EFFORTLESS EFFICIENCY: LOW-COST PRUNING OF DIFFUSION MODELS

Yang Zhang
National University of Singapore
yangzhang@u.nus.edu

Er Jin
RWTH Aachen University
Er.Jin@lfb.rwth-aachen.de

Yanfei Dong
PayPal
yanfei.dong43@gmail.com

Ashkan Khakzar
University of Oxford
ashkan.khakzar@@eng.ox.ac.uk

Philip Torr
University of Oxford
philip.torr@@eng.ox.ac.uk

Johannes Stegmaier
RWTH Aachen University
Johannes.Stegmaier@lfb.rwth-aachen.de

Kenji Kawaguchi
National University of Singapore
kenji@comp.nus.edu.sg

ABSTRACT

Diffusion models have achieved impressive advancements in various vision tasks. However, these gains often rely on increasing model size, which escalates computational complexity and memory demands, complicating deployment, raising inference costs, and causing environmental impact. While some studies have explored pruning techniques to improve the memory efficiency of diffusion models, most existing methods require extensive retraining to retain the model performance. Retraining a modern large diffusion model is extremely costly and resource-intensive, which limits the practicality of these methods. In this work, we achieve low-cost diffusion pruning without retraining by proposing a model-agnostic structural pruning framework for diffusion models that learns a differentiable mask to sparsify the model. To ensure effective pruning that preserves the quality of the final denoised latent, we design a novel end-to-end pruning objective that spans the entire diffusion process. As end-to-end pruning is memory-intensive, we further propose time step gradient checkpointing, a technique that significantly reduces memory usage during optimization, enabling end-to-end pruning within a limited memory budget. Results on state-of-the-art U-Net diffusion models SD-XL and diffusion transformers (FLUX) demonstrate that our method can effectively prune up to 20% parameters with minimal perceptible performance degradation—and notably, without the need for model retraining. We also showcase that our method can still prune on top of time step distilled diffusion models.

1 INTRODUCTION

Recently, diffusion models have made remarkable progress in various vision tasks, including text-to-image generation Brooks et al. (2023); Rombach et al. (2022); Ramesh et al. (2022); Nichol et al. (2022), image-inpainting Saharia et al. (2022a); Lugmayr et al. (2022); Corneanu et al. (2024),

Figure 1: We demonstrate that it is possible to prune state-of-the-art diffusion models up to 20%, *without retraining* after pruning, while maintaining high-quality generated images. Our pruning enables the deployment of SDXL on an 8GB GPU and FLUX on a 24GB GPU.

super-resolution Saharia et al. (2022b); Li et al. (2022), and video generation Ho et al. (2022); Luo et al. (2023); Singer et al. (2022). This progress has been accompanied by architectural evolution, from the U-Net-based Stable Diffusion 1 (SD1) Rombach et al. (2022) to the larger Stable Diffusion XL (SDXL) Podell et al. (2024), the transformer-based Stable Diffusion 3 (SD3), and most recently, the FLUX model Esser et al. (2024); Labs (2024). Although the development of models partially stems from the improvement of training techniques and architectural innovations, the performance boost is largely attributed to model size scaling. The most recent FLUX model, with 12 billion parameters, is about 13 times larger than the SD1 (860 million) developed just two years ago Labs (2024). The rapid growth in size has caused potential issues: larger models demand larger GPU and more computation during inference, limit deployment on smaller computation platforms, and substantially increase the carbon footprint.

Because of the potential issues induced by larger diffusion models, prior methods have explored ways to reduce model size and computation, Fang et al. (2023), including model distillation Meng et al. (2023); Huang et al. (2024) and model pruning Fang et al. (2023). Compared to distillation, pruning usually induces less training overhead. However, pruning diffusion models present considerable challenges, as even minor changes in continuous latent variables can degrade image quality. Consequently, pruning diffusion models often require retraining to restore performance. Fang et al. (2023) estimate that diffusion model compression can demand 10% to 20% of the original training cost. For large models, this retraining burden is substantial. For instance, training Stable Diffusion 2 requires approximately $200,000$ GPU hours on 40GB A100 GPUs AI (2022), costing nearly $\$1$M on AWS Amazon Web Services (2024). Retraining a compressed version of SD2 can, therefore, consume up to $40,000$ GPU hours. Pruning larger models, such as SDXL and FLUX, would require even more resources, posing significant challenges for resource-constrained organizations.

