AudioCache: Accelerate Audio Generation With Training-Free Layer Caching

Qingyang Shi[1,2,†], Zhicheng Du[1,2,†], Jiasheng Lu[2], Yingshan Liang[1,2]
Xinyu Zhang[1,2], Yiran Wang[1,2], Jing Peng[1], Kehong Yuan[1,*]
[1]Shenzhen International Graduate School, Tsinghua University, Shenzhen, China
[2]Huawei Technologies Co., Ltd., Shenzhen, China

***Abstract*—Diffusion models have become the primary choice in audio generation. However, their slow generation speed necessitates acceleration techniques. While current audio generation methods primarily target U-Net-based models, the Diffusion Transformer (DiT) is emerging as the trend in audio generation. As DiT costs a large amount of computational resources, we propose AudioCache: a training-free caching strategy that, for the first time to our best knowledge, accelerates DiT-based audio generation models by reusing the attention and feedforward layers of DiT during sampling. We define a reasonable statistic to characterize the degree of internal structure variation, leading to the proposal of a self-adaptive caching strategy. We achieve a 2.35x acceleration with both objective and subjective metrics remaining practically consistent. Furthermore, our method is extendable to different models and input modalities. Based on appropriate indicators and corresponding rules, this method provides a plug-and-play and training-free solution for diffusion models built on attention architectures.**

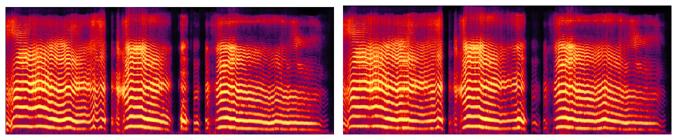
***Index Terms*—audio generation, acceleration, training-free method, caching, DiT**

I. Introduction

Diffusion models [1] have achieved significant success in audio generation [2]–[6] due to their high generation quality and diversity. Most diffusion models are based on U-Net [7], however, recent research has found that DiT [8] outperforms U-Net when used as the backbone of diffusion models. Leveraging the inherently scalable framework of transformers [9], DiT can support larger models and global receptive fields, thereby achieving superior generation quality [8]. Consequently, we believe that DiT will be the primary framework for future audio generation.

Diffusion models usually require hundreds of iterations to generate high-quality audio, which makes their slow sampling speed a limiting factor for real-time deployment in the field of audio generation. Recently, many studies have focused on accelerating diffusion models through techniques such as quantization [10], [11], pruning [12], [13], distillation [14]–[16], and optimized sampling solvers [17]–[19]. Some of these methods, such as AudioLCM [20] and ConsistencyTTA [21], have also been applied in the field of audio generation. However, these approaches often require significant training costs

*Corresponding Author: Kehong Yuan: yuankh@sz.tsinghua.edu.cn
†These authors contribute equally to this work
Work done during an intership at Huawei Technologies Co., Ltd.
Demo is available at https://sqy-qh.github.io/AudioCache/


(a) Text prompt 'A baby is crying loudly'.

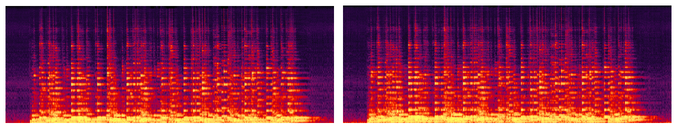

(b) Text prompt 'An acoustic guitar played in fingerstyle.'

Fig. 1. Comparison of Mel spectrograms before and after a 2.35x acceleration: the left column shows the unaccelerated version, and the right column shows the accelerated version.

to distill a new model, and they are primarily implemented on the U-Net architecture. Recently, caching methods [22], [23] have found that certain feature calculation layers do not change significantly between adjacent time steps during sampling, so these features can be stored for future reuse.

In response to the aforementioned issues, we propose an acceleration method named AudioCache for DiT-based audio generation for the first time, utilizing a training-free caching method. We reuse the attention and feedforward layers of DiT during sampling. Furthermore, we discovered that different blocks in DiT and different layers within the same block exhibit varying levels of activity during sampling, necessitating a self-adaptive caching interval. We conducted our experiments on the first open-source DiT-based audio generation model, Stable Audio Open [24], achieving a 2.35x acceleration in audio generation while maintaining virtually the same production quality. Fig. 1 shows the Mel spectrograms before and after acceleration.

