AsyncDiff: Parallelizing Diffusion Models by Asynchronous Denoising

Zigeng Chen, Xinyin Ma, Gongfan Fang, Zhenxiong Tan, Xinchao Wang*
National University of Singapore
zigeng99@u.nus.edu, xinchao@nus.edu.sg

Figure 1: We introduce a new distributed acceleration paradigm that attains a 2.8x speed-up on Stable Diffusion XL while maintaining **pixel-level consistency**, using four NVIDIA A5000 GPUs.

Abstract

Diffusion models have garnered significant interest from the community for their great generative ability across various applications. However, their typical multi-step sequential-denoising nature gives rise to high cumulative latency, thereby precluding the possibilities of parallel computation. To address this, we introduce *AsyncDiff*, a universal and plug-and-play acceleration scheme that enables model parallelism across multiple devices. Our approach divides the cumbersome noise prediction model into multiple components, assigning each to a different device. To break the dependency chain between these components, it transforms the conventional sequential denoising into an asynchronous process by exploiting the high similarity between hidden states in consecutive diffusion steps. Consequently, each component is facilitated to compute in parallel on separate devices. The proposed strategy significantly reduces inference latency while minimally impacting the generative quality. Specifically, for the Stable Diffusion v2.1, *AsyncDiff* achieves a 2.7x speedup with negligible degradation and a 4.0x speedup with only a slight reduction of 0.38 in CLIP Score, on four NVIDIA A5000 GPUs. Our experiments also demonstrate *AsyncDiff* can be readily applied to video diffusion models with encouraging performances. Code is available at `https://github.com/czg1225/AsyncDiff`

1 Introduction

Diffusion models [13] stand out in generative modeling and have significantly advanced various fields including text-to-image [43, 41, 45, 46, 72, 78] and text-to-video generation [64, 9, 61, 21, 2],

*Correspoding Author

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-3015

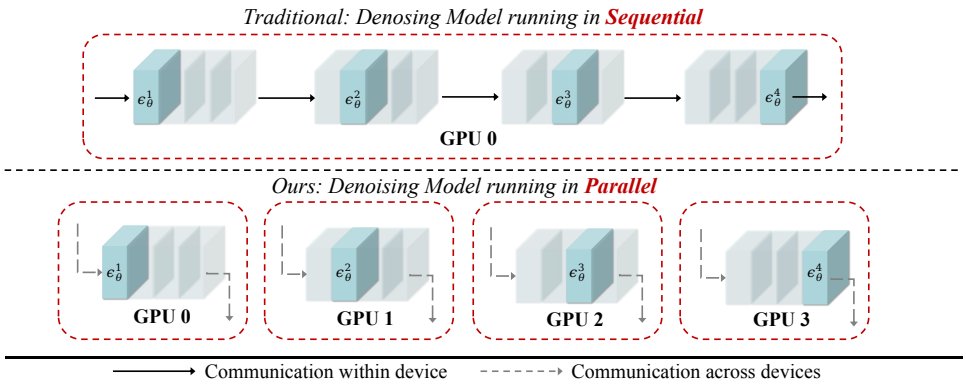

Figure 2: By preparing each component's input beforehand, we enable parallel computation of the denoising model, which substantially reduces latency while minimally affecting quality.

image translation [49, 56, 23], audio generation[22, 14, 44], style transfer[62, 4, 17], low-level vision tasks [47, 60, 40, 26, 8, 70, 3], image editing [19, 66, 51, 77], and 3D model generation [42, 18, 37], among others. However, their widespread application is hindered by the high latency inherent in their multi-step sequential denoising process. This issue becomes more pronounced as the complexity and size of the models increase to enhance generative quality.

In response to these challenges, significant research efforts are directed toward enhancing the efficiency of diffusion models. Notably, training-free acceleration methods have garnered increasing popularity due to their low cost and convenience. Numerous studies [35, 63, 76, 67, 53, 25, 33, 57, 34] improve inference speed by skipping redundant calculations in the denoising process. As computational resources grow rapidly, distributing computations across multiple devices has become a more promising approach. Recent advances [52, 24, 58] demonstrate that using distributed computing to parallelize inference effectively increases the acceleration ratio for diffusion models while maintaining acceptable generative quality. Though these methods succeed in parallelizing the diffusion models, they require iterative refining [52] or displaced patch parallelism [24], resulting in a larger number of model evaluations or low GPU utilization correspondingly.

Thus, we wish to propose a new parallel paradigm for diffusion, akin to the model parallelism in distributed computing [15, 38, 28, 16, 39, 65], which divides the denoising model into several components to be distributed on different GPUs. The primary challenge lies in the inherent sequential denoising process of diffusion models. Each step in this process depends on the completion of its predecessor, forming a dependency chain that impedes parallelization and significantly increases inference latency. Our approach seeks to disrupt this chain, allowing for the parallel execution of the denoising model while closely approximating the results of the sequential process.

In this paper, we introduce *AsyncDiff*, a universal, distributed acceleration paradigm that innovatively explores model parallelism in diffusion models. As shown in Fig 2, our method sequentially partitions the heavyweight denoising model ϵ_θ into multiple components $\{\epsilon_\theta^n\}_{n=1}^N$ based on computational load, assigning each to a separate device. Our core idea lies in decoupling the dependencies between these cascaded components by leveraging the high similarity in hidden states across consecutive diffusion steps. After the initial warm-up steps, each component takes the output from the previous component's prior step as the approximation of its original input. This transforms the traditional sequential denoising into an asynchronous process, allowing components to predict noise for different time steps in parallel. Additionally, we incorporate stride denoising to skip redundant calculations and reduce the frequency of communication between devices, further enhancing efficiency.

Through extensive testing across multiple base models, our method effectively distributes the computational burden across various devices, substantially boosting inference speed while maintaining quality. Specifically, with the text-to-image model Stable Diffusion v2.1 [43], our method achieves a 1.8x speedup with only a marginal 0.01 drop in CLIP Score [11], and a 4.0x speedup with a slight 0.38 reduction in CLIP Score on two and four NVIDIA A5000 GPUs, respectively. For video diffusion models, AnimateDiff [9] and Stable Video Diffusion [2], our approach significantly reduces latency by tens of seconds, effectively preserving video quality.

https://doi.org/10.52202/079017-3015

In summary, we present a novel distributed acceleration method for diffusion models that significantly reduces inference latency with minimal impact on generation quality. This is achieved by replacing the sequential denoising process with an asynchronous process, allowing each component of the denoising model to run independently across different devices. Extensive experiments on both image and video diffusion models strongly demonstrate the effectiveness and versatility of our method.

2 Related Works

Diffusion Models. Diffusion models have attracted significant attention due to their powerful generative capabilities across various tasks. Sohl-Dickstein et al. [54] first proposed diffusion probabilistic models. Ho et al. [13] with the introduction of Denoising Diffusion Probabilistic Models (DDPM), enhancing training efficiency and generation quality. Rombach et al. [43] advanced these models by incorporating latent spaces, enabling high-resolution image generation. Despite these advancements, the high latency of the iterative denoising process remains a limitation.

Inference Acceleration. Training-based acceleration methods focus on reducing sampling steps [48, 71, 32, 50, 69] or optimizing model architectures [27, 80, 7, 73, 68, 6]. However, these methods incur high training costs and complexity. Training-free methods are gaining popularity due to their ease of use. Some approaches develop fast solvers for SDE or ODE to improve sampling efficiency [31, 1, 30, 74, 81]. Other works [35, 63, 76, 67, 53, 25, 33, 79] observed special characteristics of diffusion models and skipped the redundant computation within the denoising process.

Parallelism. The parallelism strategy presents a promising yet underexplored approach to accelerating diffusion models. ParaDiGMS [52] implements Picard iterations for parallel sampling, yet its practical speed-up ratio is modest, and it struggles to maintain consistency with original outputs. Faster Diffusion [25] introduces encoder propagation but significantly compromises quality, and its parallelization remains theoretical. Distrifusion [24] adopts patch parallelism, dividing high-resolution images into sub-patches to facilitate parallel inference on each patch by reusing stale activation maps from each layer. However, this approach lacks flexibility across different data types or tasks, often encountering low resource utilization. Furthermore, its reliance on reusing per-layer activation maps greatly increases GPU memory demands thus introducing additional challenges for realistic applications. In contrast, our method uniquely implements model parallelism through asynchronous denoising, achieving substantial acceleration while maintaining a stable resource usage ratio and minimal impact on quality.

3 Methods

3.1 Preliminary

Diffusion models [13] are a dominant class of generative models that transform Gaussian noise into complex data distributions via a Markov process. The forward process is defined by:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I), \tag{1}$$

where $\{\beta_t\}$ progressively increases noise until the data becomes indistinguishable from noise. The reverse process, essential for data reconstruction, involves iterative denoising:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I), \tag{2}$$

where $\mu_\theta(x_t, t)$ is the predicted mean and σ_t^2 is the variance. For DDIMs [55], the reverse update is deterministic:

$$x_{t-1} = \sqrt{\frac{\alpha_{t-1}}{\alpha_t}}x_t + \sqrt{1-\alpha_{t-1}}\left(1 - \sqrt{\frac{1-\alpha_t}{\alpha_{t-1}}}\right)\epsilon_\theta(x_t, t), \tag{3}$$

where α_t is the cumulative product of $(1-\beta_t)$. These processes are computationally intensive, influencing the quality of generated samples and necessitating efficient inference methods for practical applications.

