FOLEY SOUND SYNTHESIS WITH A CLASS-CONDITIONED LATENT DIFFUSION MODEL

Robin Scheibler, Takuya Hasumi, Yusuke Fujita, Tatsuya Komatsu,
Ryuichi Yamamoto, and Kentaro Tachibana

LINE Corporation, Tokyo, Japan

ABSTRACT

We propose a competitive Foley sound synthesis system based on available components and fine-tuned on a target dataset. We reuse a text-to-audio pre-trained model composed of a latent diffusion model (LDM), trained on AudioCaps, a variational autoencoder (VAE), and a vocoder. We fine-tune the LDM on the development dataset of the DCASE 2023 Task 7 to output a latent representation conditioned on the target class number. The VAE and vocoder are then used to generate the waveform from the latent representation. To improve the quality of the generated samples, we utilize a post-processing filter that selects a subset of generated sounds to match a distribution of target class sounds. In experiments, we found that our system achieved an average Fréchet audio distance (FAD) of 4.744, which is significantly better than 9.702 produced by the baseline system of the DCASE 2023 Challenge Task 7. In addition, we perform ablation studies to evaluate the performance of the system before fine-tuning and the effect of sampling rate on the FAD.

Index Terms— Foley sound synthesis, conditional sound generation, latent diffusion, Fréchet audio distance

1. INTRODUCTION

Foley sound synthesis is the task of generating sound effects added to multimedia content to enhance the perceptual audio experience. Originally developed for cinema and television, it was conducted by skilled artisans using elaborate manual techniques and is named after Jack Donovan Foley, one of their pioneers [1]. Interestingly, Foley sound effects are perceived as more authentic than their real counterparts captured in live recordings [2]. The potential of digital signal processing for sound synthesis was recognized early, as exemplified by the famous Karplus-Strong algorithm [3]. The recent deep learning revolution has brought the realism levels of digital sound synthesis to new levels, in particular for speech [4] and music [5, 6, 7]. A few works have focused specifically on Foley sound synthesis either with video-guidance [8, 9], or without [10, 11, 12, 13]. We can also mention some niche applications like synthesizing cricket sounds as test signals for perceptual experiments [14].

Recently, text-to-audio sound generation based on diffusion models has gained traction. Following a methodology introduced for images [15], AudioLDM [16] has demonstrated impressive conditional sound generation quality. AudioLDM is composed of a latent diffusion model (LDM), a variational autoencoder (VAE), and a neural vocoder. The LDM is conditioned on a text prompt through a Contrastive Language-Audio Pretraining (CLAP) embedding. The latent representation is provided by the VAE which has learned to encode a mel-spectrogram into a compressed latent space. The neural vocoder is based on HiFi-GAN [17], and decodes a waveform from the mel-spectrogram into an uncompressed waveform. Tango [18] has been proposed to enhance the text prompting functionality of AudioLDM using an instruction-tuned large language model (LLM) instead of the CLAP embedding.

Due to this rising interest in general sound generation, a new related task was added to the DCASE 2023 Challenge to stimulate research about this challenging problem. Task 7 [19], aptly called *Foley sound synthesis*, requires participants to build a Foley sound generation model for seven sound classes: dog bark (🐕), footstep (👣), gunshot (🔫), keyboard (💻), moving motor vehicle (🚗), rain (🌧️), and sneeze/cough (🤧). The system is then used to produce 100 samples that are first evaluated in terms Fréchet audio distance. In a second stage, a subjective evaluation based on quality, accuracy, and diversity of the samples is conducted. Although text prompting models such AudioLDM or Tango have shown to be effective in fine-grained guidance for audio generation, using them for class conditional generation requires prompt engineering and trial and error. In addition, existing public models have been trained for the generation of 10 s long samples at 16 kHz, while the challenge calls for 4 s samples at 22.05 kHz.

