RFM-EDITING: RECTIFIED FLOW MATCHING FOR TEXT-GUIDED AUDIO EDITING

Liting Gao[1], *Yi Yuan*[1], *Yaru Chen*[1], *Yuelan Cheng*[1], *Zhenbo Li*[2], *Juan Wen*[2], *Shubin Zhang*[3], *Wenwu Wang*[1]

[1] Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, United Kingdom
[2] College of Information and Electrical Engineering, China Agricultural University, China
[3] Fisheries College, Ocean University of China, China

ABSTRACT

Diffusion models have shown remarkable progress in text-to-audio generation. However, text-guided audio editing remains in its early stages. This task focuses on modifying the target content within an audio signal while preserving the rest, thus demanding precise localization and faithful editing according to the text prompt. Existing training-based and zero-shot methods that rely on full-caption or costly optimization often struggle with complex editing or lack practicality. In this work, we propose a novel end-to-end efficient rectified flow matching-based diffusion framework for audio editing, and construct a dataset featuring overlapping multi-event audio to support training and benchmarking in complex scenarios. Experiments show that our model achieves faithful semantic alignment without requiring auxiliary captions or masks, while maintaining competitive editing quality across metrics.

Index Terms— Audio editing, rectified flow matching, diffusion model, CLAP score, audio editing dataset

1. INTRODUCTION

Recent advances in diffusion-based modeling have led to remarkable progress in text-to-audio (TTA) generation, with examples including denoising diffusion probabilistic model (DDPM) [1] based methods (e.g., AudioLDM [2, 3], Make-An-Audio [4,5]) and flow [6] based methods (e.g., TangoFlux [7]). Text-guided audio editing aims to modify existing audio based on natural language instructions or target descriptions while preserving the unaltered content. This enables flexible audio manipulation through prompts and supports applications in sound design, post-production, and personalized audio generation. However, research on text-guided audio editing, including training-free diffusion-inversion methods [8–10] and training-based models [11, 12], remains limited in performance and at an early stage.

Training-free audio editing typically leverages pre-trained TTA diffusion models [2, 13], inverting diffusion to recover latent noise from input audio and guiding denoising with textual prompts. Methods such as AudioEditor [8] use denoising diffusion implicit model (DDIM) inversion and null-text optimization for high-fidelity edits, while prompt-guided precise audio editing (PPAE) [9] and DDPM inversion Zero-Shot [10] manipulate cross-attention maps for localized control via semantic shifts between source and target prompts. They all introduce the Prompt-to-Prompt attention replacement mechanism [14] into audio editing to accurately align with the target text and significantly improve the CLAP score. WavCraft [15] further extends this paradigm by using large language models (LLMs) to translate prompts into expert-module instructions for flexible editing.

By contrast, AUDIT [11] trains a latent diffusion model (LDM) [16] with triplet data for instruction-guided audio editing. Non-rigid prompt editing [12] fine-tunes a diffusion model on audio-caption pairs and performs edits via interpolation in prompt embedding space. Although training-based methods [11] enable instruction following through explicit supervision, their progress is limited by the scarcity of large-scale datasets, making it difficult to accurately localize edit regions while preserving the rest, especially in complex scenarios with overlapping sounds. In contrast, training-free methods offer flexibility without labeled data, but often require costly null-text optimization in inference [8]. Furthermore, some methods depend on full captions [8–10, 12] or modified token masks [8] rather than concise editing instructions, which is time-consuming and impractical. Since audio is typically not accompanied by detailed textual descriptions, we argue that an ideal audio editing system should operate from raw audio and editing instructions, as in [11].

To address these limitations, we introduce an efficient end-to-end text-guided audio editing framework based on rectified flow matching (RFM) [17], dubbed RFM-Editing. It adopts a training paradigm that learns localized velocity fields directly from instructions rather than explicit masks or captions. To support training, we construct a large-scale audio editing dataset with overlapping multi-event audio from AudioCaps2 [18]. RFM-Editing achieves competitive performance and distributional consistency across add, remove, and replace scenarios without costly inference-time optimization, even under complex overlapping events.

