DiP-GO: A Diffusion Pruner via Few-step Gradient Optimization

Haowei Zhu[*12], **Dehua Tang**[1], **Ji Liu**[1], **Mingjie Lu**[1], **Jintu Zheng**[1], **Jinzhang Peng**[1], **Dong Li**[1], **Yu Wang**[1], **Fan Jiang**[1], **Lu Tian**[1], **Spandan Tiwari**[1], **Ashish Sirasao**[1], **Junhai Yong**[2], **Bin Wang**[†2], **Emad Barsoum**[†1]

[1]Advanced Micro Devices, Inc. [2]Tsinghua University

zhuhw23@mails.tsinghua.edu.cn;

{dehua.tang, ji.liu, jinz.peng, d.li, lu.tian, ebarsoum}@amd.com;

{yongjh, wangbins}@tsinghua.edu.cn

Abstract

Diffusion models have achieved remarkable progress in the field of image generation due to their outstanding capabilities. However, these models require substantial computing resources because of the multi-step denoising process during inference. While traditional pruning methods have been employed to optimize these models, the retraining process necessitates large-scale training datasets and extensive computational costs to maintain generalization ability, making it neither convenient nor efficient. Recent studies attempt to utilize the similarity of features across adjacent denoising stages to reduce computational costs through simple and static strategies. However, these strategies cannot fully harness the potential of the similar feature patterns across adjacent timesteps. In this work, we propose a novel pruning method that derives an efficient diffusion model via a more intelligent and differentiable pruner. At the core of our approach is casting the model pruning process into a SubNet search process. Specifically, we first introduce a SuperNet based on standard diffusion via adding some backup connections built upon the similar features. We then construct a plugin pruner network and design optimization losses to identify redundant computation. Finally, our method can identify an optimal SubNet through few-step gradient optimization and a simple post-processing procedure. We conduct extensive experiments on various diffusion models including Stable Diffusion series and DiTs. Our DiP-GO approach achieves 4.4× speedup for SD-1.5 without any loss of accuracy, significantly outperforming the previous state-of-the-art methods.

1 Introduction

Diffusion models have undergone significant advancements over the past years due to the outstanding capabilities of diffusion probabilistic models (DPMs) [1]. DPMs typically consist of two processes: the noise diffusion process and the reverse denoising process. Given their remarkable superiority in content generation, diffusion models have made significant progress in various fields of general image generation, including text-to-image generation [2, 3], layout-to-image generation [4, 5], image editing [6, 7], and image personalization [8, 9]. Furthermore, diffusion models have contributed to advancements in autonomous driving, ranging from driving dataset generation [10–12] to perception model enhancement [13, 14] through diffusion strategies. However, DPMs often incur considerable computational overhead during both the training and inference phases. The high cost of inference, due

[*]Work done during an internship at AMD.

[†]Corresponding author.

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

https://doi.org/10.52202/079017-2940

to the multi-step denoising computation during the sampling process, can significantly impact their practical application. Many efforts [15–17] have been made to improve the efficiency of diffusion models, which can be broadly divided into two types of optimization: inference sampling optimization and model structural optimization.

Sampling optimization methods reduce the number of sampling steps for generation without compromising image quality. For instance, DDIM [15] reduces these steps by exploring a non-Markovian process without requiring model retraining. LCM [18, 19] enable image generation in fewer steps with retraining requirements. Structural optimization methods [16, 17, 20, 21] aim to reduce computational overhead through efficient model design and model pruning. These methods require retraining the diffusion model, which entails significant computational overhead and large-scale datasets, making them neither convenient nor efficient. DeepCache [22] proposes a novel training-free paradigm based on the U-Net architecture in diffusion models, caching and retrieving features across adjacent denoising stages to reduce redundant computation costs. However, DeepCache only reuses the output feature of a U-Net block in a denoising step via a simple and static strategy. We believe many intermediate features remain untapped, and the simple static strategy cannot fully exploit the potential of similar feature patterns across adjacent timesteps during inference, as observed in recent studies [15, 18, 22].

To address these challenges, we introduce **Di**ffusion **P**runing via Few-step **G**radient **O**ptimization (DiP-GO), a method designed to achieve efficient model pruning with enhanced dynamism and intelligence. Our approach rethinks the diffusion model during inference by proposing a SuperNet based on standard diffusion via adding some backup connections built upon the similar features, conceptualizing the inference process as a specific SubNet derived from our proposed SuperNet. We reformulate the diffusion model pruning into a SubNet search process. By addressing the out-of-memory issue inherent in the backward process during expanded denoising timesteps using the gradient checkpoint [23] method, we introduce a plugin pruner that discovers an optimal SubNet surpassing existing methods through carefully designed optimization losses. Extensive experiments validate the effectiveness of our approach, demonstrating a 4.4$\times$ speedup on Stable Diffusion 1.5. Moreover, our method efficiently prunes the DiT model [3] without requiring retraining the diffusion model, achieving significant inference speedup. Our contribution can be summarized as follows: (1) We define a SuperNet based on standard diffusion and show how to obtain a SubNet. This transforms the diffusion optimization problem into an efficient SubNet search process without the need for retraining pretrained diffusion models. (2) We design a plugin pruner tailored specifically for diffusion models. This pruner optimizes pruning constraints while maximizing synthesis capability. Additionally, we introduce a post-processing method for the pruner to ensure that the SubNet meets specified pruning requirements. (3) We conduct extensive pruning experiments across various diffusion models, including Stable Diffusion 1.5, Stable Diffusion 2.1, Stable Diffusion XL, and DiT. Extensive experiments demonstrate the superiority of our method, achieving a notable 4.4 $\times$ speedup during inference on Stable Diffusion 1.5 without the need for retraining the diffusion model.

2 Related Work

2.1 Efficient Diffusion Models

The diffusion models, celebrated for their iterative denoising process during inference, play a pivotal role in content generation but are often hindered by time-consuming operations. To mitigate this challenge, extensive research has focused on accelerating diffusion models. Acceleration efforts typically approach the problem from two primary perspectives:

Efficient Sampling Methods. Recent works focus on reducing the number of denoising steps required for content generation. DDIM [15] achieves this by exploring a non-Markovian process related to neural ODEs. Fast high-order solvers [24, 25] for diffusion ordinary differential equations also enhance sampling speed. LCMs [18, 19] treat the reverse diffusion process as an augmented probability flow ODE (PF-ODE) problem, inspired by Consistency Models (CMs) [26], enabling generation in fewer steps. PNDM [27] emphasizes efficient sampling without retraining diffusion model. Additionally, ADD [28] combines adversarial training and score distillation to transform pretrained diffusion models into high-fidelity image generators using only single sampling steps.