We propose a pragmatic solution to this problem by combining available components to obtain a high quality Foley sound synthesis system. We modify an existing implementation of Tango[1] to enable sound-class-based guidance instead of text prompting. The class-conditioned LDM is trained using the development set of the DCASE 2023 Challenge Task that contains between 600 and 800 sounds of each of the classes. We initialize the model with a pre-trained model of Tango, which was trained with AudioCaps [20] dataset and Flan-T5 [21] LLM. The conditioning part based on Flan-T5 is replaced with a simple linear embedding layer to realize sound-class-based conditioning. Moreover, we propose a post-processing filter that selects a subset of generated samples to match a distribution of the target sound class. The post-processing filter adopts a greedy backward selection strategy that iteratively drops a sample to achieve the minimum Fréchet audio distance (FAD). Our experiments show that our system significantly outperforms the baseline system provided by the task organizers in terms of FAD. Audio samples produced by the system are available online[2].

2. BACKGROUND

Our system, like AudioLDM [16] and Tango [18], is based on the LDM originally proposed for image generation [15]. The LDM operates in the latent-space generated by a VAE pre-trained on mel-spectrograms. The generated mel-spectrograms are inverted into waveforms using the neural vocoder HiFi-GAN [17].

[1]https://github.com/declare-lab/tango
[2]http://www.robinscheibler.org/dcase23t7-samples/

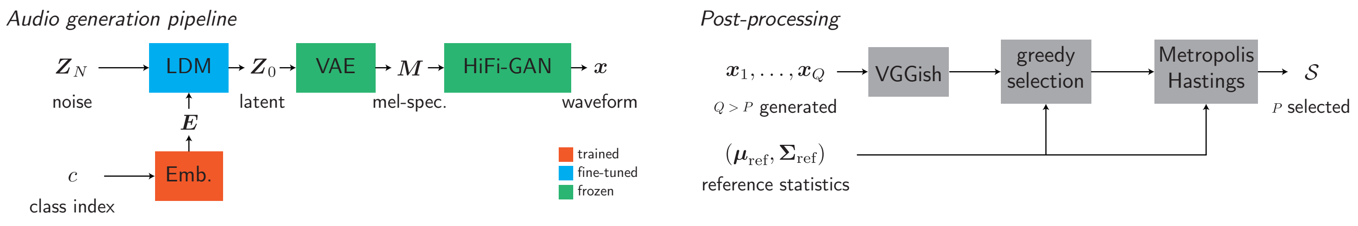

Figure 1: System overview. The audio generation pipeline (left) has three elements. The core is a latent diffusion model (LDM) with class-conditional embeddings (Emb.). We use pre-trained variational-autoencoder and HiFi-GAN vocoder for the reconstruction. The samples produced are then filtered during post-processing (right) by greedy and Metropolis-Hastings optimization.

2.1. Latent Diffusion Models

The LDM transforms a sampled Gaussian noise $\boldsymbol{Z}_N \in \mathbb{R}^{C \times \frac{T}{r} \times \frac{F}{r}}$ into a latent representation $\boldsymbol{Z}_0$ through N reverse diffusion steps with a UNet-based neural network. T is the number of mel-spectrogram frames, F is the number of mel-filter bins, C is the number of channels in latent space, and r is the compression level of VAE. The neural network receives a L-length sequence of d-dimensional embedding vectors $\boldsymbol{E} \in \mathbb{R}^{L \times d}$ transformed from the sound class indices through a linear embedding layer. The conditioner $\boldsymbol{E}$ is fed into the network through the cross-attention mechanism.

Given the latent feature $\boldsymbol{Z}_0$, the corresponding class embedding vector $\boldsymbol{E}$, and random isotropic Gaussian noise $\bar{\boldsymbol{\varepsilon}} \in \mathbb{R}^{C \times \frac{T}{r} \times \frac{F}{r}}$, the neural network is trained to minimize the following loss function $\mathcal{L}$ on the basis of the theory of denoising diffusion probabilistic models [22]:

$$\mathcal{L} = \mathbb{E}_{\bar{\boldsymbol{\varepsilon}}, \boldsymbol{Z}_0, n} \left[\| \bar{\boldsymbol{\varepsilon}} - \boldsymbol{\varepsilon}(\sqrt{\bar{\alpha}_n} \boldsymbol{Z}_0 + \sqrt{1 - \bar{\alpha}_n} \bar{\boldsymbol{\varepsilon}}, \boldsymbol{E}, n; \theta) \|_2^2 \right], \quad (1)$$

$$\bar{\alpha}_n = \prod_{n'=1}^{n} \alpha_{n'}, \quad (2)$$

$$\alpha_n = 1 - \beta_n, \quad (3)$$

where $\boldsymbol{\varepsilon}(\cdot,\cdot,\cdot;\theta)$ is the neural network that outputs the estimated noise of the same shape as $\boldsymbol{Z}_0$. The variable β_n is the variance of the Gaussian distribution in the forward process.

