D}$ **do**
    $\boldsymbol{s}$ = get_mel_spectrogram(audio);
    $\boldsymbol{x}_0 = E_{vq}(\boldsymbol{s})$, $\boldsymbol{y}$ =TextEncoder(text);
    sample t from Uniform$(1, 2, 3, \ldots, T)$;
    sample $\boldsymbol{x}_t$ from $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ based on formula (20);
    estimate $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{y})$;
    calculate loss according to formula (22)–(25);
    update network θ;
  end for
end for
return network θ.

2) Mask Transition Matrix: Previous work [13], [38] proposed introducing an additional absorbing state, such that each token either stays the same or transitions to the absorbing state. In this study, we add an additional token [MASK] into the codebook (the index is $K+1$) to represent the absorbing state, thus the model is asked to recover the original tokens from the mask tokens. The mask transition matrix $\boldsymbol{Q}_t \in \mathbb{R}^{(K+1)\times(K+1)}$ is

$$\boldsymbol{Q}_t = \begin{bmatrix} \beta_t & 0 & 0 & \cdots & 0 \\ 0 & \beta_t & 0 & \cdots & 0 \\ 0 & 0 & \beta_t & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \gamma_t & \gamma_t & \gamma_t & \cdots & 1 \end{bmatrix} \tag{16}$$

where $\gamma_t \in [0,1]$. The mask transition matrix denotes that each token either stays the same with probability $\beta_t = (1-\gamma_t)$ or transitions to an additional token [MASK] with probability γ_t. Similarly, we only store all of $\overline{\gamma}_t$ and $\overline{\beta}_t$ in advance, and we calculate $q(x_t|x_0)$ according to following formula:

$$\overline{\boldsymbol{Q}}_t\boldsymbol{c}(x_0) = \overline{\beta}_t\boldsymbol{c}(x_0) + \overline{\gamma}_t\boldsymbol{c}(K+1). \tag{17}$$

where $\overline{\gamma}_t = 1-\overline{\beta}_t$, and $\overline{\beta}_t = \prod_{t=1}^{t}\beta_t$. According to Markov transition formula, when T is enough large, $\overline{\beta}_T$ is close to 0. Thus the stationary distribution $p(\boldsymbol{x}_T)$ can be derived as:

$$p(\boldsymbol{x}_T) = [0, 0, \ldots, 1]. \tag{18}$$

The Proof can be found in Appendix B.

3) Mask and Uniform Transition Matrix: As in the previous discussion, we speculate that using the uniform transition matrix brings the reverse process is hard to learn. Using the mask transition matrix may make the model tend to focus on the mask token and ignore the context information. To combat these problems, a simple idea is to combine mask and uniform

Algorithm 2: Inference of the Diffsound Model.

Require:
Time stride Δ_t, the number of timesteps T, input text, the decoder of VQ-VAE G, network θ, stationary distribution $p(\boldsymbol{x}_T)$;
1: $t = T$, $\boldsymbol{y}$ =TextEncoder(text);
2: sample $\boldsymbol{x}_t$ from $p(\boldsymbol{x}_T)$;
3: **while** $t > 0$ **do**
4: $\quad \boldsymbol{x}_t \leftarrow$ sample from $p_\theta(\boldsymbol{x}_{t-\Delta_t}|\boldsymbol{x}_t, \boldsymbol{y})$
5: $\quad t \leftarrow (t - \Delta_t)$
6: **end while**
7: **return** $G(\boldsymbol{x}_t)$.

transition matrices. Specifically, each token has a probability of γ_t to transition to [MASK] token, a probability of $K\beta_t$ be resampled uniformly over all the K categories and a probability of $\alpha_t = 1 - K\beta_t - \gamma_t$ to stay the same token. The transition matrices $\boldsymbol{Q}_t \in \mathbb{R}^{(K+1)\times(K+1)}$ is defined as

$$\boldsymbol{Q}_t = \begin{bmatrix} \alpha_t + \beta_t & \beta_t & \beta_t & \cdots & 0 \\ \beta_t & \alpha_t + \beta_t & \beta_t & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \gamma_t & \gamma_t & \gamma_t & \cdots & 1 \end{bmatrix}. \tag{19}$$

According to previous discussions, we can calculate $q(x_t|x_0)$ according to following formula:

$$\overline{\boldsymbol{Q}}_t \boldsymbol{c}(x_0) = \overline{\alpha}_t \boldsymbol{c}(x_0) + (\overline{\gamma}_t - \overline{\beta}_t)\boldsymbol{c}(K+1) + \overline{\beta}_t. \tag{20}$$

where $\overline{\alpha}_t = \prod_{t=1}^{t} \alpha_t$, $\overline{\gamma}_t = 1 - \prod_{t=1}^{t}(1-\gamma_t)$ and $\overline{\beta}_t = (1 - \overline{\alpha}_t - \overline{\gamma}_t)/K$. Similarly, when T is enough large, $\overline{\alpha}_T$ is close to 0. Thus the stationary distribution $p(\boldsymbol{x}_T)$ can be derived as

$$p(\boldsymbol{x}_T) = [\overline{\beta}_T, \overline{\beta}_T, \ldots, \overline{\gamma}_T] \tag{21}$$

where $\overline{\beta}_T = (1 - \overline{\gamma}_T)/K$.

4) Noise Schedule: Noise schedule is used to pre-define the value of transition matrices (pre-define $\overline{\alpha}_t$, $\overline{\beta}_t$, and $\overline{\gamma}_t$ in our study). Many noise schedules have been proposed, such as the linear schedule, the consine schedule [47], and the mutual-information-based noise schedule [12]. In this study, we adapted the linear schedule for all of the experiments.

5) Loss Function: Similar to the training objective of a continuous diffusion model (9), we also train a network $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{y})$ to estimate the posterior transition distribution $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$. The network is trained to minimize the variational lower bound (VLB).

$$\mathcal{L}_{vlb} = \sum_{t=1}^{T-1} \left[D_{KL}[q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)||p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{y})] \right] + D_{KL}(q(\boldsymbol{x}_T|\boldsymbol{x}_0)||p(\boldsymbol{x}_T)) \tag{22}$$

where $p(\boldsymbol{x}_T)$ is the stationary distribution, which can be derived in advance. Note that we add conditional information $\boldsymbol{y}$ to $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{y})$ in (22). The completed training algorithm is summarized in Algorithm 1.