Project page: https://katelin-glt.github.io/RFM-Editing-Demo/

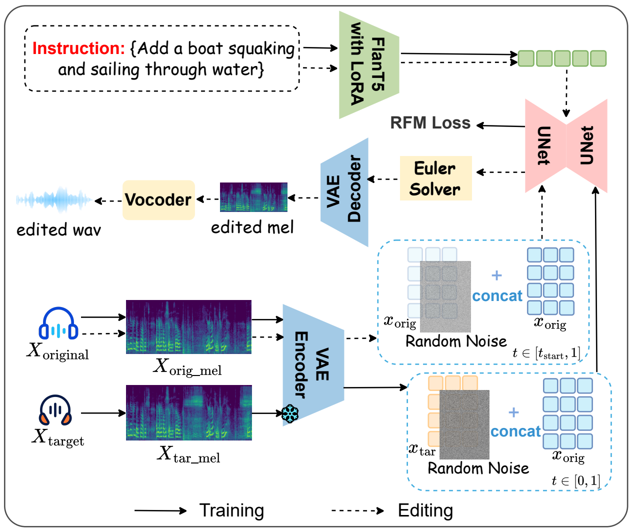


Fig. 1: The training and editing pipeline of RFM-Editing.

2. PROPOSED METHOD

Fig. 1 shows the training and inference-time editing pipeline of RFM-Editing, the first unified RFM-based instruction-guided audio editing model that jointly trains three editing tasks. Built upon the LDM [16], RFM-Editing integrates an audio feature extractor, a low-rank adaptation (LoRA [19])-tuned text encoder for instruction understanding, a U-Net for text-guided latent editing, and a BigVGAN vocoder [20] for waveform reconstruction. RFM-Editing also concatenates latent with original features channel-wise and resets reverse diffusion initialization state to preserve unedited regions.

2.1. Training with Rectified Flow Matching

RFM-Editing converts input audio to a target version based on an instruction. It is trained on original audio clips X_{original}, their edited counterparts X_{target}, and the corresponding instructions $\mathcal{I}$ by optimizing the rectified flow matching objective. Both X_{original} and X_{target} are converted into log-mel spectrograms $X_{\text{orig\_mel}} \in \mathbb{R}^{T\times F}$ and $X_{\text{tar\_mel}} \in \mathbb{R}^{T\times F}$, where T and F denote the dimensions of time and frequency, respectively. A pre-trained Variational Auto-Encoder (VAE) [21] encodes the spectrograms into latent space, where x_{orig} is the latent representation of the original audio and x_{tar} that of the target audio. To better capture editing instructions, we apply LoRA [19] to the Flan-T5 text encoder [22], freezing pre-trained weights and inserting trainable low-rank matrices into transformer layers, reducing the number of trainable parameters while improving text understanding for precise edits.

We apply the objective of RFM [17] to learn a continuous vector field that maps samples from a noisy distribution to the target distribution of the edited audio using a U-Net-based architecture. Compared to standard diffusion models relying on stochastic differential equations (SDEs), RFM formulates a deterministic ordinary differential equation (ODE) process that models a straight-line trajectory from noise ϵ to target x_{tar}, eliminating the need for fine-grained time discretization and leading to stable and efficient training. Specifically, we add random Gaussian noise ϵ to x_{tar} and obtain a perturbed latent x_t in a continuous time step $t \in [0, 1]$. The perturbed sample x_t is computed along a straight interpolation path from noise to data:

$$x_t = (1 - (1 - \sigma_{\min}) \cdot t) \cdot \epsilon + t \cdot x_{\text{tar}} \tag{1}$$

where $\sigma_{\min}$ is a small constant controlling the minimal scale of noise at $t = 0$. The time derivative of the interpolation path yields the ground-truth velocity field at any time step t:

$$\mathbf{v}_{\text{target}} = \frac{dx_t}{dt} = x_{\text{tar}} - (1 - \sigma_{\min}) \cdot \epsilon \tag{2}$$