The main contributions of our paper can be summarized as follows:

- We propose AudioCache, which applies a training-free caching method to accelerate audio generation models for the first time.
- We explore the variation patterns of different layers within DiT during the sampling process and define reasonable indicators to guide our self-adaptive caching strategy.
- We demonstrate that our method is decoupled from model

architecture and input modality, making it extendable to other audio generation tasks.

II. Related work

Diffusion Models. Diffusion models stand out among many generative models due to their ability to generate high-quality and diverse outputs. In the reverse process of a diffusion model, the noisy data is fed into a noise estimation network, which aims to progressively remove the noise and recover the original data. Existing diffusion models primarily utilize the U-Net architecture. However, DiT is emerging as a new trend. DiT is based on stacked transformers layers, which can effectively expand the model's capacity to enhance generation quality.

Acceleration of Diffusion Models. Due to the time-consuming sampling process of diffusion models, acceleration techniques are crucial. Recently, there has been a surge in research on diffusion model acceleration, including quantization, pruning, distillation, optimized sampling solvers, caching, and dynamic model inference [25]. For distillation, the Consistency Model (CM) [14] maps all points on the ODE trajectory to the origin through the consistency formula to achieve one-step generation, while the Latent Consistency Model (LCM) [15] extends CM to the latent space and achieves multi-step generation. For optimized sampling solvers, DPM-Solver++ [19] addresses the instability problem of high-order solvers, significantly reducing the sampling steps while ensuring sample quality. For caching, some time-consuming feature calculation layers show relatively small changes between adjacent time steps during the sampling process, making them suitable caching targets to improve sampling speed. Existing caching methods [22], [23], [26]–[29] are primarily applied in the field of image generation.

III. Method

A. Baseline

We use Stable Audio Open as our baseline to implement our acceleration method, which is an open-source DiT-based text-to-audio generation model capable of generating high-quality stereo sound effects and music of controllable duration. The DiT architecture of Stable Audio Open consists of 24 transformer blocks, each of which comprises a self-attention layer, a cross-attention layer, and a feedforward layer. Each layer within the transformer block is modified with layer normalization and skip residual connections.

B. Caching Strategy

Most caching strategies are training-free acceleration methods, as they reuse similar features calculated in previous sampling steps to avoid repetitive computations. These features do not change significantly between adjacent time steps in the denoising process.

In the forward propagation of each block in DiT, the main computational workload is concentrated in the self-attention layer, cross-attention layer, and feedforward layer, which is approximately two orders of magnitude larger than tensor

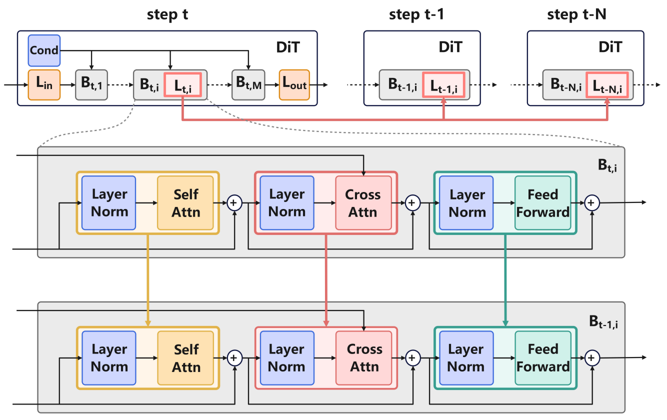


Fig. 2. *Caching strategy for DiT audio generation model.* The time step t in the sampling process varies from T to 0. In DiT, L_{in} denotes pre-processing, while L_{out} denotes post-processing. $B_{t,i}$ represents the i-th transformer block at time step t, and $L_{t,i}$ indicates a specific feature computation layer within $B_{t,i}$ (which could be a self-attention layer, a cross-attention layer, or a feedforward layer). N represents the current caching interval, meaning that the feature computation layers from subsequent time steps $t-1$ to $t-N$ will reuse the feature computation layers from time step t instead of recomputing them.

addition. Therefore, we cache the features computed by these layers at specific time steps, as shown in Fig. 2. We retain the residuals, which contain information from the previous latent z, preventing them from being cached during forward propagation. This approach avoids losing information from the previous latent z while also minimizing any speed loss [28].