In this work, we show for the first time, to the best of our knowledge, that a significant amount of parameters can still be removed without retraining the pruned model, as illustrated in Figure 1. For effective pruning that maintains generation ability throughout the denoising process, we formulate an end-to-end pruning objective to preserve the final denoised latent at the last denoising step given an initial noise input. This approach has the advantage over the alternative of minimizing the noise prediction difference between the original and the pruned model at each time step separately, which can introduce cumulative errors throughout the denoising steps and eventually lead to substantial quality degradation.

However, during end-to-end pruning, backpropagation requires storing all intermediate outputs across all diffusion steps, leading to significant memory demands. For instance, performing end-

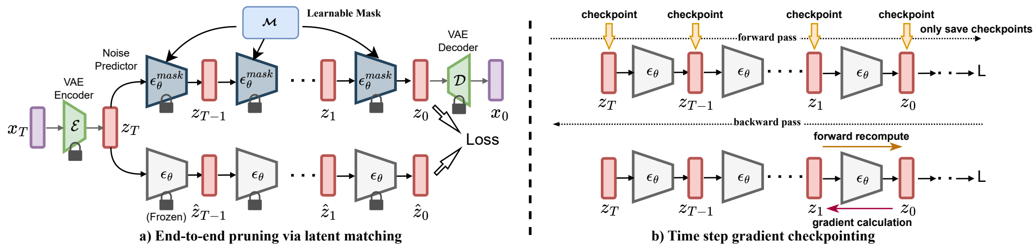


Figure 2: **Overview of end-to-end pruning framework and time step gradient checkpointing**. In a), end-to-end pruning learns a mask that applies to all denoising steps, reduces model size while preserving the final denoised latent for semantic integrity. In b), only checkpoints are stored during the forward pass. During the backward pass, we first recompute the intermediates between checkpoints at each step, then perform gradient calculation. Therefore, memory usage is reduced by T times with only one additional forward pass.

to-end pruning on SDXL requires approximately 1400 GB memory–equivalent to the capacity of 15 NVIDIA H100 GPUs. To address this, we adopt gradient checkpointing Chen et al. (2016) and propose time step gradient checkpointing, which reduces the memory usage for SDXL from 1400GB to under 30GB, with a minor overhead of an additional forward pass. An overview of end-to-end pruning and gradient checkpointing is shown in Figure 2.

Recent studies have shown that distilling the model into a student model with fewer generation steps can greatly speed up the inferenceSalimans & Ho (2022); Meng et al. (2023). We showcase that our pruning approach can be applied on top of a time-step distilled model, thereby further reducing the runtime and memory consumption of the distilled model. To the best of our knowledge, this is the first work to successfully prune time step distilled diffusion models.

Overall, our contribution is summarized as follows: **(1)** We introduce **EcoDiff**, a model-agnostic, end-to-end structural pruning framework for diffusion models that learns a differentiable neuron mask, enabling efficient pruning across various architectures. **(2)** We develop a novel diffusion step gradient checkpointing technique, significantly reducing memory requirements for end-to-end pruning to be feasible with lower computational resources. **(3)** We conduct extensive evaluations across U-Net diffusion models and diffusion transformers, demonstrating that our method can effectively prune 20% of model parameters without necessitating retraining. Additionally, we show that our approach is orthogonal to diffusion step distillation.

