This ICCV 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.

MosaicDiff: Training-free Structural Pruning for Diffusion Model Acceleration Reflecting Pretraining Dynamics

Bowei Guo Shengkun Tang Cong Zeng Zhiqiang Shen
Mohamed bin Zayed University of Artificial Intelligence
{Bowei.Guo, Shengkun.Tang, Cong.Zeng, Zhiqiang.Shen}@mbzuai.ac.ae

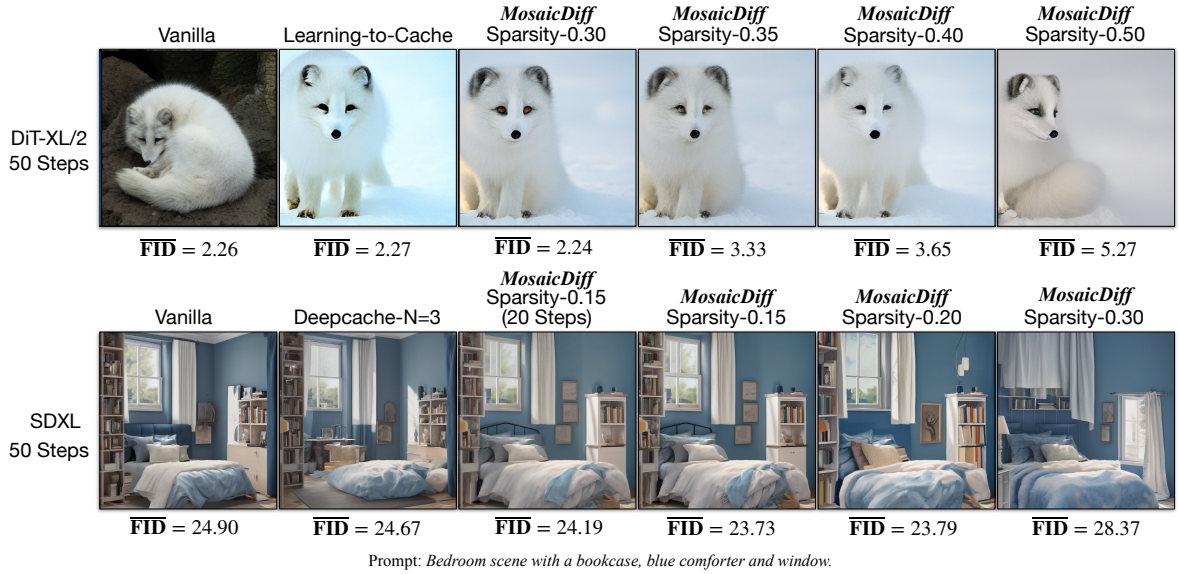


Figure 1. ***MosaicDiff*** is a post-training / training-free structural pruning technique for both transformer-based and U-Net-based diffusion models. It can achieve 0.5 pruning sparsity on linear scheduled 675M DiT-XL/2 and 0.3 pruning sparsity on scaled-linear scheduled 2.6B SDXL-base-1.0 with minimal performance degradation.

Abstract

Diffusion models are renowned for their generative capabilities, yet their pretraining processes exhibit distinct phases of learning speed that have been entirely overlooked in prior post-training acceleration efforts in the community. In this study, we introduce a novel framework called ***MosaicDiff*** *that aligns diffusion pretraining dynamics with post-training sampling acceleration via trajectory-aware structural pruning. Our approach leverages the observation that the middle, fast-learning stage of diffusion pretraining requires more conservative pruning to preserve critical model features, while the early and later, slow-learning stages benefit from a more aggressive pruning strategy. This adaptive pruning mechanism is the first to explicitly mirror the inherent learning speed variations of diffusion pretraining, thereby harmonizing the model's inner training dynamics with its accelerated sampling process. Extensive experiments on DiT and SDXL demonstrate that our method achieves significant speed-ups in sampling without compromising output quality, outperforming previous state-of-the-art methods by large margins, also providing a new viewpoint for more efficient and robust training-free diffusion acceleration. Our implementation is available at* https://github.com/bwguo105/MosaicDiff.git.

1. Introduction

Diffusion models [10, 11, 29, 33] have emerged as a powerful framework for generative tasks in unconditional image generation [11], text-guided image generation [33] and even 3D generation [31], yet their extensive computational demands, especially during pretraining, pose significant challenges for real-world applications. The high cost of diffusion pretraining has driven the community toward training-free acceleration methods, which aim to reduce sampling times without incurring additional training overhead. However, these methods often overlook the nuanced learning dynamics inherent in the pretraining process.

A closer examination of diffusion pretraining reveals a unique characteristic: the learning speed is not uniform but varies significantly across different stages. In the middle phase, the model rapidly captures coarse-grained features, while the early and later stages involve a more gradual initial movement and final refinement of the details. This trajectory, marked by a *slow-fast-slow* multi-stage learning phase, provides crucial insights into how the diffusion model evolves over time, a factor that has been largely neglected in existing acceleration approaches.

The mainstream focus on training-free acceleration or pruning [4, 19–21, 24, 38, 41], driven by the desire to bypass the costly pretraining process, has unintentionally led to ignorance of the intrinsic learning properties of diffusion models. By not considering the differential learning speeds during pretraining, these methods miss an opportunity to optimize the post-training sampling process in a way that aligns with the model's internal learning dynamics. This oversight can limit the efficiency and effectiveness of the acceleration techniques employed.

To this end, we introduce a novel trajectory-aware pruning strategy called ***MosaicDiff*** that aligns post-training / training-free acceleration with the underlying learning dynamics of the pretraining phase. By recognizing that the fast learning stage requires a more cautious approach while the slow learning stage can tolerate more aggressive pruning, our method strategically adjusts the pruning intensity along the model's learning trajectory. This alignment ensures that the post-training acceleration process is well-calibrated to the model's internal state, preserving critical features learned during the slower refinement stages. Specifically, we propose a new trajectory-aware second-order structural pruning method using SNR-aware (signal-to-noise ratio) calibration data, to identify *different sparse sub-networks from the same dense parent diffusion network* tailored to distinct learning stages.