3.2 Asynchronous Diffusion Model

Traditional diffusion models employ a sequential and synchronous denoising process. At each time step t, the noise-prediction model ϵ_θ estimates the noise ϵ_t based on the noisy image x_t and the time

https://doi.org/10.52202/079017-3015

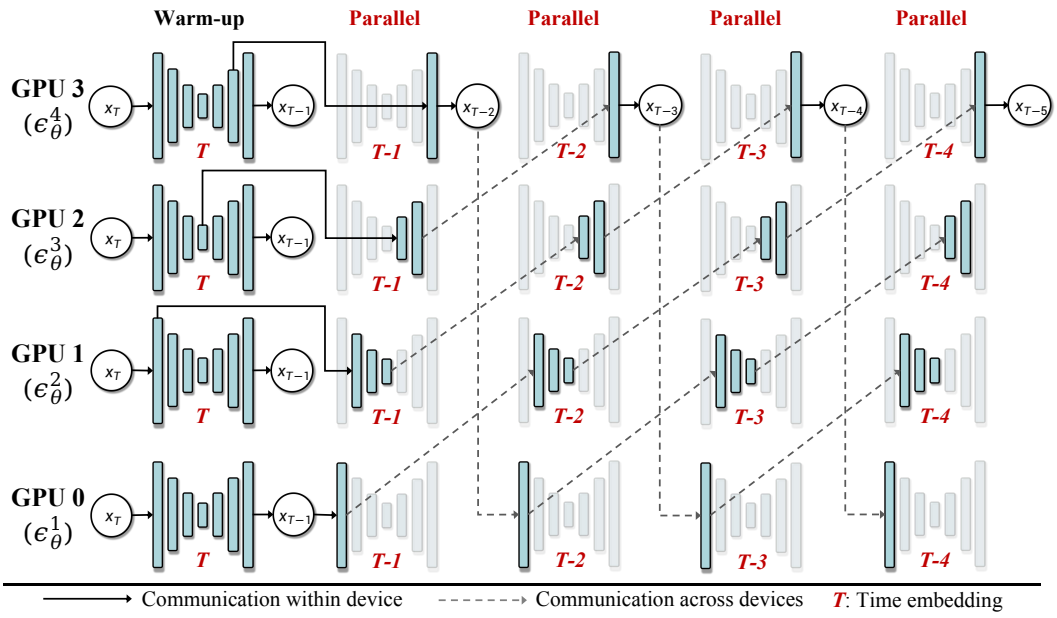

Figure 3: Overview of the asynchronous denoising process. The denoising model ϵ_θ is divided into four components $\{\epsilon_\theta^n\}_{n=1}^4$ for clarity. Following the warm-up stage, each component's input is prepared in advance, breaking the dependency chain and facilitating parallel processing.

embedding t. The image for the next step, x_{t-1}, is then generated using a sampler function $S(x_t, \epsilon_t, t)$. This process is iterative, where the generation of ϵ_t at each step is dependent on the completion of the previous denoising step, making the process slow, particularly when ϵ_θ is computationally intensive.

To address the limitations of high latency in diffusion models, leveraging multiple GPUs for distributed inference is a promising solution. Existing studies primarily focus on patch parallelism [24], where the input image is divided into patches, each processed on a different GPU. While this strategy efficiently distributes computational loads, it still retains the bottleneck of sequential denoising, as each patch must undergo the complete denoising process iteratively. In contrast, our asynchronous diffusion model innovatively introduces a model parallelism strategy. By approximating the sequential denoising as an asynchronous process, this approach enables parallel inference of the noise prediction model, effectively reducing latency and breaking the constraints of sequential execution.

Asynchronous Denoising. Figure 3 illustrates our approach to the asynchronous denoising. For a denoising process consisting of T steps, the initial w steps are designated as a warm-up phase, where w is significantly smaller than T. During this phase, the denoising model ϵ_θ operates using standard sequential inference. After warm-up steps, rather than splitting the input image, we partition the denoising model ϵ_θ into N sequential components, expressed as $\epsilon_\theta = \{\epsilon_\theta^1, \epsilon_\theta^2, ..., \epsilon_\theta^N\}$. Each component is divided to handle a comparable computational load and assigned to a distinct device. This equitable division aims to equalize the time cost of each component to approximately $l(\epsilon_\theta)/N$, thus minimizing the overall maximum latency. In this setup, original noise prediction for x_t can be represented as a cascading operation through these sub-models, defined mathematically as:

$$\epsilon_t = \epsilon_\theta(x_t, t) = \epsilon_\theta^N(\epsilon_\theta^{N-1}(\dots \epsilon_\theta^2(\epsilon_\theta^1(x_t, t), t) \dots, t), t). \tag{4}$$

Although each device can independently compute its assigned component, the dependency chain persists because the input for each component $\epsilon_{\theta,n}$ is derived from the output from its preceding component $\epsilon_{\theta,n-1}$. Therefore, despite the distribution of model components across multiple devices, full parallelization is constrained by these sequential dependencies.

Our principal innovation is to break the dependency between cascaded components by utilizing hidden features from previous steps. Observations indicate that the hidden states of each block in the denoising model always exhibit substantial similarity across adjacent time steps. Leveraging this, each component at time step t can take the output from the preceding component at time step $t-1$ as the approximation of its original input. Specifically, the n-th component $\epsilon_\theta^n(, t)$ receives the output of

https://doi.org/10.52202/079017-3015

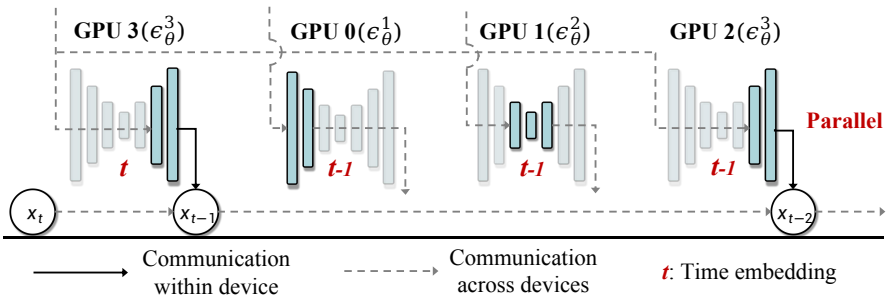

Figure 4: Illustration of stride denoising. The model ϵ_θ is divided into three components $\{\epsilon_\theta^n\}_{n=1}^3$, with a stride S of 2 for clarity. Components ϵ_θ^1 and ϵ_θ^2 are skipped at time step t. A single parallel batch results in the completion of denoising for two steps, producing x_{t-1} and x_{t-2}.

$\epsilon_\theta^{n-1}(\cdot, t-1)$. This alteration allows the noise prediction for x_t to be represented as follows:

$$\epsilon_t = \epsilon_\theta^N(\epsilon_\theta^{N-1}(\dots\epsilon_\theta^2(\epsilon_\theta^1(x_{t+N-1}, t+N-1), t+N-2)\dots, t+1), t). \tag{5}$$

In this new framework, noise prediction ϵ_t is derived from components executed across N previous time steps. This transforms the denoising process from sequential to asynchronous, as the prediction of noise ϵ_t already begins before denoising at step $t+1$ is completed. At each time step, the N components are running as parts of the noise prediction model for the next N steps. Specifically, the n-th component ϵ_θ^n, computed in parallel at time t, contributes to the noise prediction for the future time step $t-N+n$. Figure 3 depicts this asynchronous process using a U-net model with N set to 4. The strong resemblance of hidden states between consecutive diffusion steps enables the asynchronous process to closely mimic the denoising results of the original sequential process.

Model Parallelism. By transitioning to an asynchronous denoising strategy, the dependencies among components within the same time step are eliminated. This adjustment allows each component's input for time step t to be prepared in advance, enabling the N split components to be processed concurrently across multiple devices. Once computed, the outputs from each component must be stored and then broadcasted to other devices to facilitate parallel processing for subsequent time steps. In contrast, in the traditional sequential denoising process, the time cost for each step accumulates as follows:

$$C_{seq}(t) = C(\epsilon_\theta^1) + C(\epsilon_\theta^2) + \ldots + C(\epsilon_\theta^N). \tag{6}$$

By adopting asynchronous denoising to enable parallel computation of each component, the cost for each time step is now given by:

$$C_{asy}(t) = \max(C(\epsilon_\theta^1), C(\epsilon_\theta^2), ..., C(\epsilon_\theta^N)) + C(\text{comm.}), \tag{7}$$

where $\max()$ represents taking the maximum value, and $C(\text{comm.})$ indicates the communication cost across multiple GPUs. As the model components are equally divided by computational load, their time costs are similar, allowing us to approximate the overall cost of each time step as:

$$C_{asy}(t) \approx \frac{C_{seq}(t)}{N} + C(\text{comm.}). \tag{8}$$

Since the communication overhead $C(\text{comm.})$ is generally much lower than the model's execution time, it leads to significant overall cost reductions. Moreover, increasing N further reduces time costs but complicates the accurate approximation of the original denoising process.