To help the model distinguish between editable and non-editable content, we provide the original latent x_{orig} as an additional condition by concatenating it with the noisy latent x_t along the channel dimension (as illustrated in Fig. 1). This enables the model to directly access the unedited input during both training and inference, helping preserve the unchanged regions while only applying edits where instructed. RFM-Editing learns a continuous vector field $\mathbf{v}^*_\theta(x_t \oplus x_{\text{orig}}, t, E_{\mathcal{I}})$ that predicts the direction from x_t to the target latent x_{tar}, conditioned on the instruction embedding $E_{\mathcal{I}}$. The model is trained by minimizing a mean squared error (MSE) loss between the predicted and target velocity fields:

$$\mathcal{L}_{\text{RFM}} = \mathbb{E}_{x_{\text{tar}}, x_{\text{orig}}, t, \epsilon}\left[\left\|\mathbf{v}^*_\theta(x_t \oplus x_{\text{orig}}, t, E_{\mathcal{I}}) - \mathbf{v}_{\text{target}}\right\|_2^2\right] \tag{3}$$

This loss guides the optimization of the model parameters by encouraging the predicted vector field to align with the target velocity at each sampled timestep.

2.2. Instruction-Driven Editing

At inference, we leverage the trained model to perform audio editing conditioned on the original audio X_{original} and a textual instruction $\mathcal{I}$, without requiring the full target description, as done in [8, 10]. $X_{\text{orig\_mel}}$ is first encoded by the VAE into a latent representation x_{orig}, while the instruction is embedded into a vector $E_{\mathcal{I}}$ using the LoRA-tuned Flan-T5 encoder.

Instead of initializing the sampling process from pure Gaussian noise, we adopt a more flexible initialization strategy inspired by DDPM/DDIM inversion. Since audio editing aims to preserve most of the original content rather than synthesizing entirely new audio from scratch, the initial state should retain partial information from the original input to better preserve unedited regions. Specifically, we define the starting point x_{start} along the rectified interpolation path from noise ϵ to the original audio latent x_{orig}:

$$x_{\text{start}} = (1 - (1 - \sigma_{\min}) \cdot t_{\text{start}}) \cdot \epsilon + t_{\text{start}} \cdot x_{\text{orig}} \tag{4}$$

where t_{start} is a small adjustable parameter and we set $t_{\text{start}} = 0.01$ in our model. This facilitates faithful editing by preserving non-editable regions during denoising, resulting in better consistency between the edited and original audio. Thus, the sampling interval becomes $t \in [t_{\text{start}}, 1]$. At each step t, the noisy latent x_t is concatenated with the original latent x_{orig} along the channel dimension and passed to the trained U-Net, along with the current time step t and instruction embedding $E_{\mathcal{I}}$. The U-Net predicts the instantaneous velocity field $\mathbf{v}^*_\theta(x_t \oplus x_{\text{orig}}, t, E_{\mathcal{I}})$, which is used by a continuous-time Euler solver to iteratively update the latent:

$$x_{t+\Delta t} = x_t + \Delta t \cdot \mathbf{v}^*_\theta(x_t \oplus x_{\text{orig}}, t, E_{\mathcal{I}}). \tag{5}$$

Iterating this update until $t = 1$, we obtain the target latent x^*_{tar}. Finally, x^*_{tar} is decoded by the VAE decoder to reconstruct the log-mel spectrogram of the edited audio. The vocoder [20] is then used to convert the spectrogram into a waveform, producing the final edited audio output.