Efficient Structural Methods. Other efforts concentrate on reducing the computational overhead associated with each denoising step. Previous methods [16, 17, 22] have typically conducted extensive

empirical studies to identify and remove non-critical layers from U-Net architectures to achieve faster networks. BK-SDM [16] customizes three efficient U-Nets by strategically removing residual and attention blocks. Derived from BK-SDM, KOALA [17] develops two efficient U-Nets of varying sizes tailored for SD-XL applications. Diff-pruning [20] employs Taylor expansion over pruned timesteps to pinpoint essential layer weights, optimizing model efficiency without sacrificing performance. DeepCache [22] enhances inference efficiency by reusing predictions from blocks in previous timesteps within the U-Net architecture. LAPTOP-Diff [21] tackles optimization problems with a one-shot pruning approach, incorporating normalized feature distillation to streamline retraining processes. T-GATE [29] not only reduces computation overhead but also marginally lowers FID scores by omitting text conditions during fidelity-improvement stages.

In addition to the two primary acceleration methods, other strategies such as distillation [28, 30, 31], early stopping [32], and quantization [33] are commonly employed to enhance performance and efficiency. However, most of these strategies necessitate retraining pretrained models. Our method falls under the category of efficient structural methods by focusing on reducing inference time at each timestep. Importantly, these efficiency gains are achieved without retraining the diffusion model.

2.2 Model Optimization

Network Pruning. The taxonomy of pruning methodologies typically divides into two main categories: unstructured pruning methods [34–36] and structural pruning methods [37–40]. Unstructured pruning methods involve masking parameters without structural constraints by zeroing them out, often requiring specialized software or hardware accelerators. In contrast, structured pruning methods generally remove regular parameters or substructures from networks. Recent works have been interested in accelerating transformers. Dynamic skipping blocks, which involve selectively removing layers while maintaining the overall structure, have emerged as a paradigm for transformer compression [41–44]. However, applying structural pruning techniques to diffusion modeling poses unique challenges that necessitate reevaluating conventional pruning methods.

3 Methodology

In this study, we introduce the Diffusion Pruner via Few-step Gradient Optimization (DiP-GO), which utilizes a neural network to predict whether to skip or keep each computational block during inference. Our primary objective is to identify the optimal subset of computational blocks that facilitate denoising with minimal computational overhead. As illustrated in Figure 2, our method comprises three main components: a neural network pruner, optimization losses, and a post-process algorithm to derive the pruned model based on the predictions of pruner. The neural network pruner is designed with learnable queries inspired by DETR [45] to predict the state of each block. Our proposed optimization losses include sparsity and consistency constraints for generation quality, guiding the pruner to accurately assess the importance of each block. In this Section, we first revisit the framework of diffusion models in Section 3.1, emphasizing their potential for exploring pruned networks. In Section 3.2, we introduce a SuperNet based on diffusion models and demonstrate how to derive a SubNet or pruned network from it for inference acceleration, highlighting the challenges in achieving an optimal SubNet. Section 3.3 details our method, including the neural network pruner, optimization losses, and post-process algorithm for obtaining a SubNet that meets pruning requirements. Finally, we provide insights into the training and inference processes of our method.

3.1 Preliminary

We begin with a brief introduction to diffusion models. Diffusion models are structured to learn a series of sequential state transitions with the goal of iteratively refining random noise sampled from a known prior distribution towards a target distribution x_0 that matches the data distribution. During the forward diffusion process, the transition from x_{t-1} to x_t is initially determined by a forward transition function, which can be described as follows:

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

where the hyperparameter $\{\beta_t \in (0,1)\}_{t=1}^{T}$ increases with each successive time step t.

 https://doi.org/10.52202/079017-2940

To generate samples from a learned diffusion model, it involves a series of reverse state transitions from $\boldsymbol{x}_T \rightarrow \cdots \rightarrow \boldsymbol{x}_0$ to denoise random noise $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ into the clean data point $\boldsymbol{x}_0$. At each timestep, the denoised output $\boldsymbol{x}_{t-1}$ is predicted by approximating the noise prediction network, which is conditioned on the time embedding t and the previous data point $\boldsymbol{x}_t$:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \frac{1}{\sqrt{a_t}}(x_t - \frac{\beta_t}{\sqrt{1-\overline{a}_t}} z_\theta(x_t, t)), \beta \mathbf{I}) \tag{2}$$

where the covariance constant $\beta_t = 1 - \alpha_t$, $\overline{a}_t = \prod_{i=1}^{T} \alpha_i$, and $z_\theta(x_t, t)$ are the parameterized deep neural networks. With the reverse Markov chain, we can iteratively sample from the learnable transition kernel $x_{t-1} \sim p_\theta(x_{t-1}|x_t)$ until $t = 1$.

Diffusion modes typically require multi-step conditional sampling to gradually obtain the target sample point x_0. However, recent studies [15, 18, 22] have highlighted that multi-step inference processes involve substantial redundant feature computations, particularly in noise prediction networks like UNet and Transformer. For example, in Stable Diffusion 1.4 models with 25 steps, Multiply-Accumulate Operations (MACs) of UNet can comprise up to 87.2% of the total computational load [16]. This underscores significant potential for accelerating inference by effectively eliminating these redundancies. In this work, we propose accelerating the diffusion model by integrating a differentiable pruning network designed to identify and remove these redundant computations.

3.2 SuperNet and SubNet of Diffusion Model

Our goal is to identify and remove unimportant blocks during inference to accelerate the process. To achieve this, we introduce a SuperNet based on the diffusion model. This SuperNet is designed to facilitate block removal while ensuring the pruned model maintains inference capability through additional connections. Our approach effectively eliminates unimportant blocks during inference, essentially deriving a SubNet from the SuperNet by skipping these unnecessary components. Thus, the pruning process can be conceptualized as a SubNet search within the SuperNet framework.

How to Construct a SuperNet. Recent studies [15, 18, 22] have observed that diffusion models often exhibit similar feature patterns across adjacent timesteps during inference. Building on this insight, we enhance the standard diffusion model's inference phase by introducing additional connections from the current timestep to the previous one. These connections serve as backups for blocks that may be removed, ensuring each block retains valid inputs even if its dependent blocks are eliminated for acceleration. Specifically, for all inputs of each block across all timesteps except the inital step during inference, we establish a backup input connection to the corresponding block in the previous timestep, as illustrated in Figure 1.