Even in our scenario where diffusion pretraining is not performed, our study demonstrates that it is possible to infer the learning speed characteristics from the sampling strategy itself. Through a combination of empirical studies and theoretical analysis, we connect the effective learning dynamics of the pretraining process to these insights, using them to guide our trajectory-aware pruning strategy. Our proposed methodology bridges the gap between pretraining behavior and post-training acceleration, allowing us to optimize the sampling process without the need for retraining.

Our experimental results show that this alignment strategy is particularly effective, especially at higher and more challenging pruning ratios, where it surpasses all previous state-of-the-art methods including Learning-to-Cache [23], DiP-GO [46], DeepCache [24] and Diff-Pruning [5]. This study is the first to systematically integrate the learning speed variations from the pretraining phase into the design of a post-training acceleration method, thereby providing a new solution for training-free diffusion model acceleration. We summarize our contributions as follows:

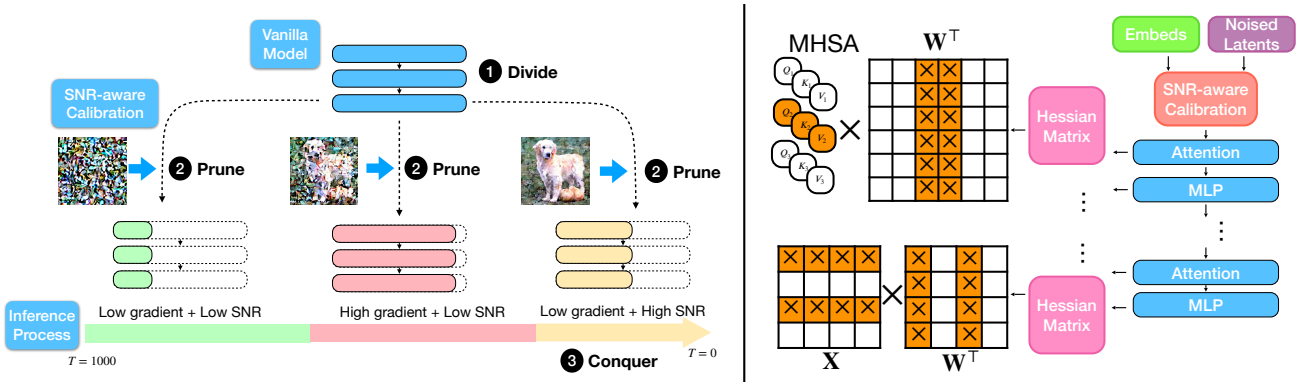
- We introduce a pruning strategy that aligns training-free / post-training acceleration with the varying learning speeds of diffusion pretraining without actual pretraining.
- We propose a novel approach to identify stage-specific sparse networks, applying aggressive pruning during slow-learning phases and conservative pruning during fast-learning phases.
- Our extensive experiments demonstrate state-of-the-art generation and acceleration performance, particularly at high pruning ratios, outperforming existing training-free methods by significant margins.

2. Related work

Efficient Diffusion Models. Diffusion models exhibit exceptional generative performance but face high computational costs due to their iterative denoising process [39]. Inference acceleration methods mainly aim to either reduce the number of sampling steps or optimize computations per step. Step-reduction approaches such as DDIM [38], DPM-Solver [19], and Consistency Models [40] reformulate the diffusion process, while knowledge distillation approaches [25, 35] transfer multi-step denoising capabilities from larger teacher models to compact students. Meanwhile, per-step optimization strategies employ architectural compression methods, including structural pruning [4, 12, 45], model distillation [13, 26], and early stopping [22, 41]. Additional techniques, such as quantization [8, 15, 16, 37] and feature caching [23, 24, 46], further reduce computation and memory usage. Finally, trajectory stitching [28] combines models of varying complexities during inference stages without degrading generation.

Noise Schedule Optimization. Noise schedules are critical hyperparameters that directly influence learning speeds throughout the diffusion process. Prior studies [17, 27, 30] have demonstrated that these schedules significantly affect the training outcomes of diffusion models. For example, Choi et al. [1] assign weights to emphasize important training steps based on noise schedules, enhancing training performance. However, existing works often overlook that noise schedules similarly impact the inference stage, essentially an unguided extension of the training process. Leveraging insights from pretrained model dynamics, we propose utilizing the step-wise importance indicated by the noise schedule to accelerate and enhance the sampling process.

Structural Pruning. Structural pruning has been widely applied to large neural networks, such as large language models (LLMs) [6, 14, 42], to efficiently accelerate inference by removing entire substructures (e.g., neurons, channels, or layers). However, pruning methods for diffusion models are still in their early stages due to their itera-



(a) Main pipeline of Divide, Prune and Conquer.

(b) Second-order structural pruning for sub diffusion networks.

Figure 2. **Overview of *MosaicDiff*.** (a) **Main framework:** We first divide the inference process into three distinct stages according to a quantitative analysis of pretraining dynamics. For each stage, we utilize SNR-aware calibration data to perform second-order structural pruning, obtaining subnetworks with varying degrees of sparsity. Finally, we integrate these subnetworks to enable efficient inference across all timesteps. (b) **Second-order structural pruning:** To practically implement pruning on diffusion models, we feed SNR-aware calibration data into the pretrained model, computing Hessian matrices for each Attention and MLP layer. We then derive saliency scores from these Hessians to prune less important weight columns, corresponding to heads in multi-head self-attention (MHSA) layers and neurons in intermediate MLP layers.

tive nature and heightened sensitivity to parameter reduction [29]. Recent works, such as Diff-Pruning [4], utilize improved Taylor pruning to identify redundant structures, while EcoDiff [45] employs a training-based differentiable mask for model sparsification. However, these approaches still yield limited performance and are constrained to low sparsity levels, primarily due to their use of a uniformly pruned model across all diffusion timesteps. This approach overlooks the inherent step-wise importance dynamics in the diffusion sampling process, potentially missing further efficiency gains.