3. EXPERIMENTS

3.1. Datasets

We construct an instruction-based audio editing dataset using AudioCaps2 [18]. The DeepSeek API is used to count sound events in each caption. Audio clips with more than three events are excluded, as they tend to be noisy and less suitable for training, and those containing only one event as single-event clips for composition. We mix each audio X with two random single-event clips A and B to create overlapping and semantically meaningful examples, yielding six instruction-conditioned triplets: $\langle X, X+A, \text{Add A}\rangle$, $\langle X, X+B, \text{Add B}\rangle$, $\langle X+A, X, \text{Remove A}\rangle$, $\langle X+B, X, \text{Remove B}\rangle$, $\langle X+A, X+B, \text{Replace A with B}\rangle$, $\langle X+B, X+A, \text{Replace B with A}\rangle$, which are used as model inputs during training. Each example has a target caption used only for evaluation. The original captions are retained for future research.

To ensure high-quality supervision, we compute the CLAP similarity [23] between each audio and its caption, and retain only samples where both the original and edited pairs achieve a CLAP similarity above 0.35. The final full dataset contains 95,616 samples per task type, yielding 234,639, 26,103, and 26,103 samples for training, validation, and testing, with each split balanced across task types. We also provide a relatively smaller subset with 54,123, 6,021, and 6,021 samples for training, validation, and testing.

3.2. Experimental Settings and Baselines

We train the model on log-mel spectrograms with 1024 time frames and 64 mel-frequency bins extracted from 10 seconds audio clips sampled at 16kHz. The model uses a U-Net backbone with cross-attention to Flan-T5 text encoder, and is conditioned via classifier-free guidance. We adopt a velocity-based rectified flow with linear noise–data interpolation. Training is conducted for 100 epochs on A100 GPUs with a learning rate of 5×10^{-5}. During inference, Euler integration [24] is used with 200 sampling steps. Validation is based on CLAP similarity of 1000 randomly selected validation samples at each epoch during training, and the best checkpoint is saved according to the highest CLAP score.

Table 1: Quantitative evaluation of edited audio.

Method	FD ↓	FAD ↓	KL ↓	IS ↑
AudioEditor [8]	<u>14.24</u>	**2.01**	4.07	**8.40**
AUDIT [11]	32.62	7.22	9.99	<u>6.59</u>
Zero-Shot [10]	25.77	3.86	4.09	5.04
RFM-Editing	15.00	2.95	<u>2.90</u>	4.90
RFM-Editing$_{\text{full}}$	**13.27**	<u>2.50</u>	**2.77**	5.27

We compare RFM-Editing against three baselines: AudioEditor [8], Zero-Shot [10] and AUDIT [11]. RFM-Editing leverages velocity-based RFM to achieve text-guided edits across diverse tasks. We use a pre-trained CLAP [23] to compute the cosine similarity between the edited audio and the target caption to measure semantic alignment. To assess overall quality, distributional consistency, and efficiency, we report Frechet Distance (FD), Frechet Audio Distance (FAD), Kullback–Leibler (KL) divergence, Inception Score (IS) [2], and the average editing time for each audio clip. FD, KL and IS are computed using PANNs [25] that extracts both semantic embeddings and class logits, while FAD is measured with VGGish [26], which captures low-level perceptual audio features. RFM-Editing refers to training on the subset, while RFM-Editing$_{\text{full}}$ denotes training on the full dataset.

3.3. Results

Table 1 reports the quantitative comparison in terms of edited audio quality, including FD, FAD, KL, and IS. We observe that RFM-Editing trained with the subset already achieves competitive performance, substantially outperforming AUDIT [11] and the Zero-Shot [10] across most metrics. When trained on the full dataset, RFM-Editing$_{\text{full}}$ further improves and achieves the best FD and KL scores, indicating higher distributional consistency and better alignment with the target semantics distribution in feature space. These results suggest that velocity-based RFM enables more stable modeling of global distributional transitions and stronger generalization to diverse semantic shifts specified by instructions in audio editing. Although RFM-Editing obtains only moderate IS, this is expected as the task emphasizes faithful alignment with editing instructions rather than maximizing output diversity.

