This CVPR Workshop paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

SmoothCache: A Universal Inference Acceleration Technique for Diffusion Transformers

Joseph Liu
Roblox
josephliu@roblox.com

Joshua Geddes
Queen's University
j.geddes@queensu.ca

Ziyu Guo
Roblox
zguo@roblox.com

Haomiao Jiang
Roblox
haomiaojiang@roblox.com

Mahesh Kumar Nandwana
Roblox
mnandwana@roblox.com

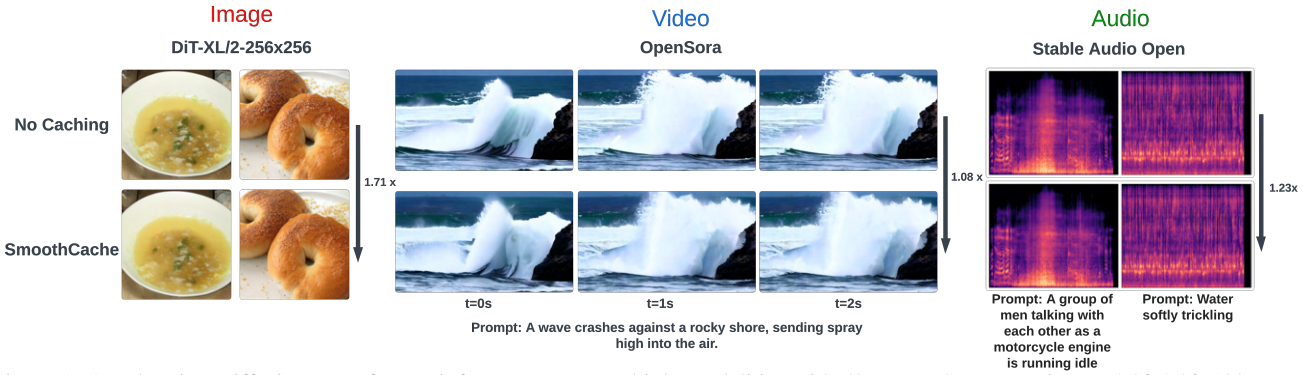


Figure 1. Accelerating Diffusion Transformer inference across multiple modalities with 50 DDIM Steps on DiT-XL-256x256, 100 DPM-Solver++(3M) SDE steps for a 10s audio sample (spectrogram shown) on Stable Audio Open, 30 Rectified Flow steps on Open-Sora 480p 2s videos.

Abstract

Diffusion Transformers (DiT) have emerged as powerful generative models for various tasks, including image, video, and speech synthesis. However, their inference process remains computationally expensive due to the repeated evaluation of resource-intensive attention and feed-forward modules. To address this, we introduce SmoothCache[1], a model-agnostic inference acceleration technique for DiT architectures. SmoothCache leverages the observed high similarity between layer outputs across adjacent diffusion timesteps. By analyzing layer-wise representation errors from a small calibration set, SmoothCache adaptively caches and reuses key features during inference. Our experiments demonstrate that SmoothCache achieves 8% to 71% speed up while maintaining or even improving generation quality across diverse modalities. We showcase its effectiveness on DiT-XL for image generation, Open-Sora for text-to-video, and Stable Audio Open for text-to-audio, highlighting its potential to enable real-time applications and broaden the accessibility of powerful DiT models.

[1]Code can be found at https://github.com/Roblox/SmoothCache

1. Introduction

In the rapidly evolving landscape of generative modeling, Diffusion models [8, 32] have emerged as a pivotal force, offering unparalleled capabilities for creating rich and diverse content. Diffusion transformers (DiT) [26, 34], in particular, harnessing the scalable architecture of transformers, demonstrated significant advancements in various domains, including but not limited to the creation of images [3], audio [5, 13, 15], video [12, 22, 41], and 3D models [24].

The central challenge limiting the broader adoption of DiT's is the computational intensity of their inference process. The cost of running the entire diffusion pipeline lies primarily in the denoising steps, and improving denoising efficiency directly contributes to a faster diffusion pipeline. To address this, inference optimizations have been researched from two directions: (1) reducing the number of sampling steps, and (2) lowering the inference cost per step.

The use of advanced solvers [17, 19] have been proposed to reduce the number of timesteps required for sampling, thereby accelerating inference. Techniques such as knowledge distillation [29], architecture optimizations [14], pruning [6], and quantization [7, 31] aim to reduce the computational complexity of individual denoising steps.

While these methods have shown promise, there remains a need for techniques that can further accelerate DiT inference *across diverse modalities* without compromising generation quality. Caching [1, 16, 21, 38] has emerged as a potential solution by exploiting the inherent redundancy in the diffusion process. Previous and concurrent work on DiT caching has investigated uniform scheduling [30], training a caching pattern with a dataset [20], and exploiting qualitative characteristics about a candidate model to create a caching pattern [4, 40]. However, these caching techniques are limited by either overly simplistic or model-specific strategies or reliance on costly retraining procedures.

To overcome these limitations, we introduce SmoothCache, a simple and universal caching scheme capable of speeding up a diverse spectrum of Diffusion Transformer models. SmoothCache leverages the observation that layer outputs from adjacent timesteps in any DiT-based model exhibits high cosine similarities [20, 30], suggesting potential computational redundancies. By carefully leveraging layer-wise representation errors from a small calibration set, SmoothCache adaptively determines the optimal caching intensity at different stages of the denoising process. This allows for the reuse of key features during inference without significantly impacting generation quality. SmoothCache is designed with generality in mind, and can be applied to any DiT architecture without model-specific assumptions or training while still achieving performance gains over uniform caching.