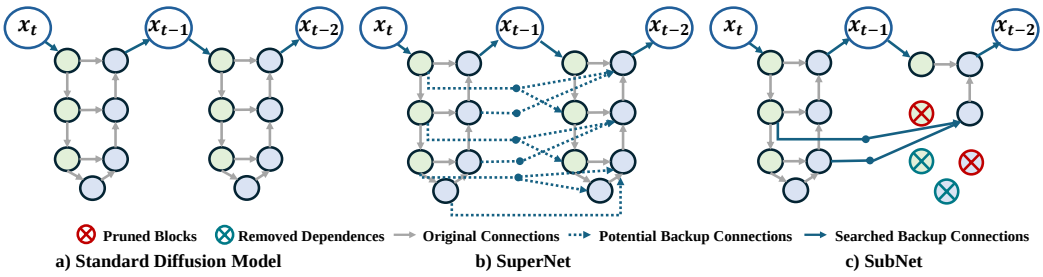


Figure 1: Overview of the SuperNet and SubNet. Standard diffusion models execute the full inference path step by step. In our framework, we propose a SuperNet based on the original flow and integrate backup connections to facilitate block removal. This allows the partial inference SubNet to efficiently eliminate redundant computational costs.

How to Obtain a SubNet. To construct the SuperNet for the standard diffusion model, we introduce additional connections that ensure a valid SubNet selects either the original input connection or the backup input connection, but not both simultaneously. This design principle mandates that if a dependent block is pruned, its original input connection is also eliminated to reflect the block's removal. Conversely, if the dependent block is retained, the backup input connection is removed to maintain efficient inference, as depicted in Figure 1.

https://doi.org/10.52202/079017-2940

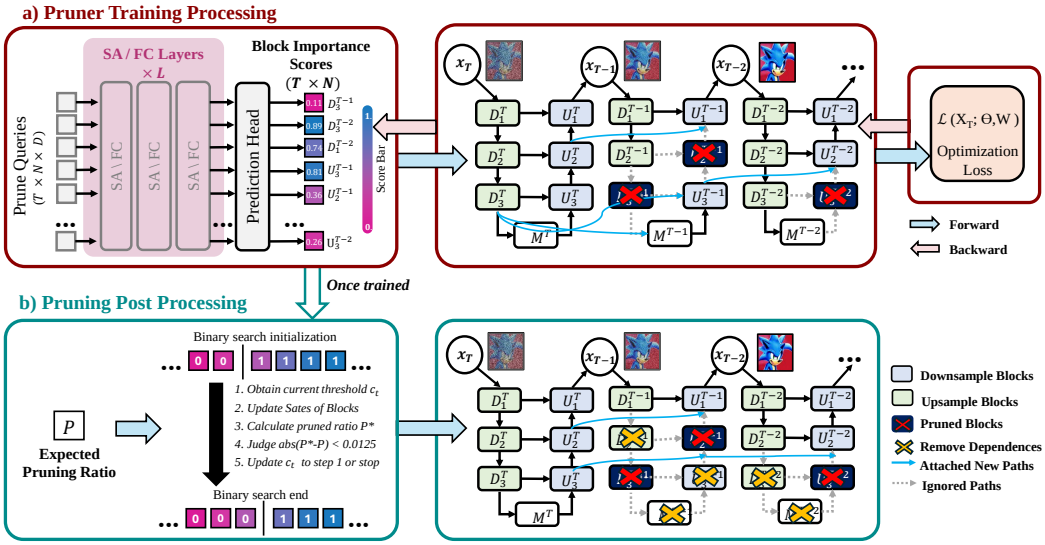


Figure 2: Overview of our diffusion pruner. a) DiP-GO employs a pruner network to learn the importance scores of blocks in the diffusion sampling process. It takes $N \times T$ queries as input and passes them through stacked self-attention (SA) and fully connected (FC) layers to capture the structural information in existing diffusion models. The network predicts the partial inference paths based on the $N \times T$ importance scores and is optimized by consistent and sparse loss. b) Once trained, the pruner network is discarded. We can infer the optimal partial inference path with expected computational costs via post-processing based on the predicted importance scores.

We draw inspiration from the Lottery Ticket Hypothesis (LTH) [46], which posits the existence of a sub-network capable of achieving comparable performance to the original over-parameterized network for a given task, but with fewer unnecessary weights. Moreover, prior work [22] has explored manually removing redundant computations by caching features across adjacent steps. Thus, our approach seeks to identify an optimal SubNet from the SuperNet, maximizing diffusion model acceleration while minimizing any loss in generation quality.

Hard to Obtain an Optimal SubNet. The challenge of obtaining an optimal SubNet is compounded by the large number of blocks expanded during inference. In a diffusion pipeline with $N \times T$ blocks (where N is the number of blocks per timestep and T is the number of timesteps), each block's decision to be kept or removed results in $2^{N \times T}$ possible configurations. For instance, a 50-step PLMS setup [27], considering 9 blocks in the U-Net, yields 2^{450} choices ($> 10^{135}$). Traditional search methods like random search and genetic algorithms [47] often struggle in such vast search spaces. Gradient-based optimization offers a promising approach to tackle this challenge. However, there are significant hurdles to overcome. First, effectively modeling discrete block states (kept or removed) with parametric methods poses difficulties. Second, training the entire model, comprising both the parametric model and the expanded diffusion model with denoising timesteps, risks encountering out-of-memory (OOM) issues.

3.3 Our DiP-GO Approach

In this study, we introduce a diffusion pruner network designed to predict importance scores for all blocks during reverse sampling as depicted in Figure 2. To optimize the pruner network effectively, we employ two key optimization losses: consistency and sparsity losses, leveraging few-step gradient optimization. Addressing the OOM issue inherent in such computations, we implement gradient checkpointing and half-precision floating-point representation techniques, enabling efficient search processes on a single GPU. Once the pruner network trained, we extract predicted importance scores for all blocks. Subsequently, we devise a post-processing algorithm to utilize these scores, generating pruned SubNets of diffusion models that satisfy specific pruning criteria.

Pruner Network. Our pruner network comprises three main components: $N \times T$ learnable queries, a query encoder, and a prediction head. We design the learnable queries to match the number of all blocks during inference. These queries are optimized with sparsity and consistency loss constraints to

https://doi.org/10.52202/079017-2940

learn the contextual information necessary for predicting the importance score of each block. For the query encoder, we provide two options: a simple version with several stacked linear layers, and a more complex version with several stacked self-attention layers to facilitate interaction among the learnable queries. Our experiments demonstrate that both versions can effectively obtain optimal SubNets in various diffusion models under different pruning requirements. The prediction head consists of $N \times T$ simple branches, each containing two stacked linear layers followed by a softmax operation. The final linear layer has a dimension of 2, and the softmax output represents the importance scores of a block. During training or inference, the query embeddings are transformed into output embeddings via the query encoder. These embeddings are then independently decoded into binary vectors by the multi-layer prediction head, resulting in $N \times T$ importance scores for all blocks.