Classifier-free guidance [23] can be used to boost the fidelity of the sound class. Using this technique, the backward process to obtain $\boldsymbol{Z}_{n-1}$ from $\boldsymbol{Z}_n$ can be written

$$\boldsymbol{Z}_{n-1} = \frac{1}{\sqrt{\alpha_n}} \left(\boldsymbol{Z}_n - \frac{\beta_n}{\sqrt{1 - \bar{\alpha}_n}} \tilde{\boldsymbol{\varepsilon}}_n \right) + \sigma_n \tilde{\boldsymbol{\varepsilon}}_n \quad (4)$$

$$\tilde{\boldsymbol{\varepsilon}}_n = w \boldsymbol{\varepsilon}(\boldsymbol{Z}_n, \boldsymbol{E}, n; \theta) + (1 - w) \boldsymbol{\varepsilon}(\boldsymbol{Z}_n, \boldsymbol{O}, n; \theta), \quad (5)$$

where σ_n^2 is the variance of the Gaussian distribution in the reverse process, respectively. The symbol $\boldsymbol{O}$ represents the all zero embedding vector used for unconditional inference, and w is a parameter of the guidance scale. Note that, to accelerate the sampling speed at inference time, denoising diffusion implicit models (DDIM) [24] sampling is used.

2.2. Variational autoencoder and neural vocoder

A VAE is used to compress a mel-spectrogram $\boldsymbol{M} \in \mathbb{R}^{T \times F}$ into the latent space parametrized by mean and variance $\mu, \sigma \in \mathbb{R}^{C \times \frac{T}{r} \times \frac{F}{r}}$. The VAE is composed of a stack of CNN-based encoders. In the submitted system pipeline, the latent $\boldsymbol{Z}_0$ produced by the LDM is fed into the decoder of VAE to reconstruct a mel-spectrogram $\boldsymbol{M}$.

To reconstruct a waveform $\boldsymbol{x} \in \mathbb{R}^{T'}$ from a mel-spectrogram $\boldsymbol{M}$ given by the VAE, the generator of HiFi-GAN [17] can be used, where T' is a length of the waveform. The module repeatedly upsamples the mel-spectrograms by a transposed convolution followed by multi-receptive field fusion (MRF). The MRF is composed of residual blocks, where each block processes the inputs by convolutions of multiple kernel sizes and dilations to capture the temporal feature by various receptive fields.

3. PROPOSED SYSTEM OVERVIEW

An overview of our submitted system is depicted in Fig. 1. Our system adopts a similar pipeline with Tango [18], where a latent generator based on LDM, a latent-to-mel decoder using VAE, and a mel-to-wav vocoder are cascaded. Our LDM accepts a sound class index c as a conditioner instead of a text prompt. We use pre-trained VAE and HiFi-GAN models used in AudioLDM [16] to reconstruct a waveform from the latent representation. After the audio generation pipeline, a post-processing filter is employed to drop irrelevant samples to match the distribution of a target sound class. In the following subsections, we describe our implementation of the modules.

3.1. Sound-class-based Conditioning

When training, we initialize our model with a pre-trained checkpoint of Tango. The checkpoint is designed to receive a sequence of embedding vectors $\boldsymbol{E}$ from the Flan-T5 text encoder. We replace the text encoder with a linear embedding layer that projects a sound class index c into a d-dimentional vector. Unlike Tango, we jointly train the conditioner with the main network of LDM. Although the cross-attention mechanism for conditioning accepts a sequence of embedding vectors, which is designed to accept a text sequence, we use a single target class embedding vector as $\boldsymbol{E} \in \mathbb{R}^{1 \times d}$ in this work.