We show that SmoothCache with general training-free caching scheme is able to speedup the performance across image, video, audio domains to match or exceed SOTA caching scheme for each dedicated domain. We demonstrate that SmoothCache is also compatible with various common solvers in diffusion transformers.

1.1. Related Work

Diffusion models [8, 32] have emerged as a powerful class of generative models, capable of producing high-quality samples across various domains such as images, audio, and video. Initally, U-Net architectures were favored for this denoising step due to their strong ability to capture both global and local information in the latent space, crucial for reconstructing fine details in the data [27, 28].

However, the inherent limitations of U-Nets, particularly their struggle to scale effectively to high-dimensional data and long sequences, led to the exploration of alternative architectures. The Diffusion Transformer (DiT) [26] was proposed as a solution to U-Nets to exploit the scalability of the transformer architecture [34]. Transformers, renowned for their ability to handle long-range dependencies and their inherent scalability, have proven highly effective in various sequence-based tasks. DiT successfully adapted these advantages to the diffusion framework, demonstrating strong performance in generating not only images but also extending to diverse domains like speech [5, 13, 15], video [12, 22, 41], and 3D generation [24].

Efficient Diffusion Models. Diffusion models require many function evaluations to iteratively remove noise from data, and thus quickly scale in computational costs. Many previous works have focused on reducing the number of diffusion steps required to achieve high quality samples by exploring accelerated noise schedulers and fast ODE solvers [17–19, 33]. However, not all tasks might be suitable for faster solvers with less steps, as they might rely on inversion for tasks like image-editing [25], and while work has been done to extend this to other solvers [9, 35], future solvers will face similar issues with adoption. Traditional acceleration methods such as such as quantization [7, 14, 31] and compression/distillation techniques [6, 14, 29] could improve model performance, but these techniques often require extensive retraining, or specific architectures and parameterization around a specific task in order to realize efficiency gains.

Diffusion Model Caching. Caching has emerged as a promising technique for accelerating DiT inference by exploiting computational redundancies in the denoising process. For U-Net based architectures, DeepCache [21] leverages cross-timestep similarities by caching and reusing up sampling feature maps, while other methods use focus on caching blocks of layers [38], cross-attention modules [16], or intermediate noise states [1]. However, in addition to exploiting specific properties of the U-Net architecture, these models only perform these analyses on image diffusion models, leveraging biases and assumptions in those tasks that might not necessarily generalize to other modalities. For Diffusion Transformers, further research has been conducted to apply caching to image [4, 20, 30] and video [40] DiT diffusion models. Again, these methods only study how their methods accelerate diffusion inference in their specific modalities, with very little attempt to show that these methods generalize beyond their modality of interest. FORA [30] for example might have extensive acceleration gains in image diffusion, but our preliminary investigations shows that it does not work on Audio or Video diffusion tasks. Likewise, Pyramid-Attention Broadcast [40] leverages specific qualities of video diffusion, such as its cache-able cross-attention layer in order to accelerate diffusion, on top of other GPU parallelization tricks, but Fig. 2 shows this assumption breaks in other modalities. Moreover, not all caching methods are post-training. Training-based caching methods like Learning-to-Cache [20] that require further training for specific diffusion step configurations, which is not viable in cases where access to source training data might not be available, such as in open source models trained on proprietary (non-public) data.

SmoothCache attempts to address these limitations by

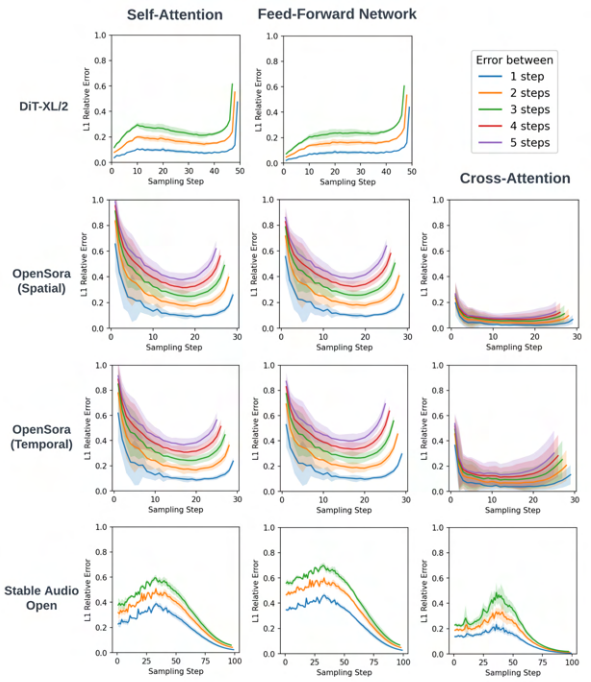


Figure 2. **L1 Relative Error Curves of different architecture components**. Curves are plotted with 95% confidence intervals from 10 calibration samples from all components explored in this paper and scaled to the same y-axis range. Note that OpenSora has distinct spatial and temporal diffusion blocks.

introducing a training-free caching approach, that makes no underlying assumption about the specific task or modality at hand, outperforming uniform schedules and even matching or beating handcrafted caching techniques reported in literature. We also present multi-modal results to show that this technique does generalize across modalities, and how it adapts to various model, sampling, and solver configurations with just one calibration inference pass and a single hyper-parameter α.

2. Method

In this section, we describe the preliminary setup, base assumptions, observations and math fomulation of the proposed SmoothCache method.