2 Methodology

2.1 End-to-End Pruning Objective

Conventional diffusion model pruning approaches consider only the changes in noise prediction at each denoising step Fang et al. (2023). However, this can cause the error to accumulate throughout the entire denoising process and eventually lead to significant distortion. In contrast, we formulate an end-to-end pruning objective that considers the entire denoising process. The pruning objective is to learn a set of masking parameters $\boldsymbol{\mathcal{M}} = [\boldsymbol{\mathcal{M}}_1, ..., \boldsymbol{\mathcal{M}}_N]$ for N target layer to minimize the difference between the final denoised latent z_0 generated by the unmasked latent denoising backbone ϵ_θ and the predicted $\hat{z_0}$ generated by the masked backbone $\epsilon_\theta^{\text{mask}}$ under the same text prompt x and initial noisy latent z_T. To formulate the end-to-end pruning objective, we first define the complete denoising process as $\mathcal{F}$ based on Equation 6:

$$z_0 = f(f(\dots f(z_i, y, 1), y, 2), \dots, y, T) = \mathcal{F}(z_T, y, T) \tag{1}$$

We omit T for the subsequent discussion as it is a constant. The denoising process iteratively refines and denoises the latent representation, starting from the initial time step T and proceeding to $t = 0$, resulting in the final denoised latent z_0. Based on Equation 1, we summarize the pruning objective as follows:

$$\arg\min_{\boldsymbol{\mathcal{M}}} \mathbb{E}_{z_T, y \sim \mathcal{C}} \left[\|\mathcal{F}_{\epsilon_\theta}(z_T, y) - \mathcal{F}_{\epsilon_\theta^{\text{mask}}}(z_T, y, \boldsymbol{\mathcal{M}})\|_2 \right] + \beta \|\boldsymbol{\mathcal{M}}\|_0 \tag{2}$$

where $z_T \sim \mathcal{N}(0,1)$ is the initial noise, $\mathcal{C} = \{y^i\}_{i=1}^{N}$ is the dataset consisting of conditions for conditioned generation, $\|\boldsymbol{\mathcal{M}}\|_0 = \sum_{j=1}^{|\boldsymbol{\mathcal{M}}|} \mathbb{I}(\mathcal{M}_j \neq 0)$ denotes as L_0 norm for sparsity, and β is regularization coefficient of the sparsity regularization. Algorithm 1 shows the procedure of end-to-end pruning using gradient-based optimization.

2.2 Continuous Relaxation of Discrete Masking

Our pruning objective in Equation 2 involves the calculation of L_0 norm, which is not differentiable. Hence, we adopt continuous relaxation of the sparse optimization by applying hard-discrete sampling originally proposed by Louizos et al. (2018). For each continuous masking variable $\hat{\mathcal{M}} \in [0,1]$, we adapt the hard-discrete sampling as follows: $s = \sigma\left((\log(u+\delta) - \log(1-u+\delta) + \lambda)/\alpha\right)$, $\bar{s} = s(\zeta - \gamma) + \gamma$, $\hat{\mathcal{M}} = \min(1, \max(0, \bar{s}))$, where σ is the sigmoid function, $u \sim U(0,1)$ is sampled from a uniform distribution, λ controls the likelihood of masking, α is the temperature parameter, ζ and γ stretch s. Additionally, δ controls the steepness, where a higher value of δ closely resembles a step function, and a lower value suggests a distribution closer to a sigmoid function. Figure 4 illustrates the behavior of the hard-discrete distribution compared to the standard sigmoid function. Hence, we can instead learn to optimize $\boldsymbol{\lambda} \in \mathbb{R}^{|\boldsymbol{\mathcal{M}}|}$. The first term in Equation 2 can be formatted as a reconstruction loss $\mathcal{L}_E$:

$$\mathcal{L}_E(\boldsymbol{\lambda}) = \sum_{y}\sum_{z_T} \|\mathcal{F}_{\epsilon_\theta}(z_T, y) - \mathcal{F}_{\epsilon_\theta^{\text{mask}}}(z_T, y, \hat{\boldsymbol{\mathcal{M}}}(\boldsymbol{\lambda}))\|_2 \tag{3}$$