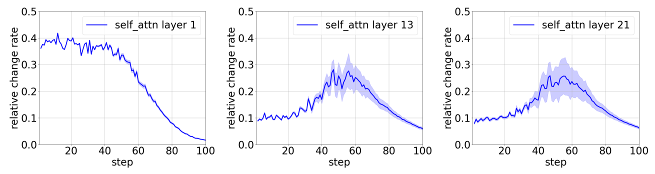
C. Self-adaptive Caching Strategy

After discussing what to cache, we then address when to cache. We observe that at different stages of the denoising process, the activity of different blocks in DiT varies, and the variation patterns of different layers within a single block also differ. Therefore, we should define appropriate statistics to characterize their changing patterns.

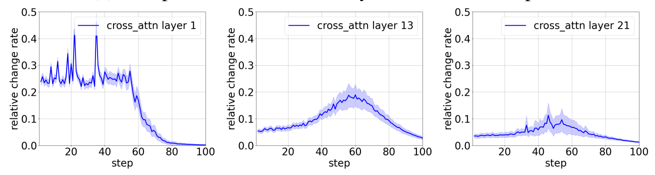
Inspired by [26], We define the L1 relative change rate of a certain layer $L_{t,i}$ in DiT as shown in Equation (1) to characterize the variation patterns of that layer at time step t. $l_{t,i}$ is the input latent to $L_{t,i}$ and c is the embedding of the condition.

$$\mathcal{E}_{L,t,i} = \frac{\|L_{t,i}(l_{t,i}, c) - L_{t+1,i}(l_{t+1,i}, c)\|_1}{\|L_{t,i}(l_{t,i}, c)\|_1} \tag{1}$$

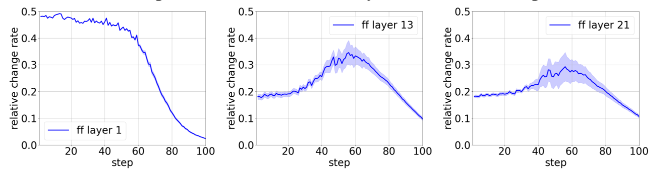
In Fig. 3, we present the mean and variance of the L1 relative change rate for various feature calculation layers under the influence of different prompts as the sampling time steps advance. Our observations indicate that the variance in these layers' responses to various prompts is minimal. This suggests that the changing patterns within these layers, as measured by the L1 relative change rate, exhibit consistency across different prompts. Since the L1 relative change rate remains decoupled from the prompt variations, it is well-defined for cache strategy base on model structure only. Regarding our baseline model, during the initial phase of the contour generation [32] in the denoising process, the layers situated in the front blocks



(a) Graphs of self attention layers at different depths.



(b) Graphs of cross attention layers at different depths.



(c) Graphs of feedforward layers at different depths.

Fig. 3. The curve represents the average L1 relative change rate of different layers in DiT of Stable Audio Open during denoising process, and the light colored area shows the standard deviation of the L1 relative change rate corresponding to different prompts. The curve is plotted using 5000 prompts randomly selected from the AudioCaps [30] and Song Describer [31] datasets.

are more active. As the process transitions into the detail generation stage [32], the layers in the middle and rear blocks become more active. In the end of the denoising process, all layers demonstrate a tendency towards convergence.

Based on the above conclusions, we need to use self-adaptive caching intervals for different layers at different time steps. Our self-adaptive caching strategy, as outlined in Algorithm 1, is described as follows: Since the aforementioned rule is not affected by prompts, we can calculate the L1 relative change rate of each layer in advance. As sampling progresses, when the cumulative L1 relative change rate, as defined in Equation (2), of a certain layer i from the last caching moment t_1 to the current time step t reaches the set threshold δ, indicating that the cumulative change of that layer has become sufficiently large, the features of that layer are recalculated, and the cache is updated. Otherwise, the cached values are reused.

$$\mathcal{E}_{sum,L,t,i} = \sum_{\tau=t_1}^{t} \frac{\|L_{\tau,i}(l_{\tau,i},c) - L_{\tau+1,i}(l_{\tau+1,i},c)\|_1}{\|L_{\tau,i}(l_{\tau,i},c)\|_1} \tag{2}$$