3.2. FAD-oriented Post-processing Filter

The quality of the samples produced by the system, while acceptable, can be improved by over-generating and filtering. For this task, a target sample quality metric is necessary. The FAD metric used in the challenge is an obvious choice. The FAD is computed as follows. First, VGGish [25] embeddings of both the reference and generated samples are computed. The embeddings are computed for segments of 16,000 samples with half-overlap. This produces 10 embedding vectors per 4 s of generated audio. We note that the challenge abuses the metric a little bit since the VGGish model was trained on 16 kHz data, while the challenge uses 22.05 kHz. The mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ of the embedding vectors of both reference and generated audio are computed and their Fréchet distance [26] is

$$\begin{aligned}\mathrm{FAD}(\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r, \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g) = \\ \|\boldsymbol{\mu}_r - \boldsymbol{\mu}_g\|^2 + \mathrm{tr}\left(\boldsymbol{\Sigma}_r + \boldsymbol{\Sigma}_g - 2(\boldsymbol{\Sigma}_r\boldsymbol{\Sigma}_g)^{1/2}\right). \quad (6)\end{aligned}$$

To obtain P samples, we first generate Q samples, with $Q > P$. Then, we first reduce the number of samples by greedy selection. We start with the set of all Q samples, denoted $\mathcal{S} = \{1, \ldots, Q\}$. At each iteration, we remove sample k whose absence decreases the FAD most, i.e.,

$$k = \arg\min_{\ell \in \mathcal{S}} \mathrm{FAD}(\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r, \bar{\boldsymbol{\mu}}_g^{(\mathcal{S}-\ell)}, \bar{\boldsymbol{\Sigma}}_g^{(\mathcal{S}-\ell)}), \quad (7)$$

where $\bar{\boldsymbol{\mu}}_g^{(\mathcal{S}-\ell)}$ and $\bar{\boldsymbol{\Sigma}}_g^{(\mathcal{S}-\ell)}$ are the mean and covariance matrix, respectively, after removing the ℓth sample. Then, we update $\mathcal{S} \leftarrow \mathcal{S} - \{k\}$, where "$-$" here is the set difference operator. We repeat until the size of $\mathcal{S}$ is P, or no sample can be removed without the FAD increasing.

If we still have more than P samples, we apply the Metropolis-Hastings (MH) algorithm [27] to find a good sub-set of P elements. We initialize the algorithm by uniformly sampling at random 100 subsets of P samples and picking the one with lowest FAD. At each iteration of the algorithm, we randomly swap two samples. We first pick at random one of the current P samples. Then, we pick one of the discarded samples with probability inversely proportional to the embedding distance to the first sample. We swap the two samples and evaluate the FAD. If it decreases, we accept the change. If it increases, we only accept the change with a small probability that decreases over time with a linear schedule. Otherwise, we reject the change. The subset with the lowest FAD over all iterations is returned by the algorithm.

We note that such a filtering system allows to achieve an arbitrarily small FAD, at the cost of generating an increasingly large number of samples. In our final system where $P = 100$, we set $Q = 200$ to strike a balance between FAD performance and generation time.

4. EXPERIMENTS

4.1. Effect of Sampling Rate on FAD

Our generative model operates at 16 kHz and thus requires upsampling to 22.05 kHz to match the dataset. This means that the frequency band from 8 kHz to 11.025 kHz will be empty. We test the effect of this by downsampling the development dataset to 16 kHz and then back up to 22.05 kHz. Table 1 shows the FAD of the development dataset, i.e., computed with test set as reference[3], before and after this operation. The FAD increases by less than 1, which seems acceptable for our purpose. The effect varies by class, and surprisingly the FAD decreases for the *rain* class.

[3]The mean vector and covariance matrix of the VGGish embeddings of the test set were provided by the task organizers for the purpose of computing the FAD with respect to the hidden test set.