6) Reparameterization Trick: Recent works [13], [47] found that approximating some surrogate variables, such as $p_\theta(\hat{\boldsymbol{x}}_0|\boldsymbol{x}_t, \boldsymbol{y})$ gives better results comparing with directly predicting the posterior $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$. In this study, we follow the reparameterization trick proposed in [13], which lets the Diffsound model predict the noiseless token distribution $p_\theta(\hat{\boldsymbol{x}}_0|\boldsymbol{x}_t, \boldsymbol{y})$ at each reverse step, and then compute $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{y})$ according to the following formula:

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{y}) = \sum_{\hat{\boldsymbol{x}}_0=1}^{K} q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \hat{\boldsymbol{x}}_0) p_\theta(\hat{\boldsymbol{x}}_0|\boldsymbol{x}_t, \boldsymbol{y}). \tag{23}$$

Based on the formula (23), an auxiliary denoising objective loss is introduced, which encourages the network to predict $p_\theta(\hat{\boldsymbol{x}}_0|\boldsymbol{x}_t, \boldsymbol{y})$:

$$\mathcal{L}_{x_0} = -\log p_\theta(\hat{\boldsymbol{x}}_0|\boldsymbol{x}_t, \boldsymbol{y}). \tag{24}$$

Experimental results indicate that combining $\mathcal{L}_{x_0}$ and $\mathcal{L}_{vlb}$ could get better performance. Thus our final loss function is defined as:

$$\mathcal{L} = \lambda \mathcal{L}_{x_0} + \mathcal{L}_{vlb} \tag{25}$$

where λ is a hyper-parameter to control the weight of the auxiliary loss $\mathcal{L}_{x_0}$.

7) Fast Inference Strategy: We can see that the inference speed of the Diffsound model is related to the number of timesteps, T. By leveraging the reparameterization trick, we can skip some steps in the Diffsound model to achieve a faster inference. Usually, we sample the spectrogram tokens in the chain of $\boldsymbol{x}_T, \boldsymbol{x}_{T-1}, \boldsymbol{x}_{T-2}, \ldots, \boldsymbol{x}_0$. Thus, we can use a larger time stride Δ_t by sampling the spectrogram tokens in the chain of $\boldsymbol{x}_T, \boldsymbol{x}_{T-\Delta_t}, \boldsymbol{x}_{T-2\Delta_t}, \ldots, \boldsymbol{x}_0$. Similar to (23), with the fast inference strategy, $p_\theta(\boldsymbol{x}_{t-\Delta_t}|\boldsymbol{x}_t, \boldsymbol{y})$ can be computed as:

$$p_\theta(\boldsymbol{x}_{t-\Delta_t}|\boldsymbol{x}_t, \boldsymbol{y}) = \sum_{\hat{\boldsymbol{x}}_0=1}^{K} q(\boldsymbol{x}_{t-\Delta_t}|\boldsymbol{x}_t, \hat{\boldsymbol{x}}_0) p_\theta(\hat{\boldsymbol{x}}_0|\boldsymbol{x}_t, \boldsymbol{y}). \tag{26}$$

Note that we make sure the last step is $\boldsymbol{x}_0$ in our experiments. We found this strategy makes the inference stage more efficient, which only causes little decrease to quality. The whole inference algorithm is summarized in Algorithm 2.

C. Pre-Training Diffsound on AudioSet Dataset

Recently, image generation has got great success, one of the reasons is that they collect large-scale text-image data, *e.g.*, CogView [25] collects more than 30 million high-quality (Chinese) text-image pairs. However, collecting large-scale text-audio pairs is time-consuming. AudioSet [14] is the largest open-source dataset in the audio field, but it only provides the event labels for each audio. To utilize these data, we propose a mask-based text generation method to generate text descriptions according to event labels.

1) Mask-Based Text Generation Method (MBTG): Our method is based on our observation of how humans write text descriptions for audio: Humans first listen to the audio to find out which events happen, and then they add some detailed descriptions to compose a sentence. For example, if the label is

Algorithm 3: Pre-Training the Diffsound Model on AudioSet.

Require:
The audio and its corresponding label $\{\boldsymbol{A}, \boldsymbol{L}\}$, the number of training epoch n, initial network parameters θ;
Count the number of sound events for each audio-label pair;
Split $\{\boldsymbol{A}, \boldsymbol{L}\}$ into two subset according the number of events, $\{\boldsymbol{A}_{SES}, \boldsymbol{L}_{SES}\}, \{\boldsymbol{A}_{MES}, \boldsymbol{L}_{MES}\}$;
for $i = 1$ to n **do**
  for (audio, l) in $\{\boldsymbol{A}_{SES}, \boldsymbol{L}_{SES}\}$ **do**
    text = MBTG(l);
    Train the model θ according to (text, audio);
  end for
end for
for $i = 1$ to $2 \cdot n$ **do**
  for (audio, l) in $\{\boldsymbol{A}_{MES}, \boldsymbol{L}_{MES}\}$ **do**
    text = MBTG(l);
    Train the model θ according to (text, audio);
  end for
end for
return Diffsound θ.

"dog barks, man speaks", one can generate the text description like that "a dog is barking when a man is speaking" or "a dog barks after a man speaks over". The first sentence indicates that the events of dog barking and man speaking are simultaneously happening. The latter shows that we first listen to a man speaks, and then a dog barks. Although the keywords are the same, the generated two texts correspond to different audio due to different detailed descriptions. It means that automatically generating text descriptions according to the label information is a tough task. Instead of generating specific text, we propose to use '[MASK]' token to replace the detailed description. We can generate text descriptions like that "[MASK] [MASK] dog bark [MASK] man speaking [MASK]". We expect the model to learn the relationship of events rather than directly obtain it from the text description. The generation algorithm is easy to implement. We randomly insert $m \in \{1, 2\}$ mask tokens on either side of the label.

2) Curriculum Learning Strategy: We found that it is easier to generate audio that only includes a single event than to audio that includes multiple events. To help the Diffsound model learn better, we mimic the human learning process by letting the Diffsound model learn from easy samples, and gradually advance to complex samples and knowledge. Thus, a curriculum learning [48] strategy is proposed. Specifically, we rank the AudioSet according to the number of events in the audio, and then we split the AudioSet into two subsets: one only includes the audio of a single event (we refer to it as the Single Event Set (SES)), and the other includes the audio of multiple events (we refer to it as the Multiple Event Set (MES)). We first train the Diffsound model on the SES in the first few epochs. After that we train the Diffsound model on the MES. The whole algorithm is summarized in Algorithm 3.

V. Dataset and Data Pre-Processing

AudioSet [14] and AudioCaps [8] dataset are used in our experiments. In the following, we first introduce the AudioSet and AudioCaps datasets, then we discuss the details of data pre-processing.