Algorithm 1: Self-adaptive Caching Strategy

```
Input: set of layers requiring caching $D$ (self-attn, cross-attn,
       ff), current time step $t$, latent $z_t$, DiT depth $M$,
       threshold $\delta$, sampling solver $Solver$, condition $c$
Output: latent $z_{t-1}$
$l_{t,1} = L_{in}(z_t)$//pre-processing
for $i$ = 1,2,...,$M$ do
    for $L$ in $D$ do
        if $\mathcal{E}_{sum,L,t,i} > \delta$ then
            calculate $L_{t,i}(l_{t,i}, c)$
            store $L_{t,i}(l_{t,i}, c)$ in $Cache_{L,t,i}$
            $l_{t,i+1} = l_{t,i} + L_{t,i}(l_{t,i}, c)$ //residual connection
        else
            retrieve $L_{t,i}(l_{t,i}, c)$ from $Cache_{L,t,i}$
            $l_{t,i+1} = l_{t,i} + L_{t,i}(l_{t,i}, c)$
$DiT_t = L_{out}(l_{t,M+1})$ //post-processing,$DiT_t$ is model output
$z_{t-1} = Solver(z_t, DiT_t)$
return $z_{t-1}$
```

IV. Experiment

A. Experimental Settings

Model and Datasets. We choose Stable Audio Open as our baseline model, which is the first text-to-audio DiT model. Stable Audio Open uses a T5 [33] text encoder. For our test datasets, we use Audiocaps and Song Describer. The Audiocaps test dataset contains 4,875 captions, while the Song Describer dataset without singing includes 586 captions. For all datasets, each audio corresponds to multiple captions, so we generate an audio for each caption for evaluation purposes.

Implementations. For both our baseline model and accelerated model, we use DPM++ 3M SDE as the solver to perform inference over 100 time steps and test the generation results with various self-adaptive cache thresholds δ. For the Audiocaps dataset, we generate 47 seconds of 44.1 kHz dual-channel audio and then trim it to 10 seconds (discarding the silent parts at the end). For the Song Describer dataset, we generate 47 seconds of 44.1 kHz dual-channel audio.

Objective Evaluation. For quantitative generation quality assessment, we continue the three evaluation metrics of Stable Audio Open: FD_{openl3} [34], KL_{passt} [35], [36] and $CLAP_{score}$ [37]. For FD_{openl3}, all audio is evaluated in a dual-channel format with a 44.1 kHz sampling rate. For 16kHz single-channel audio generated by other models, upsampling to 44.1 kHz is required, and the single channel is duplicated to simulate dual-channel.

Subjective Evaluation. We use a 5-level Likert scale ranging from 20 to 100 to implement our Mean Opinion Score(MOS) questionnaire, which includes evaluations of audio quality score (MOS-Q) and text fidelity score (MOS-F). 20 users participate in this study.

B. Main Results

Table I and Table II shows our main experimental results. We present the results of AudioCache using different self-adaptive threshold δ. Objective indicators and subjective test results indicate that δ=0.3 can achieve a good acceleration ratio while maintaining the original generation quality.

C. Ablation studies

For ablation experiments, our method tests on models with different DiT architectures, input modalities, samplers, and sampling steps, demonstrating that our method is decoupled from the model, input modality, sampler, and sampling steps. In the following experiments, we default to using the optimal threshold δ=0.3, which balances generation performance and acceleration ratio.

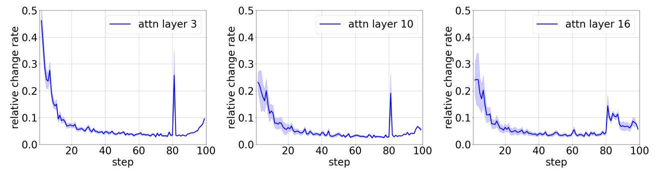
Table III shows the results of our method applied to the text-to-audio and video-to-audio models of Make-An-Audio 3 [38], which is a flow matching model. Make-An-Audio 3 employs a different DiT architecture than Stable Audio Open. The text-to-audio model of Make-An-Audio 3 uses a CLAP [39] text encoder and DiT architecture, while the video-to-audio model uses a CAVP video encoder and DiT architecture. In Fig. 4, we plot the L1 relative change rate in different attention layers of the two DiT models as a function of sampling steps. It can be seen that the variation pattern is minimal under inputs of different prompts or videos, as the variance of the L1 relative change rate is very small, indicating that this variation pattern is decoupled from the input. We achieve a 2-3x acceleration for both models using AudioCache with almost no loss of generation quality. This proves that our acceleration method is decoupled from the DiT model architecture and input modality.