While AudioEditor [8] attains the lowest FAD by local attention manipulation, its performance on global semantic consistency shown by KL is limited due to over-editing. AUDIT [11] suffers from poor fidelity and distributional scores after being trained for 100 epochs on the constructed dataset, whereas Zero-Shot [10] achieves competitive performance but lacks distributional consistency. In contrast, our approach achieves a more balanced performance across all metrics.

In Table 2, RFM-Editing demonstrates a clear advantage in faithfulness of the edited audio to the target captions and the editing efficiency. AudioEditor [8] achieves the best CLAP score and alignment with the target text due to attention replacement mechanism [14], but its editing is nearly an order of magnitude slower than ours due to inference-time

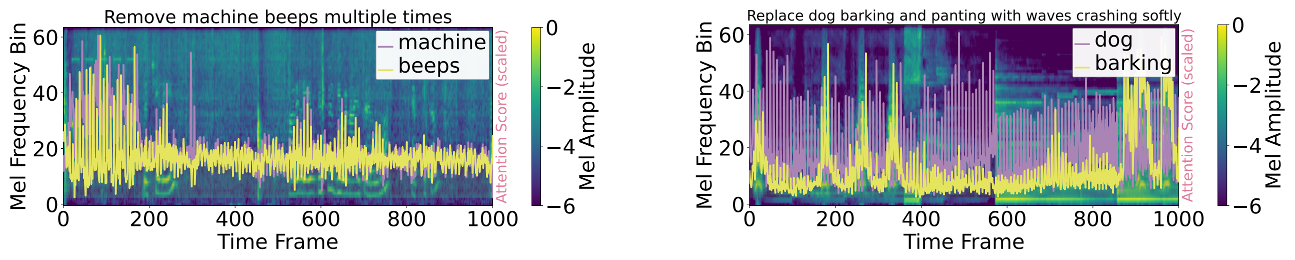
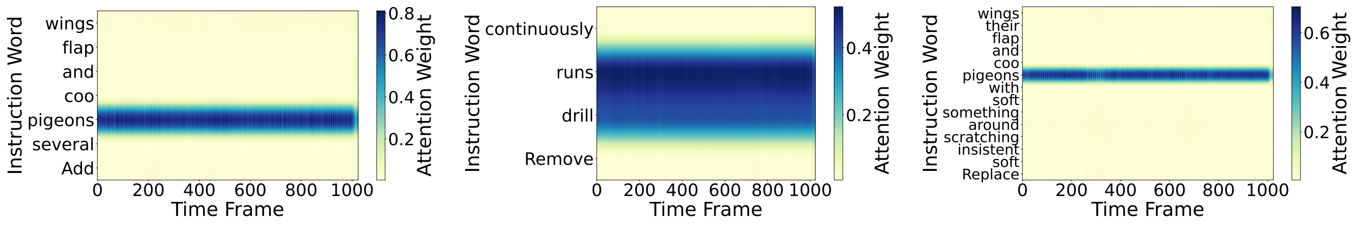


Fig. 2: Visualizations of dynamic cross-attention trajectories of specific tokens in remove and replace tasks in RFM-Editing.



(a) Add (b) Remove (c) Replace

Fig. 3: Visualization of token-wise cross-attention distributions between the instruction sequence and audio features.

Table 2: Comparison of editing fidelity and efficiency.

Method	Prompt	CLAP ↑	Editing Time (s) ↓
AudioEditor [8]	caption & modified tokens	**0.4579**	101.87
AUDIT [11]	**instruction**	0.1113	11.00
Zero-Shot [10]	caption	0.4333	12.52
RFM-Editing	**instruction**	0.4250	**10.97**
RFM-Editing$_{\text{full}}$	**instruction**	0.4398	11.27

Table 3: Effect of diffusion start time initialization.

t_{start}	CLAP ↑	FD ↓	FAD ↓	KL ↓	IS ↑
0	0.4216	17.97	2.45	2.96	4.27
0.001	0.4224	17.94	2.48	**2.94**	4.27
0.01	**0.4249**	17.38	2.52	3.06	4.34
0.1	0.3799	**16.80**	**1.49**	4.47	**5.24**

optimization, significantly degrading user experience. Moreover, it requires both full captions and modified token indices and the Zero-Shot [10] depends on target captions rather than concise editing instructions, which highlights the superiority of instruction-driven methods such as [11] and RFM-Editing.