A. AudioSet

An ontology comprising 527 sound classes is used in the large-scale audio dataset known as AudioSet. More than 2 million 10 seconds audio snippets from YouTube videos are included in the AudioSet collection. There are roughly 1.9 M audio clips in our training set because some audio clips can no longer be downloaded. Each audio clip may have one or more labels for the presented audio events.

B. AudioCaps

AudioCaps is the largest audio captioning dataset currently available with around 50 k audio clips sourced from AudioSet. AudioCaps includes three sets: training, validation, test sets. There are 49256, 494, and 957 audio clips in our training, validation and test sets. Each audio clip in the training set contains one human-annotated caption, while each contains five captions in the validation and test set. We use the AudioCaps training set to train our models. We evaluate our methods on the AudioCaps validation set.

C. Data Pre-Processing

All audio clips in the two datasets are converted to 22.05 k Hz and padded to 10 seconds long. Log mel-spectrograms extracted using a 1024-points Hanning window with 256-points hop size and 80 mel bins are used as the input features. Finally, we can extract a 860×80 mel-spectrogram from 10 seconds audio.

VI. Evaluation Metric

In this study, we investigate Humans Mean Opinion Score (MOS) and objective assessment metrics.

A. MOS

We randomly choose 15 sets of generated sound clips by AR token-decoder and Diffsound model. Each set includes one text description, one real sample, 1–2 generated sounds by AR token-decoder, and 1–2 generated sounds by the Diffsound model. We let 10 people give the grades for these sounds in three aspects: relevance, fidelity, and intelligibility. Note that the test person never knows whether the sound is real or generated. We ask people to give 0–5 grades on the three aspects. Finally, we use the average score of the three aspects as the MOS.

B. Objective Assessment Metrics

Human evaluation of the performance of the sound generation model is an expensive and tedious procedure. Thus, designing a

proper evaluation metric that can measure the gap between generated and real samples is very important for the generation task. In this paper, we employed three objective quality assessment metrics: Fréchet Inception Distance (FID), Kullback-Leibler (KL) divergence, and Audio Caption Loss. We also conducted a group of ablation experiments to validate that our proposed metrics are effective.

1) FID: Fréchet Inception Distance (FID) [15] is often used to evaluate the fidelity of the generated samples in the image generation domain. Iashin and Rahtu [3] also use FID as one metric to evaluate the generated sound. FID is defined by the distance between the distributions of the pre-classification layer's features of InceptionV3 [49] between fake and real samples, and InceptionV3 is usually pre-trained on ImageNet [50]. The mathematics definition of FID as follow:

$$||\boldsymbol{m}_r - \boldsymbol{m}_f||_2^2 + Tr\left(\boldsymbol{C}_r + \boldsymbol{C}_f - 2(\boldsymbol{C}\boldsymbol{C}_f)^{\frac{1}{2}}\right) \tag{27}$$

where $\boldsymbol{m}_r$ and $\boldsymbol{m}_f$ denote the mean of features extracted from real and generated samples. $\boldsymbol{C}_r$ and $\boldsymbol{C}_f$ are the covariance matrix of features extracted from real and generated samples. Tr denotes the trace of the matrix.

Considering the difference between images and spectrograms, we adapt the InceptionV3 architecture [49] for the spectrogram and train the model on the AudioSet dataset [14]. Specifically, we modify the input convolutional layer and change the number of channels from 3 to 1. We do not use the max pooling operations, in order to preserve spectrogram resolution at higher layers. We train the InceptionV3 on the Audioset with a batch size of 64 mel-spectrograms using Adam optimizer, the learning rate is 3×10^{-3} with weight decay is 1×10^{-3}.

2) KL: For the text-to-sound generation task, one important thing is to evaluate the relevance between generated samples and text descriptions. Considering a sound comprises of multiple events, we can use a pre-trained classifier (pre-trained InceptionV3 on the AudioSet dataset) to get the probability of generated and real samples, and then calculate Kullback-Leibler (KL) divergence between the two probability distributions.

3) Audio Caption Loss: Text-to-sound generation task can be seen as a reverse audio caption [51], [52] task. Intuitively, if the generated sample has high fidelity and relevance with text description $\boldsymbol{d}$, the generated sample can be translated to $\hat{\boldsymbol{d}}$ using an audio caption system, and the difference between $\boldsymbol{d}$ and $\hat{\boldsymbol{d}}$ should be small. Thus we propose to turn to the metrics in audio caption field for the text-to-sound generation task. Specifically, we first train an audio caption transformer (ACT) [53] on AudioCaps dataset [8]. We follow the basic model structure proposed in [53]. The difference is that we use a 860×80 log-mel-spectrogram as input. Then we use SPICE [54] and CIDEr [55], which are common evaluation metrics for audio caption tasks, to evaluate the quality of generated samples. The SPICE score guarantees the generated captions are semantically close to the audio, while the CIDEr score guarantees captions are syntactically fluent. The higher SPICE and CIDEr scores indicate better generation quality.

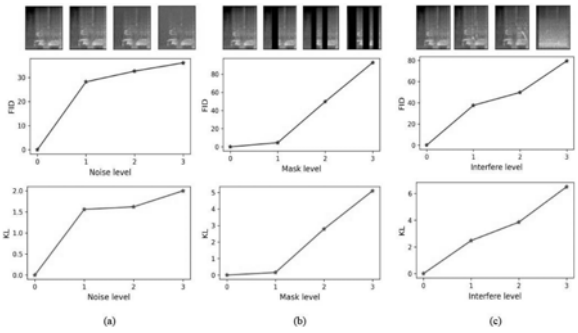

Fig. 3. FID and KL are evaluated for **(a)**: add Gaussian noise, **(b):** mask part of audio content, **(c):** mix with other interfere sound. The first row is the simple visualization of how the spectrograms changed when different disturbance level is used. The disturbance level rises from zero and increases to the highest level. We can see that the FID and KL scores capture the disturbance level very well.

C. The Effectiveness of FID and KL

We try to validate whether the FID and KL scores can measure the gap between the generated and real samples. To realize this, we randomly choose a set of audio $\boldsymbol{X}_o$ from the AudioCaps [8] validation set and try to generate a new set $\boldsymbol{X}_f$ from three different aspects, respectively: add Gaussian noise, mask part of the audio content, and add interfering sound. Finally, we calculate the FID and KL scores between $\boldsymbol{X}_o$ and $\boldsymbol{X}_f$ to verify whether the FID and KL metrics are sensitive to these factors. The visualization results are shown in Fig. 3.