2.1. Preliminaries

The diffusion process transforms data $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ by adding Gaussian noise over T steps, producing a sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$. This is modeled as a Markov chain, where each step follows:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), \quad (1)$$

where β_t is the noise variance at timestep t and $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$ due to the cumulative effect of the noise added during the diffusion steps. The goal of the reverse process is to recover the original data $\mathbf{x}_0$ from the noisy sample $\mathbf{x}_T$. This is learned via a neural network that parameterizes the reverse transition:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(t)), \quad (2)$$

where $\mu_\theta(\mathbf{x}_t, t)$ is the predicted mean and $\Sigma_\theta(t)$ is the variance, both learned by the model. The full generative process is defined as:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t). \quad (3)$$

The generative model is trained by minimizing the variational bound on the negative log likelihood, learning to denoise $\mathbf{x}_t$ at each timestep and ultimately generate samples from the learned distribution. Traditionally, $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ has been approximated with U-Net style model architectures, but recent works have begun to use Diffusion Transformer (DiT) architectures, which have shown to scale better, especially for more complex tasks such as video generation. DiT architectures consist of repeated blocks containing the Self-attention, Cross-attention, and Feed-forward layers in the traditional Transformer, which are usually the computational bottleneck for both model training and inference.

A key property of diffusion models, which has driven the development of caching techniques, is the high cosine similarity between layer outputs at adjacent timesteps. This pattern of similarity is observed across a variety of generative models and solvers, spanning different modalities such as image, video, and speech diffusion. This high similarity suggests that there are computational redundancies within the diffusion process that can be leveraged to improve efficiency.

2.2. SmoothCache

The objective of SmoothCache is to provide a training-free, model-agnostic strategy of caching and reusing layer outputs such that minimal error is introduced. However, we believe a single optimal static scheme that caches cross-timestep layer output may not exist across different model architectures, solvers, and modalities. For example, we examine the average representation error between layer outputs of consecutive time steps as shown in Fig. 2. For layer output L at timestep t the representation error k timesteps behind is defined as $E(L_t, L_{t+k}) = \frac{\|L_t - L_{t+k}\|_1}{\|L_t\|_1}$.

We observe that layers generally exhibit higher differences in later time steps in the DiT-XL label-to-image model, while layers in the Open-Sora text-to-video model are more sensitive in the first and last diffusion time steps. This discovery emphasizes a need to apply the similarity principle in a generalizable technique, such that different models benefit differently based on error curves.

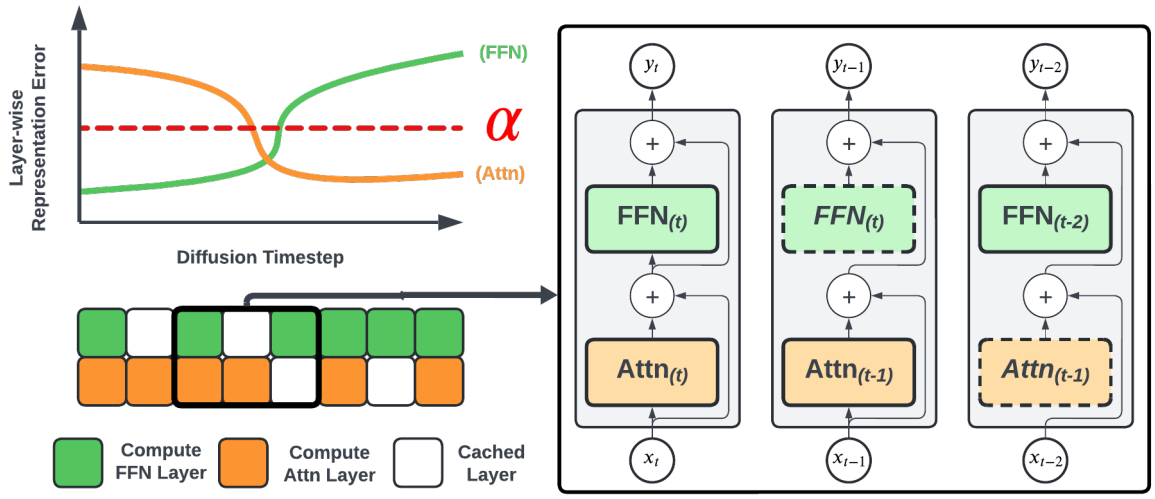


Figure 3. **Illustration of SmoothCache.** When the layer representation loss obtained from the calibration pass is below some threshold α, the corresponding layer is cached and used in place of the same computation on a future timestep. The figure on the left shows how the layer representation error impacts whether certain layers are eligible for caching. The error of the attention (attn) layer is higher in earlier timesteps, so our schedule caches the later timesteps accordingly. The figure on the right shows the application of the caching schedule to the DiT-XL architecture. The output of the attn layer at time $t-1$ is cached and re-used in place of computing FFN $t-2$, since the corresponding error is below α. This cached output is introduced in the model using the properties of the residual connection.

Let L_t represent the output of some layer at timestep t, and let L_{t-k} represent the output of the same layer at some future diffusion timestep $t-k$. By the cross-timestep layer similarity observation, $L_{t-k} \sim L_t$. Thus instead of computing L_{t-k} during the diffusion process, we can approximate the function by using the previously computed L_t. Any time the layer is computed, our method stores L_t in a cache that can be accessed and used in place of future layer computations. We apply caching to the aforementioned computational bottlenecks at the output that precedes a residual connection, shown in Fig. 3. This includes Self-attention and Feed-forward layers in the DiT-XL model, Self-attention, Cross-attention, and Feed-forward layers in the StableAudio model, and Self-attention, Cross-attention, and Feed-forward layers in both the spatial and temporal blocks in the OpenSora model as shown in Fig. 4. Fig. 5 shows that these components comprise of nearly all of the compute that occurs during generation. We also note that the compute distribution varies from model to model.