Table 1: Fréchet audio distance (FAD) of the development dataset under several selection procedure and sampling frequencies. The column *full* is the full development set. *100 random* indicates that we chose at random a 100 samples from each class. The result at different stages of the post-filtering a under *optimization*. The latter is further divided into the result of the greedy optimization, which may have more than 100 sample, the random initialization of MH (+*rand.*), and the final MH stage.

class	full	100 random		optimized (22 kHz)		
	22 kHz	22 kHz	16 kHz	greedy†	+rand.	+MH
🐕	1.144	1.883	2.954	0.609	0.802	0.793
👣	2.072	2.388	3.846	0.715	0.862	0.837
🔫	2.606	3.036	4.368	0.667	0.899	0.887
💻	2.772	3.210	3.067	0.441	0.460	0.460
🚗	4.324	5.426	7.950	1.210	1.281	1.281
🌧️	3.007	3.624	3.423	0.783	0.902	0.902
🤧	0.400	0.768	1.367	0.188	0.268	0.265
avg.	2.332	2.904	3.854	0.660	0.782	0.775

† More than 100 samples

4.2. FAD-based Post-filtering

We evaluate the post-filtering described in Section 3.2 on the development dataset of the DCASE 2023 Task 7. We evaluate the FAD at different stages of the post-filtering pipeline and show the results in Table 1, under the *optmized* column. We see that the greedy stage is very effective and reduces the FAD to 23.5 % of the random selection. However, there may still be more than 100 samples at this stage. Selecting 100 samples out the ones selected by the greedy approach increases slightly the FAD by about 0.22 points. The effect of the MH algorithm is not always effective, but can further reduce the FAD by up to 0.03 points in the best case.

4.3. Foley Sound Synthesis

4.3.1. Models and Hyperparameters

HiFi-GAN and VAE: We used pre-trained checkpoints of HiFi-GAN and VAE used in [16]. The HiFi-GAN model was trained with AudioSet [28]. All the training data were segmented or padded into 10 seconds and resampled to 16 kHz, i.e., $T' = 160,000$. Each audio sample was transformed into a 64-dim Mel-spectrogram ($F = 64$) with a window length of 1024, and a hop length of 160. The number of frames T was 1024 by padding 24 frames to avoid further padding with downsampling operations in VAE and LDM. The VAE model was trained with AudioSet [28], AudioCaps [20], Freesound [4], and BBCSFX [5]. The compression level r was 4, and the number of channels C was 8.

Baselines: We use two baselines. 1) The challenge baseline systems [19], itself based on [12]. It uses a VQ-VAE for compression to latent space and HiFi-GAN for signal reconstruction. For latent generation, it uses an auto-regressive model based on Pixel-SNAIL [29]. This model operates at 22.05 kHz. 2) We also evaluate the direct use of Tango [18] for the task. We condition the generation on text prompts designed for each of the target classes — 🐕: "A dog barking", 👣: "Footsteps", 🔫: "Gun shot", 💻: "Typing

[4]https://freesound.org/

[5]https://sound-effects.bbcrewind.co.uk

Table 2: Fréchet audio distance (FAD) with baseline and our systems. ’raw’ indicates the system without the FAD filter, i.e., the first 100 samples from the audio generation pipeline were used. ’filtered’ indicates our submitted system with the FAD filter.

class	baseline	Tango	ours	
	[19]	[18]	raw	filtered
🐕	13.411	6.031	5.835	3.816
👣	8.109	11.889	11.209	8.227
🔫	7.951	8.082	5.790	3.427
💻	5.230	11.014	3.698	2.758
🚗	16.108	14.636	11.440	6.837
🌨️	13.337	8.550	7.031	5.399
🤧	3.770	9.450	3.658	2.741
avg.	9.702	9.450	6.952	4.744

a keyboard”, 🚗: “motor vehicle moving”, 🌨️: “Heavy rain”, and 🤧: “woman sneezing.” Since Tango produces 10 s long samples at 16 kHz, we truncate them to 4 s and upsample to 22.05 kHz.

Proposed System: We initialized our LDM using a checkpoint of Tango [6]. The model used the conditioning vector dimension $d = 1024$. The initial checkpoint was trained with AudioCaps [20]. For fine-tuning, we used the DCASE2023 Task7 development set. Since the audio data were sampled at 22.05 kHz and segmented in four seconds, we resampled them to 16 kHz and padded them into 10 seconds. We set $N = 1000$ forward diffusion steps for fine-tuning. Our LDM was fine-tuned with the AdamW optimizer with an initial learning rate of 3e-5 and a linear decay learning rate scheduler. We fine-tuned the model for 100k training iterations, with an effective batch size of 42 using seven A100 GPUs. In the inference phase, we used DDIM [24] for 100 sampling steps and a classifier-free guidance scale of $w = 3$. As our model produces a 10-second audio segment at a 16 kHz sampling rate, we extracted the first four-second segment and resampled it to 22.05 kHz to fit the challenge rule.