Stride Denoising. While asynchronous denoising reduces latency by parallelizing the denoising model, it completes only one denoising step at a time. To enhance efficiency, we introduce stride denoising, which completes multiple denoising steps simultaneously through a single parallel computation. The diagram is illustrated in Figure 4, where we set the stride to 2 for clarity. Unlike the continuous broadcasting of hidden states at each time step, stride denoising broadcasts them every two steps. As depicted, at time step t, we conduct denoising alone, and at time step $t-1$, we compute and broadcast the hidden states for the next parallel computation round. Consequently, the hidden states from time step t are not required, allowing us to skip the calculations for ϵ_θ^1 and ϵ_θ^2 at this step. In this stride, only $\epsilon_\theta^3(\cdot, t)$, $\epsilon_\theta^1(\cdot, t-1)$, $\epsilon_\theta^2(\cdot, t-1)$, and $\epsilon_\theta^3(\cdot, t-1)$ need computing, all receiving the

https://doi.org/10.52202/079017-3015

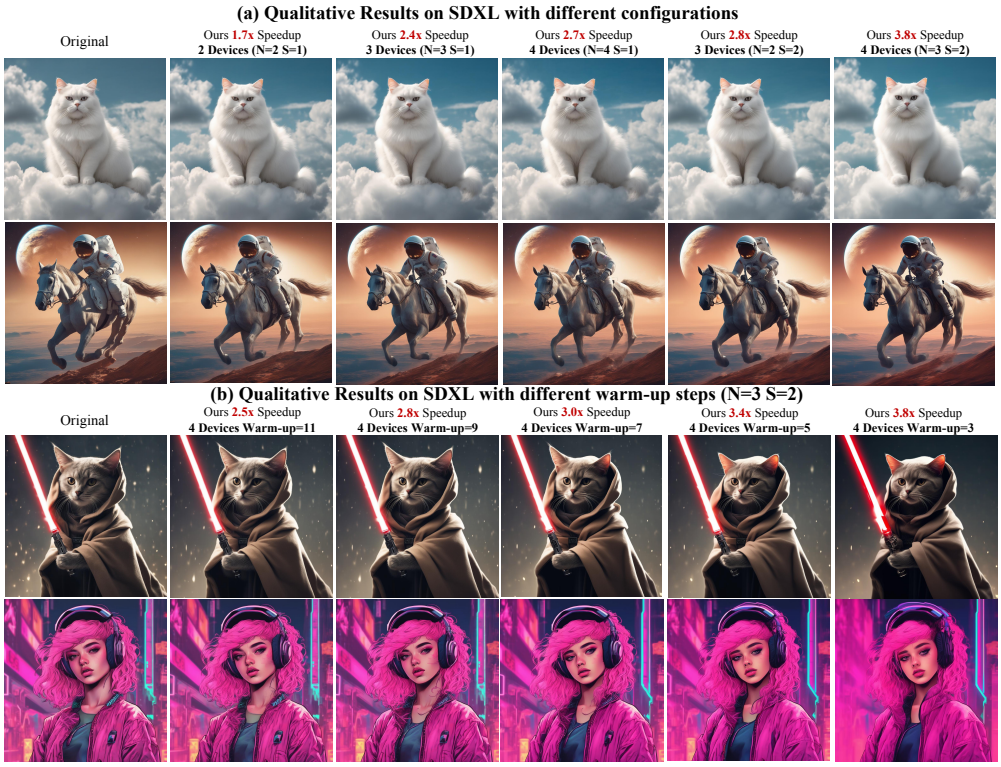

Figure 5: Qualitative Results. (a) Our method significantly accelerates the denoising process with minimal impact on generative quality. (b) Increasing warm-up steps achieves pixel-level consistency with the original output while maintaining a high speed-up ratio.

previously broadcast hidden states, enabling their parallel processing. Both $\epsilon_\theta^3(\cdot, t)$ and $\epsilon_\theta^3(\cdot, t-1)$ share the same feature from $\epsilon_\theta^2(\cdot, t+1)$, so the stride should be kept small to maintain quality. Stride denoising effectively reduces both computational load and communication demands by decreasing the parallel computing rounds needed to complete the process. Compared to the significant improvements it brings in efficiency, the quality sacrifice is minimal and can be entirely compensated for by slightly increasing the warm-up steps. We also illustrate the full schematic of it in Appendix Figure 7.

Multi-Device Communication. Parallel inference of the model necessitates efficient communication between devices, as each component ϵ_θ^n must access the cached hidden state from the preceding component ϵ_θ^{n-1}, which resides on a different device. Post each parallel computation batch, each device stores the current hidden state needed for the next parallel batch. These states, encompassing all component outputs, are then broadcast to all participating devices before the next parallel computation batch. Although each component ϵ_θ^n primarily uses the cached output of ϵ_θ^{n-1} for its input, it may require residual features [10] from other components. Therefore, it's crucial to broadcast the stored states from every component across all devices before each round of parallel computation.

4 Experiments

4.1 Implementation Details

Base models. We validated the broad applicability of *AsyncDiff* through extensive testing on several diffusion models. For text-to-image tasks, we experimented with three versions of Stable Diffusion: SD 1.5, SD 2.1 [43], and Stable Diffusion XL (SDXL) [41]. Additionally, we explored the effectiveness of *AsyncDiff* on video diffusion models using Stable Video Diffusion (SVD) [2] and AnimateDiff [9]. All models were evaluated using 50 DDIM steps. We facilitated communication across multiple GPUs using the *broadcast* operation from *torch.distributed*, powered by the *NVIDIA Collective Communication Library* (NCCL) backend.

Table 1: Quantitative evaluations of *AsyncDiff* on three text-to-image diffusion models, showcasing various configurations. 'N' indicates the number of components into which the model is divided, and 'S' represents the denoising stride. *MACs* quantifies the computational load per device for generating a single image throughout the denoising process.

Base Model	Configuration	Devices	MACs↓	latency↓	Speed up↑	CLIP Score↑	FID↓	LPIPS↓
SD 2.1 (Text-to-Image)	Original Model	1	76T	5.51s	1.0x	31.60	27.89	–
	+ Ours (N=2 S=1)	2	38T	3.03s	1.8x	31.59	27.79	0.2121
	+ Ours (N=3 S=1)	3	25T	2.41s	2.3x	31.56	28.00	0.2755
	+ Ours (N=4 S=1)	4	19T	2.10s	2.6x	31.40	28.28	0.3132
	+ Ours (N=2 S=2)	3	19T	1.82s	3.0x	31.43	28.55	0.3458
	+ Ours (N=3 S=2)	4	13T	1.35s	4.0x	31.22	29.41	0.3778
SD 1.5 (Text-to-Image)	Original Model	1	34T	2.70s	1.0x	30.63	29.96	–
	+ Ours (N=2 S=1)	2	17T	1.52s	1.8x	30.62	29.94	0.1988
	+ Ours (N=3 S=1)	3	11T	1.23s	2.2x	30.58	29.87	0.2645
	+ Ours (N=4 S=1)	4	9T	1.01	2.6x	30.52	30.10	0.3073
	+ Ours (N=2 S=2)	3	9T	0.94s	2.9x	30.46	30.98	0.3232
	+ Ours (N=3 S=2)	4	6T	0.72s	3.7x	30.17	30.89	0.3811
SDXL (Text-to-Image)	Original Model	1	299T	13.81s	1.0x	32.33	27.43	–
	+ Ours (N=2 S=1)	2	150T	8.00s	1.7x	32.21	27.79	0.2509
	+ Ours (N=3 S=1)	3	100T	5.84s	2.4x	32.05	28.03	0.2940
	+ Ours (N=4 S=1)	4	75T	5.12s	2.7x	31.90	29.12	0.3157
	+ Ours (N=2 S=2)	3	75T	4.91s	2.8x	31.70	28.99	0.3209
	+ Ours (N=3 S=2)	4	49T	3.65s	3.8x	31.40	30.27	0.3556

Table 2: Quantitative evaluations of the effect of increasing warm-up steps. More warm-up steps can achieve pixel-level consistency with the original output while slightly reducing processing speed.