To determine whether to use a previously cached output, we define the following problem. Let t represent the current timestep, $t+k$ represent the timestep when the cache was previously filled, and i_j represent the jth layer of type i, where $i \in S = \{attn, ffn, ...\}$ depending on the model architecture. Our method is guided by the key hypothesis that caching is effective if the loss between the computed and cached outputs $\mathcal{L}(L_{i_j,t}, L_{i_j,t+k})$, is bounded by some layer-dependent hyper parameter $\alpha_{i_j} > 0$.

Computing $\mathcal{L}(L_{i_j,t}, L_{i_j,t+k})$, is not possible without evaluating $L_{i_j,t}$, which removes any benefit from skipping the layer computation. For a given DiT architecture, we remarkably observe that the difference in layer representation error for two different samples is within a negligibly small threshold, as shown the 95% confidence interval of the plots generated with 10 calibration samples in Fig. 2. This finding suggests that the error curve for a specific model input $\mathcal{L}(L_{i_j,t}, L_{i_j,t+k})$ can be closely approximated by the average error curve for an adequately large set of calibration inputs. In other words, if $\tilde{L}_{i_j,t}$ represents the calibration output for layer $L_{i_j,t}$, then $\mathcal{L}(L_{i_j,t}, L_{i_j,t+k}) \sim \mathcal{L}(\tilde{L}_{i_j,t}, \tilde{L}_{i_j,t+k})$.

A hyper parameter search to find all α_{i_j} has a expo-

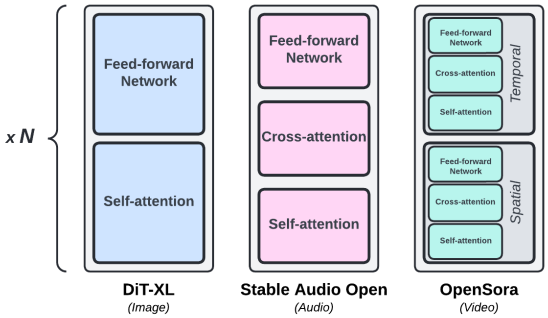


Figure 4. **SmoothCache-Eligible Layers of candidate models**. This visualization highlights the targeted layers that precede residual connections in a DiT block for each architecture. Each model contains N DiT blocks. In the original DiT-XL model, Self-attention and Feed-forward layers are cached. In the Stable Audio Open model, Self-attention, Cross-attention, and Feed-forward layers are cached. In the Open Sora model, Self-attention, Cross-attention, and Feed-forward layers across both the temporal and spatial partitions of the DiT block.

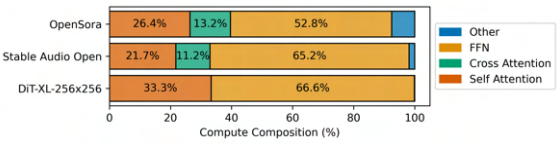


Figure 5. **Layer Compute Composition of candidate models**. These are computed from the MACs of the default model configurations without SmoothCache applied. Note that in all candidate models, SmoothCache eligible layers comprise at least 90% of compute time.

nential search space based on the number of layers and is significantly costly. In order to simplify the caching problem, we define a single hyper parameter $\alpha > 0$ to guide caching for all layers. We define $\mathcal{L}$ as the average L1 relative error of all N layers of type i, which is selected in order to compare true representation errors between layers for all types i and and positional depths j in the network. Additionally, we recognize caching specific layers can introduce errors in future layers of the same type in the network. For example, if we approximate $L_{i_{j-1},t}$ with $L_{i_{j-1},t+k}$, this may introduce noise such that the calibration error $\mathcal{L}(\tilde{L}_{i_j,t}, \tilde{L}_{i_j,t+k})$ no longer correctly approximates the true error $\mathcal{L}(L_{i_j,t}, L_{i_j,t+k})$, which leads to poor caching decisions. In order to mitigate the cascading impact of caching layers, we group caching decisions for all layers of type i, such that all j layers in $L_{i,t+k}$ approximate $L_{i,t}$. Thus, cached output $L_{i_j,t+k}$ is used in place of computing $L_{i_j,t}$ when

$$\mathcal{L}(L_{i_j,t}, L_{i_j,t+k}) \sim \frac{1}{N}\sum_{j=1}^{N}\frac{\|\tilde{L}_{i_j,t} - \tilde{L}_{i_j,t+k}\|_1}{\|\tilde{L}_{i_j,t}\|_1} < \alpha \quad (4)$$

Using cached outputs is computationally inexpensive, and allows for significant speedup of the diffusion inference process. As Eq. 4 makes no prior assumptions about the specific properties of the particular diffusion process being cached, it can be applied across multiple architectures and modalities, only requiring a brief linear search to find an optimal α hyperparameter for each sampling configuration. Additionally, because caching decisions are only dependent on calibration error, they do not change at model runtime. This ensures compatibility with existing graph compilation optimizations.

3. Experiments

3.1. Experiment Setup

Models, Datasets, and Solvers In order to demonstrate the effectiveness of our technique across DiT architectures and prove the model-agnostic claim, we evaluate SmoothCache on multiple candidate diffusion models across a variety of modalities, across different numbers of diffusion steps.

Text to Image Generation Firstly, we select the original DiT-XL-256×256 label to image diffusion model, using the original released model weights [2]. Since the DiT family of models is trained on the ImageNet1k dataset, we generate 50,000 256×256 images for evaluation. We test SmoothCache with the DDIM solver, and use classifier free guidance with a scale of 1.5.

Text to Video Generation Secondly, we select the OpenSORA text to video model. We use the pre-trained OpenSORA v1.2 model, and evaluate its performance using VBench evaluation protocol using videos generated from the 946 prompts from the VBench prompt suite on 2 second 480p videos with a 9:16 aspect ratio. We use flash attention and the bfloat16 datatype for inference. We test SmoothCache using the Rectified Flow [17] solver for 30 sampling steps across 1000 Euler steps, using the default CFG scale of 7.0.