3.4. Ablation and Visualization

We analyze the effect of initialization for diffusion start time on editing behavior in Table 3. Increasing t_{start} preserves more of the original audio, but reduces the editing strength, which results in improved perceptual quality, as reflected by the lowest FAD and highest IS when $t_{\text{start}} = 0.1$, but simultaneously leads to poor semantic alignment with the target captions, as indicated by an extremely low CLAP score. By contrast, setting $t_{\text{start}} = 0.01$ offers the best trade-off, yielding the best CLAP while still maintaining competitive audio quality.

To intuitively illustrate the effectiveness of RFM-Editing, we visualize the cross-attention weights within the diffusion network. Successful editing requires the model to accurately localize the time frames of sound events to be removed or replaced, while also attending to newly added events in the instruction. We present two complementary visualizations: dynamic normalized cross-attention trajectories between selected instruction tokens and audio features in Fig. 2, and token-wise cross-attention heatmaps that illustrate the relative attention distribution of different tokens throughout the generation process in Fig. 3.

In Fig. 2, the cross-attention weights between specific tokens and audio features reveal how the model focuses on key segments for editing. Tokens “beeps” and “barking” associated with transient events exhibit sharp attention peaks that align with actual sound occurrences, whereas the token “dog” describing a persistent source maintains high attention over longer time spans. This observation reflects the true temporal structure of the audio and indicates that the model can accurately localize target events without time-aligned masks, which is crucial for precise and effective audio editing.

Furthermore, the heatmaps in Fig. 3 (a), (b) and (c) show that the model consistently attends to the key parts of the instruction across all tasks, ensuring accurate and instruction-aligned editing outcomes. Interestingly, we observe that in replacement tasks, if the model assigns greater attention to the events to be removed rather than the newly introduced ones, the editing quality tends to degrade. In addition, the quality of the instruction has a substantial impact on the results, highlighting the critical role of prompts in editing tasks.

4. CONCLUSION

We have presented RFM-Editing, the first rectified flow matching framework for instruction-guided audio editing without captions or masks, along with a new dataset. Experiments show that RFM-Editing can automatically localize instruction-relevant time frames, achieving faithful alignment with target semantics and precise editing. Results highlight rectified flow matching as a practical paradigm, and suggest future work on leveraging language prompting capabilities.

5. ACKNOWLEDGMENT

This work was partially supported by a research scholarship from the China Scholarship Council (CSC) and a studentship from the University of Surrey. For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) license to any Author Accepted Manuscript version arising from this work.

6. REFERENCES

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

[2] H. Liu, Z. Chen, Y. Yuan, X. Mei, X. Liu, D. Mandic, W. Wang, and M. D. Plumbley, “AudioLDM: text-to-audio generation with latent diffusion models,” in *Proceedings of the 40th International Conference on Machine Learning*, 2023, pp. 21 450–21 474.

[3] H. Liu, Y. Yuan, X. Liu, X. Mei, Q. Kong, Q. Tian, Y. Wang, W. Wang, Y. Wang, and M. D. Plumbley, “AudioLDM 2: Learning holistic audio generation with self-supervised pretraining,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 2871–2883, 2024.

[4] R. Huang, J. Huang, D. Yang, Y. Ren, L. Liu, M. Li, Z. Ye, J. Liu, X. Yin, and Z. Zhao, “Make-An-Audio: Text-to-audio generation with prompt-enhanced diffusion models,” in *International Conference on Machine Learning*. PMLR, 2023, pp. 13 916–13 932.

[5] J. Huang, Y. Ren, R. Huang, D. Yang, Z. Ye, C. Zhang, J. Liu, X. Yin, Z. Ma, and Z. Zhao, “Make-An-Audio 2: Temporal-enhanced text-to-audio generation,” *arXiv preprint arXiv:2305.18474*, 2023.