Configuration	SD 2.1			SD 1.5			SDXL		
	Speedup↑	CLIP↑	LPIPS↓	Speedup↑	CLIP↑	LPIPS↓	Speedup↑	CLIP↑	LPIPS↓
Original Model	1.0x	31.60	–	1.0x	30.63	–	1.0x	32.33	–
Warm-up = 3	3.5x	31.26	0.3289	3.3x	30.16	0.3676	3.8x	31.40	0.3556
Warm-up = 5	3.1x	31.27	0.2769	3.0x	30.14	0.3304	3.4x	31.60	0.2993
Warm-up = 7	2.9x	31.32	0.2309	2.7x	30.10	0.2839	3.0x	31.77	0.2521
Warm-up = 9	2.7x	31.40	0.1940	2.5x	30.17	0.2354	2.8x	31.92	0.2095
Warm-up = 11	2.4x	31.45	0.1628	2.4x	30.22	0.1927	2.5x	32.01	0.1740

Dataset and Evaluation Metrics. We assess the zero-shot generation capability using the MS-COCO 2017 [29] validation set, which comprises 5,000 images and captions. For image generation, quality is measured by the CLIP Score (on ViT-g/14) [11] and Fréchet Inception Distance (FID) [12], with LPIPS [75] used to check consistency with original outputs. In video generation, quality is evaluated by averaging the CLIP Score across all frames of a video. We also report MACs per device and latency to gauge efficiency comprehensively. All latency measurements were conducted on NVIDIA A5000 GPUs equipped with NVLINK Bridge.

4.2 Experimental Results on Image Diffusion Models

Improvements on Base Models. Table 1 displays our acceleration outcomes for three fundamental image diffusion models under various configurations. In this context, 'N' represents the number of segments into which the denoising model is divided, and 'S' denotes the stride of denoising for each parallel computation batch. Our approach, *AsyncDiff*, not only significantly accelerates processing but also minimally impacts generative quality. The speedup ratio is almost proportional to the number of devices used, demonstrating efficient resource utilization. Visualization results in Figure 5 (a) illustrate the high generative quality achieved even with substantially reduced latency. Although achieving pixel-level consistency with the original output is challenging at high acceleration ratios, the generated image still effectively conveys the semantic information in the prompt, which is crucial for generative results.

Pixel-level Consistency by Warm-up. In Table 2, we explore the balance between pixel-level consistency and processing speed by adjusting the warm-up steps in the diffusion models. As the initial steps of these models play a crucial role in reconstructing the global structure based on text prompts [76], a modest increase in warm-up steps can significantly enhance consistency with the

https://doi.org/10.52202/079017-3015

Table 3: Quantitative comparison with other parallel acceleration methods. To ensure a fair comparison with Distrifusion, we increased the warm-up steps in our method to match the speedup ratio of Distrifusion, allowing us to fairly compare generation quality and resource costs.

Method	Speed up↑	Devices	MACs↓	Memory↓	CLIP Score↑	FID↓	LPIPS↓
Original Model	1.0x	1	76T	5240MB	31.60	27.87	–
Faster Diffusion	1.6x	1	57T	9692MB	30.84	29.95	0.3477
Distrifusion	1.6x	**2**	**38T**	6538MB	**31.59**	27.89	**0.0178**
Ours (N=2 S=1)	1.6x	**2**	44T	**5450MB**	**31.59**	**27.79**	0.0944
Distrifusion	2.3x	4	**19T**	7086MB	31.43	27.97	0.2710
Ours (N=2 S=2)	2.3x	**3**	20T	**5516MB**	**31.49**	**27.71**	**0.2117**
Distrifusion	2.7x	8	**10T**	7280MB	31.31	28.12	0.2934
Ours (N=3 S=2)	2.7x	**4**	14T	**5580MB**	**31.40**	**28.03**	**0.1940**

Figure 6: Qualitative Comparison with Distrifusion on SD2.1. At the same acceleration ratio, *AsyncDiff* outperforms in generating higher quality and more consistent images with the original.

original images. Figure 5(b) illustrates this trend with qualitative comparisons of generative results on SDXL using gradually increasing warm-up steps. Increasing the warm-up steps to 9 achieves visual indistinguishability from the original output while maintaining an impressive 2.8x acceleration ratio.

Comparison with Acceleration Baselines. We evaluated our *AsyncDiff* method on SD 2.1 against two other parallel acceleration methods: Faster Diffusion [25] and Distrifusion [24]. Faster Diffusion employs encoder propagation but compromises significantly on generative quality. As its parallelism maintains theoretical and lacks a multi-device implementation, we cannot measure its realistic latency with more than one GPU. Its ideal speed-up on 2 devices is about 1.9x. Distrifusion, on the other hand, uses patch parallelism for distributed acceleration but faces potential issues with low resource utilization and high GPU memory demands.

According to Table 3, our method achieves the same operational speed using only 4 GPUs and 3 GPUs as Distrifusion does with 8 GPUs and 4 GPUs, respectively. Additionally, our method requires almost the same amount of memory as the original setup, whereas Distrifusion significantly increases memory requirements, posing extra challenges for practical applications. In terms of generative quality, *AsyncDiff* and Distrifusion both mirror the original diffusion model's performance at a 1.6x acceleration ratio. However, at higher speedup ratios of 2.3x and 2.7x, our method demonstrates significantly superior generative quality. Qualitative comparisons in Fig 6 further show that *AsyncDiff* maintains better pixel-level consistency with the original input compared to Distrifusion.

4.3 Experimental Results on Video Diffusion Models

As presented in Table 4, we conducted experiments with different configurations on two video diffusion models: SVD [2] (25 frames), and AnimentDiff [9] (16 frames), to demonstrate the efficacy

Table 4: Quantitative evaluations of *AsyncDiff* on text-to-video and image-to-video diffusion models. We present the results with various configurations.

Base Model	Configuration	Devices	MACs↓	latency↓	Speed up↑	CLIP Score↑
AnimateDiff (Text-to-Video)	Original Model	1	786T	43.5s	1.0x	30.65
	+ Ours (N=2 S=1)	2	393T	24.5s	1.8x	30.65
	+ Ours (N=3 S=1)	3	262T	19.1s	2.3x	30.54
	+ Ours (N=2 S=2)	3	197T	14.2s	3.0x	30.32
	+ Ours (N=3 S=2)	4	131T	11.5s	3.8x	30.20
SVD (Image-to-Video)	Original Model	1	3221T	184s	1.0x	26.88
	+ Ours (N=2 S=1)	2	1611T	101s	1.8x	26.66
	+ Ours (N=3 S=1)	3	1074T	80s	2.3x	26.56
	+ Ours (N=4 S=1)	4	805T	68s	2.7x	26.19

Table 5: Effect of stride denoising on SD 2.1. Stride denoising significantly lowers overall latency and the communication cost while only slightly compromising the generative quality

Configuration	MACs↓	Latency↓	Speedup↑	Communication		CLIP Score↑
				Nums↓	Latency↓	
AsyncDiff (3 devices) w/o stride denoising	25T	2.41s	2.3x Faster	49 times	0.23s(9.5%)	**31.56**
AsyncDiff (3 devices) w/ stride denoising	**19T**	**1.82s**	**3.0x Faster**	**25 times**	**0.12s(6.6%)**	31.43
AsyncDiff (4 devices) w/o stride denoising	19T	2.10s	2.6x Faster	49 times	0.40s(19.0%)	**31.40**
AsyncDiff (4 devices) w/ stride denoising	**13T**	**1.35s**	**4.0x Faster**	**25 times**	**0.10s(7.4%)**	31.22

of our method. Video generation, often constrained by exceptionally high latency and substantial computation load, greatly benefits from our approach. For a 50-step video diffusion model, *AsyncDiff* significantly reduces latency—by tens or even hundreds of seconds—while preserving the quality of generated content. Qualitative results shown in the Appendix. D further corroborate the effectiveness of our method. *AsyncDiff* achieves an impressive acceleration ratio of over three times while still producing videos that closely match the prompt descriptions, ensuring the rationality of actions and details. These findings highlight the substantial potential of *AsyncDiff* in accelerating the inference process of video diffusion models.

4.4 Effect of Stride Denoising

We introduce stride denoising to further enhance the efficiency of the asynchronous denoising process. Stride denoising completes multiple steps simultaneously through a single parallel computation, reducing the number of parallel rounds and communication frequency across devices. For a diffusion process with T steps and warm-up step W, the number of broadcasts decreases from $T - W$ to $(T - W)//2$ with a stride of 2. This strategy also reduces the computational load on each device by skipping unnecessary calculations. Table 5 shows the effects of stride denoising in our parallel framework with 3 and 4 devices. Stride denoising significantly lowers overall latency and the proportion of communication time, especially as the number of devices used increases. While stride denoising slightly impacts generation quality, this effect is minimal and can be mitigated by a modest increase in warm-up steps, preserving efficiency and maintaining quality.