Optimization Losses. The k-th predicted binary vector of importance score, denoted as $\boldsymbol{s}^k$, represents the likelihood of its corresponding block being removed or kept in the denoising process. A gate $\boldsymbol{g} \in \{0,1\}^{TN}$ is derived based on $\boldsymbol{s}$, where $\boldsymbol{g}^k = 0$ or $\boldsymbol{g}^k = 1$ indicate removing or keeping the k-th computation block, respectively. Only the blocks that are kept according to $\boldsymbol{g}$ will be calculated in the denoising process. However, directly converting predicted probabilities $\boldsymbol{s}$ into discrete gates $\boldsymbol{g}$ with $\arg\max$ is non-differentiable. To address this issue, we utilize the Straight-Through (ST) Estimator [48] to approximate the real gradient $\nabla_\theta \boldsymbol{g}$ with the gradient of the soft prediction $\nabla_\theta \boldsymbol{s}$. To encourage both high-fidelity predictions and minimal computation block usage, we design our training objective function as a combination of consistent loss $\mathcal{L}_c$ and sparse loss $\mathcal{L}_s$, formulated as follows:

$$\mathcal{L}(\boldsymbol{x}_T; \theta, W) = \mathcal{L}_c + \alpha_s \mathcal{L}_s = \begin{cases} f(\boldsymbol{x}_0^p, \boldsymbol{x}_0^{gt}) + \frac{\alpha_s}{NT} \sum_k^{NT} \gamma^k \boldsymbol{g}^k & \text{if } sparsity < \tau \\ f(\boldsymbol{x}_0^p, \boldsymbol{x}_0^{gt}) & \text{if } sparsity \geq \tau \end{cases} \tag{3}$$

Here, α_s represents a hyperparameter used to balance the consistent and sparse losses. θ and W denote the pruner network and pretrained diffusion model, respectively. $f(\cdot)$ denotes a distance function that evaluates the consistency between the generated clean data point $\mathbf{x}_0^p$ from partial inference of the pruned SubNet and the $\mathbf{x}_0^{gt}$ from full inference. This function can be any distance measure, and in this work, we utilize a negative SSIM loss [49]. The sparse loss encourages minimal computational usage and is weighted by the computational flops proportion γ^k of the k-th block, thereby imposing a greater penalty on heavier blocks. The calculation of γ takes into account the cascading relationships between blocks. Specifically, when a block is pruned, the associated dependent blocks will also pruned. Therefore, the flops reduction from pruning a block includes the block itself and its dependent blocks. We denote the flops reduction ratio after pruning the k-th block as γ^k. The flops ratio γ is in the range $[0, 1]$. The sparse loss is only introduced when the sparsity (pruning ratio) is below a certain threshold τ. This compound loss controls the trade-off between efficiency (block usage) and accuracy (generation quality).

Post-Processing Algorithm. After training the pruner network, our diffusion pruner is able to predict which computation blocks during inference contribute less to generation quality based on the importance scores for all the blocks. As the importance scores are continuous values in inference phase, they can not be utilized directly to identify which blocks should be removed to meet given pruning requirements. Therefore, we present a post-process algorithm to obtain an appropriate threshold for these importance score to meet the pruning requirements as shown in Algorithm 1 in Appendix B. Considering the required pruning sparsity, we use bisection lookup to select the appropriate threshold value to identify which blocks should be removed to meet the pruning ratio. Specifically, the blocks whose important scores below the threshold should be removed and the kept blocks should update their input connections as mentioned in Section 3.2 to maintain the pruned model inference. Thus a pruned model met the pruning ratio has been obtained.

Training and Inference Details. In the training phase, the prompt inputs are fed into the diffusion model to obtain two kinds of outputs, one is generated by the baseline diffusion model and the other is generated by the pruned model obtained via the current predictions of the pruner network. Then our proposed losses are utilized to optimize the pruner network to enable distinguishing the less important blocks. In the pruner's network, we initialize the weight of the last linear layer's output channel to 0 and its bias to 1. This setup ensures that at the beginning of training, the consistency loss is 0 and the sparsity loss is 1, facilitating smooth training. As training progresses, the sparsity loss gradually decreases while the number of pruned blocks increases, causing the consistency loss to rise. To maintain network fidelity after pruning, we switch to training only with the consistency loss once the sparsity loss reaches 0.2, continuing until training is complete. Once the pruner is well trained, we can obtain pruned models to meet the pruning requirements via our post-process algorithm.

4 Experiments

4.1 Experimental Setup

Pre-trained Model and Datasets. We select four official pretrained Diffusion Models (i.e., SD-1.5 [2], SD-2.1 [2], SD-XL [50] and DiT [3]) to evaluate our approach. The SD series models are constructed on the U-Net [51] and the DiT is constructed on the transformer [52]. We utilize a subset of the DiffusionDB [53] dataset comprising 1000 samples to train our pruner network, utilizing only textual prompts. Following previous works [29, 22], we evaluate the DiP-GO on three public datasets, i.e., PartiPrompts [54], MS-COCO 2017 [55] and ImageNet [56].

Evaluation Metrics. We employ the Fréchet Inception Distance (FID) [57] metrics to assess the quality of images created by the generative models. FID quantifies the dissimilarity between the Gaussian distributions of synthetic and real images. A lower FID score indicates a closer resemblance to real images in the generative model. Additionally, we utilize the CLIP Score [58] (ViT-g/14) to evaluate the relational compatibility between images and text.

Implementation Details. For Stable Diffusion models, we utilize the SGD optimizer with a cosine learning schedule for 1000 steps of training. The batch size, learning rate, and weight decay are set to 1, 0.1, and 1×10^{-4}, respectively. The hyperparameters α_s, τ, and the query embedding dimension D, along with the encoder layer number L, are set to 1, 0.2, 512, and 1, respectively. For the Diffusion Transformer model, we use the same experimental configuration as for the stable diffusion model, except that the learning rate set to 10. To evaluate the inference efficiency, we evaluate the Multiply Accumulate Calculation (MACs), Parameters (Params), and Speedup for all models with batch size of 1 in the PyTorch 2.1 environment on the AMD MI250 platform. Besides, we report MACs in those tables, which refer to the totals MACs for all steps.

Table 1: Comparison with PLMS, BK-SDM and DeepCache on SD-1.5. We utilize prompts in PartiPrompt and COCO2017 validation set.