Table IV shows the results of AudioCache under different samplers. Our method maintains the generation quality almost unchanged even after accelerating by 2-3 times, demonstrating that our acceleration method is decoupled from the sampler.

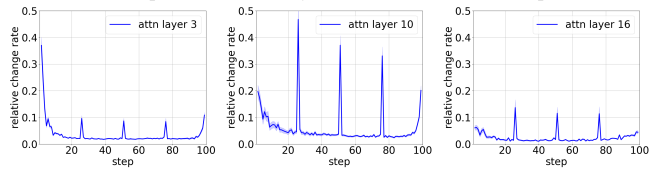
Table V shows the results of AudioCache at different sampling steps. When the number of sampling steps increases, the threshold can be appropriately increased to improve the acceleration ratio while ensuring the generation effect, demonstrating that our acceleration method is decoupled from the sampling steps.

V. Conclusion

In this paper, we apply the training-free caching method to accelerate audio generation models for the first time. We define the L1 relative change rate to explore the variation patterns of different layers in DiT during the inference process, which guides us in designing a self-adaptive caching strategy. Our method achieves a 2.35x acceleration while maintaining practical consistency in both objective and subjective metrics. Our approach is decoupled from the DiT model architecture, input modality, sampler, and sampling steps, demonstrating the plug-and-play nature of our method.



(a) Graph of attention layers of T2A at different depths.



(b) Graph of attention layers of V2A model at different depths.

Fig. 4. The curve represents the average L1 relative change rate of different layers in DiT of Make-An-Audio 3 during denoising process, and the light colored area shows the standard deviation of the L1 relative change rate corresponding to different prompts.

TABLE III
Text-to-audio generation on AudioCaps test dataset, video-to-audio generation on VGGSound test dataset. For Make-An-Audio 3, comparison of our self-adaptive caching method and baseline model in generating 16kHz single-channel 10s audio.

Model	Method	Model	Speed	FD (↓)	KL (↓)	CLAP (↑)
Stable Audio Open	/	T2A	/	78.24	2.14	0.29
Stable Audio Open	AudioCache(δ=0.3)	T2A	x2.35	80.30	2.15	0.28
make-an-audio 3	/	T2A	/	189.28	1.51	0.42
make-an-audio 3	AudioCache(δ=0.1)	T2A	x2.41	193.21	1.50	0.42
make-an-audio 3	/	V2A	/	57.15	3.48	/
make-an-audio 3	AudioCache(δ=0.1)	V2A	x2.50	59.06	3.35	/

TABLE IV
Text-to-audio generation on AudioCaps test dataset. Comparison of our self-adaptive caching method on different samplers in generating 44.1kHz dual-channel 10s audio.

Sampler	Method	Speed	FD (↓)	KL (↓)	CLAP (↑)
DPM ++ 3M SDE	Stable Audio Open	/	78.24	2.14	0.29
DPM ++ 3M SDE	AudioCache(δ=0.3)	x2.35	80.30	2.15	0.28
Heun	Stable Audio Open	/	79.03	2.16	0.30
Heun	AudioCache(δ=0.3)	x2.39	82.05	2.17	0.30

TABLE V
Text-to-audio generation on AudioCaps test dataset. Comparison of our self-adaptive caching method on different steps in generating 44.1kHz dual-channel 10s audio.

Step	Method	Speed	FD (↓)	KL (↓)	CLAP (↑)
100	Stable Audio Open	/	78.24	2.14	0.29
100	AudioCache(δ=0.3)	x2.35	80.30	2.15	0.28
200	Stable Audio Open	/	79.89	2.15	0.30
200	AudioCache(δ=0.4)	x2.85	82.75	2.14	0.31
500	Stable Audio Open	/	80.58	2.14	0.28
500	AudioCache(δ=0.6)	x3.24	82.76	2.16	0.27

TABLE I
Text-to-audio generation on AudioCaps test dataset. Comparison of AudioCache and baseline model Stable Audio Open in generating 44.1kHz dual-channel 10s audio.