4.5 Compatibility with Various Samplers

With the recent rise of advanced sampling algorithms for diffusion models, a key concern is whether the acceleration method can adapt to various samplers. AsyncDiff is a universal method that can be combined with different samplers, such as the DDIM sampler [55] and DPM-Solver [31]. In Table 7, we present the quantitative evaluation of AsyncDiff on SD 2.1 using the DDIM sampler. Compared to using fewer DDIM steps, our method achieves significantly better generation quality at similar speeds, with the improvement becoming more pronounced as speedup increases. Table 6 presents the quantitative evaluation of AsyncDiff on SD 2.1 with the DPM-Solver sampler. At the same speedup ratio, AsyncDiff significantly enhances generation quality compared to the baseline. Qualitative results are also provided in the Appendix figures, demonstrating that our method achieves considerable acceleration while maintaining high consistency with the original output.

https://doi.org/10.52202/079017-3015

Table 6: Quantitative evaluations of AsyncDiff using DPM-Solver sampler on SD 2.1

Method	Speed up ↑	MACs ↓	CLIP Score ↑	FID ↓
DPM-Solver 25steps	1.0x	76T	31.57	28.37
DPM-Solver 15steps	1.6x	46T	31.52	28.89
Ours (N=2 S=1)	1.6x	**38T**	**31.58**	**27.71**
DPM-Solver 10steps	2.2x	30T	31.29	29.28
Ours (N=3 S=1)	2.2x	**25T**	**31.36**	**28.20**

Table 7: Quantitative evaluations of AsyncDiff using DDIM sampler on SD 2.1

Method	Speed up ↑	MACs ↓	CLIP Score ↑	FID ↓
Original	1.0x	76T	31.60	27.89
DDIM 27steps	1.8x	41T	31.53	28.43
Our AsyncDiff (N=2 S=1)	1.8x	**38T**	**31.59**	**27.79**
DDIM 21steps	2.3x	32T	31.46	29.09
Our AsyncDiff (N=3 S=1)	2.3x	**25T**	**31.56**	**28.00**
DDIM 15steps	3.0x	23T	31.26	30.12
Our AsyncDiff (N=2 S=2)	3.0x	**19T**	**31.43**	**28.55**
DDIM 11steps	4.0x	17T	30.99	32.25
Our AsyncDiff (N=3 S=2)	4.0x	**13T**	**31.22**	**29.41**

Table 8: Acceleration Ratio and Latency on Different GPUs

GPU	FP16 Compute	Original	N=2 S=1	N=3 S=1	N=2 S=2	N=3 S=2
NVIDIA RTX A5000	117 TFLOPS	1.0x(5.51s)	1.8x(3.03s)	2.3x(2.41s)	3.0x(1.82s)	4.0x(1.35s)
NVIDIA RTX 3090	71 TFLOPS	1.0x(5.61s)	1.8x(3.20s)	2.1x(2.65s)	2.9x(1.91s)	3.5x(1.60s)
NVIDIA RTX 2080Ti	54 TFLOPS	1.0x(8.20s)	1.7x(4.91s)	2.0x(4.08s)	2.8x(2.94s)	3.5x(2.35s)

5 Efficiency Analysis on Different Devices

As a hardware-friendly and versatile method, our acceleration technique delivers strong performance on a wide range of GPUs. We tested inference speeds on the professional-grade NVIDIA RTX A5000, as well as the consumer-grade NVIDIA RTX 2080 Ti and NVIDIA RTX 3090 GPUs. As shown in Table 8, our method achieved a high acceleration ratio across all three GPUs. Furthermore, our method can be applied as long as the devices have basic communication capabilities.

6 Conclusion

In this paper, we propose a new parallel paradigm, *AsyncDiff*, to accelerate diffusion models by leveraging model parallelism across multiple devices. We split the denoising model into several components, each assigned to a different device. We transform the conventional sequential denoising into an asynchronous process by exploiting the high similarity of hidden states between consecutive time steps, enabling each component to compute in parallel. Our method has been comprehensively validated on three image diffusion models (SD 2.1, SD 1.5, SDXL) and two video diffusion models (SVD, AnimateDiff). Extensive experiments demonstrate that our approach significantly accelerates inference with only a marginal impact on generative quality. This work investigates the practical application of model parallelism in diffusion models, establishing a new baseline for future research in distributed diffusion models.

Acknowledgement

This project is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (Award Number: MOE-T2EP20122-0006).

References

[1] Fan Bao, Chongxuan Li, Jun Zhu, and Bo Zhang. Analytic-dpm: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. *arXiv preprint arXiv:2201.06503*, 2022.

[2] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[3] Zigeng Chen, Chaowei Liu, Yuan Yuan, Michael Bi Mi, and Xinchao Wang. Metaisp: Efficient raw-to-srgb mappings with merely 1m parameters. In Kate Larson, editor, *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 686–694. International Joint Conferences on Artificial Intelligence Organization, 8 2024. Main Track.

[4] Jiwoo Chung, Sangeek Hyun, and Jae-Pil Heo. Style injection in diffusion: A training-free approach for adapting large-scale diffusion models for style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8795–8805, 2024.

[5] Keyan Ding, Kede Ma, Shiqi Wang, and Eero P Simoncelli. Image quality assessment: Unifying structure and texture similarity. *IEEE transactions on pattern analysis and machine intelligence*, 44(5):2567–2581, 2020.

[6] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16091–16101, 2023.

[7] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. *Advances in neural information processing systems*, 36, 2024.

[8] Lanqing Guo, Chong Wang, Wenhan Yang, Siyu Huang, Yufei Wang, Hanspeter Pfister, and Bihan Wen. Shadowdiffusion: When degradation prior meets diffusion model for shadow removal. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14049–14058, 2023.

[9] Yuwei Guo, Ceyuan Yang, Anyi Rao, Yaohui Wang, Yu Qiao, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. *arXiv preprint arXiv:2307.04725*, 2023.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021.

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

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

[14] 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*, pages 13916–13932. PMLR, 2023.

[15] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019.

[16] Zhihao Jia, Matei Zaharia, and Alex Aiken. Beyond data and model parallelism for deep neural networks. *Proceedings of Machine Learning and Systems*, 1:1–13, 2019.

[17] Yongcheng Jing, Yining Mao, Yiding Yang, Yibing Zhan, Mingli Song, Xinchao Wang, and Dacheng Tao. Learning graph neural networks for image style transfer. In *ECCV*, 2022.

[18] Animesh Karnewar, Andrea Vedaldi, David Novotny, and Niloy J Mitra. Holodiffusion: Training a 3d diffusion model using 2d images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18423–18433, 2023.

 https://doi.org/10.52202/079017-3015

[19] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6007–6017, 2023.

[20] Junjie Ke, Qifei Wang, Yilin Wang, Peyman Milanfar, and Feng Yang. Musiq: Multi-scale image quality transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5148–5157, 2021.

[21] Levon Khachatryan, Andranik Movsisyan, Vahram Tadevosyan, Roberto Henschel, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Text2video-zero: Text-to-image diffusion models are zero-shot video generators. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15954–15964, 2023.

[22] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

[23] Bo Li, Kaitao Xue, Bin Liu, and Yu-Kun Lai. Bbdm: Image-to-image translation with brownian bridge diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern Recognition*, pages 1952–1961, 2023.

[24] Muyang Li, Tianle Cai, Jiaxin Cao, Qinsheng Zhang, Han Cai, Junjie Bai, Yangqing Jia, Ming-Yu Liu, Kai Li, and Song Han. Distrifusion: Distributed parallel inference for high-resolution diffusion models. *arXiv preprint arXiv:2402.19481*, 2024.

[25] 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.

[26] Xin Li, Yulin Ren, Xin Jin, Cuiling Lan, Xingrui Wang, Wenjun Zeng, Xinchao Wang, and Zhibo Chen. Diffusion models for image restoration and enhancement–a comprehensive survey. *arXiv preprint arXiv:2308.09388*, 2023.

[27] 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.

[28] Zhuohan Li, Lianmin Zheng, Yinmin Zhong, Vincent Liu, Ying Sheng, Xin Jin, Yanping Huang, Zhifeng Chen, Hao Zhang, Joseph E Gonzalez, et al. {AlpaServe}: Statistical multiplexing with model parallelism for deep learning serving. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*, pages 663–679, 2023.

[29] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[30] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*, 2022.

[31] 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.

[32] 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.

[33] Zhaoyang Lyu, Xudong Xu, Ceyuan Yang, Dahua Lin, and Bo Dai. Accelerating diffusion models via early stop of the diffusion process. *arXiv preprint arXiv:2205.12524*, 2022.

[34] 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.

[35] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. *arXiv preprint arXiv:2312.00858*, 2023.

[36] Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal processing letters*, 20(3):209–212, 2012.

[37] Norman Müller, Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulo, Peter Kontschieder, and Matthias Nießner. Diffrf: Rendering-guided 3d radiance field diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4328–4338, 2023.

[38] Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R Devanur, Gregory R Ganger, Phillip B Gibbons, and Matei Zaharia. Pipedream: generalized pipeline parallelism for dnn training. In *Proceedings of the 27th ACM symposium on operating systems principles*, pages 1–15, 2019.

[39] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2021.

[40] Ozan Özdenizci and Robert Legenstein. Restoring vision in adverse weather conditions with patch-based denoising diffusion models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[41] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

[42] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

[43] 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.

[44] Ludan Ruan, Yiyang Ma, Huan Yang, Huiguo He, Bei Liu, Jianlong Fu, Nicholas Jing Yuan, Qin Jin, and Baining Guo. Mm-diffusion: Learning multi-modal diffusion models for joint audio and video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10219–10228, 2023.

[45] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22500–22510, 2023.

[46] 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.

[47] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE transactions on pattern analysis and machine intelligence*, 45(4):4713–4726, 2022.

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

[49] Hiroshi Sasaki, Chris G Willcocks, and Toby P Breckon. Unit-ddpm: Unpaired image translation with denoising diffusion probabilistic models. *arXiv preprint arXiv:2104.05358*, 2021.