1) Add Gaussian Noise: Intuitively, if the generated sample contains too much noise, which may not be very well. Thus, we add Gaussian noise to $\boldsymbol{X}_o$ to generate a new set $\boldsymbol{X}_f$, and then we calculate the FID and KL scores between $\boldsymbol{X}_o$ and $\boldsymbol{X}_f$. As Fig. 3(a) shows, the FID and KL scores gradually increase when we add more noise, which indicates that FID and KL metrics are sensitive to extra noise.

2) Mask Part of Audio Content: If the generated sample only contains part of the acoustic information compared to the real sample, the generated sample is suboptimal. Thus, we mask part of the audio content to generate a new set $\boldsymbol{X}_f$. As Fig. 3(b) shows, when we gradually increase the proportion of masked parts, FID and KL scores also increase.

3) Add Interfering Sound: The generated sound should not contain irrelevant acoustic information with the text description. We randomly choose an interfering audio from the AudioCaps training set, and then we directly mix the interfering sound with $\boldsymbol{X}_o$ to build a new set $\boldsymbol{X}_f$. According to Fig. 3(c), we can see that the FID and KL scores gradually increase when the interfering sound increases.

VII. Experimental Setup

A. Implementation Details

Our proposed text-to-sound generation framework is not trained end-to-end. We train each part separately. For text encoder, we directly use the pre-trained BERT or CLIP models. We first train VQ-VAE and vocoder. Then we train the token-decoder with the help of the text encoder and pre-trained VQ-VAE. Note that the VQ-VAE and text encoder are fixed when we train the

token-decoder. In the following, we will introduce the details of network structure and training strategy.

1) VQ-VAE: In this study, our VQ-VAE's spectrogram encoder E_{vq}, spectrogram decoder G, and discriminator D follow the setting of VQ-GAN [3], [24], which is a variant version of VQ-VAE [7]. For codebook $\boldsymbol{Z}$, the dimension of each vector n_z is set as 256, and the codebook size K is set as 256. VQ-VAE converts 860×80 spectrogram into 53×5 tokens. We train our VQ-VAE model on AudioCaps and AudioSet datasets. We find that training on AudioSet can achieve better performance. Unless specifically stated, we default to using the VQ-VAE pre-trained on AudioSet. The learning rate is fixed and determined as a product of a base learning rate, a number of GPUs, and a batch size. In our experiments, the base learning rate is set as 1×10^{-6}, and Adam optimizer [56] is used. We train VQ-VAE with batches of 20 mel-spectrograms on 8 Nvidia V100 GPUs. The training takes about 9 days on the AudioSet. To stabilize the training procedure, we zero out the adversarial part of the loss in formula (4) (set $\lambda_d = 0$) for the first 2 training epochs, after that the adversarial loss is used, and $\lambda_d = 0.8$.

2) Autoregressive Token-Decoder: Inspired by the success in autoregressive sound generation [3], [4], we follow the SOTA backbone of the video-to-sound generation task, and employ a transformer-based network to learn the mapping from text to the spectrogram tokens. Specifically, the autoregressive token-decoder is a 19-layer 16-head transformer with a dimension of 1024. We use one dense layer to map the text features into the transformer's hidden dimension space (1024), so that the text features can be forward into the transformer. The output of the transformer is passed through a K-way softmax classifier. The base learning rate is 1×10^{-6}, and the AdamW optimizer [57] is used. The batch size is set as 16 for each GPU. The model is trained until the loss on the validation set has not improved for 2 consecutive epochs. Training the autoregressive token-decoder takes about 2 days on 8 Nvidia P40 GPUs.

3) Diffsound Model: For a fair comparison with the autoregressive token-decoder under similar parameters, we also built a 19-layer 16-head transformer with a dimension of 1024 for the Diffsound model. Each transformer block contains a full attention, a cross attention to combine text features and a feed-forward network block. The current timestep t is injected into the network with Adaptive Layer Normalization [58](AdaLN) operator.

a) Noise Schedule Setting: For the uniform transition matrix, we linearly increase $\overline{\beta}_t$ from 0 to 0.1, and decrease $\overline{\alpha}_t$ from 1 to 0. For the mask transition matrix, we linearly increase $\overline{\gamma}_t$ from 0 to 1, and decrease $\overline{\beta}_t$ from 1 to 0. For the mask and uniform transition matrix, we linearly increase $\overline{\gamma}_t$ and $\overline{\beta}_t$ from 0 to 0.9 and 0.1, and decrease $\overline{\alpha}_t$ from 1 to 0.

b) Training Details: For the default setting, we set timesteps $T = 100$ and loss weight $\lambda = 1e-4$ in formula (25). The mask and uniform transition matrix is used, because we find it can get the best generation performance. We optimize our network using AdamW [57] with $\beta_1 = 0.9$ and $\beta_2 = 0.94$. The basic learning rate is 3×10^{-6}, and batch size is 16 for each GPU. We train the Diffsound model on the AudioCaps, which takes about 2 days on 16 Nvidia V100 GPUs (the number of training epochs is set to 400). If we pre-train the Diffsound model on the AudioSet, we use 32 Nvidia V100 GPUs, which takes about 8 days (the number of training epochs is set to 200).

TABLE I
THE MEAN OPINION SCORE COMPARASION BETWEEN AR AND DIFFSOUND MODELS. GT DENOTES THE GROUND TRUTH SOUND

Model	Relevance↑	Fidelity↑	Intelligibility↑	MOS↑
GT	4.307	4.167	3.873	4.116
AR	2.747	2.7	2.913	2.786
Diffsound	**3.833**	**3.487**	**3.36**	**3.56**

The bold entities indicate the best performance.

4) Vocoder: We rely on the official implementation of the MelGAN [6]. During training, the model inputs a random sequence of 8192 audio samples (the sample rate is 22050). The vocoder is trained for 200 epochs with a batch size of 256 mel-spectrograms on one P40 GPU for approximately 20 days. Considering the time complexity, we do not use all of the AudioSet data, we randomly choose 40% audio clips to train the MelGAN.

5) The Duration of the Generated Sound: Consider that each of the clips from the AudioCaps dataset contains 10 seconds of audio. We fixed the duration of the generated sound to 10 seconds to ensure a fair comparison of generated and real sound. As a result, the number of generated mel-spectrogram tokens is fixed at 265 for both the AR token-decoder and the Diffsound model (10 seconds audio corresponding to 80×860 mel-spectrogram, and the spectrogram encoder in the VQ-VAE model including 16 downsampling operation for both time and frequency dimensions, thus 10 seconds audio can be approximated to $5 \times 53 = 265$ tokens).