3. *MosaicDiff*

Framework Overview. Figure 2a illustrates the main framework of ***MosaicDiff***, which contains three main phases *Divide, Prune and Conquer*. Starting from a pretrained vanilla large diffusion model, we need to determine both the pruning stages and the corresponding sparsity levels. In the *Divide* phase, the inference trajectory is split into segments according to the strategy introduced in Section 3.2. Within each segment, a *Prune* step applies second-order structural pruning guided by SNR-aware calibration data (Sections 3.3 and 3.4). Finally, the *Conquer* step merges these pruned sub-networks for the final sampling, ensuring that the accelerated inference remains consistent with the original model's trajectory.

3.1. Preliminary

Diffusion models comprise a forward training process and a reverse sampling process. The forward process learns the features in images by gradually adding Gaussian noise to them on the basis of a Markov chain. Generally, diffusion models add noise according to a pre-defined and fixed hyper-parameter schedule $\beta_1, ..., \beta_T$. Thus, we can formulate the forward process as:

$$q(x_{1:T} \mid x_0) := \prod_{t=1}^{T} q(x_t \mid x_{t-1}), \tag{1}$$

$$q(x_t \mid x_{t-1}) := \mathcal{N}\Big(x_t;\ \sqrt{1-\beta_t}x_{t-1},\ \beta_t \mathbf{I}\Big). \tag{2}$$

If further define $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$, we can sample x_t at an arbitrary timestep t using a closed form:

$$q(x_t \mid x_0) = \mathcal{N}\Big(x_t;\ \sqrt{\bar{\alpha}_t}x_0,\ (1-\bar{\alpha}_t)\mathbf{I}\Big), \tag{3}$$

which equals to:

$$x_t(x_0,\ \epsilon) = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{4}$$

Thus, we can calculate signal-to-noise ratio (SNR) of the image at the timestep t by:

$$\mathrm{SNR}(t) = \frac{\bar{\alpha}_t}{1-\bar{\alpha}_t}. \tag{5}$$

The reverse process of diffusion models generates images by gradually denoising pure Gaussian noises from a distribution defined as $p(x_T) = \mathcal{N}(x_T;\ \mathbf{0}, \mathbf{I})$ according to a similar Markov chain:

$$p_\theta(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t), \tag{6}$$

$$p_\theta(x_{t-1} \mid x_t) := \mathcal{N}\Big(x_{t-1};\ \mu_\theta(x_t, t),\ \Sigma_\theta(x_t, t)\Big). \tag{7}$$

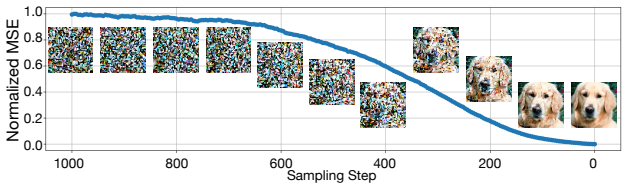


Figure 3. Change in image MSE over sampling steps: In the early stage $T \in (600, 1000)$, the MSE decreases slowly with images remaining largely noisy, in the middle stage $T \in (200, 600)$, denoising accelerates and images converge rapidly, and in the final stage $T \in (0, 200)$, MSE reduction slows, indicating only subtle perceptual refinements.

Set $\Sigma_\theta(x_t, t) = \sigma^2 \mathbf{I}$, where σ^2 can be directly calculated through β_t, and reparameterize $\mu_\theta(x_t, t)$ as:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\Big). \tag{8}$$

Then what neural networks really predict at timestep t is the noise $\epsilon_\theta(x_t, t)$. The loss function is:

$$\mathcal{L}(\theta) = \mathbb{E}_{x_0,\epsilon}\Big[\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|^2\Big]. \tag{9}$$

3.2. Stage Division for Reverse Process

Empirical Observation. A key observation driving ***MosaicDiff*** is that diffusion pretraining exhibits distinct phases of learning speed, which are reflected in the reverse denoising steps. Prior research has empirically shown that the importance of different timesteps in diffusion models varies significantly. Some studies [1, 27, 28] emphasize this by analyzing the evolution of the SNR throughout the process, while others reach a similar conclusion by comparing feature similarities along the generation trajectory.

To harness these differences, we partition the reverse process into multiple stages based on timestep intervals, assigning each stage its own tailored sparsity budget. This strategic segmentation allows us to align the pruning intensity with the learning dynamics observed during pretraining. Our approach specifically focuses on monitoring the changes in Mean Squared Error (MSE) between the intermediate latents representations $\hat{x}_t$ and the final output latents $\hat{x}_0$, i.e.:

$$\text{MSE}(t) = \frac{1}{d}\|\hat{x}_t - \hat{x}_0\|_2^2, \tag{10}$$

where d is the dimension of latents $\hat{x}_t$ and $\hat{x}_0$. Using outputs from the DiT-XL/2 model as an example, we plot the MSE trend across sampling steps t. As shown in Figure 3, the MSE decreases gradually during the early stages of generation, with the associated gradients remaining nearly constant. In contrast, during the intermediate stages, the MSE rapidly diminishes while the gradients increase significantly. Finally, in the later steps of the reverse process, the rate of MSE reduction slows again, and the gradients subside to lower levels. We claim that the expected MSE and gradients for each step can be derived via a closed-form solution, which provides theoretical support as the following theorem for our stage-specific pruning strategy.