[50] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042*, 2023.

[51] Yujun Shi, Chuhui Xue, Jiachun Pan, Wenqing Zhang, Vincent YF Tan, and Song Bai. Dragdiffusion: Harnessing diffusion models for interactive point-based image editing. *arXiv preprint arXiv:2306.14435*, 2023.

[52] Andy Shih, Suneel Belkhale, Stefano Ermon, Dorsa Sadigh, and Nima Anari. Parallel sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

[53] Junhyuk So, Jungwon Lee, and Eunhyeok Park. Frdiff: Feature reuse for exquisite zero-shot acceleration of diffusion models. *arXiv preprint arXiv:2312.03517*, 2023.

[54] 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.

 https://doi.org/10.52202/079017-3015

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

[56] Xuan Su, Jiaming Song, Chenlin Meng, and Stefano Ermon. Dual diffusion implicit bridges for image-to-image translation. *arXiv preprint arXiv:2203.08382*, 2022.

[57] Zhenxiong Tan, Xinyin Ma, Gongfan Fang, and Xinchao Wang. Litefocus: Accelerated diffusion inference for long audio synthesis. *arXiv preprint arXiv:2407.10468*, 2024.

[58] Zhenxiong Tan, Xingyi Yang, Songhua Liu, and Xinchao Wang. Video-infinity: Distributed long video generation. *arXiv preprint arXiv:2406.16260*, 2024.

[59] Jianyi Wang, Kelvin CK Chan, and Chen Change Loy. Exploring clip for assessing the look and feel of images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 2555–2563, 2023.

[60] Jianyi Wang, Zongsheng Yue, Shangchen Zhou, Kelvin CK Chan, and Chen Change Loy. Exploiting diffusion prior for real-world image super-resolution. *arXiv preprint arXiv:2305.07015*, 2023.

[61] Jiuniu Wang, Hangjie Yuan, Dayou Chen, Yingya Zhang, Xiang Wang, and Shiwei Zhang. Modelscope text-to-video technical report. *arXiv preprint arXiv:2308.06571*, 2023.

[62] Zhizhong Wang, Lei Zhao, and Wei Xing. Stylediffusion: Controllable disentangled style transfer via diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7677–7689, 2023.

[63] 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. *arXiv preprint arXiv:2312.03209*, 2023.

[64] Jay Zhangjie Wu, Yixiao Ge, Xintao Wang, Stan Weixian Lei, Yuchao Gu, Yufei Shi, Wynne Hsu, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7623–7633, 2023.

[65] Yuanzhong Xu, HyoukJoong Lee, Dehao Chen, Blake Hechtman, Yanping Huang, Rahul Joshi, Maxim Krikun, Dmitry Lepikhin, Andy Ly, Marcello Maggioni, et al. Gspmd: general and scalable parallelization for ml computation graphs. *arXiv preprint arXiv:2105.04663*, 2021.

[66] Binxin Yang, Shuyang Gu, Bo Zhang, Ting Zhang, Xuejin Chen, Xiaoyan Sun, Dong Chen, and Fang Wen. Paint by example: Exemplar-based image editing with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18381–18391, 2023.

[67] Xingyi Yang and Xinchao Wang. Hash3d: Training-free acceleration for 3d generation. *arXiv preprint arXiv:2404.06091*, 2024.

[68] Xingyi Yang, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Diffusion probabilistic model made slim. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22552–22562, 2023.

[69] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. *arXiv preprint arXiv:2311.18828*, 2023.

[70] Fanghua Yu, Jinjin Gu, Zheyuan Li, Jinfan Hu, Xiangtao Kong, Xintao Wang, Jingwen He, Yu Qiao, and Chao Dong. Scaling up to excellence: Practicing model scaling for photo-realistic image restoration in the wild. *arXiv preprint arXiv:2401.13627*, 2024.

[71] Zongsheng Yue, Jianyi Wang, and Chen Change Loy. Resshift: Efficient diffusion model for image super-resolution by residual shifting. *Advances in Neural Information Processing Systems*, 36, 2024.

[72] Chenshuang Zhang, Chaoning Zhang, Mengchun Zhang, and In So Kweon. Text-to-image diffusion model in generative ai: A survey. *arXiv preprint arXiv:2303.07909*, 2023.

[73] Dingkun Zhang, Sijia Li, Chen Chen, Qingsong Xie, and Haonan Lu. Laptop-diff: Layer pruning and normalized distillation for compressing diffusion models. *arXiv preprint arXiv:2404.11098*, 2024.

[74] Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. *arXiv preprint arXiv:2204.13902*, 2022.

[75] 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.

[76] Wentian Zhang, Haozhe Liu, Jinheng Xie, Francesco Faccio, Mike Zheng Shou, and Jürgen Schmidhuber. Cross-attention makes inference cumbersome in text-to-image diffusion models. *arXiv preprint arXiv:2404.02747*, 2024.

[77] Zhixing Zhang, Ligong Han, Arnab Ghosh, Dimitris N Metaxas, and Jian Ren. Sine: Single image editing with text-to-image diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6027–6037, 2023.

[78] Shihao Zhao, Dongdong Chen, Yen-Chun Chen, Jianmin Bao, Shaozhe Hao, Lu Yuan, and Kwan-Yee K Wong. Uni-controlnet: All-in-one control to text-to-image diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

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

[80] Yang Zhao, Yanwu Xu, Zhisheng Xiao, and Tingbo Hou. Mobilediffusion: Subsecond text-to-image generation on mobile devices. *arXiv preprint arXiv:2311.16567*, 2023.

[81] Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. Dpm-solver-v3: Improved diffusion ode solver with empirical model statistics. *Advances in Neural Information Processing Systems*, 36, 2024.

 https://doi.org/10.52202/079017-3015

❉ In this document, we provide supplementary materials that extend beyond the scope of the main manuscript, constrained by space limitations.

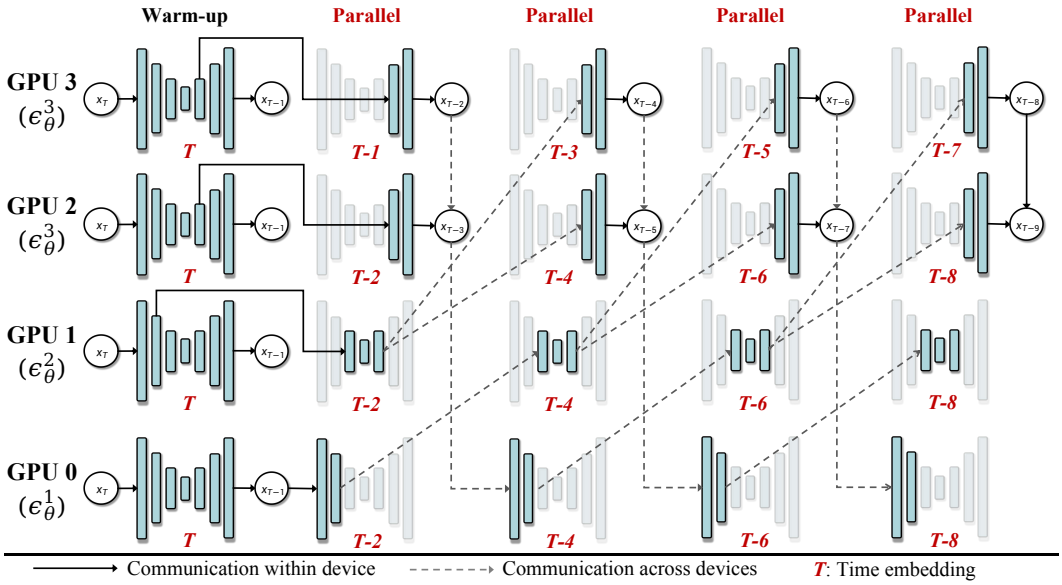

Figure 7: Schematic of the asynchronous diffusion model with stride denoising. The model ϵ_θ is divided into three components $\{\epsilon_\theta^n\}_{n=1}^3$, with a stride S of 2 for clarity. A single parallel batch results in the completion of denoising for two steps

A More Implementation Details.

Model Segmentation. In our method, we partition the cumbersome denoising model into multiple components, each assigned to a different device. After successfully parallelizing the computation of each component, the time cost for each time step now corresponds to the maximum latency among these components. To optimize parallel processing efficiency, we partition the model into segments that each carry a roughly equal computational load. This arrangement allows all modules to finish their computations nearly simultaneously, making full use of available computational resources. The segmentation strategy is sequential except for SDXL [41]. For the denoising U-net within the SDXL module, we group its first and last blocks into a single segment and apply sequential splitting to the remaining blocks. This is because SDXL has specific needs for high-frequency details, and res connections typically contain abundant high-frequency information.