Text to Audio Generation Lastly, we select the Stable Audio Open text to audio model. We follow the same inference protocol as described in the original paper [5]. We run inference Stable Audio Open with DPM-Solver++ (3M) SDE for 100 steps with classifier free guidance scale of 7.0.

Evaluation Metrics We use a variety of standard metrics in each domain to demonstrate the efficiency to quality trade off. We record the Multiply-Accumulate Operations (MACs) and latency of the full diffusion process. We derive the acceleration ratio from baseline sampling latencies. To measure generation quality, we evaluate the candidate models on common metrics in the corresponding domain. For DiT-XL-256×256, we generate 50,000 images with the specified configuration and report the FID, IS, and sFID. For OpenSORA, we evaluate the performance of the model using VBench [10], Learned Perceptual Image Patch Similarity (LPIPS) [39], Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) [37], and generate 946 videos based on the VBench suite prompts. LPIPS, PSNR and SSIM are computed relative to the non-cached videos. We report the final scaled VBench score. Lastly, for Stable Audio Open, we use the exact same evaluation protocol described in the Stable Audio Open technical report [5] using the same evaluation code[3], reporting CLAP, $\text{FD}_{\text{OpenL3}}$ and KL_{PaSST} metrics for AudioCaps [11], MusicCaps without singing prompts [2], and Song Describer without singing prompts [23]. All speed results are measured on a single H100-80G GPU, and averaged across 50 runs.

Implementation Details As mentioned previously, we apply caching to layers that precede residual connections. In the DiT-XL-256×256 model, this includes the Self-attention and Feed-forward modules in the DiT block. In OpenSora V1.2, we cache the temporal Self-attention,

[2] https://github.com/facebookresearch/DiT
[3] https://github.com/Stability-AI/stable-audio-metrics

Cross-attention and Feed-forward modules as well as their equivalent spatial variants (for a total of 6 types of modules). In Stable Audio Open, we cache the Self-attention, Cross-attention and Feed-forward modules. For each architecture, solver, and preset number of diffusion steps, we obtain the representation errors $L(F_{i_j,t}, F_{i_j,t+k})$ by computing differences in layer outputs for some random samples. For all modalities, we use 10 samples for calibration. Ablations show that the choice of the number of calibration samples does not matter much, something that can be observed in Fig. 2. We also choose which samples to generate the error curves based on whether we did conditional generation during calibration or not, which we only do for OpenSora and Stable Audio Open. We fix $k \in \{1, 2, 3\}$ for DiT-XL-256×256 and Stable Audio Open as we determine the layer representation error to grow too large past a difference of 4 timesteps. In OpenSORA, k goes up to 5 in particular due to relatively low error in the cross-attention components of the network. For DiT, we calibrate on samples generated unconditionally using the null prompt. We calibrate OpenSora on conditionally generated 480p 2s videos with randomly sampled prompts from VidProM [36]. For Stable Audio Open, we calibrate using randomly sampled prompts from the AudioCaps validation set.

3.2. Results

We present the results of SmoothCache on DiT-XL-256×256, OpenSora, and Stable Audio Open in Tables 1, 2, and 3 respectively. In order to provide a fair comparison between acceleration and quality trade offs, we compare the default solver used in the model, such as DDIM or DPM++, with a SmoothCache implementation with specified hyper parameters. We find two configurations with identical acceleration ratios show SmoothCache to perform better on various quality metrics, indicating superior performance over the typical acceleration/quality tradeoff. For all results, we run 5 trials and report the mean and standard deviation for each metric.

3.2.1 Quantitative results

Comparing against existing literature In order to further investigate the true effectiveness of our technique, we compare SmoothCache to existing DiT caching techniques in literature developed concurrently with SmoothCache, such Fast-Forward Caching (FORA) [30] and Learning-to-Cache (L2C) [20] for Label-to-Image generation. We note that the above methods relies on specific properties of the relevant model architecture, and does not necessarily translate across different modalities, while SmoothCache is agnostic to those considerations as it models the caching scheme directly off the observed error curves. FORA does not work in OpenSora or Stable Audio Open due to the difference in the error curves as seen in Fig. 2 and hence we do not report the results here. We see that SmoothCache outperforms FORA across different inference times at 50 sampling steps, yielding similar performance at lower inference time or better performance with the same inference time. The one exception here is L2C, which requires leveraging the full ImageNet training set to learn a policy for a given number of sampling steps. Changing the number of sampling steps requires a full retraining, and L2C has a theoretical maximum of a 2× speedup because the caching policy is only learned with skipping every other step, limitations that SmoothCache does not have.

Table 1. Results For DiT-XL-256×256 on using DDIM Sampling, sorted by TMACs. Note that L2C is not training free.