Method	Pruning Type	PartiPrompts			COCO2017		
		MACs ↓	Speedup ↑	CLIP Score ↑	MACs ↓	Speedup ↑	CLIP Score ↑
PLMS - 50 steps	Baseline	16.94T	1.00×	29.51	16.94T	1.00×	30.30
BK-SDM - Base	Structured	11.19T	1.49×	28.88	11.19T	1.45×	29.47
PLMS - 25 steps	Fast Sampler	8.47T	2.04×	29.33	8.47T	1.91×	29.99
PLMS - Skip - Interval=2	Structured	8.47T	2.04×	19.74	8.47T	1.91×	16.78
DeepCache	Structured	6.52T	2.15×	29.46	6.52T	2.11×	30.23
Ours (w/ Pruned-0.80)	Structured	**3.38T**	**4.43**×	**29.51**	**3.38T**	**4.40**×	**30.29**
BK-SDM - Small	Structured	10.88T	1.75×	27.94	10.88T	1.68×	27.96
PLMS - 15 steps	Fast Sampler	5.08T	2.89×	28.58	5.08T	2.59×	29.39
Ours (w/ Pruned-0.85)	Structured	**2.54T**	**5.52**×	**29.07**	**2.54T**	**5.46**×	**29.84**

Table 2: Comparison of computational complexity, inference speed, CLIP Score and FID on the MS-COCO 2017 validation set on SD-2.1.

Inference Method	MACs↓	Speedup↑	CLIP Score ↑	FID-5K ↓
SD-2.1-50 steps [2]	38.04T	1.00×	31.55	27.29
SD-2.1-20 steps [2]	15.21T	2.49×	**31.53**	27.83
Ours (w/ Pruned-0.7)	11.42T	3.02×	31.50	**25.98**
Ours (w/ Pruned-0.8)	7.61T	3.81×	30.92	27.69

4.2 Comparison with State-of-the-Art Methods on Different Base Models

Stable Diffusion on PartiPrompt and COCO2017. We compare our method with the state-of-the-art (SOTA) compression methods on Stable Diffusion 1.5 (SD-1.5), and the results are summarized in Table 1. Compared to the SOTA DeepCache [22], our approach demonstrates significant performance improvements, achieving nearly $2\times$ fewer MACs while maintaining better CLIP Scores. Our method can achieve $4.4\times$ speedup compared to the baseline model. Furthermore, our method does not require training the diffusion model, which preserves the pre-trained knowledge of the diffusion model. Also, we apply our method on the SD-2.1 model to verify the effectiveness, as shown in Table 2 , our method achieves significant acceleration while maintaining generation quality, demonstrating its superiority.

Diffusion Transformers on ImageNet. To the best of our knowledge, we are the first to apply pruning to DiT [3] model. Therefore, we have replicated a training-free acceleration method, DeepCache

https://doi.org/10.52202/079017-2940

Table 3: Comparison of pruning type, computational complexity, FID and inference speed on the ImageNet validation datasets on DiT. * denotes the results reproduced with diffusers [59].

Method	Pruning Type	MACs ↓	FID-50K ↓	Speedup ↑
DiT-XL/2-250 steps	-	29.66T	2.27	1.00×
DiT-XL/2*-250 steps	Baseline	29.66T	2.97	1.00×
DiT-XL/2*-110 steps	Fast Sampler	13.05T	3.06	2.13×
DiT-XL/2*-100 steps	Fast Sampler	11.86T	3.17	2.46×
DeepCache(DiT-XL/2*)-N=2	Structured Pruning	15.88T	3.07	1.76×
Ours (DiT-XL/2* w/ Pruned-0.6)	Structured Pruning	11.86T	**3.01**	**2.43×**
DiT-XL/2*-70 steps	Fast Sampler	8.30T	3.35	3.49×
DeepCache(DiT-XL/2*)-N=5	Structured Pruning	6.77T	3.20	3.44×
Ours (DiT-XL/2* w/ Pruned-0.75)	Structured Pruning	7.40T	**3.14**	**3.60×**

with intervals = 2 and 5, on DiT for comparison. The results in Table 3 show that our method can speed up the original DiT model by a factor of 2.4 with minimal performance loss, while DeepCache has a lower speedup ratio when applied to the DiT model. This can be attributed to DeepCache's overreliance on pre-defined structures, whereas our method can automatically learn the optimal pruning strategy for the given model, thereby achieving superior performance.

4.3 Compatibility with Fast Sampler

We investigate the compatibility of DiP-GO with methods that prioritize reducing sampling steps using faster samplers: DDIM [15], DPM-Solver [25], and LCM [18]. As shown in Table 4, it indicates that our method further improves computational efficiency on existing fast samplers. Specifically, we reduce MACs by a factor of 5 on the SD-1.5 with DDIM sampler and by 3.36× on the SD-2.1 with DPM-Solver. Our method achieves nearly unchanged performance with significant acceleration. Additionally, our method benefits from information redundancy in multi-step optimization processes, showing relatively limited acceleration performance on fewer-step LCM due to its low redundancy in features across adjacent timesteps.

Table 4: Comparison with PLMS, SSIM, and LCM samplers. We evaluate the effectiveness of our methods on COCO2017 validation set.

Sampler	Base Model		Ours	
	MACs ↓	CLIP Score ↑	MACs ↓	CLIP Score ↑
DDIM (SD-1.5 w/ 50 steps)	16.94T	30.30	3.38T	30.29
DPM (SD-2.1 w/ 50 steps)	38.04T	31.55	11.42T	31.50
DPM (SD-2.1 w/ 25 steps)	19.02 T	31.59	9.51T	31.52
LCM (SD-XL w/ 4 steps)	11.95T	31.92	11.58T	31.30

4.4 Ablation Study

Compared with Different Consistent Constraints. We further compare other alternatives explored for our consistent loss designs, we further scrutinize additional options, including L1, L2, SSIM, and L1+SSIM losses, as depicted in Table 5. The results demonstrate that SSIM emerges as the most effective choice, boasting the highest CLIP-Score. In contrast, the L1 loss function often results in image blurring or distortion due to its sensitivity to pixel-level differences, the L2 loss may yield overly smoothed images by penalizing squared differences between pixels. Conversely, the combination of L1+SSIM loss attempts to address these limitations but can complicate the training process and suffer from trade-offs. Therefore, SSIM emerges as the preferred choice in our consistent loss designs, offering superior accuracy and stability while preserving image quality.

Table 5: Comparison with different consistent loss types. Here we conduct pruning experiments with 80% sparsity on COCO2017 validation using SD-1.5.