[6] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, “Flow matching for generative modeling,” *arXiv preprint arXiv:2210.02747*, 2022.

[7] C.-Y. Hung, N. Majumder, Z. Kong, A. Mehrish, A. A. Bagherzadeh, C. Li, R. Valle, B. Catanzaro, and S. Poria, “TangoFlux: Super fast and faithful text to audio generation with flow matching and CLAP-ranked preference optimization,” *arXiv preprint arXiv:2412.21037*, 2024.

[8] Y. Jia, Y. Chen, J. Zhao, S. Zhao, W. Zeng, Y. Chen, and Y. Qin, “AudioEditor: A training-free diffusion-based audio editing framework,” in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2025, pp. 1–5.

[9] M. Xu, C. Li, D. Zhang, D. Su, W. Liang, and D. Yu, “Prompt-guided precise audio editing with diffusion models,” in *Proceedings of the 41st International Conference on Machine Learning*, 2024, pp. 55 126–55 143.

[10] H. Manor and T. Michaeli, “Zero-shot unsupervised and text-based audio editing using ddpm inversion,” *arXiv preprint arXiv:2402.10009*, 2024.

[11] Y. Wang, Z. Ju, X. Tan, L. He, Z. Wu, J. Bian *et al.*, “AUDIT: Audio editing by following instructions with latent diffusion models,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 71 340–71 357, 2023.

[12] F. Paissan, L. Della Libera, Z. Wang, M. Ravanelli, P. Smaragdis, C. Subakan *et al.*, “Audio editing with non-rigid text prompts,” in *Proceedings of INTERSPEECH 2024*, 2024.

[13] J. Xue, Y. Deng, Y. Gao, and Y. Li, “Auffusion: Leveraging the power of diffusion and large language models for text-to-audio generation,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2024.

[14] A. Hertz, R. Mokady, J. Tenenbaum, K. Aberman, Y. Pritch, and D. Cohen-or, “Prompt-to-prompt image editing with cross-attention control,” in *The Eleventh International Conference on Learning Representations*.

[15] J. Liang, H. Zhang, H. Liu, Y. Cao, Q. Kong, X. Liu, W. Wang, M. Plumbley, H. Phan, and E. Benetos, “WavCraft: Audio editing and generation with natural language prompts.” ICLR 2024 Workshop on LLM Agents, 2024.

[16] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-resolution image synthesis with latent diffusion models,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 684–10 695.

[17] X. Liu, C. Gong, and Q. Liu, “Flow straight and fast: Learning to generate and transfer data with rectified flow,” *arXiv preprint arXiv:2209.03003*, 2022.

[18] C. D. Kim, B. Kim, H. Lee, and G. Kim, “AudioCaps: Generating Captions for Audios in The Wild,” in *NAACL-HLT*, 2019.

[19] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, “LoRA: Low-rank adaptation of large language models.” *ICLR*, vol. 1, no. 2, p. 3, 2022.

[20] S.-G. Lee, W. Ping, B. Ginsburg, B. Catanzaro, and S. Yoon, “BigVGAN: A universal neural vocoder with large-scale training,” *arXiv preprint arXiv:2206.04658*, 2022.

[21] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.

[22] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma *et al.*, “Scaling instruction-finetuned language models,” *Journal of Machine Learning Research*, vol. 25, no. 70, pp. 1–53, 2024.

[23] B. Elizalde, S. Deshmukh, M. Al Ismail, and H. Wang, “CLAP learning audio concepts from natural language supervision,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[24] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, “Score-based generative modeling through stochastic differential equations,” *arXiv preprint arXiv:2011.13456*, 2020.

[25] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, “PANNs: Large-scale pretrained audio neural networks for audio pattern recognition,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[26] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold *et al.*, “CNN architectures for large-scale audio classification,” in *International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2017, pp. 131–135.