Post-processing: For each sound class, we generated $Q = 200$ samples with the aforementioned audio generation pipeline. Then the FAD filter is applied to reduce the number of samples to $P = 100$.

4.3.2. Results

Table 2 shows FAD of the baselines and our system with respect to the evaluation set. First, we observe that the Tango model, which was trained on a much larger dataset, but did not include the Task 7 development set, performs slightly better than the baseline. This shows the effectiveness of a large, diverse dataset. It is likely that further prompt engineering would improve the result. The improvement seems to come from large reductions of FAD on the dog bark and rain classes. We note that both the baseline and Tango could be improved by the post-filtering, which we did not try.

Our fine-tuned model conditioned on class embeddings performs better accross the board, with the exception of footstep, where the baseline is better. However, informal listening tests revealed the samples to be of good quality regardless. We also point out that the keyboard class nearly saturates the lower bound FAD of 100 random samples of the development set at 16 kHz (see Table 1). We also see that the moving motor vehicle class is fairly difficult, as suggested by the FAD of this class in the development set.

With the FAD filter, the FADs were significantly reduced regardless of the sound classes. Specifically, we achieve a reduction of 32 % of the average FAD compared to the raw outputs. The results demonstrate that the proposed audio generation pipeline can generate class-specific audio samples with sufficient diversity, and that the proposed FAD filter can select a subset of them with the statistics of the target sound class.

[6]https://huggingface.co/declare-lab/tango

5. CONCLUSION

We proposed a system based on class-conditioned latent diffusion model for the DCASE2023 Task7: Foley sound synthesis. We efficiently make use of existing models that we adapt to the task and pair with some signal processing for adaptation and output filtering. We fine-tune pre-trained model for text-to-audio generation on the development set of Task 7, and modifies its conditioning mechanism to use class embedding vectors. Our post-filtering system uses greedy and global optimization to select a set of samples matching the statistics of the target evaluation set and decrease the FAD. We found this technique very successful. However, we also noticed during preliminary experiments that the audio quality of samples selected this way did not always match the FAD score obtained. This highlights the importance of generating good samples in the first stage of the system. Overall, our submission system achieved significantly better FAD scores than both the challenge baseline and an out-of-the-box text-to-audio model.

An important aspect that is yet to be understood is how to leverage unlabelled data for pre-training, since training sets for sound effects may be small depending on the target class.

6. REFERENCES

[1] S. Pauletto, “Foley performance and sonic implicit interactions,” in *The Body in Sound, Music and Performance: Studies in Audio and Sonic Arts*, L. O Keefe and I. Nogueira, Eds. Abingdon, Oxon: Routledge, 2023.

[2] L. M. Heller and L. Wolf, “When hybrid sound effects are better than real recordings,” *Proc. Meet. Acoust.*, vol. 46, no. 1, p. 050002, May 2022.

[3] K. Karplus and A. Strong, “Digital synthesis of plucked-string and drum timbres,” *Computer Music Journal*, vol. 7, pp. 43–55, 1983.

[4] X. Tan, T. Qin, F. Soong, and T.-Y. Liu, “A survey on neural speech synthesis,” July 2021, arXiv:2106.15561 [cs, eess].

[5] C. Hernandez-Olivan and J. R. Beltrán, “Music Composition with Deep Learning: A Review,” in *Advances in Speech and Music Technology: Computational Aspects and Applications*, ser. Signals and Communication Technology, A. Biswas, E. Wennekes, A. Wieczorkowska, and R. H. Laskar, Eds. Cham: Springer International Publishing, 2023, pp. 25–50.

[6] A. Agostinelli, T. I. Denk, Z. Borsos, *et al.*, “MusicLM: Generating music from text,” Jan. 2023, arXiv:2301.11325 [cs, eess].

[7] M. Pasini and J. Schlüter, “Musika! Fast infinite waveform music generation,” in *Proc. ISMIR*, Bengaluru, IN, Dec. 2022.