The L_0 complexity loss $\mathcal{L}_0$ given the hard-discrete parameter $\boldsymbol{\lambda}$ can be described as:

$$\mathcal{L}_0(\boldsymbol{\lambda}) = \sum_{j=1}^{|\boldsymbol{\lambda}|} \text{Sigmoid}\left(\log \lambda_j - \beta \log \frac{-\gamma}{\zeta}\right) \approx C\|\boldsymbol{\lambda}\|_1, \tag{4}$$

for a constant C. Detailed derivation is in Appendix D. Finally, the end-to-end pruning loss for $\boldsymbol{\lambda}$ is formulated as: $\mathcal{L}(\boldsymbol{\lambda}) = \mathcal{L}_E(\boldsymbol{\lambda}) + \beta\mathcal{L}_0(\boldsymbol{\lambda}) = \mathcal{L}_E(\boldsymbol{\lambda}) + \beta\|\boldsymbol{\lambda}\|_1$. After learning the continuous mask control variable $\boldsymbol{\lambda}$, we obtain the final discrete mask $\boldsymbol{\mathcal{M}}$ by applying a threshold τ: $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}) = \mathbb{I}(\boldsymbol{\lambda} > \tau)$, where τ is selected to achieve a desired sparsity ratio.

2.3 Time Step Gradient Checkpointing

To perform the end-to-end pruning shown in Algorithm 1, the backpropagation needs to traverse all diffusion steps. This necessitates storing all intermediate variables across each step, leading to a substantial increase in memory usage during mask optimization. To address this memory challenge, we propose a novel time-step gradient checkpointing technique.

Traditional gradient checkpointing reduces memory usage by storing selected intermediate outputs and recomputing forward passes between checkpoints Chen et al. (2016); Gruslys et al. (2016). For a model with N layers, aggressive checkpointing achieves $O(1)$ memory complexity but increases the runtime complexity from $O(N)$ to $O(N^2)$. A more balanced approach places checkpoints every $\sqrt{N}$ layers, yielding sublinear memory savings at the cost of an additional forward pass, thereby maintaining $O(N)$ runtime complexity. However, traditional checkpointing only stores intermediate values within a single forward pass, limiting its utility for diffusion models, which require multiple model forward passes across time steps. To address this, we propose *time step gradient checkpointing*, which stores intermediate denoised latent $\hat{z}$ after each denoising step. This approach reduces memory demands across diffusion steps while preserving computation efficiency. We present Algorithm 2 to show how to apply time step gradient checkpointing to calculate the loss gradient w.r.t. the mask control variable $\boldsymbol{\lambda}$.

3 Experiments

3.1 Setup

Model: we prune state-of-the-art (SOTA) latent diffusion models, SDXL, and FLUX, representing various latent diffusion architectures Rombach et al. (2022); Podell et al. (2024); Labs (2024). For

Models	Pruning Ratio	Methods	GFLOP ↓	#Params ↓	MS COCO			Flickr 30K		
					FID ↓	CLIP ↑	SSIM ↑	FID ↓	CLIP ↑	SSIM ↑
SDXL U-Net Architecture	0%	Original	478K	2.6B	35.50	0.31	1	49.34	0.34	1
	10%*	DeepCache	430K	2.6B	36.63	0.31	0.73	49.37	0.34	0.76
	10%	DiffPruning		2.3B	108.96	0.22	0.31	97.60	0.26	0.33
	10%	**EcoDiff** (Ours)		2.3B	33.75	0.31	0.53	41.35	0.34	0.52
	20%*	DeepCache	382K	2.6B	36.66	0.31	0.81	50.25	0.34	0.83
	20%	DiffPruning		2.1B	404.87	0.05	0.26	438.82	0.04	0.27
	20%	**EcoDiff** (Ours)		2.1B	34.41	0.31	0.50	42.84	0.33	0.53
FLUX (Step-Distilled) DiT Architecture	0%	Original	281K	11.9B	30.99	0.33	1	39.70	0.35	1
	10%*	DeepCache	253K	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	10%	DiffPruning		10.7B	34.63	0.32	0.26	41.16	0.34	0.25
	10%	**EcoDiff** (Ours)		10.7B	32.16	0.32	0.37	42.56	0.34	0.377
	15%*	DeepCache	239K	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	15%	DiffPruning		10.1B	106.56	0.28	0.22	120.52	0.30	0.22
	15%	**EcoDiff** (Ours)		10.1B	31.76	0.30	0.36	43.25	0.33	0.36