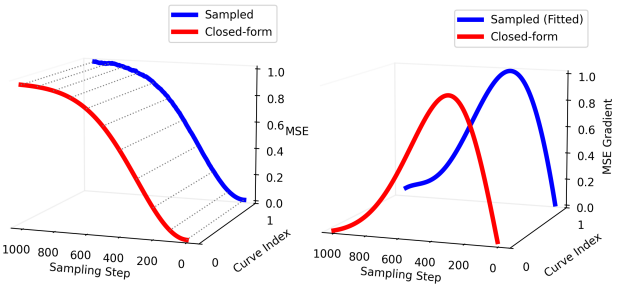


Figure 4. MSE and gradient curves comparison under Linear Schedule. *Left*: MSE calculated from our closed-form approximation closely matches the sampled results. *Right*: Gradients derived from our closed-form expression align with empirically sampled gradients.

Theorem 1. *With the $\bar{\alpha}_t$ from the noise scheduler, the expectation of MSE and gradient can be formulated as :*

$$\mathbb{E}\Big[\mathit{MSE}(t)\Big] = \frac{1}{d}\Big[(1-\sqrt{\bar{\alpha}_t})^2\|\hat{x}_0\|_2^2 + (1-\bar{\alpha}_t)\|\mathbf{I}\|_2^2\Big], \tag{11}$$

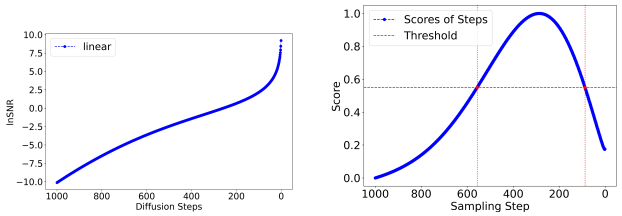
$$\mathbb{E}\Big[\mathit{Grad}(t)\Big] = \frac{1}{d}\Big[(\delta_t + 2(\sqrt{\bar{\alpha}_{t-1}} - \sqrt{\bar{\alpha}_t}))\|\hat{x}_0\|_2^2 - \delta_t\|\mathbf{I}\|_2^2\Big]. \tag{12}$$

Define $\delta_t := \bar{\alpha}_t - \bar{\alpha}_{t-1}$. Detailed derivations and proofs are provided in Appendix. In Figure 4, we compare the closed-form MSE and gradient curves with their empirical counterparts. The close alignment between the theoretical and observed curves validates our equations and supports our approach.

Quantitative Importance Scores. We define an importance score, $score(t)$, for each sampling step to quantify its contribution during inference. Our formulation is based on the above observation that the gradient of MSE is indicative of convergence speed - the larger the gradient, the more rapidly the step approaches the final output, ensuring a lower sparsity level s in our pruning algorithm. Moreover, as shown in Figure 5a, the SNR significantly increases during the final sampling steps. Although the gradients in these stages are comparably small to those in the early steps, even slight changes in the MSE can have a disproportionately large impact on the final image quality. To capture this effect, we incorporate the SNR (from Equation (5)) into our importance score as follows:

$$score(t) = \mathbb{E}\Big[\text{Grad}(t)\Big] + \lambda \ln \text{SNR}(t) \tag{13}$$

where λ is a hyperparameter that controls the influence of the SNR, making the final $score(t)$ entirely dependent on



(a) SNR trend of Linear schedule. (b) Final scores of sampling steps.

Figure 5. Influence of SNR on Final Scores. (a) Change in SNR across sampling steps, showing a sharp increase during the final steps. (b) Final scores computed combining SNR. A threshold of $M = 0.55$ clearly divides the curve into three stages.

the noise schedule $\bar{\alpha}_t$. Figure 5b shows the final score curves for DiT-XL/2 under a linear schedule. The results of SDXL under a scaled linear schedule are in Appendix.

Stage and Sparsity Decider. After computing the final $score$ of a diffusion model based on $\bar{\alpha}_t$, we define:

$$threshold := M \cdot \max_t(score(t)), \tag{14}$$

where $M \in (0, 1)$ is a hyper-parameter. As illustrated in Figure 5b, this threshold divides the score curve into three segments, each corresponding to a pruning stage with a fixed sparsity s_i, where $i \in \{0, 1, 2\}$. We then compute the average score $\overline{score}_i$ for each segment and set:

$$s_i \propto 1 - \overline{score}_i. \tag{15}$$

Intuitively, segments with higher average scores receive lower sparsity, preserving more parameters in timesteps deemed critical. Based on Equation (15), we assign each specific sparsity level to each segment by leveraging both our experimental results and the observed learning dynamics presented above. Details are presented in the Appendix.

3.3. SNR-Aware Calibration Dataset

After partitioning the reverse process into stages and determining the corresponding sparsities, we build an SNR-aware calibration dataset for pruning. Our objective is to have each model specialize in a specific stage, excelling at inferring on latents that fall within a targeted SNR range. To achieve this, we first resize each image from a standard calibration set (e.g., ImageNet-1K) and encode it into a latent representation. Next, we randomly select a timestep t within the stage of interest and add noise to the latent according to Equation (4) to ensure that the resulting latent meets the desired SNR value, i.e., SNR(t). The noised latent, along with its associated timestep t and the corresponding class label or caption, is then packaged to be considered in the Hessian computation for pruning.

For models using classifier-free guidance (CFG) during inference, we mimic unconditional generation by providing a null-label calibration example alongside each noised latent. Specifically, when CFG is enabled, we duplicate the latent and t (concatenating them) and pair this with a null label to ensure the calibration process accurately reflects the inference setup. This comprehensive calibration approach, resizing, encoding, random timestep selection, noise addition, and appropriate pairing with labels (or null labels under CFG) ensures that the model receives precise Hessian information for each linear layer. This, in turn, guides the following stage-specific second-order structural pruning for optimal acceleration.

3.4. Second-Order Structural Pruning

In contrast to existing compression methods, we employ the Hessian matrix to assess the importance of substructures more precisely for different sampling stages, resulting in a training-free, fine-tuning-free pruning algorithm that can be seamlessly applied to our accelerating diffusion framework.