Schedule	Steps	FID (↓)	sFID (↓)	IS (↑)	TMACs	Latency (s)
L2C	50	2.27 ± 0.04	4.23 ± 0.02	245.8 ± 0.7	278.71	6.85
No Cache	50	2.28 ±0.03	4.30 ±0.02	241.6 ± 1.1	365.59	8.34
Ours (α = 0.08)	50	**2.28 ±0.03**	**4.29 ±0.02**	**241.8 ± 0.9**	**336.37**	**7.62**
FORA (n=2)	50	2.65 ±0.04	4.69 ±0.03	238.5 ± 1.1	190.25	5.17
Ours (α = 0.18)	50	**2.65 ±0.04**	**4.65 ±0.03**	**238.7 ± 1.1**	**175.65**	**4.85**
FORA (n=3)	50	3.31 ±0.05	5.71 ±0.06	230.1 ± 1.3	131.81	4.12
Ours (α = 0.22)	50	**3.14 ±0.05**	**5.19 ±0.04**	**231.7 ± 1.0**	**131.81**	**4.11**
No Cache	30	2.66 ±0.04	4.42 ±0.03	234.6 ± 1.0	219.36	4.88
FORA (n=2)	30	3.79 ±0.04	5.72 ±0.05	222.2 ± 1.2	117.08	3.13
Ours (α = 0.35)	30	**3.72 ±0.04**	**5.51 ±0.05**	**222.9 ± 1.0**	**117.08**	**3.13**
No Cache	70	2.17 ±0.02	4.33 ±0.02	242.3 ± 1.6	511.83	11.47
FORA (n=2)	70	2.36 ±0.02	4.46 ±0.03	242.2 ± 1.3	263.43	7.15
Ours (α = 0.08)	70	**2.37 ±0.02**	**4.29 ±0.03**	**242.6 ± 1.5**	**248.8**	**6.9**
FORA (n=3)	70	2.80 ±0.02	5.38 ±0.04	238.0± 1.2	175.77	5.61
Ours (α = 0.12)	70	**2.68 ±0.02**	**4.90 ±0.04**	**238.8 ± 1.3**	**175.77**	**5.62**

Examining inference speedup/quality tradeoff Stable Audio Open and DiT-XL/2-256×256 provides the highest speed/quality tradeoff compared to OpenSora, which gives around a 10% speedup latency wise and around 16-22% reduction in MACs comapred to an almost 20-60% speedup for the other modalities, whose discrepancy could be attributed by the larger overhead of non-DiT components (since inference latency is measured end-to-end). We also observe that the amount of inference speed/quality tradeoff looks directly correlated to the error deviation between calibration samples which can be seen in Fig. 2, with the higher variance in error among individual samples across timesteps for OpenSora versus that for Stable Audio Open and DiT-XL/2-256×256.

3.2.2 Qualitative results

We show visual examples for all modalities. We show 256×256 images for image generation, using null-

Table 2. Results For OpenSora on Rectified Flow (30 steps).

Schedule	VBench (%) (↑)	LPIPS (↓)	PSNR (↑)	SSIM (↑)	TMACs	Latency (s)
No Cache	79.36 ±0.19	-	-	-	1612.1	28.43
Ours (α = 0.02)	78.76 ±0.38	0.5852 ± 0.0352	11.08 ± 0.96	0.5085 ± 0.0315	1388.5	26.57
Ours (α = 0.03)	78.10 ±0.51	0.5347 ± 0.1119	12.62 ± 2.81	0.5601 ± 0.0812	1321.1	26.17

Table 3. Results For Stable Audio Open on DPMSolver++(3M) SDE on 3 datasets.

	AudioCaps			MusicCaps (No Singing)			Song Describer (No Singing)				
Schedule	FD_{OpenL3} (↓)	KL_{PaSST} (↓)	CLAP (↑)	FD_{OpenL3} (↓)	KL_{PaSST} (↓)	CLAP (↑)	FD_{OpenL3} (↓)	KL_{PaSST} (↓)	CLAP (↑)	TMACs	Latency (s)
No Cache	81.7 ± 6.8	2.13 ± 0.02	0.287 ± 0.003	82.7 ± 2.1	0.931 ± 0.012	0.467± 0.001	105.2 ± 6.3	0.551 ±0.024	0.421 ± 0.003	209.82	5.65
Ours ($\alpha = 0.15$)	84.5 ± 6.7	2.15 ± 0.02	0.285 ± 0.003	85.9 ± 2.3	0.942 ± 0.012	0.467 ± 0.001	106.2 ± 6.6	0.555 ± 0.024	0.420 ± 0.003	170.75	4.59
Ours ($\alpha = 0.30$)	89.6 ± 6.3	2.17 ± 0.02	0.271 ± 0.003	82.0 ± 1.5	0.962 ± 0.012	0.448 ± 0.001	131.3 ± 5.9	0.596 ± 0.028	0.392 ± 0.003	136.16	3.72



Figure 6. **SmoothCache results on DiT-XL/2-256×256** for unconditional generation with 50 DDIM sampling steps on ImageNet-1k for thresholds 0.08 and 0.18, as well as for Static Caching.

conditional prompts. For audio, we show the log-mel spectrogram in order to visualize the waveform for conditional prompt generation across the 3 datasets we measure. For video, we show the first, middle and last frame of 2 second 480p 24 fps videos at a 9:16 aspect ratio.

We see that for DiT-XL/2-256×256 outputs in Fig. 6, even though there is a drop of FID with SmoothCache applied, that there is some noticeable difference in performance quality when there is a higher threshold applied, but there exists a threshold where the quality of the model performance is visually indistinguishable from the model without caching. This shows the fine-granularity in inference speed/quality tradeoff that SmoothCache affords, while static caching as shown in FORA can only yield the lower quality SmoothCache threshold. We observe particularly that this degradation is less pronounced in Stable Audio Open in Fig. 7, where the audio waveforms look visually identical, and when listened to do not have perceptible differences. This is consistent with the qualitative results which show only a minor degradation across all measured metrics across all datasets. For OpenSora, we see more significant differences in when caching in Fig. 8, with noticeable artifacting when the higher caching threshold is applied. This is consistent with the quantitative results, and suggests this architecture/modality is more sensitive to caching than the other modalities, which makes sense due to the complex spatial/temporal modelling that other modalities explored do not have to deal with.

3.3. Ablations

We ablate our investigations by showing how model performance varies across different step sizes, and show the robustness of SmoothCache for the DiT-XL-256×256 model.