Model	Speed	FD (↓)	KL (↓)	CLAP (↑)	MOS-Q(↑)	MOS-F(↑)
Stable Audio Open	/	78.24	2.14	0.29	86.50±1.22	90.63±1.08
AudioCache(δ=0.2)	x1.78	79.57	2.14	0.28	86.88±1.15	90.50±1.10
AudioCache(δ=0.3)	x2.35	80.30	2.15	0.28	87.13±1.27	89.88±1.12
AudioCache(δ=0.4)	x2.80	91.47	2.20	0.25	83.07±1.21	87.04±1.18

TABLE II
Text-to-audio generation on Song Describer dataset. Comparison of AudioCache and baseline model Stable Audio Open in generating 44.1kHz dual-channel 47s audio.

Model	Speed	FD (↓)	KL (↓)	CLAP (↑)	MOS-Q(↑)	MOS-F(↑)
Stable Audio Open	/	96.51	0.55	0.41	86.43±1.35	88.29±1.18
AudioCache(δ=0.2)	x1.78	97.44	0.57	0.42	87.14±1.42	88.29±1.35
AudioCache(δ=0.3)	x2.35	98.07	0.58	0.41	86.29±1.38	88.00±1.25
AudioCache(δ=0.4)	x2.80	103.14	0.65	0.38	82.29±1.24	86.29±1.20

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[2] Huan Liao, Haonan Han, Kai Yang, Tianjiao Du, Rui Yang, Zunnan Xu, Qinmei Xu, Jingquan Liu, Jiasheng Lu, and Xiu Li, "Baton: Aligning text-to-audio model with human preference feedback," 2024.

[3] Zach Evans, CJ Carr, Josiah Taylor, Scott H Hawley, and Jordi Pons, "Fast timing-conditioned latent audio diffusion," *arXiv preprint arXiv:2402.04825*, 2024.

[4] Zach Evans, Julian D Parker, CJ Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons, "Long-form music generation with latent diffusion," *arXiv preprint arXiv:2404.10301*, 2024.

[5] Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao, "Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models," in *International Conference on Machine Learning*. PMLR, 2023, pp. 13916–13932.

[6] Jiawei Huang, Yi Ren, Rongjie Huang, Dongchao Yang, Zhenhui Ye, Chen Zhang, Jinglin Liu, Xiang Yin, Zejun Ma, and Zhou Zhao, "Make-an-audio 2: Temporal-enhanced text-to-audio generation," *arXiv preprint arXiv:2305.18474*, 2023.

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer, 2015, pp. 234–241.

[8] William Peebles and Saining Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4195–4205.

[9] A Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[10] Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao, "Post-training quantization for vision transformer," *Advances in Neural Information Processing Systems*, vol. 34, pp. 28092–28103, 2021.

[11] Junhyuk So, Jungwon Lee, Daehyun Ahn, Hyungjun Kim, and Eunhyeok Park, "Temporal dynamic quantization for diffusion models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[12] Gongfan Fang, Xinyin Ma, and Xinchao Wang, "Structural pruning for diffusion models," 2023.

[13] Thibault Castells, Hyoung-Kyu Song, Bo-Kyeong Kim, and Shinkook Choi, "Ld-pruner: Efficient pruning of latent diffusion models using task-agnostic insights," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 821–830.

[14] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever, "Consistency models," *arXiv preprint arXiv:2303.01469*, 2023.

[15] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao, "Latent consistency models: Synthesizing high-resolution images with few-step inference," *arXiv preprint arXiv:2310.04378*, 2023.

[16] Koichi Saito, Dongjun Kim, Takashi Shibuya, Chieh-Hsin Lai, Zhi Zhong, Yuhta Takida, and Yuki Mitsufuji, "Soundctm: Uniting score-based and consistency models for text-to-sound generation," *arXiv preprint arXiv:2405.18503*, 2024.

[17] Jiaming Song, Chenlin Meng, and Stefano Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

[18] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu, "Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps," *Advances in Neural Information Processing Systems*, vol. 35, pp. 5775–5787, 2022.

[19] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu, "Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models," *arXiv preprint arXiv:2211.01095*, 2022.