Problem Formulation. Given a linear layer with calibration input $\mathbf{X} \in \mathbb{R}^{b\times n}$ and weight $\mathbf{W} \in \mathbb{R}^{m\times n}$, our method aims to find the compressed weight $\widehat{\mathbf{W}}$ at the pruning sparsity s, which causes the least error compare with the original output, evaluated by the square error:

$$\text{argmin}_{\widehat{\mathbf{W}}} \|\mathbf{X}\widehat{\mathbf{W}}^\top - \mathbf{X}\mathbf{W}^\top\|_2^2 \tag{16}$$

This objective can be decomposed into independent minimization jobs across the m rows of $\mathbf{W}$. However, since we focus on structural pruning, the indices we remove at all rows should be the same, i.e. we should prune entire columns. Define a column mask $\mathbf{M} \in \mathbb{R}^n$, which only contains value 0 and 1. $\mathbf{M}_i = 1$ means column i need to be pruned and $\mathbf{M}_i = 0$ means column i to be preserved, where $i \in [0, n)$. We need to prune weight $\mathbf{W}$ to sparsity s, so $\|\mathbf{M}\|_1 = \lfloor s \cdot n \rfloor$. Therefore, our goal becomes to find the optimal mask $\mathbf{M}$, so that pruning out $\mathbf{W}_{:,\mathbf{M}}$ will result in the least error.

Saliency Score and Weight Update. After second-order derivative, the Hessian matrix for the ℓ_2-minimization problem in Equation (16) can be calculated as $\mathbf{H} = \mathbf{X}^\top\mathbf{X} \in \mathbb{R}^{n\times n}$. We further solve Equation (16) by extending formulas from Optimal Brain Surgeon (OBS) [7]. Then, we can obtain the saliency score of the optimal mask:

$$\text{argmin}_{\mathbf{M}} \sum_{i=0}^{m-1} \mathbf{W}_{i,\mathbf{M}} \cdot \left(\mathbf{H}^{-1}_{\mathbf{M},\mathbf{M}}\right)^{-1} \cdot \mathbf{W}^\top_{i,\mathbf{M}} \tag{17}$$

After eliminating the columns according to the mask, we can also compute an update δ for the remaining weights to compensate for pruning to further reduce errors:

$$\delta = -\mathbf{W}_{:,\mathbf{M}} \cdot \left(\mathbf{H}^{-1}_{\mathbf{M},\mathbf{M}}\right)^{-1} \cdot \mathbf{H}^{-1}_{\mathbf{M},:} \tag{18}$$

Lastly, we multiply the weight matrix with the mask $\mathbf{W} \odot (\mathbf{1} - \mathbf{M})$ to ensure the pruned columns are exactly zeros.

Pruned Structures. In our work with diffusion models, both the Transformer and U-Net architectures are examined under two pruned configurations: (1) head pruning within the multi-head self-attention, and (2) the reduction of the intermediate dimension in MLP modules.

4. Experiments

4.1. Setup

Models. We validate our methods on two mainstream latent diffusion models, leveraging both transformer [43] and U-Net [34] architectures: 1) DiT [29] is a widely adopted transformer-based diffusion model available in multiple parameter configurations. We focus on pruning the 256 $\times$ 256 linear scheduled DiT-XL/2 variant, which consists of 675 million parameters, to demonstrate the effectiveness of our approach. 2) SDXL-base-1.0 [30] is a state-of-the-art U-Net-based text-to-image diffusion model with 2.6 billion parameters. We employ this model to present the scalability of our methods on large-scale architectures and prove the compatibility with scaled-linear scheduled models.

Datasets and Metrics. For DiT, following previous work, we conduct experiments on ImageNet-1K [2] at the resolution of 256 $\times$ 256. For SDXL, we use the MS COCO 2017 [18] for quantitative evaluation. We follow the evaluation of DiT, we generate 50,000 images and evaluate their quality using the Fréchet Inception Distance (FID) [9], computed with ADM's TensorFlow evaluation suite [3], consistent with prior work. Additionally, we report Inception Score (IS) and Precision-Recall as complementary metrics. For SDXL, we include FID-5K calculated by torchmetrics to assess image quality, CLIP-Score [32] calculated by ViT-B-16 to measure text-image alignment, and SSIM [44] to quantify output differences compared to the original model outputs. The MACs is evaluated using PyTorch-OpCounter, and the latency is tested when generating 8 images on a single RTX 4090, which we conduct five times and compute the average.

Baselines. We compare our method against state-of-the-art sampling acceleration techniques. For DiT, we first compare our method with Diff-Pruning [5] by utilizing their official implementation. We also compare against DiP-GO [46], a training-based compression approach, to highlight our effectiveness at larger sampling steps when using the conventional DDPM [11]. We report the results of DiP-GO from their original paper. Moreover, to show the superior performance of our method on acceleration in a training-free manner, we compare our approach with Learning-to-Cache (L2C) [23], a caching-based method that requires training. We also report the results of L2C from their paper. For SDXL, we first compare with pruning approaches Diff-Pruning and Eco-Pruning [45]. Because Eco-Pruning targets memory reduction more than compute speed, we briefly include its reported results from the original paper. Moreover, in addition to pruning methods, we also compare with the training-free caching approach DeepCache [24].

Implementation Details. On DiT, we randomly choose 1024 images from ImageNet-1K as calibration for each pruning stage. On SDXL, we randomly select 1024 images and paired captions from MS COCO 2017 training dataset as calibration. For both models, we select threshold $M = 0.55$ and influence of SNR $\lambda = 0.01$. All experiments are conducted on NVIDIA RTX 4090 GPU.