VIII. RESULTS AND ANALYSIS

In this section, we conduct experiments to verify the effectiveness of our text-to-sound generation framework. Table I shows the MOS comparison between the generated and real sound. We can see that our text-to-sound generation framework could achieve good MOS performance (the MOS both large than 2.5) regardless of whether the token-decoder is AR or Diffsound model. Let's start with a detailed comparison between the AR token-decoder and our proposed Diffsound model. Then we conduct ablation studies for the Diffsound model.

A. The Comparison Between the AR Decoder and Diffsound

1) Subjective and Objective Metrics: Table I shows the subjective metrics (MOS) comparison between the AR token-decoder and Diffsound model. We can see that our Diffsound model significantly improves the MOS compared to the AR token-decoder, *e.g.*, MOS 3.56 *v.s* 2.786. Due to many audio clips including background noise, the intelligibility of Ground Truth is relatively low. Table II shows the objective metrics comparison between the AR decoder and the Diffsound. Firstly, by comparing rows 1 and 2, we can see that using the CLIP model as the text encoder brings better generation performance than the BERT model. Secondly, we can see that using the VQ-VAE trained on the AudioSet dataset brings better generation quality

TABLE II
THE OBJECTIVE METRICS COMPARISON BETWEEN AR DECODER AND DIFFSOUND

Model	CB	TE	FID↓	KL↓	SPICE↑	CIDEr↑
AR	Caps	BERT	18.01	6.8	0.055	0.1
	Caps	CLIP	17.94	5.98	0.082	0.2
	AudioSet	CLIP	16.87	5.31	0.088	0.22
Diffsound	Caps	CLIP	13.47	4.95	0.093	0.28
	AudioSet	CLIP	**9.76**	**4.21**	**0.103**	**0.36**

CB denotes that we train the codebook on audioset or audiocaps (caps for short) datasets. TE denotes the type of text encoder.
The bold entities indicate the best performance.

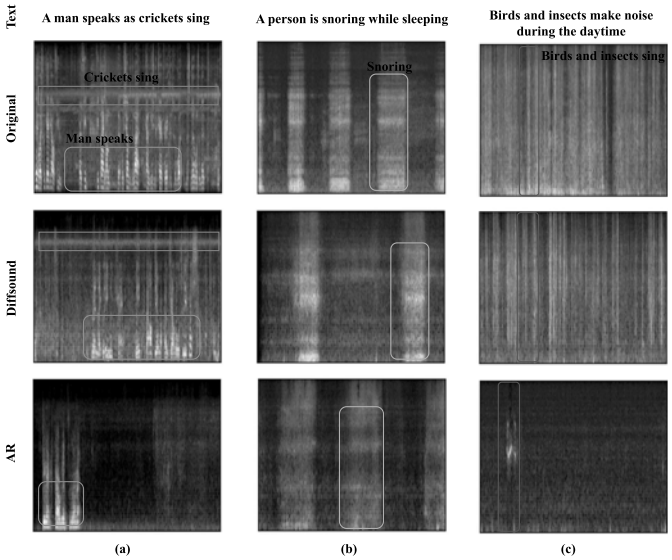

Fig. 4. The visualization of generated samples by the Diffsound model and AR token-decoder. The first line is the text input. The second line is the mel-spectrograms of real audio. The last two lines are the mel-spectrograms of generated audio by Diffsound model and AR token-decoder.

than on the AudioCaps dataset, we speculate that training on a large-scale dataset can improve the ability of VQ-VAE. Lastly, by comparing the AR token-decoder and Diffsound model, we can see that the Diffsound model gets better performance on all of the metrics, *e.g.*, FID 9.76 *v.s* 16.87, KL 4.21 *v.s* 5.31, CIDEr 0.36 *v.s* 0.22. In summary, the objective and subjective metrics both indicate the effectiveness of our Diffsound model.

2) The Generation Speed: Generation speed is also an important metric to evaluate the generation model. To investigate the generation speed between AR token-decoder and Diffsound model, we conducted a group of ablation experiments, and the results are shown in Table III. Note that we conducted these experiments on a single Nvidia P40 GPU. We fix the generated sound duration as 10 seconds. We only calculate the time to generate the mel-spectrograms and ignore the vocoder's costs due to the vocoder is the same for all models. Firstly, we can see that using the AR token-decoder to generate a mel-spectrogram needs about 23 seconds, but using our Diffsound model only takes about 5 seconds. Moreover, our Diffsound model also has better generation performance. Secondly, the generation speed of our Diffsound model can be further improved by using less number of timesteps T and larger time stride Δ_t but decreases the generation quality. The fastest generation speed is obtained when the number of timesteps $T = 25$ and $\Delta_t = 7$. The Diffsound model only needs 0.53 seconds to generate a mel-spectrogram, which is 43 times faster than the AR token-decoder with a similar FID score.

3) Visualization: Fig. 4 shows some generated samples by the Diffsound model and AR token-decoder. We can see that the Diffsound model can generate a scene with complete semantics compared to the AR token-decoder, *e.g.*, Fig. 4(a) shows that the sound generated by AR token-decoder only includes the man speaks event and the crickets sing is missing. Furthermore, as Fig. 4(b) and (c) show, our Diffsound model has better detailed modelling ability than the AR token-decoder.

B. Ablation Study for Diffsound Model

1) Impact of Different Transition Matrices for Diffsound Model: In this section, we explore the impact of three transition matrices on Diffsound model: uniform transition matrix, mask transition matrix, mask and uniform transition matrix. Table II presents the results of Diffsound model with three different transition matrices. We can see that the best results are obtained when the mask and uniform matrix is used and $\overline{\gamma}_T = 0.9$. From Table IV, it can be observed that using the combined mask and uniform transition matrix outperforms using the uniform transition matrix. One possible explanation is that: using a uniform transition matrix may make the reverse process hard to learn; and only using the mask transition matrix may make the model tend to focus on the mask token and ignore the context information.

2) Impact of the Final Mask Rate $\overline{\gamma}_T$ for the Mask and Uniform Transition Matrix: We conduct ablation studies to investigate the impact of the final mask rate ($\overline{\gamma}_T$) on the mask and uniform transition matrix. Results are shown on Table IV (lines 2–6). Experiments show that using $\overline{\gamma}_T = 0.9$ outperforms other settings.