Table 1: **Quantitative analysis of pruned diffusion models on MS COCO and Flickr 30K datasets.** Our pruned model achieves comparable or noticeable better FID scores (on SDXL) than the original diffusion model, demonstrating high semantic fidelity and image quality, even without explicitly preserving structural similarity. For DeepCache, we consider speedup instead of pruning ratio (marked with $*$). Lower FID and higher CLIP scores indicate improved performance. The SSIM score is calculated between images generated by the pruned and original models. GFLOP is calculated with 5 denoising steps for FLUX and 50 for SDXL.

FLUX, we employ the time-step distillation model, *FLUX.1-schnell*. **Pruning details**: we only use text prompts from GCC3M Sharma et al. (2018) for training. **Baselines**: we compare our method with the SOTA diffusion pruning methods, DiffPruning and DeepCache Ma et al. (2024b); Fang et al. (2023). For DeepCache, we tune for acceleration, since it cannot compress the model. **Evaluation metrics**: we select Fréchet Inception Distance (FID) Heusel et al. (2017), CLIP score Radford et al. (2021), and Structural Similarity Index Measure (SSIM) Wang et al. (2004). We evaluate SSIM by comparing the pruned and original (unmasked) models. We use pretrained CLIP encoder *ViT-B-16* to calculate the CLIP score. For quantitative evaluation, we use the MS COCO and Flickr 30k datasets Lin et al. (2014); Young et al. (2014), randomly selecting 5,000 image-caption pairs from each. Additionally, we measure computation in Giga Floating Point Operation (GFLOP) and the total number of parameters in the pruned models. **Miscellaneous:** To ensure reproducibility, we use the Hugging Face Diffusers and Accelerator library Face (2021; 2022). All experiments are conducted on a single NVIDIA H100 GPU with 80G VRAM. More details in Appendix E

3.2 Main Results

To show the advantage of EcoDiff, we compare our method with several SOTA compression methods Fang et al. (2023); Ma et al. (2024b). We conduct quantitative evaluations as shown in Table 1. For FLUX, at pruning ratios of 10% and 15%, the FID scores on the MS COCO dataset increase by only 1.77 and 0.77, respectively. For the pruned SDXL model, the FID scores decrease by 1.75 and 1.09 at pruning ratios of 10% and 20%, respectively, indicating an improvement in generative quality. Similarly, for the Flickr30K, we observed that our pruned SDXL model shows a noticeable decrease in FID scores, with drops of 8 and 6.7 at pruning ratios of 10% and 20%, respectively. For the pruned FLUX model, we observed only a slight increase in FID scores, with values of 2.86 and 3.55 at pruning ratios of 10% and 15%, respectively. Nonetheless, it remains superior to DiffPruning across most FID scores. Notably, DeepCache is not applicable to FLUX due to its exclusive compatibility with the U-Net architecture. We also observe low SSIM scores for all our pruned models compared to conventional pruning methods Fang et al. (2023), with all values below 0.55. This suggests that EcoDiff can generate high-quality images without mimicking the original model. As shown in Figure 6, our generated images' semantic fidelity and fine-grained quality are noticeably superior to those of the baseline methods. Additionally, we observe that sometimes the pruned models produce images with better semantic meaning than the original diffusion model, which suggests the potential for our framework to investigate or even improve the semantic understanding of diffusion models. Additional results on SD2 are in Appendix H. Results of using light retraining after pruning can be found in Appendix J.