4.2. Main results

Result Comparison on DiT. We present a comprehensive comparison between our proposed method and baseline approaches, as summarized in Table 1 and Table 2. First, we compare our method with Diff-Pruning under identical sparsity constraints. Without additional fine-tuning, our method significantly outperforms Diff-Pruning, achieving an FID of 2.24 compared to 180.76 obtained by Diff-Pruning using 50 sampling steps. As the number of sampling steps decreases, the performance gap widens further. Specifically, with only 20 and 10 sampling steps, Diff-Pruning attains FID scores of 223.8 and 270.26, respectively, while our ***MosaicDiff*** achieves substantially lower FID scores of 3.20 and 12.28.

Moreover, our method surpasses both the vanilla DiT and DiP-GO at 100 and 70 sampling steps, achieving lower MACs and FID scores, thereby producing higher-quality images with reduced computational cost. Furthermore, ***MosaicDiff*** consistently outperforms L2C across all evaluated step counts, achieving superior results in IS, FID, and Precision. Notably, under 20-step sampling using the fast DDIM sampler, our method achieves an FID of 3.20, outperforming both L2C and the uncompressed baseline model. While the original DiT achieves slightly higher Recall, the minor reductions observed in both L2C and ***MosaicDiff*** indicate a negligible impact on generative diversity. These results highlight the superior performance of ***MosaicDiff*** and demonstrate the effectiveness of the proposed approach.

Result Comparison on SDXL. To show the scalability and compatibility on other architectures, we evaluate our method on a larger U-Net based model, SDXL and provide the result comparison in Table 3 and 4. Without further fine-tuning, Diff-Pruning continues to underperform across all metrics, achieving FID scores of 108.96 and 404.87 at sparsity of 10% and 20%, respectively. In contrast, our training-free method consistently outperforms EcoDiff in terms of FID, CLIP, and SSIM. These results indicate that our method not only produces higher-quality images but also maintains better semantic alignment with textual prompts and better preserves the visual similarity to the original SDXL baseline. We further compare our approach with DeepCache in Table 4. To ensure a fair compar-

Table 1. Result comparison of ***MosaicDiff*** and other baselines on DiT-XL/2 using DDIM sampler. ***MosaicDiff*** consistently outperforms all baselines across different sampling step configurations in terms of FID, IS, and Precision, while maintaining competitive Recall. Moreover, ***MosaicDiff*** achieves the highest speedup and the lowest MACs, demonstrating its superior efficiency in accelerated sampling.

Method	Steps	MACs (T)	Latency (s)	Speedup	IS ↑	FID ↓	Precision ↑	Recall ↑
Vanilla DiT-XL/2	250	28.61	19.20	1.00×	243.4	2.14	80.70	60.57
Vanilla DiT-XL/2	50	5.72	3.83	1.00×	238.6	2.26	80.16	**59.89**
Diff-Pruning-0.3	50	4.10	2.98	1.29×	4.68	180.76	7.24	20.26
Learning-to-Cache	50	4.36	3.01	1.27×	244.1	2.27	80.94	58.76
MosaicDiff (Ours) Pruned-0.33	50	3.92	2.90	1.32×	**267.8**	**2.24**	**82.01**	57.31
Vanilla DiT-XL/2	20	2.29	1.53	1.00×	223.5	3.48	78.76	**57.07**
Diff-Pruning-0.3	20	1.64	1.20	1.28×	2.99	223.80	2.92	12.38
Learning-to-Cache	20	1.78	1.23	1.24×	227.0	3.46	79.15	55.62
MosaicDiff (Ours) Pruned-0.30	20	1.64	1.20	1.28×	**266.7**	**3.20**	**81.13**	53.67
Vanilla DiT-XL/2	10	1.14	0.77	1.00×	158.3	12.38	66.78	**52.82**
Diff-Pruning-0.3	10	0.82	0.63	1.22×	2.14	270.26	0.93	9.53
Learning-to-Cache	10	1.04	0.69	1.12×	156.3	12.79	66.21	52.15
MosaicDiff (Ours) Pruned-0.30	10	0.79	0.58	1.33×	**174.6**	**12.28**	**66.95**	49.40

Table 2. Result comparison with DiP-GO on DiT-XL/2.

Method	Steps	MACs (T)	FID ↓
Vanilla DiT-XL/2	100	11.86	3.17
DiP-GO Pruned-0.6	-	11.86	3.01
MosaicDiff (Ours) Pruned-0.25	100	8.52	**2.67**
Vanilla DiT-XL/2	70	8.30	3.35
DiP-GO Pruned-0.75	-	6.77	3.14
MosaicDiff (Ours) Pruned-0.25	70	5.99	**3.01**

Table 3. Comparison of ***MosaicDiff*** with existing pruning methods on SDXL at different sparsity levels.

Method	Sparsity	FID↓	CLIP ↑	SSIM ↑
Vanilla SDXL	0	24.90	0.32	1
Diff-Pruning	10%	108.96	0.22	0.31
EcoDiff	10%	33.75	0.31	0.53
MosaicDiff (Ours)	10%	**23.18**	**0.32**	**0.67**
Diff-Pruning	20%	404.87	0.05	0.26
EcoDiff	20%	34.41	0.31	0.5
MosaicDiff (Ours)	20%	**23.79**	**0.32**	**0.64**
EcoDiff	24%	61.00	-	-
MosaicDiff (Ours)	30%	**28.37**	0.31	0.53

Table 4. Comparison of ***MosaicDiff*** Pruned-0.15 with DeepCache on SDXL across different sampling step configurations.

Method	Steps	MACs (T)	FID ↓
Vanilla SDXL	50	159.60	24.90
MosaicDiff (Ours)	50	135.66	**23.73**
DeepCache-N=2	50	93.23	24.88
MosaicDiff (Ours)	25	72.04	**24.04**
DeepCache-N=3	50	59.66	24.67
MosaicDiff (Ours)	20	57.63	**24.17**
DeepCache-N=5	50	37.26	24.43
MosaicDiff (Ours)	10	28.82	**24.32**

ison with DeepCache's aggressive step-caching strategy, we evaluate our method at reduced sampling steps. As shown, our approach achieves lower FID scores at all step configurations while also requiring fewer MACs. These results highlight the superior efficiency and improved image quality delivered by our method on the SDXL architecture.