3) Number of Timesteps: We conduct ablation studies to investigate the impact of the number of timesteps T of the training and inference stages for the Diffsound model, with the results shown in Table V. In this study, considering the generation speed, we set the maximum number of timesteps T as 100 in both training and inference. We can see that using more number of timesteps in the training and inference stage could get better performance. However using more number of timesteps will cost more time in the inference stage. Furthermore, we can find that it still maintains a good performance when dropping 75 inference steps (*e.g.*, training step is 100, but inference step is 25), which gives us a direction to boost the generation speed.

C. The Effectiveness of Pre-Training the Diffsound Model on AudioSet

In this section, we validate whether pre-training the Diffsound on the AudioSet dataset can improve the generation performance. Note that considering the time complexity, we only use about 45% AudioSet training set. Table VI shows the experimental results. We can see that using the pre-trained

TABLE III
THE GENERATION SPEED COMPARISON BETWEEN THE AR DECODER AND DIFFSOUND

Token-decoder	Timestep (T)	Time stride Δ_t	FID (↓)	KL (↓)	SPICE (↑)	CIDEr (↑)	Speed(spec/s) (↓)
AR	-	-	16.87	5.31	0.054	0.16	23.24
Diffsound	25	7	16.76	4.68	0.088	0.25	**0.53**
		5	16.51	4.7	0.088	0.26	0.59
		3	12.54	4.38	0.099	0.32	0.72
		1	10.91	4.2	0.104	0.34	1.49
	50	7	15.26	4.66	0.094	0.29	0.67
		5	14.04	4.54	0.092	0.28	0.82
		3	11.06	4.25	0.102	0.32	1.15
		1	10.48	4.24	0.104	0.35	2.77
	100	7	11.87	4.35	0.103	0.34	1.02
		5	12.71	4.44	0.095	0.30	1.28
		3	10.13	4.3	0.103	0.35	1.86
		1	**9.76**	**4.21**	**0.103**	**0.36**	4.96

Timesteps T denotes the start step in inference stage. Time stride $\Delta_t > 1$ indicates that we use fast inference strategy.
The bold entities indicate the best performance.

TABLE IV
ABLATION STUDY FOR THREE DIFFERENT TRANSITION MATRICES

ID	Matrix	Mask rate	FID↓	KL↓	SPICE↑	CIDEr↑
1	U	0	10.14	4.31	0.101	0.35
2	MU	0.1	10.63	4.47	0.093	0.29
3		0.3	10.64	4.35	0.099	0.34
4		0.5	10.75	4.31	0.096	0.32
5		0.7	9.84	4.37	0.102	0.34
6		0.9	**9.76**	**4.21**	**0.103**	**0.36**
7	M	1	11.5	4.46	0.103	0.34

Furthermore, we also discuss the effect of the final mask rate γ_T on mask and uniform matrix. U denotes the uniform matrix, M denotes the mask matrix.
The bold entities indicate the best performance.

TABLE V
ABLATION STUDY ON TRAINING AND INFERENCE STEPS

		Training step		
		25	50	100
Inference step	25	12.22	11.58	10.91
	50	-	11.08	10.48
	100	-	-	**9.76**

Each column uses the same training steps while each row uses the same inference steps. We only report the FID in this table.
The bold entities indicate the best performance.

TABLE VI
THE EFFECTIVENESS OF PRE-TRAINING THE DIFFSOUND ON AUDIOSET. CL DENOTES THE CURRICULUM LEARNING STRATEGY

Token-decoder	PR	CL	FID↓	KL↓	SPICE↑	CIDEr↑
Diffsound	✗	✗	9.76	4.21	0.103	0.36
	✓	✗	8.78	4.15	0.101	0.35
	✓	✓	**8.27**	**4.11**	**0.105**	**0.36**

PR denotes the pre-training strategy.
The bold entities indicate the best performance.

Diffsound model can improve the generation performance in terms of FID and KL (such as lowering the score of FID and KL from 9.76 and 4.21 to 8.78 and 4.15). We can see that when pre-training and curriculum learning strategies are both used, the best performance is obtained (such as FID score of 8.27, KL score of 4.11, and SPICE score of 0.105), which shows that the curriculum learning strategy is also important. We believe that performance can be further improved when we use more data.

TABLE VII
CHOOSE HIGH-RELEVANCE SOUND WITH TEXT DESCRIPTION BASED ON AUDIO CAPTION LOSS

Token-decoder	Top_k	FID↓	KL↓	SPICE↑	CIDEr↑
AR	10	16.87	5.31	0.088	0.22
	5	16.28	5.22	0.094	0.26
	2	**15.72**	**5.03**	**0.145**	**0.41**
Diffsound	10	9.76	4.21	0.103	0.36
	5	9.83	4.03	0.164	0.52
	2	10.14	**3.86**	**0.218**	**0.71**

Top_k denotes that we keep the top k samples according to the SPICE and CIDEr scores. We totally generate 10 samples for each text examples.
The bold entities indicate the best performance.

D. Choose High-Relevance Sound With Text Description Based on Audio Caption Loss

Due to the random sample process in the inference stage of the Diffsound model and AR token-decoder, the generated sound may be different in multiple sampling processes even using the same text description. We generated 10 samples for each text in this study. To quickly choose high-relevance samples with the input text, we can rank the samples according to the sum of their SPICE and CIDEr scores, and then we only keep a subset of them. As Table VII shows, if we only keep the top 2 samples for each text, the AR method's KL score will improve from 5.31 to 5.03, and the Diffsound method's KL score will improve from 4.21 to 3.86. We conjecture that this strategy can help us quickly choose high-relevance samples with the text description. Furthermore, we also observe that the FID score of the Diffsound will slightly increase when top_k from 10 to 2. We think one of the reasons is that the Diffsound model generates some high-fidelity samples but these samples are irrelevant to the text description.

IX. CONCLUSION

In this work, we present a framework for text-to-sound generation tasks, and propose a novel non-autoregressive token-decoder (Diffsound) based on discrete diffusion model, which significantly improves the generation performance and speed compared to the AR token-decoder. We also explore a simple pre-training strategy to further improve the performance of

Diffsound model. To effectively evaluate the quality of generated samples, we designed three objective evaluation metrics for this task. Both objective and subjective metrics verified the effectiveness of Diffsound model.

This work still has some limitations that need to be addressed in our future work, *e.g.*, our generation framework is not end-to-end, we separately train the VQ-VAE, the token decoder, and the vocoder, which may not be optimal. In the future, we will explore an end-to-end sound generation framework.