Examining the Caching/Sample Step Pareto Front We attempt to show via comparison with static caching that our technique yields a better Pareto front along different sampling steps through the image generation results. We note that all caching techniques have varying performance as you vary the number of solver sampling steps, and we show that despite to the minimal assumptions SmoothCache makes about the sampler and model architecture, SmoothCache at least matches up to other caching strategies if not outright beating them across multiple fronts. Table 1 shows that even with lower or higher number of sampling steps, SmoothCache outperforms FORA across multiple inference speed/quality points at different number of sampling steps. We observe that SmoothCache outperforms Static Caching for multiple sampling configurations. While L2C slightly outperforms SmoothCache on DDIM sampling for DiT-XL, SmoothCache does not require expensive training over ImageNet1k, and generalizes to different model architectures and sampling configurations.

Calibration sample size We note that the choice of the number of calibration samples does not adversely affect the caching schedule generated. Empirically, we observe that 10 samples for all 3 models investigated in this paper is usually enough to reliably regenerate the same caching sched-

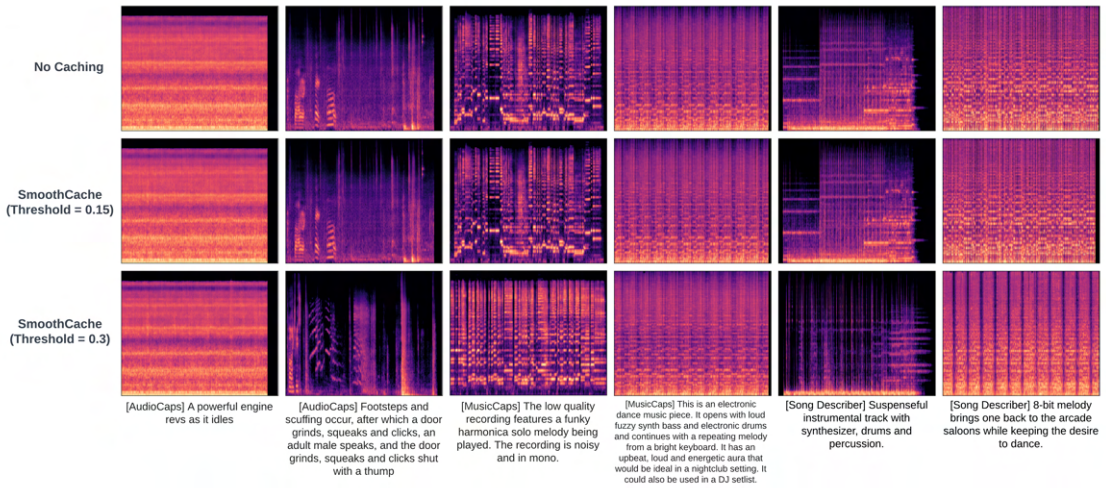


Figure 7. **SmoothCache Results on Stable Audio Open** for threshold 0.15 and 0.3. Log-Mel Spectrograms are shown.

ule given the same α. We however note that the confidence interval of the different error curves vary from modality to modality as seen in Fig. 2.

4. Limitations and Future Work

The main limitation of the SmoothCache technique is its reliance on the repeated DiT block architecture, particularly the residual connections following the aforementioned computational bottleneck layers. These connections allow for the output y_t to be approximated by some linear function of input and previously cached layers $f(x_{t+k}) + x_t$, a feature we consider essential for its caching effectiveness. Additionally, the performance gains from SmoothCache are strongly linked to the computational intensiveness of the candidate layers. Consequently, it may yield smaller performance improvements when applied to DiT networks with limited depth or width.

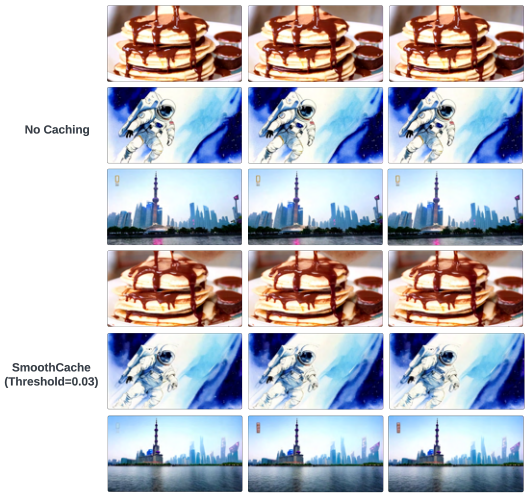


Figure 8. **SmoothCache Results on OpenSora** for threshold 0.03 for 2s 480p videos. We show the first, middle and last frame of each video. We use the following prompts, in order from top to bottom: (1) `Chocolate sauce is poured slowly over a stack of fluffy pancakes.` (2) `an astronaut floating in space with the earth in the background` (3) `The bund Shanghai, pan right`

A secondary limitation lies in the assumption that errors from approximating outputs of earlier layers have minimal impact on the loss function guiding caching decisions for deeper layers. For instance, if the first Self-attention layer is approximated using a cached result from a previous timestep, caching the second Self-attention layer could introduce further output discrepancies, even if the calibration loss suggests low error. This is because the calibration loss is computed when no caching is performed, which may not fully model true approximation errors during SmoothCache-enabled inference. We address this issue by grouping caching and computation decisions for layers of the same type at each timestep. However, this does not fully resolve dependency issues between different layer types, leaving room for further optimization in future work.

We also highlight a phenomenon that future work can investigate where the pareto front of inference speed/quality seems to correlate with the variance in error across different calibration samples, with architectures/modalities that have higher variance between sample error curves having narrower fronts than those which have lower variance.