Table 5. Ablation study on SNR-aware calibration dataset.

Calibration Dataset	IS ↑	FID ↓	Precision ↑	Recall ↑
w/o SNR-aware	227.3	3.96	76.42	53.73
w/ SNR-aware	**266.1**	**3.22**	**81.48**	**57.18**

Table 6. Ablation study on the stage division strategy.

Strategy	Divider 1	Divider 2	IS↑	FID ↓
None	-	-	68.5	40.89
Uniform	667	333	80.6	33.21
$M = 0.45$	600	50	253.2	3.36
$M = 0.55$	550	100	**266.7**	**3.20**
$M = 0.70$	500	130	242.6	3.71

4.3. Ablation study

SNR-Aware Calibration Dataset. In this section, we highlight the importance of incorporating our SNR-aware calibration dataset. As shown in Table 5, utilizing SNR-aware calibration leads to a substantial improvement in generation quality, boosting the IS by over 40 points and reducing the FID by 0.74. Additionally, both Precision and Recall benefit from the SNR-aware calibration, demonstrating its effectiveness in enhancing both fidelity and diversity. These results clearly underscore the importance of leveraging SNR-aware data during the calibration process to achieve superior overall performance.

Stage Division and Choice of M. We emphasize the importance of dividing the denoising process into distinct stages based on the model's sensitivity at different timesteps. We compare the proposed stage division (Ours) with two alternatives: no stage division (None) and uniform stage division and choice of M in Equation 14. As shown in Table 6, applying uniform sparsity across all timesteps without any stage division leads to a significantly degraded FID of 40.89. Performance improves when introducing a simple uniform stage division, but our proposed stage division achieves the best results across all metrics. Notably, in all cases, the overall sparsity is kept constant at 0.3 and sam-

Table 7. Ablation study on the sparsity choice strategy.

Strategy	Sparsity Stage 1	Stage 2	Stage 3	IS↑	FID↓
Uniform	0.3	0.3	0.3	80.6	33.21
Sparser Stage 1	0.9	0.04	0.1	245.8	3.53
Sparser Stage 2	0.6	0.15	0.1	176.4	10.20
Non SNR refined	0.6	0.04	0.4	239.5	4.18
MosaicDiff (Ours)	0.6	0.04	0.1	**266.1**	**3.22**

pling steps are 20 on DiT-XL/2, highlighting the importance of aligning the pruning strategy with the noise schedule to maximize performance.

Sparsity Selection. In this part, we compare the performance of different sparsity choices at every stage according to Equation (15). As shown in Table 7, we fix $M = 0.55$ with stage divider at steps 450 and 900 and keeping a total sparsity of all stages at 0.3 with sampling steps at 20. The results indicate that increasing the sparsity in Stage 2 and 3 leads to a noticeable degradation in performance. In contrast, increasing the sparsity in Stage 1 has minimal impact on FID and IS, suggesting that early-stage pruning is less detrimental to overall generation quality.

4.4. Analysis

Accuracy-Efficiency Tradeoff. Figure 6 illustrates the trade-off between FID and latency, providing a more comprehensive comparison of our method against the state-of-the-art caching algorithm and the vanilla model. Our approach consistently achieves superior performance across different sparsity levels. Specifically, for a given latency, ***MosaicDiff*** achieves lower FID scores compared to both baselines. Conversely, for a target FID, ***MosaicDiff*** consistently requires less latency, demonstrating its efficiency. The results highlight that ***MosaicDiff*** maintains high generation quality while reducing computational cost, even under aggressive sparsity configurations, demonstrating its effectiveness in balancing accuracy and efficiency.

Compatibility with Existing Acceleration Methods. We demonstrate the compatibility of our approach with various existing acceleration strategies, including caching methods, fast samplers and step-distilled models. As shown in Table 8, combining our pruning technique with DeepCache substantially reduces MACs and improves FID scores. Integrating with the fast sampler DPM-Solver++ [20] further enhances image quality and reduces latency. Additionally, our method seamlessly applies to step-distilled models like SDXL-Turbo [36], simultaneously lowering computational cost and improving generation quality (see Appendix for more details). These results highlight the versatility and broad applicability of ***MosaicDiff*** in conjunction with complementary acceleration methods.

Runtime of Compression. As a training-free approach, the primary computational cost of ***MosaicDiff*** is Hessian matrix calculation. Table 9 shows that our method prunes a full three-stage DiT-XL/2 (675M parameters) within 30

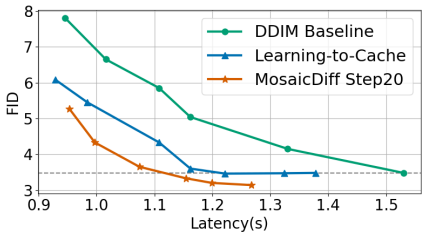


Figure 6. Accuracy-Efficiency Tradeoff for DiT-XL/2 with 20 sampling steps. The dashed line represents the FID of the vanilla DDIM baseline at 20 sampling steps.

Table 8. Performance of ***MosaicDiff*** at sparsity 0.15 when combined with existing acceleration strategies on SDXL.

Method	Steps	MACs (T)	FID ↓
DeepCache-N=3	50	59.66	24.61
Ours + DeepCache-N=3	50	50.71	**23.88**
DPM-solver++	20	67.80	24.96
Ours + DPM-solver++	20	57.63	**24.01**
SDXL-Turbo	4	13.56	30.93
Ours + SDXL-Turbo	4	11.53	**30.08**

Table 9. Runtime of ***MosaicDiff*** and other compression methods.