[20] Huadai Liu, Rongjie Huang, Yang Liu, Hengyuan Cao, Jialei Wang, Xize Cheng, Siqi Zheng, and Zhou Zhao, "Audiolcm: Text-to-audio generation with latent consistency models," *arXiv preprint arXiv:2406.00356*, 2024.

[21] Yatong Bai, Trung Dang, Dung Tran, Kazuhito Koishida, and Somayeh Sojoudi, "Accelerating diffusion-based text-to-audio generation with consistency distillation," *arXiv preprint arXiv:2309.10740*, 2023.

[22] Xinyin Ma, Gongfan Fang, and Xinchao Wang, "Deepcache: Accelerating diffusion models for free," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 15762–15772.

[23] Senmao Li, Taihang Hu, Fahad Shahbaz Khan, Linxuan Li, Shiqi Yang, Yaxing Wang, Ming-Ming Cheng, and Jian Yang, "Faster diffusion: Rethinking the role of unet encoder in diffusion models," *arXiv preprint arXiv:2312.09608*, 2023.

[24] Zach Evans, Julian D Parker, CJ Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons, "Stable audio open," *arXiv preprint arXiv:2407.14358*, 2024.

[25] Enshu Liu, Xuefei Ning, Zinan Lin, Huazhong Yang, and Yu Wang, "Oms-dpm: Optimizing the model schedule for diffusion probabilistic models," in *International Conference on Machine Learning*. PMLR, 2023, pp. 21915–21936.

[26] Felix Wimbauer, Bichen Wu, Edgar Schoenfeld, Xiaoliang Dai, Ji Hou, Zijian He, Artsiom Sanakoyeu, Peizhao Zhang, Sam Tsai, Jonas Kohler, et al., "Cache me if you can: Accelerating diffusion models through block caching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 6211–6220.

[27] Pratheba Selvaraju, Tianyu Ding, Tianyi Chen, Ilya Zharkov, and Luming Liang, "Fora: Fast-forward caching in diffusion transformer acceleration," *arXiv preprint arXiv:2407.01425*, 2024.

[28] Pengtao Chen, Mingzhu Shen, Peng Ye, Jianjian Cao, Chongjun Tu, Christos-Savvas Bouganis, Yiren Zhao, and Tao Chen, "delta-dit: A training-free acceleration method tailored for diffusion transformers," *arXiv preprint arXiv:2406.01125*, 2024.

[29] Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang, "Learning-to-cache: Accelerating diffusion transformer via layer caching," *arXiv preprint arXiv:2406.01733*, 2024.

[30] Chris Dongjoo Kim, Byeongchang Kim, Hyunmin Lee, and Gunhee 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)*, Jill Burstein, Christy Doran, and Thamar Solorio, Eds., Minneapolis, Minnesota, June 2019, pp. 119–132, Association for Computational Linguistics.

[31] Ilaria Manco, Benno Weck, Seungheon Doh, Minz Won, Yixiao Zhang, Dmitry Bogdanov, Yusong Wu, Ke Chen, Philip Tovstogan, Emmanouil Benetos, et al., "The song describer dataset: a corpus of audio captions for music-and-language evaluation," *arXiv preprint arXiv:2311.10057*, 2023.

[32] Binxu Wang and John J Vastola, "Diffusion models generate images like painters: an analytical theory of outline first, details later," *arXiv preprint arXiv:2303.02490*, 2023.

[33] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.

[34] Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi, "Frechet audio distance: A metric for evaluating music enhancement algorithms," *arXiv preprint arXiv:1812.08466*, 2018.

[35] Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez, "Simple and controllable music generation," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[36] Khaled Koutini, Jan Schlüter, Hamid Eghbal-Zadeh, and Gerhard Widmer, "Efficient training of audio transformers with patchout," *arXiv preprint arXiv:2110.05069*, 2021.

[37] Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick, and Shlomo Dubnov, "Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[38] Peng Gao, Le Zhuo, Ziyi Lin, Chris Liu, Junsong Chen, Ruoyi Du, Enze Xie, Xu Luo, Longtian Qiu, Yuhang Zhang, et al., "Lumina-t2x: Transforming text into any modality, resolution, and duration via flow-based large diffusion transformers," *arXiv preprint arXiv:2405.05945*, 2024.

[39] Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick, and Shlomo Dubnov, "Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.