Loss Type	L1	L2	SSIM	L1 + SSIM
CLIP Score↑	29.94	29.71	**30.29**	29.77

Effect of Gradient Optimization. As the traditional search algorithm can also obtain SubNets from our proposed SuperNet. It is crucial to validate whether traditional search-based algorithms yield

positive effectiveness. We assess two search algorithms: random search and genetic algorithm-based search [47] in Table 6. We have iterated the search 1000 times using the first 500 images of the test set as a calibration dataset. Remarkably, we observe that the search time of traditional search algorithms is significantly longer than the training time of our method due to a large number of evaluations. Moreover, due to the vast search space, traditional search algorithms struggle to achieve satisfactory results. Additionally, traditional search algorithms lack the "once-for-all" characteristic, requiring re-execution when faced with deployment scenarios demanding different computational resources. In contrast, leveraging the parametric pruner network, our method achieves superior performance with reduced running time and is more adaptable to diverse development scenarios.

Table 6: Comparison of cost time, computational complexity and CLIP-Score between Random Search and GA search strategies on Stable Diffusion 1.5.

Method	Cost GPU Hours ↓	Pruning Ratio	MACs ↓	CLIP Score ↑
PLMS-50 steps	-	-	16.94T	30.30
Random Search	25	0.80	2.96T	28.73
GA Search	25	0.80	3.34T	29.37
Ours	2.3	0.80	3.38T	**30.29**
Random Search	24	0.85	2.90T	27.22
GA Search	24	0.85	2.73T	28.61
Ours	2.2	0.85	2.54T	**29.84**
Random Search	23	0.90	1.94T	24.07
GA Search	23	0.90	2.04T	25.14
Ours	2.2	0.90	1.69T	**28.72**

Qualitative Analysis of Increased Prune Ratio. In Figure 3, we visualize the generated images as we increase the pruning ratio. With the increase in pruning ratio, the model's inference speed significantly improves, allowing us to achieve up to a fourfold increase in inference speed. However, as the pruning ratio increases, some patterns in the image content deviate from those in the original images. Nevertheless, our main objects in the figures consistently adhere to the textual conditions. Subtle changes in background details typically do not compromise image quality, as quantitatively analyzed in Table 1.

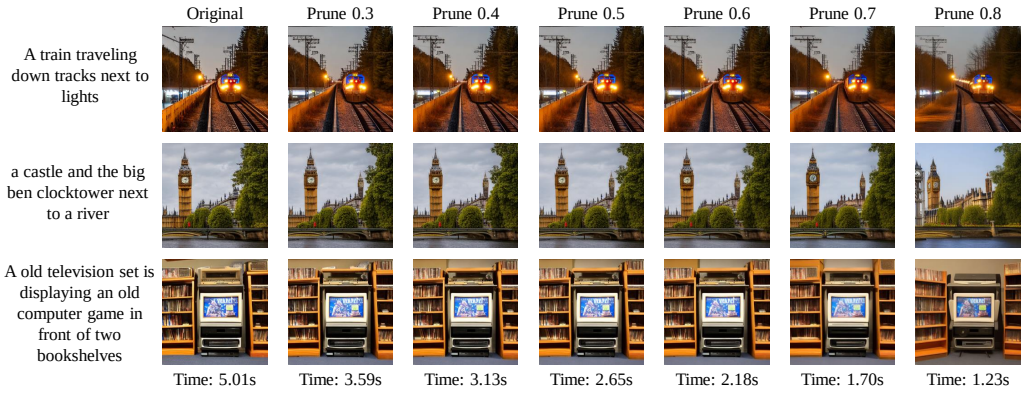


Figure 3: Visualization of generated images. It shows evolving patterns as pruning ratios increase. Despite these changes, main objects in the images remain consistent with the textual conditions.

5 Conclusion

This work explores resolving diffusion accelerating tasks by reducing redundant feature calculations across adjacent timesteps. We present a novel diffusion pruning framework and cast the model pruning process as a SubNet search problem. Our approach introduces a plugin pruner network that identifies an optimal SubNet through few-step gradient optimization. Results on a wide range of Stable Diffusion (SD) and DiT series models verify the effectiveness of our method. We achieve a 4.4× speedup on Stable Diffusion 1.5 and effectively prune the DiT model with few step optimizations.

https://doi.org/10.52202/079017-2940

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 33:6840–6851, 2020.

[2] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10684–10695, 2022.

[3] 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.

[4] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, pages 3836–3847, 2023.

[5] Chengyou Jia, Minnan Luo, Zhuohang Dang, Guang Dai, Xiaojun Chang, Mengmeng Wang, and Jingdong Wang. Ssmg: Spatial-semantic map guided diffusion model for free-form layout-to-image generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 2480–2488, 2024.

[6] Xi Chen, Lianghua Huang, Yu Liu, Yujun Shen, Deli Zhao, and Hengshuang Zhao. Anydoor: Zero-shot object-level image customization. *arXiv preprint arXiv:2307.09481*, 2023.

[7] 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.

[8] 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.

[9] Nupur Kumari, Bingliang Zhang, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Multi-concept customization of text-to-image diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1931–1941, 2023.

[10] Xiaofan Li, Yifu Zhang, and Xiaoqing Ye. Drivingdiffusion: Layout-guided multi-view driving scene video generation with latent diffusion model. *arXiv preprint arXiv:2310.07771*, 2023.

[11] Ruiyuan Gao, Kai Chen, Enze Xie, Lanqing Hong, Zhenguo Li, Dit-Yan Yeung, and Qiang Xu. Magicdrive: Street view generation with diverse 3d geometry control. *arXiv preprint arXiv:2310.02601*, 2023.

[12] Kai Chen, Enze Xie, Zhe Chen, Lanqing Hong, Zhenguo Li, and Dit-Yan Yeung. Integrating geometric control into text-to-image diffusion models for high-quality detection data generation via text prompt. *arXiv: 2306.04607*, 2023.

[13] Jiacheng Chen, Ruizhi Deng, and Yasutaka Furukawa. Polydiffuse: Polygonal shape reconstruction via guided set diffusion models. *ArXiv*, abs/2306.01461, 2023.

[14] Zhendong Wang, Jianmin Bao, Wengang Zhou, Weilun Wang, Hezhen Hu, Hong Chen, and Houqiang Li. Dire for diffusion-generated image detection. *arXiv preprint arXiv:2303.09295*, 2023.

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

[16] Bo-Kyeong Kim, Hyoung-Kyu Song, Thibault Castells, and Shinkook Choi. Bk-sdm: Architecturally compressed stable diffusion for efficient text-to-image generation. *ICML Workshop on Efficient Systems for Foundation Models (ES-FoMo)*, 2023.

[17] Youngwan Lee, Kwanyong Park, Yoohrim Cho, Yong Ju Lee, and Sung Ju Hwang. Koala: Self-attention matters in knowledge distillation of latent diffusion models for memory-efficient and fast image synthesis. *arXiv preprint arXiv:2312.04005*, 2023.