APPENDIX A
THE PROOF OF FORMULA (14)

We use mathematical induction to prove formula (14). We have following conditional information:

$$\beta_t \in [0,1], \alpha_t = 1 - K\beta_t, \overline{\alpha}_t = \prod_{i=1}^{t} \alpha_i, \overline{\beta}_t = (1-\overline{\alpha}_t)/K. \tag{28}$$

Now we want to prove that $\overline{\boldsymbol{Q}}_t \boldsymbol{c}(x_0) = \overline{\alpha}_t \boldsymbol{c}(x_0) + \overline{\beta}_t$. Firstly, when $t = 1$, we have:

$$\overline{\boldsymbol{Q}}_1 \boldsymbol{c}(x_0) = \begin{cases} \overline{\alpha}_1 + \overline{\beta}_1, & x = x_0 \\ \overline{\beta}_1, & x \neq x_0 \end{cases} \tag{29}$$

which is clearly hold. Suppose the formula (14) holds at step t, then for $t = t+1$, we have:

$$\overline{\boldsymbol{Q}}_{t+1} \boldsymbol{c}(x_0) = \boldsymbol{Q}_{t+1} \overline{\boldsymbol{Q}}_t \boldsymbol{c}(x_0).$$

Now we consider two conditions:

1) when $x = x_0$ in step $t+1$, we have two situations in step t, that is, $x = x_0$ and $x \neq x_0$, so that we have:

$$\begin{aligned} \boldsymbol{Q}_{t+1} \boldsymbol{c}(x_0)_{(x)} &= (\overline{\alpha}_t + \overline{\beta}_t)(\alpha_{t+1} + \beta_{t+1}) + (K-1)\overline{\beta}_t \beta_{t+1} \\ &= \overline{\alpha}_{t+1} + \overline{\alpha}_t \beta_{t+1} + \overline{\beta}_t \alpha_{t+1} + K\overline{\beta}_t \beta_{t+1} \\ &= \overline{\alpha}_{t+1} + \overline{\alpha}_t \beta_{t+1} + \overline{\beta}_t \alpha_{t+1} + (1-\overline{\alpha}_t)\beta_{t+1} \\ &= \overline{\alpha}_{t+1} + \frac{(1-\overline{\alpha}_t)}{K} \alpha_{t+1} + \frac{(1-\alpha_{t+1})}{K} \\ &= \overline{\alpha}_{t+1} + \overline{\beta}_{t+1}. \end{aligned} \tag{30}$$

2) when $x \neq x_0$ in step $t+1$, we also have two situations in step t, that is, $x = x_0$ and $x \neq x_0$, so that we have:

$$\begin{aligned} \boldsymbol{Q}_{t+1} \boldsymbol{c}(x_0)_{(x)} &= \overline{\beta}_t(\alpha_{t+1} + \beta_{t+1}) + \overline{\beta}_t \beta_{t+1}(K-1) + \overline{\alpha}_t \beta_{t+1} \\ &= \overline{\beta}_t(\alpha_{t+1} + \beta_{t+1} + \beta_{t+1}(K-1)) + \overline{\alpha}_t \beta_{t+1} \\ &= \overline{\beta}_t + \overline{\alpha}_t \beta_{t+1} \\ &= \frac{(1-\overline{\alpha}_t)}{K} + \frac{\overline{\alpha}_t(1-\alpha_{t+1})}{K} \\ &= \overline{\beta}_{t+1}. \end{aligned} \tag{31}$$

The proof of formula (14) is completed.

Furthermore, we can see that when t is enough large, $\overline{\alpha}_t$ is close to 0. The formula (14) changes to $\overline{\boldsymbol{Q}}_t \boldsymbol{c}(x_0) = 1/K$. Thus we can derive the stationary distribution $p(\boldsymbol{x}_T) = [1/K, 1/K, \ldots, 1/K]$.

APPENDIX B
THE PROOF OF FORMULA (17)

We also use mathematical induction to prove formula (17). We have following conditional information:

$$\gamma_t \in [0,1], \beta_t = 1 - \gamma_t, \overline{\gamma}_t = 1 - \overline{\beta}_t, \overline{\beta}_t = \prod_{i=1}^{t} \beta_i. \tag{32}$$

Now we want to prove that $\overline{\boldsymbol{Q}}_t \boldsymbol{c}(x_0) = \overline{\beta}_t \boldsymbol{c}(x_0) + \overline{\gamma}_t \boldsymbol{c}(K+1)$. Firstly, when $t = 1$, we have:

$$\overline{\boldsymbol{Q}}_1 \boldsymbol{c}(x_0) = \begin{cases} \overline{\beta}_1, & x = x_0 \\ \overline{\gamma}_1, & x = K+1 \end{cases} \tag{33}$$

which is clearly hold. Suppose the formula (17) is hold at step t, then for $t = t+1$, we have:

$$\overline{\boldsymbol{Q}}_{t+1} \boldsymbol{c}(x_0) = \boldsymbol{Q}_{t+1} \overline{\boldsymbol{Q}}_t \boldsymbol{c}(x_0)$$

Now we consider two conditions:

1) when $x = x_0$ in step $t+1$, we only have one situation in step t, that is, $x = x_0$, so that we have:

$$\begin{aligned} \boldsymbol{Q}_{t+1} \boldsymbol{c}(x_0)_{(x)} &= \beta_{t+1} \overline{\beta}_t \\ &= \overline{\beta}_{t+1}. \end{aligned} \tag{34}$$

2) when $x = K+1$ in step $t+1$, we have two situations in step t, that is, $x = x_0$ and $x = K+1$, so that we have:

$$\begin{aligned} \boldsymbol{Q}_{t+1} \boldsymbol{c}(x_0)_{(x)} &= \overline{\gamma}_t + \overline{\beta}_t \gamma_{t+1} \\ &= 1 - \overline{\beta}_t + \overline{\beta}_t \gamma_{t+1} \\ &= 1 - \overline{\beta}_t(1 - \gamma_{t+1}) \\ &= 1 - \overline{\beta}_{t+1} \\ &= \overline{\gamma}_{t+1}. \end{aligned} \tag{35}$$

The proof of formula (17) is completed.

Similarly, when t is enough large, $\overline{\beta}_t$ is close to 0. The formula (17) can be written as $\overline{\boldsymbol{Q}}_t \boldsymbol{c}(x_0) = \overline{\gamma}_t \boldsymbol{c}(K+1)$, where $\overline{\gamma}_t = 1$. Thus we can derive the stationary distribution $p(\boldsymbol{x}_T) = [0, 0, \ldots, 1]$.