[8] S. Andreu and M. Villanueva Aylagas, "Neural Synthesis of Sound Effects Using Flow-Based Deep Generative Models," *Proc. AAAI Conf. Artif. Intell. Interact. Digit. Entertain.*, vol. 18, no. 1, pp. 2–9, Oct. 2022.

[9] S. Ghose and J. J. Prevost, "AutoFoley: Artificial synthesis of synchronized sound tracks for silent videos with deep learning," *IEEE Trans. Multimedia*, vol. 23, pp. 1895–1907, June 2021.

[10] A. Barahona-Rıos and S. Pauletto, "Synthesising knocking sound effects using conditional WaveGAN," in *Proc. 17th Sound and Music Computing Conference*, Torino, IT, June 2020.

[11] S. Li, L. Zhang, C. Dong, *et al.*, "FastFoley: Non-autoregressive Foley Sound Generation Based on Visual Semantics," in *Man-Machine Speech Communication*, ser. Communications in Computer and Information Science, L. Zhenhua, G. Jianqing, Y. Kai, and J. Jia, Eds. Singapore: Springer Nature, 2023, pp. 252–263.

[12] X. Liu, T. Iqbal, J. Zhao, *et al.*, "Conditional sound generation using neural discrete time-frequency representation learning," in *Proc. MLSP*, Oct. 2021, pp. 1–6.

[13] S. Pascual, G. Bhattacharya, C. Yeh, J. Pons, and J. Serrà, "Full-band general audio synthesis with score-based diffusion," in *Proc. ICASSP*, Rhodes, GR, June 2023, pp. 1–5.

[14] M. Oliveira, V. Almeida, J. Silva, and A. Ferreira, "Analysis and re-synthesis of natural cricket sounds assessing the perceptual relevance of idiosyncratic parameters," in *Proc. ICASSP*, Rhodes, GR, June 2023, pp. 1–5.

[15] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. IEEE ICCV*, 2022, pp. 10 684–10 695.

[16] H. Liu, Z. Chen, Y. Yuan, *et al.*, "AudioLDM: Text-to-audio generation with latent diffusion models," *arXiv preprint arXiv:2301.12503*, 2023.

[17] J. Kong, J. Kim, and J. Bae, "HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 17 022–17 033.

[18] D. Ghosal, N. Majumder, A. Mehrish, and S. Poria, "Text-to-audio generation using instruction tuned LLM and latent diffusion model," *arXiv preprint arXiv:2304.13731*, 2023.

[19] K. Choi, J. Im, L. Heller, *et al.*, "Foley sound synthesis at the DCASE 2023 challenge," *arXiv preprint arXiv:2304.12521*, 2023.

[20] C. D. Kim, B. Kim, H. Lee, and G. Kim, "AudioCaps: Generating captions for audios in the wild," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, June 2019, pp. 119–132.

[21] H. W. Chung, L. Hou, S. Longpre, *et al.*, "Scaling instruction-finetuned language models," *arXiv preprint arXiv:2210.11416*, 2022.

[22] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[23] J. Ho and T. Salimans, "Classifier-free diffusion guidance," *arXiv preprint arXiv:2207.12598*, 2022.

[24] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *Proc. ICML*, 2021. [Online]. Available: https://openreview.net/forum?id=St1giarCHLP

[25] S. Hershey, S. Chaudhuri, D. P. W. Ellis, *et al.*, "CNN architectures for large-scale audio classification," in *Proc. ICASSP*, New Orleans, LA, USA, Mar. 2017, pp. 131–135.

[26] D. C. Dowson and B. V. Landau, "The Fréchet distance between multivariate normal distributions," *Journal of Multivariate Analysis*, vol. 12, no. 3, pp. 450–455, Sept. 1982.

[27] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, June 1953.

[28] J. F. Gemmeke, D. P. Ellis, D. Freedman, *et al.*, "Audio Set: An ontology and human-labeled dataset for audio events," in *Proc. ICASSP*. New Orleans, LA, USA: IEEE, 2017, pp. 776–780.

[29] X. I. Chen, N. Mishra, M. Rohaninejad, and P. Abbeel, "PixelSNAIL: An improved autoregressive generative model," in *Proc. ICML*. PMLR, July 2018, pp. 864–872.