Time Shifting. We introduce a technique called time shifting. Following the warm-up steps, the time embedding for each step is shifted back by one step. For instance, in a 50-step asynchronous denoising process with a warm-up of 2 steps, the original sequence of time embeddings is $\{50, 49, 48, 47, ..., 3, 2, 1\}$. With time shifting, this sequence is adjusted to $\{50, 49, 49, 48, ..., 3, 2\}$. In certain extreme cases, asynchronous denoising might leave residual noise in the output. Time shifting addresses this by adjusting the time embeddings backward, enhancing the denoising effect. It's important to note that time shifting is not a standard component of our method but is employed optionally. The quantitative results presented in this paper are achieved without the use of time shifting.

Stride Denoising. To further enhance efficiency, we introduce stride denoising, which completes multiple denoising steps simultaneously through a single parallel computation. Figure 7 illustrates the full schematic of applying stride denoising to *AsyncDiff*. In this depiction, the denoising model ϵ_θ is divided into three components $\epsilon_\theta^n{}_{n=1}^3$, and for clarity, the stride S is set to 2. Unlike the continuous broadcasting of hidden states at each time step, stride denoising broadcasts them every two steps. As depicted, at time step $\{T-1, T-3, T-5, T-7\}$, we conduct denoising alone, and at time step $\{T-2, T-4, T-6, T-8\}$, we compute and broadcast the hidden states for the next parallel computation round. Consequently, the hidden states from time step $\{T-1, T-3, T-5, T-7\}$

https://doi.org/10.52202/079017-3015

are not required, allowing us to skip the calculations for ϵ_θ^1 and ϵ_θ^2 at these steps. Stride denoising effectively reduces both computational load and communication demands by decreasing the parallel computing rounds needed to complete the process. Compared to the significant improvements it brings in efficiency, the quality sacrifice is minimal and can be entirely compensated for by slightly increasing the warm-up steps.

B More Analysis.

Time cost. In Table 9, we present the time costs associated with model running and inter-device communication when using *AsyncDiff* on SD 2.1. Generally, communication expenses constitute only a minor fraction of the total time cost, demonstrating that *AsyncDiff* is an effective distributed acceleration technique suitable for practical application. It is important to note that as the number of devices increases, the time needed for data broadcasting between devices also rises, thereby increasing the proportion of communication costs. However, employing stride denoising can substantially reduce these costs by decreasing the number of parallel rounds needed to complete the denoising process.

Table 9: Time cost comparisons on SD 2.1. 'Ratio' in this table represents the proportion of communication cost to overall latency. All measurements were conducted on NVIDIA A5000 GPUs equipped with NVLINK Bridge

Config	Time Cost			
	Overall	Running	Comm.	Ratio
N=2 S=1	3.03s	2.90s	0.13s	4.30%
N=3 S=1	2.41s	2.18s	0.23s	9.54%
N=4 S=1	2.10s	1.80s	0.30s	14.29%
N=2 S=2	1.82s	1.70s	0.12s	6.59%
N=3 S=2	1.35s	1.25s	0.10s	7.40%

Speedup Ratio. We also evaluate the acceleration ratio on SD 2.1 with varying numbers of denoising steps. As indicated in Table 10, *AsyncDiff* significantly enhances processing speed, even with a denoising procedure consisting of only 25 steps. When the number of steps extends to 100, our approach achieves a speedup of up to 4.3x, surpassing the ratio of devices employed.

Table 10: Acceleration ratio on SD 2.1 under different num of denoising steps

Config	Speedup↑		
	25steps	50steps	100steps
Origin	1.0x (2.89s)	1.0x (5.51s)	1.0x (10.96s)
N=2 S=1	1.7x (1.70s)	1.8x (3.03s)	1.8x (6.04s)
N=3 S=1	2.1x (1.35s)	2.3x (2.41s)	2.3x (4.71s)
N=4 S=1	2.4x (1.21s)	2.6x (2.10s)	2.7x (4.01s)
N=2 S=2	2.7x (1.05s)	3.0x (1.82s)	3.2x (3.39s)
N=3 S=2	3.4x (0.86s)	4.0x (1.35s)	4.3x (2.52s)

C More Quantitative Results.

To thoroughly assess the quality of images produced following acceleration, we provide quantitative analyses on three base models (SD 2.1 [43], SD 1.5 [43], SDXL [41]) using four additional metrics: the full reference metric, DISTS [5], and no-reference metrics including MUSIQ [20], CLIP-IQA [59], and NIQE [36]. The experimental results in Table 11 demonstrate that our method significantly reduces inference latency while maintaining a high level of quality in diffusion model-generated images. On SD 1.5, our approach not only accelerates the inference process but also brings the image quality closer to the natural distribution.

https://doi.org/10.52202/079017-3015

Table 11: Quantitative evaluations of *AsyncDiff* on three text-to-image diffusion models using more metrics including DISTS [5], MUSIQ [20], CLIP-IQA [59], and NIQE [36].

Base Model	Configuration	Devices	DISTS↓	MUSIQ↑	CLIP-IQA↑	NIQE↓
SD 2.1	Original Model	1	–	69.95	0.6653	3.9675
	+ **Ours** (N=2 S=1)	2	0.1041	69.55	0.6539	3.8850
	+ **Ours** (N=3 S=1)	3	0.1280	69.04	0.6441	3.9438
	+ **Ours** (N=4 S=1)	4	0.1419	68.58	0.6365	3.9724
	+ **Ours** (N=2 S=2)	3	0.1556	68.03	0.6158	3.5761
	+ **Ours** (N=3 S=2)	4	0.1689	67.13	0.5986	3.6761
SD 1.5	Original Model	1	–	71.98	0.6534	3.5517
	+ **Ours** (N=2 S=1)	2	0.1169	72.21	0.6569	3.7448
	+ **Ours** (N=3 S=1)	3	0.1434	71.73	0.6481	3.8023
	+ **Ours** (N=4 S=1)	4	0.1599	71.51	0.6442	3.8620
	+ **Ours** (N=2 S=2)	3	0.1668	71.14	0.6323	3.9613
	+ **Ours** (N=3 S=2)	4	0.1905	69.42	0.6070	4.1047
SDXL	Original Model	1	–	71.58	0.6633	4.0743
	+ **Ours** (N=2 S=1)	2	0.1038	70.56	0.6498	4.1139
	+ **Ours** (N=3 S=1)	3	0.1211	69.88	0.6389	4.1585
	+ **Ours** (N=4 S=1)	4	0.1391	67.70	0.6056	4.0927
	+ **Ours** (N=2 S=2)	3	0.1329	69.56	0.6222	4.1685
	+ **Ours** (N=3 S=2)	4	0.1527	68.16	0.5955	4.2745

D More Qualitative Results

Qualitative Results on Image Diffusion Models. As depicted in Figure 8, we present further qualitative results for SD 2.1 and SDXL under various configurations. The speedup achieved is nearly proportional to the number of devices utilized, indicating efficient resource usage by our method. Moreover, the images generated by our approach closely match the text descriptions and are of high quality.

Qualitative Results on Video Diffusion Models. We present qualitative evaluations of *AsyncDiff* applied to the video diffusion models. Figures 9, 10, and 11 illustrate the generated results using our method on the text-to-video model AnimateDiff [9]. Figure 12 displays results from applying our method to the image-to-video model SVD [2]. For a 50-step video diffusion model, *AsyncDiff* markedly decreases latency—saving tens or even hundreds of seconds—while maintaining the integrity and quality of the generated videos.

E Limitations

As a distributed acceleration framework, *AsyncDiff* necessitates frequent communication between devices throughout the denoising process. Consequently, if the devices lack the capability to communicate effectively or have subpar communication infrastructure, our method may not perform optimally. Additionally, *AsyncDiff* operates as a plug-and-play acceleration solution that depends on pre-trained diffusion models. Therefore, if the baseline quality of the original diffusion models is unsatisfactory, achieving high-quality results with our method could be challenging.

F Societal impacts

In this paper, we introduce a universal distributed acceleration approach for diffusion models. This method substantially speeds up the inference phase of diverse diffusion models by fully leveraging computational resources. It holds significant potential for practical applications, particularly in computationally intensive generation tasks like video and speech generation.