REFERENCES

[1] P. Chen, Y. Zhang, M. Tan, H. Xiao, D. Huang, and C. Gan, "Generating visually aligned sound from videos," *IEEE Trans. Image Process.*, vol. 29, pp. 8292–8302, 2020.

[2] Y. Zhou, Z. Wang, C. Fang, T. Bui, and T. L. Berg, "Visual to sound: Generating natural sound for videos in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3550–3558.

[3] V. Iashin and E. Rahtu, "Taming visually guided sound generation," in *Proc. Brit. Mach. Vis. Conf.*, 2021, pp. 1–15.

[4] X. Liu, T. Iqbal, J. Zhao, Q. Huang, M. D. Plumbley, and W. Wang, "Conditional sound generation using neural discrete time-frequency representation learning," in *Proc. IEEE Int. Workshop Mach. Learn. Signal Process.*, 2021, pp. 1–6.

[5] Q. Kong, Y. Xu, T. Iqbal, Y. Cao, W. Wang, and M. D. Plumbley, "Acoustic scene generation with conditional SampleRNN," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 925–929.

[6] K. Kumar et al., "Mel-GAN: Generative adversarial networks for conditional waveform synthesis," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2019, vol. 32, pp. 14910–14921.

[7] A. V. D. Oord et al., "Neural discrete representation learning," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 6309–6318.

[8] C. D. Kim, B. Kim, H. Lee, and G. Kim, "AudioCaps: Generating captions for audios in the wild," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2019, vol. 1 pp. 119–132.

[9] H. Chen, W. Xie, A. Vedaldi, and A. Zisserman, "VGGSound: A large-scale audio-visual dataset," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2020, pp. 721–725.

[10] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. Int. Conf. Ad. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 6840–6851.

[11] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2256–2265.

[12] J. Austin, D. Johnson, J. Ho, D. Tarlow, and R. v. d. Berg, "Structured denoising diffusion models in discrete state-spaces," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 17981–17993.

[13] S. Gu et al., "Vector quantized diffusion model for text-to-image synthesis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10696–10706.

[14] J. F. Gemmeke et al., "Audio set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2017, pp. 776–780.

[15] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 6629–6640.

[16] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1060–1069.

[17] Z. Zhang, Y. Xie, and L. Yang, "Photographic text-to-image synthesis with a hierarchically-nested adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6199–6208.

[18] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–35.

[19] S.-H. Lee, H.-W. Yoon, H.-R. Noh, J.-H. Kim, and S.-W. Lee, "Multi-spectrogan: High-diversity and high-fidelity spectrogram generation with adversarial style combination for speech synthesis," *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 14, pp. 13198–13206, 2021.

[20] H. Guo, F. K. Soong, L. He, and L. Xie, "A new GAN-based end-to-end TTS training algorithm," in *Proc. Interspeech*, 2019, pp. 1288–1292.

[21] J. N. Hurle, S. Lattner, and G. Richard, "DrumGAN: Synthesis of drum sounds with timbral feature conditioning using generative adversarial networks," in *Proc. 21st Int. Soc. Music Inform. Retrieval Conf. (ISMIR)*, 2020.

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

[23] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *Preprint*, 2018.

[24] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 12873–12883.

[25] M. Ding et al., "CogView: Mastering text-to-image generation via transformers," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 19822–19835.

[26] N. Parmar et al., "Image transformer," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4055–4064.

[27] A. Razavi, A. V. d. Oord, and O. Vinyals, "Generating diverse high-fidelity images with VQ-VAE-2," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2019, vol. 32, pp. 14866–14876.

[28] A. Ramesh et al., "Zero-shot text-to-image generation," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 8821–8831.

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

[30] Y. Wang et al., "Tacotron: Towards end-to-end speech synthesis," in *Proc. Interspeech*, 2017, pp. 4006–4010.

[31] P. Esser, R. Rombach, A. Blattmann, and B. Ommer, "ImageBART: Bidirectional context with multinomial diffusion for autoregressive image synthesis," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 3518–3532.

[32] P. Dhariwal and A. Nichol, "Diffusion models beat GANS on image synthesis," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 3518-3532.

[33] A. Q. Nichol et al., "GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 16784–16804.

[34] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "DiffWave: A versatile diffusion model for audio synthesis," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–17.

[35] M. Jeong, H. Kim, S. J. Cheon, B. J. Choi, and N. S. Kim, "Diff-TTS: A denoising diffusion model for text-to-speech," in *Proc. InterSpeech*, 2021, pp. 3605–3609.

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

[37] S.-G. Lee et al., "PriorGrad: Improving conditional denoising diffusion models with data-driven adaptive prior," in *Poc. Int. Conf. Learn. Representations*, 2022, pp. 1–19.

[38] E. Hoogeboom, D. Nielsen, P. Jaini, P. Forré, and M. Welling, "Argmax flows and multinomial diffusion: Learning categorical distributions," *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 12454–12465, 2021.

[39] J. D. M.-W. C. Kenton and L. K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.

[40] A. Radford et al., "Learning transferable visual models from natural language supervision," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 8748–8763.

[41] X. Tan, T. Qin, F. Soong, and T.-Y. Liu, "A survey on neural speech synthesis," 2021, *arXiv:2106.15561*.

[42] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1125–1134.

[43] F. Schmidt, "Generalization in generation: A closer look at exposure bias," in *Proc. 3rd Workshop Neural Generation Transl.*, 2019, pp. 157–167.

[44] D. Griffin and J. Lim, "Signal estimation from modified short-time Fourier transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 2, pp. 236–243, Apr. 1984.

[45] A. V. d. Oord et al., "WaveNet: A generative model for raw audio," 2016, *arXiv:1609.03499*.

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

[47] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 8162–8171.

[48] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 41–48.

[49] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2818–2826.

[50] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[51] M. Wu, H. Dinkel, and K. Yu, "Audio caption: Listen and tell," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 830–834.

[52] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: An audio captioning dataset," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2020, pp. 736–740.

[53] X. Mei, X. Liu, Q. Huang, M. D. Plumbley, and W. Wang, "Audio captioning transformer," in *Proc. Detection Classification Acoust. Scenes Events Workshop*, 2021, pp. 211–215.

[54] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "SPICE: Semantic propositional image caption evaluation," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 382–398.

[55] R. Vedantam, C. L. Zitnick, and D. Parikh, "CIDER: Consensus-based image description evaluation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 4566–4575.

[56] D. P. Kingma and J. Ba, "ADAM: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[57] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, *arXiv:1711.05101*.

[58] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.