5. Conclusion

We introduce SmoothCache, a training-free caching technique that works across multiple diffusion solvers and modalities. Using the layer representation error of a calibration inference pass, SmoothCache identifies redundancies in the diffusion process, enabling caching and reuse of output feature maps, reducing the number of computationally expensive layer operations. Our evaluations show it at least matches or exceeds the performance of existing modality-specific caching methods.

References

[1] Shubham Agarwal, Subrata Mitra, Sarthak Chakraborty, Srikrishna Karanam, Koyel Mukherjee, and Shiv Saini. Approximate caching for efficiently serving diffusion models. *arXiv preprint arXiv:2312.04429*, 2023. 2

[2] Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. Musiclm: Generating music from text, 2023. 5

[3] Junsong Chen, Jincheng YU, Chongjian GE, Lewei Yao, Enze Xie, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In *The Twelfth International Conference on Learning Representations*, 2024. 1

[4] Pengtao Chen, Mingzhu Shen, Peng Ye, Jianjian Cao, Chongjun Tu, Christos-Savvas Bouganis, Yiren Zhao, and Tao Chen. δ-dit: A training-free acceleration method tailored for diffusion transformers. *arXiv preprint arXiv:2406.01125*, 2024. 2

[5] Zach Evans, Julian D Parker, CJ Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons. Stable audio open. *arXiv preprint arXiv:2407.14358*, 2024. 1, 2, 5

[6] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models, 2023. 1, 2

[7] Yefei He, Luping Liu, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. Ptqd: Accurate post-training quantization for diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024. 1, 2

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1, 2

[9] Seongmin Hong, Kyeonghyun Lee, Suh Yoon Jeon, Hyewon Bae, and Se Young Chun. On exact inversion of dpm-solvers, 2023. 2

[10] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 5

[11] Chris Dongjoo Kim, Byeongchang Kim, Hyunmin Lee, and Gunhee Kim. Audiocaps: Generating captions for audios in the wild. In *NAACL-HLT*, 2019. 5

[12] PKU-Yuan Lab and Tuzhan AI etc. Open-sora-plan, 2024. 1, 2

[13] Matthew Le, Apoorv Vyas, Bowen Shi, Brian Karrer, Leda Sari, Rashel Moritz, Mary Williamson, Vimal Manohar, Yossi Adi, Jay Mahadeokar, et al. Voicebox: Text-guided multilingual universal speech generation at scale. *Advances in neural information processing systems*, 36, 2024. 1, 2

[14] Yanyu Li, Huan Wang, Qing Jin, Ju Hu, Pavlo Chemerys, Yun Fu, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Snapfusion: Text-to-image diffusion model on mobile devices within two seconds. *Advances in Neural Information Processing Systems*, 36, 2024. 1, 2

[15] Huadai Liu, Rongjie Huang, Xuan Lin, Wenqiang Xu, Maozong Zheng, Hong Chen, Jinzheng He, and Zhou Zhao. Vitts: visual text-to-speech with scalable diffusion transformer. *arXiv preprint arXiv:2305.12708*, 2023. 1, 2

[16] Haozhe Liu, Wentian Zhang, Jinheng Xie, Francesco Faccio, Mengmeng Xu, Tao Xiang, Mike Zheng Shou, Juan-Manuel Perez-Rua, and Jürgen Schmidhuber. Faster diffusion via temporal attention decomposition, 2024. 2

[17] Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. 1, 2, 5

[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*, 35:5775–5787, 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. 1, 2

[20] 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. 2, 6

[21] 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*, pages 15762–15772, 2024. 2

[22] Xin Ma, Yaohui Wang, Gengyun Jia, Xinyuan Chen, Ziwei Liu, Yuan-Fang Li, Cunjian Chen, and Yu Qiao. Latte: Latent diffusion transformer for video generation. *arXiv preprint arXiv:2401.03048*, 2024. 1, 2

[23] Ilaria Manco, Benno Weck, Seungheon Doh, Minz Won, Yixiao Zhang, Dmitry Bogdanov, Yusong Wu, Ke Chen, Philip Tovstogan, Emmanouil Benetos, Elio Quinton, György Fazekas, and Juhan Nam. The song describer dataset: a corpus of audio captions for music-and-language evaluation. In *Machine Learning for Audio Workshop at NeurIPS 2023*, 2023. 5

[24] Shentong Mo, Enze Xie, Ruihang Chu, Lanqing Hong, Matthias Niessner, and Zhenguo Li. Dit-3d: Exploring plain diffusion transformers for 3d shape generation. *Advances in neural information processing systems*, 36:67960–67971, 2023. 1, 2

[25] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6038–6047, 2023. 2

[26] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023. 1, 2

[27] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image

synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 2

[28] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022. 2

[29] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022. 1, 2

[30] 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. 2, 6

[31] Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. Post-training quantization on diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1972–1981, 2023. 1, 2

[32] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015. 1, 2

[33] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. 2

[34] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 1, 2

[35] Jiangshan Wang, Junfu Pu, Zhongang Qi, Jiayi Guo, Yue Ma, Nisha Huang, Yuxin Chen, Xiu Li, and Ying Shan. Taming rectified flow for inversion and editing, 2024. 2

[36] Wenhao Wang and Yi Yang. Vidprom: A million-scale real prompt-gallery dataset for text-to-video diffusion models. In *Thirty-eighth Conference on Neural Information Processing Systems*, 2024. 6

[37] Zhou Wang and A.C. Bovik. A universal image quality index. *IEEE Signal Processing Letters*, 9(3):81–84, 2002. 5

[38] 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*, pages 6211–6220, 2024. 2

[39] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 5

[40] Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. *arXiv preprint arXiv:2408.12588*, 2024. 2

[41] Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all, 2024. 1, 2