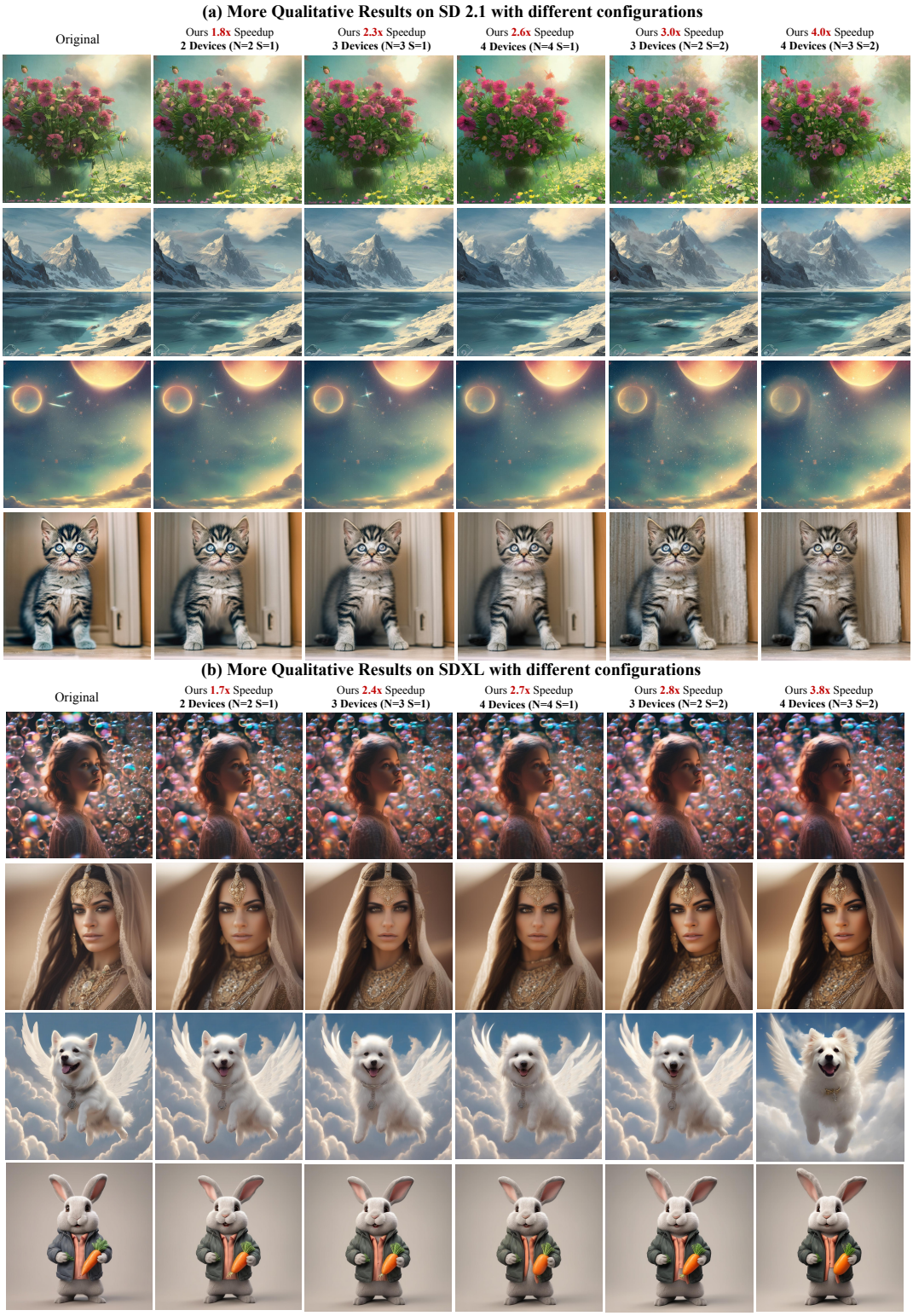

Figure 8: Qualitative results on SD 2.1 and SDXL with different configurations.Our method maintains excellent generation quality even when achieving speedups of up to four times.

https://doi.org/10.52202/079017-3015

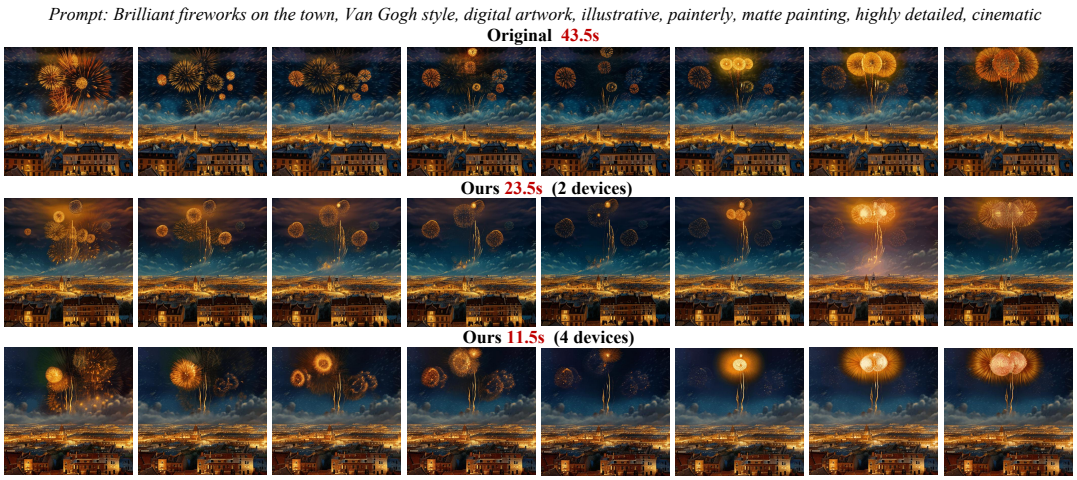

Figure 9: Qualitative results on AnimateDiff (1)

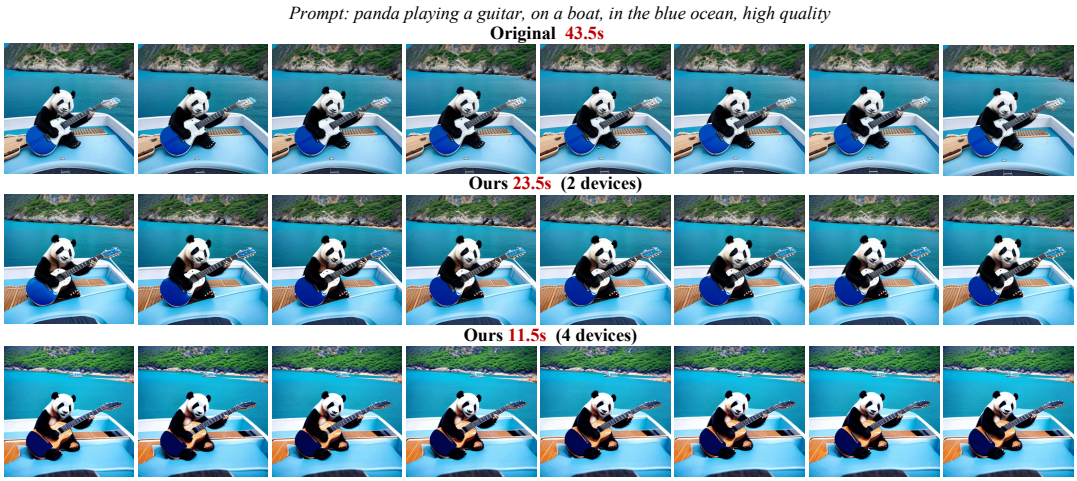

Figure 10: Qualitative results on AnimateDiff (2)

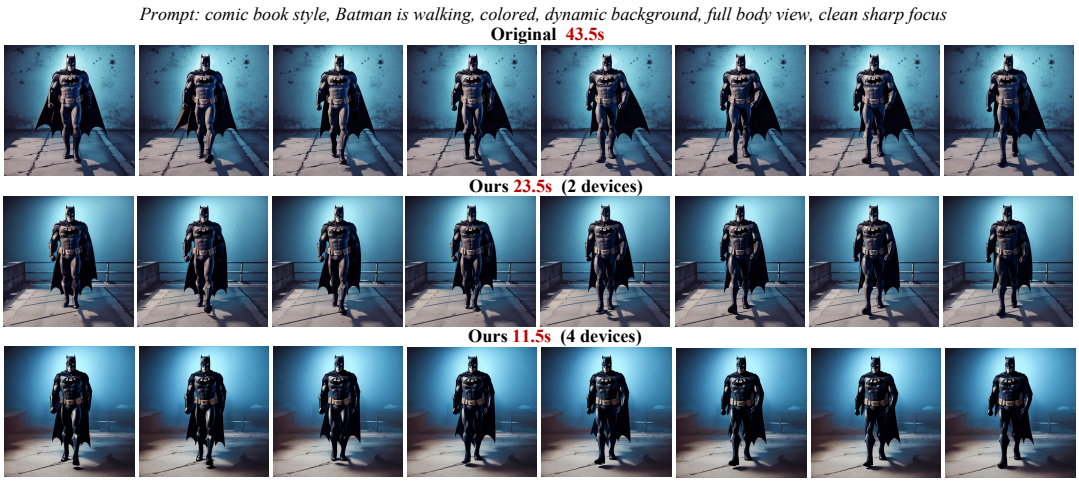

Figure 11: Qualitative results on AnimateDiff (3)

https://doi.org/10.52202/079017-3015

Figure 12: Qualitative results on Stable Video Diffusion

https://doi.org/10.52202/079017-3015

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in our abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: We discuss the limitation of our work in the Appendix.E.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not include theoretical results

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide a detailed description of our method along with extensive experimental results.

   Guidelines:

https://doi.org/10.52202/079017-3015

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We offer the full code along with relevant instructions.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide all the details about the experiment in our paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provide the details about initialization and dataset split.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the details about the computation resources we used in the experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

https://doi.org/10.52202/079017-3015

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We strictly adhere to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the societal impacts in the Appendix.F.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The creators or original owners of assets (e.g., code, data, models) used in the paper are properly credited, and the license and terms of use are explicitly mentioned and properly adhered to.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: New assets introduced in the paper are well documented, and the documentation is provided alongside the assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

https://doi.org/10.52202/079017-3015

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.