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

[19] Simian Luo, Yiqin Tan, Suraj Patil, Daniel Gu, Patrick von Platen, Apolinário Passos, Longbo Huang, Jian Li, and Hang Zhao. Lcm-lora: A universal stable-diffusion acceleration module, 2023.

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

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

[22] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[23] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

[24] 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.

[25] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models, 2023.

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

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

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

[29] 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.

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

[31] David Berthelot, Arnaud Autef, Jierui Lin, Dian Ang Yap, Shuangfei Zhai, Siyuan Hu, Daniel Zheng, Walter Talbott, and Eric Gu. Tract: Denoising diffusion models with transitive closure time-distillation. *arXiv preprint arXiv:2303.04248*, 2023.

[32] 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.

[33] Yefei He, Luping Liu, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. Ptqd: Accurate post-training quantization for diffusion models, 2023.

[34] Sejun Park, Jaeho Lee, Sangwoo Mo, and Jinwoo Shin. Lookahead: A far-sighted alternative of magnitude-based pruning. *arXiv preprint arXiv:2002.04809*, 2020.

[35] Xin Dong, Shangyu Chen, and Sinno Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. *Advances in neural information processing systems*, 30, 2017.

[36] Victor Sanh, Thomas Wolf, and Alexander Rush. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in neural information processing systems*, 33:20378–20389, 2020.

[37] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets, 2017.

 https://doi.org/10.52202/079017-2940

[38] Sara Elkerdawy, Mostafa Elhoushi, Abhineet Singh, Hong Zhang, and Nilanjan Ray. To filter prune, or to layer prune, that is the question. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[39] Xiaohan Ding, Guiguang Ding, Yuchen Guo, and Jungong Han. Centripetal sgd for pruning very deep convolutional networks with complicated structure. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4943–4953, 2019.

[40] Liyang Liu, Shilong Zhang, Zhanghui Kuang, Aojun Zhou, Jing-Hao Xue, Xinjiang Wang, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Group fisher pruning for practical network compression. In *International Conference on Machine Learning*, pages 7021–7032. PMLR, 2021.

[41] Ji Liu, Dehua Tang, Yuanxian Huang, Li Zhang, Xiaocheng Zeng, Dong Li, Mingjie Lu, Jinzhang Peng, Yu Wang, Fan Jiang, Lu Tian, and Ashish Sirasao. Updp: A unified progressive depth pruner for cnn and vision transformer, 2024.

[42] Minjia Zhang and Yuxiong He. Accelerating training of transformer-based language models with progressive layer dropping. *Advances in Neural Information Processing Systems*, 33:14011–14023, 2020.

[43] Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning*, pages 2793–2803. PMLR, 2021.

[44] Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33:9782–9793, 2020.

[45] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[46] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[47] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[48] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[49] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[50] 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.

[51] 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.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[53] Zijie J. Wang, Evan Montoya, David Munechika, Haoyang Yang, Benjamin Hoover, and Duen Horng Chau. DiffusionDB: A large-scale prompt gallery dataset for text-to-image generative models. *arXiv:2210.14896 [cs]*, 2022.

[54] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2(3):5, 2022.

[55] 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.

[56] 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.

[57] 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.

[58] 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.

[59] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, Steven Liu, William Berman, Yiyi Xu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models, 2023.

[60] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, et al. Pixart-a: Fast training of diffusion transformer for photorealistic text-to-image synthesis. *arXiv preprint arXiv:2310.00426*, 2023.

[61] Jaemin Cho, Abhay Zala, and Mohit Bansal. Dall-eval: Probing the reasoning skills and social biases of text-to-image generation models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3043–3054, 2023.

[62] Ranjita Naik and Besmira Nushi. Social biases through the text-to-image generation lens. In *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, pages 786–808, 2023.

[63] Candace Ross, Boris Katz, and Andrei Barbu. Measuring social biases in grounded vision and language embeddings. *arXiv preprint arXiv:2002.08911*, 2020.

[64] Siwei Lyu. Deepfake detection: Current challenges and next steps. pages 1–6, 2020.

 https://doi.org/10.52202/079017-2940

A Memory Optimization Details

Gradient Checkpointing. Due to the multi-step Markovian nature of sampling in diffusion models, updating the entire sampling process using gradient accumulation incurs significant memory costs, even with a batch size of 1. To mitigate this issue, we employ gradient checkpointing and half-precision floating-point training to reduce memory consumption. The core idea behind gradient checkpointing is to selectively preserve a portion of activation values during forward propagation, discarding the rest. During backpropagation, the gradients of the discarded activation values are computed using the saved gradients of the preserved nodes, effectively reducing memory usage. Additionally, we use gradient accumulation, wherein gradients computed over multiple iterations are accumulated and then backpropagated in a single batch for parameter updates, thus allowing for larger batch sizes under limited memory usage.

B Pseudo Code

Here, we show the details of our proposed post-process algorithm via pseudo code as followings.

Algorithm 1 Diffusion Pruner

Input: A pretrained diffusion model M, importance scores S, a pruning ratio p
Output: The pruned diffusion model M^*

left $\leftarrow 0.0$
right $\leftarrow 1.0$
while True **do**
    current $\leftarrow$ (left + right)$/2$
    $S^* \leftarrow S$
    for t in $[0, 1, 2, ..., T]$ **do**
        for each block score s in S^* **do**
            if $s <$ current **then**
                $s \leftarrow 0$
            end if
        end for
    end for
    update_scores_of_blocks (S^*) // remove dependent blocks to set them zeros.
    $p^*, M^* \leftarrow$ prune_diffusion_model(S^*, M) // obtain the pruned ratio and the pruned model.
    if abs$(p^* - p) < 0.0125$ **then**
        break
    else if $p^* < p$ **then**
        left $\leftarrow$ current
    else
        right $\leftarrow$ current
    end if
    $\mathcal{T} \leftarrow \mathcal{T}/2$
end while
return M^*

C Additional Experiment Results

Figure 4: Visualization of DiT model generated images: samples using DDIM-250 steps (uplink) and pruned 60% MACs (downlink). The speedup ratio here is $2.4\times$.

C.1 More Qualitative Results

Comparison with DiT Baselines. We provide the original unpruned DiT model and a version pruned by 0.6 ratio to generate comparison images in Figure 4. It can be observed that the plots generated by the pruned model are almost identical to those produced by the original model. Although there are slight differences in details, such as the appearance of the dog's eyes, these do not significantly affect the overall image quality.

Comparison with SD Baselines. We provide qualitative comparisons with the SD baseline and DeepCache, as shown in Figure 5. Our method demonstrates superior image-text consistency and image quality compared to existing methods.