4 Conclusion

In this work, we introduce **EcoDiff**, a structural pruning method for diffusion models that uses differentiable neuron masking. We design an end-to-end pruning objective to consider the generation ability across all denoising steps and preserve the final denoised latent instead of final denoised images. With our novel pruning objective, we create a more flexible pruning target and achieve effective pruning that maintains image quality and semantics. To address the high memory demands of end-to-end pruning with gradient optimization, we propose time step gradient checkpointing, which reduces memory usage by up to 50 times compared to standard training. Results on the most recent SDXL and FLUX models show that EcoDiff is model-agnostic and can effectively prune large diffusion models with a reasonable computing budget. Furthermore, our approach removes up to 20% of model parameters without performance loss or the need for retraining—a substantial improvement over prior works. Additionally, we show that we can prune on top of time step distilled models, further reducing their latency and deployment requirement. Overall, our work establishes a new standard for diffusion model pruning, highlighting the high parameter redundancy in diffusion models. Future work can build on our work to achieve a higher compression level by incorporating retraining or other compression techniques.

REFERENCES

Stability AI. Stable diffusion v2 model card, 2022. URL `https://huggingface.co/stabilityai/stable-diffusion-2`.

Inc. Amazon Web Services. Aws pricing, 2024. URL `https://aws.amazon.com/pricing/`. Accessed: [13.11.2024].

Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *CVPR*, pp. 18392–18402, 2023.

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

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

Ciprian Corneanu, Raghudeep Gadde, and Aleix M Martinez. Latentpaint: Image inpainting in latent space with diffusion models. pp. 4334–4343, 2024.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *ICLR*, 2024.

Hugging Face. Accelerate: A library for quick, easy distributed and mixed precision training, 2021. URL `https://github.com/huggingface/accelerate`.

Hugging Face. Diffusers: State-of-the-art diffusion models for image and audio generation in pytorch, 2022. URL `https://github.com/huggingface/diffusers`.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. In *NeurIPS*, 2023.

Gongfan Fang, Hongxu Yin, Saurav Muralidharan, Greg Heinrich, Jeff Pool, Jan Kautz, Pavlo Molchanov, and Xinchao Wang. Maskllm: Learnable semi-structured sparsity for large language models. *NeurIPS*, 2024.

Audrunas Gruslys, Rémi Munos, Ivo Danihelka, Marc Lanctot, and Alex Graves. Memory-efficient backpropagation through time. *NeurIPS*, 29, 2016.

Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Lingjie Liu, and Joshua M Susskind. Boot: Data-free distillation of denoising diffusion models with bootstrapping. In *ICML 2023 Workshop on Structured Probabilistic Inference* $\{\backslash\&\}$ *Generative Modeling*, 2023.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

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. In *NeurIPS*, pp. 6626–6637, 2017.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

Yi-Ting Hsiao, Siavash Khodadadeh, Kevin Duarte, Wei-An Lin, Hui Qu, Mingi Kwon, and Ratheesh Kalarot. Plug-and-play diffusion distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13743–13752, 2024.

Tao Huang, Yuan Zhang, Mingkai Zheng, Shan You, Fei Wang, Chen Qian, and Chang Xu. Knowledge diffusion for distillation. *NeurIPS*, 36, 2024.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

Woosuk Kwon, Sehoon Kim, Michael W. Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. A fast post-training pruning framework for transformers. In *NeurIPS*, 2022a.

Woosuk Kwon, Sehoon Kim, Michael W Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. A fast post-training pruning framework for transformers. *NeurIPS*, 2022b.

Black Forest Labs. Flux. `https://blackforestlabs.ai/announcing-black-forest-labs/`, 2024. Accessed: [02.11.2024].

Haoying Li, Yifan Yang, Meng Chang, Shiqi Chen, Huajun Feng, Zhihai Xu, Qi Li, and Yueting Chen. Srdiff: Single image super-resolution with diffusion probabilistic models. *Neurocomputing*, 479:47–59, 2022.