Model	Resolution	Method	Hardware	GPU hours ↓
DiT-XL/2	256 × 256	L2C	RTX 4090	16.3
		MosaicDiff	RTX 4090	**0.5**
SD-1.5	512 × 512	DiP-GO	MI250	2.5
		MosaicDiff	RTX 4090	**0.8**
SDXL	1024 × 1024	*MosaicDiff*	RTX 4090	6.0

minutes using a single RTX 4090 GPU. In contrast, L2C requires over 16 hours of router training on identical hardware. DiP-GO takes approximately 2.5 hours to train a pruner for SD-1.5 (865M parameters) on an AMD MI250 GPU [46], whereas our method prunes it in just 0.8 hours on one RTX 4090. Furthermore, ***MosaicDiff*** compresses the significantly larger SDXL model (2.6B parameters) at higher resolution in merely 6 hours. These results highlight the practicality, speed and ease-of-integration advantages of ***MosaicDiff*** compared to training-based or fine-tuning-dependent methods for accelerating diffusion models.

5. Conclusion

We have presented ***MosaicDiff***, a training-free structural pruning framework applicable to both transformer-based and U-Net-based diffusion models. By leveraging the varying learning speeds across the denoising process, our approach assigns varying sparsity levels to different sampling stages. Extensive experiments demonstrate that the proposed method consistently outperforms state-of-the-art compression methods, confirming its effectiveness in accelerating diffusion models without sacrificing generation quality. While this approach introduces slightly more memory overhead due to the need to store multiple pruned subnetworks corresponding to different sampling stages, we leave exploring and optimizing this for future work.

References

[1] Jooyoung Choi, Jungbeom Lee, Chaehun Shin, Sungwon Kim, Hyunwoo Kim, and Sungroh Yoon. Perception prioritized training of diffusion models, 2022. 2, 4

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[3] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. 6

[4] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. In *Advances in Neural Information Processing Systems*, pages 16716–16728. Curran Associates, Inc., 2023. 2, 3

[5] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. In *Advances in Neural Information Processing Systems*, 2023. 2, 6

[6] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot, 2023. 2

[7] B. Hassibi, D.G. Stork, and G.J. Wolff. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, pages 293–299 vol.1, 1993. 5

[8] Yefei He, Luping Liu, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. Ptqd: Accurate post-training quantization for diffusion models. *arXiv preprint arXiv:2305.10657*, 2023. 2

[9] 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, 2018. 6

[10] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 1

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 1, 6

[12] Bo-Kyeong Kim, Hyoung-Kyu Song, Thibault Castells, and Shinkook Choi. On architectural compression of text-to-image diffusion models. 2023. 2

[13] Bo-Kyeong Kim, Hyoung-Kyu Song, Thibault Castells, and Shinkook Choi. Bk-sdm: A lightweight, fast, and cheap version of stable diffusion. In *European Conference on Computer Vision*, pages 381–399. Springer, 2024. 2

[14] Eldar Kurtic, Elias Frantar, and Dan Alistarh. Ziplm: Inference-aware structured pruning of language models, 2023. 2

[15] Muyang Li*, Yujun Lin*, Zhekai Zhang*, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. Svdquant: Absorbing outliers by low-rank components for 4-bit diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025. 2

[16] Xiuyu Li, Long Lian, Yijiang Liu, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. Q-diffusion: Quantizing diffusion models. *arXiv preprint arXiv:2302.04304*, 2023. 2

[17] Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed, 2024. 2

[18] 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. 6

[19] 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. 2

[20] 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. 8

[21] 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. 2

[22] 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. 2

[23] Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-cache: Accelerating diffusion transformer via layer caching, 2024. 2, 6

[24] 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, 6

[25] Chenlin Meng, Ruiqi Gao, Diederik P Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. *arXiv preprint arXiv:2210.03142*, 2022. 2

[26] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14297–14306, 2023. 2

[27] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models, 2021. 2, 4

[28] Zizheng Pan, Bohan Zhuang, De-An Huang, Weili Nie, Zhiding Yu, Chaowei Xiao, Jianfei Cai, and Anima Anandkumar. T-stitch: Accelerating sampling in pre-trained diffusion models with trajectory stitching. *arXiv*, 2024. 2, 4

[29] 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, 3, 6

[30] 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, 2023. 2, 6

[31] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 1

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 6

[33] 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. 1

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 6

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

[36] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *European Conference on Computer Vision*, pages 87–103. Springer, 2024. 8

[37] 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. 2

[38] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 2

[39] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019. 2

[40] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023. 2

[41] Shengkun Tang, Yaqing Wang, Caiwen Ding, Yi Liang, Yao Li, and Dongkuan Xu. Adadiff: Accelerating diffusion models through step-wise adaptive computation. In *European Conference on Computer Vision*, pages 73–90. Springer, 2024. 2

[42] Shengkun Tang, Oliver Sieberling, Eldar Kurtic, Zhiqiang Shen, and Dan Alistarh. Darwinlm: Evolutionary structured pruning of large language models. *arXiv preprint arXiv:2502.07780*, 2025. 2

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 6

[44] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. 6

[45] Yang Zhang, Er Jin, Yanfei Dong, Ashkan Khakzar, Philip Torr, Johannes Stegmaier, and Kenji Kawaguchi. Effortless efficiency: Low-cost pruning of diffusion models. *arXiv preprint arXiv:2412.02852*, 2024. 2, 3, 6

[46] Haowei Zhu, Dehua Tang, Ji Liu, Mingjie Lu, Jintu Zheng, Jinzhang Peng, Dong Li, Yu Wang, Fan Jiang, Lu Tian, Spandan Tiwari, Ashish Sirasao, Jun-Hai Yong, Bin Wang, and Emad Barsoum. Dip-go: A diffusion pruner via few-step gradient optimization. *arXiv preprint arXiv:2410.16942*, 2024. 2, 6, 8