Pruning Gate Visualization. Our method exhibits a specific pattern of pruning ratios with respect to the timesteps. As shown in Figure 6, fewer blocks are pruned during the middle denoising stage (approximately between steps 65 and 150), as this is when the image content is rapidly being generated. Conversely, the pruning ratio in the latter stage is higher since the content has already taken shape.

Feature Similarity Analysis. Recent studies have confirmed feature similarity across adjacent time steps [22, 29]. We also conducted an analysis of feature similarity between adjacent steps in fast samplers. Specifically, we sampled 200 images from the COCO2017 validation set and calculated the average cosine similarity between the features of the penultimate upsampling block across all steps for two typical fast samplers, resulting in a similarity matrix, shown in Figure 7. The heatmap in Figure 7 highlights the high degree of similarity between features at consecutive time steps.

C.2 More Quantitative Results

More Ablations. We conducted an ablation study on α and present the results in Table 7. Our method achieves the best performance when $\alpha = 1.0$. we also conducted an ablation study on γ. Without γ, pruning 80% on SD1.5 resulted in a CLIP score of 29.50 (w/ γ: 30.29).

More Baseline Comparison. We also evaluated our method on PixArt-α [60], achieving excellent pruning results, as shown in Table 8 below. Our method exhibits minimal performance loss on PixArt-α with a 0.4 pruning ratio.

https://doi.org/10.52202/079017-2940

Figure 5: A qualitative comparison with existing methods is provided. We compare our method (prune 0.75) with DeepCache (N=4).

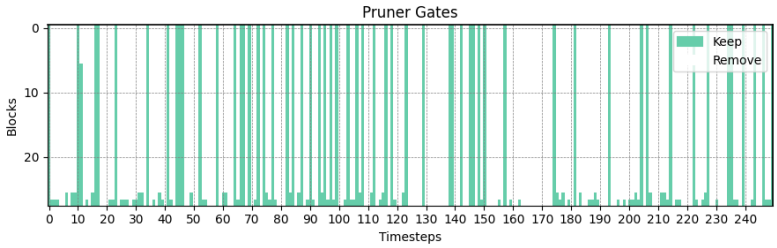


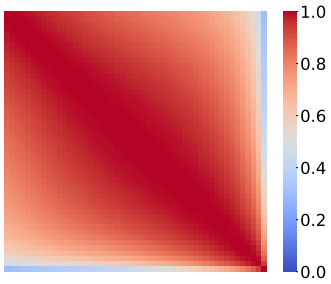
Figure 6: The visualization results of the pruning gates for DiT-XL/2 at 250 steps with a pruning ratio of 0.75.

D Limitations

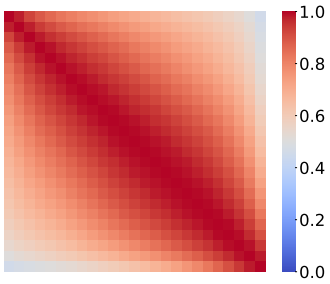
A limitation of our method arises from its training process of the pruner network. Our method necessitates tuning an additional pruner network for the pre-trained diffusion model. This may entail users investing additional time when adapting our method to specific diffusion models. For example, we train DiP-GO for SD-1.5 on a single AMD Instinct MI250 GPU for $\sim$ 2.5 hours. However, we note that the introduced time is small compared to training a lightweight diffusion model. Besides, same as existing work, our method struggles to maintain performance with extremely high pruning ratios, presenting a challenge for deploying diffusion models in scenarios with severely limited computational resources.

E Social Impact

Generative models have demonstrated promising results in content generation [50, 60, 2]. However, due to the high inference costs, current methods struggle to achieve rapid application and deployment. Our approach introduces an efficient acceleration method for diffusion models, enabling nearly lossless speedup. Moreover, our method does not require retraining of the pretrained models and is compatible with various diffusion models, making it highly generalizable. This makes it suitable for rapid deployment of generative models on mobile and edge devices.



(a) DDIM (SD-1.5 w/ 50 steps)



(b) DPM (SD-2.1 w/ 25 steps)

Figure 7: Feature similarity across adjacent time steps in fast samplers.

Table 7: Comparison of different α values. Pruning experiments with 80% pruning ratio were conducted on COCO2017 validation using SD-1.5.

α	0.1	0.5	1.0	2.0
CLIP Score↑	29.77	29.93	**30.29**	30.17

Table 8: Comparison with a 20-step DPM-Solver sampler for diffusion transformer model. We evaluate the effectiveness of our methods on COCO2017 validation set.

Method ↓	**MACs** ↓	**Speedup** ↑	**CLIP Score** ↑
PixArt-α w/ 20-step DPM	85.65 T	1.0 ×	30.43
Pruned-0.4 (Ours)	51.39 T	1.6 ×	30.41

Nevertheless, since generative models are pretrained on large-scale internet datasets, the data they generate may contain inherent social biases and stereotypes [61–63]. Additionally, there is a risk of misuse, such as in the creation of DeepFakes [64], which could pose significant social harm. While reducing the usage cost, it is crucial to prevent the low-cost generative models from being misused, leading to negative societal impacts. Therefore, it is necessary to establish relevant laws and regulations, create a well-regulated community environment, and provide guidelines to ensure responsible dissemination and use of generative models.

https://doi.org/10.52202/079017-2940

NeurIPS Paper Checklist

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 abstract provides a concise summary of the main findings and contributions of the paper, while the introduction elaborates on the problem statement and research objectives, thereby clarifying the contributions.

   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: In Limitation Section in Appendix D, we expound upon the limitations of the work conducted and provide a brief discussion thereof.

   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: [No]

Justification: None.

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: In Section 4.1, we introduced the details of experimental setup and model training to ensure reproducibility.

   Guidelines:

   - 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**

https://doi.org/10.52202/079017-2940

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: [No]

Justification: The code will be released after it successfully passes our company's internal review.

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: In Section 4.1, we introduced the details of experimental setup and model training and testing.

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: [No]

Justification: The experiments conducted in our paper do not involve the use of error bars or statistical significance analysis, thus this aspect is not applicable to our study.

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: For our experiments, we furnished detailed specifications of the GPU models used along with their corresponding tasks. Furthermore, we included specific information regarding the model training batch size and the number of training iterations.

   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.
   - 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 have carefully reviewed the NeurIPS Code of Ethics and adhere to its principles.

   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 both potential positive societal impacts and negative societal impacts of the work performed in Appendix.

https://doi.org/10.52202/079017-2940

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: Our 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, such as code, data, or models, used in the paper, are properly credited. Additionally, the license and terms of use associated with these assets are explicitly mentioned and respected in accordance with ethical and legal standards.

    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: [NA]

    Justification: Our paper does not release new 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)?

    Answer: [NA]

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

    Guidelines:

https://doi.org/10.52202/079017-2940

- 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.