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 *ECCV*, pp. 740–755, 2014.

Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. In *ICLR*, 2018.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *CVPR*, pp. 11461–11471, 2022.

Zhengxiong Luo, Dayou Chen, Yingya Zhang, Yan Huang, Liang Wang, Yujun Shen, Deli Zhao, Jingren Zhou, and Tieniu Tan. Videofusion: Decomposed diffusion models for high-quality video generation. In *CVPR*, pp. 10209–10218. IEEE Computer Society, 2023.

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.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *NeurIPS*, 2023.

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

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *CVPR*, pp. 15762–15772, 2024b.

Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *CVPR*, pp. 14297–14306, 2023.

Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob Mcgrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In *ICML*, pp. 16784–16804. PMLR, 2022.

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. In *ICLR*, 2024.

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. *ICML*, 2021.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

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

Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH*, pp. 1–10, 2022a.

Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4713–4726, 2022b.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *ICLR*, 2022.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Association for Computational Linguistics*, pp. 2556–2565, 2018.

Noam M. Shazeer. Glu variants improve transformer. *ArXiv*, abs/2002.05202, 2020. URL `https://api.semanticscholar.org/CorpusID:211096588`.

Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. In *ICLR*, 2022.

Junhyuk So, Jungwon Lee, and Eunhyeok Park. Frdiff: Feature reuse for universal training-free acceleration of diffusion models. *arXiv preprint arXiv:2312.03517*, 2023.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

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*, pp. 600–612, 2004.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. In *Transactions of the Association for Computational Linguistics*, pp. 67–78, 2014.

Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *CVPR*, pp. 9194–9203, 2018.

Yang Zhang, Yanfei Dong, and Kenji Kawaguchi. Investigating layer importance in large language models. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 469–479, 2024a.

Yang Zhang, Yawei Li, Xinpeng Wang, Qianli Shen, Barbara Plank, Bernd Bischl, Mina Rezaei, and Kenji Kawaguchi. Finercut: Finer-grained interpretable layer pruning for large language models. *arXiv preprint arXiv:2405.18218*, 2024b.

A Related Work

Efficient diffusion model: Several works focus on enhancing the efficiency of the diffusion models at inference time. DDIM formulates the diffusion processes with non-Markovian transformations, reducing the required diffusion steps for high-quality generation Song et al. (2021). The Latent Diffusion Model adopts a two-stage generation process, performing the diffusion process in latent space and decoding it back to the image domain using a Variational Autoencoder (VAE) Rombach et al. (2022); Kingma & Welling (2014). Besides, some prior works accelerate diffusion inference by sharing intermediate variables to reduce redundant computations Ma et al. (2024b;a); So et al. (2023). However, these methods are often limited to specific architectures and do not generalize well to newer diffusion models incorporating multi-modal attention. Early stopping methods attempt to terminate the diffusion process once satisfactory generation is reached, but can sacrifice fine-grained details Lyu et al. (2022). Distillation-based approaches train a student model that are either smaller or can generate with fewer diffusion steps Salimans & Ho (2022); Meng et al. (2023); Gu et al. (2023); Hsiao et al. (2024). Nevertheless, model distillation requires extensive retraining.

Model pruning: Pruning reduces model size, thereby lowering both memory requirements for loading the model and computation demands during inference. Recent advances in model pruning have predominantly focused on large language models (LLMs), where techniques such as unstructured and semi-structured pruning eliminate connections between neurons, and structured pruning targets neurons, attention heads, and layers Kwon et al. (2022a); Fang et al. (2024); Zhang et al. (2024b;a); Yu et al. (2018); Ma et al. (2023); Kwon et al. (2022b); Sun et al. (2023). In contrast, pruning methods for diffusion models remain relatively underexplored. Recent works have explored pruning diffusion models. Castells et al. (2024) perform occlusion-based pruning that calculates an importance score for occluding a part through exhaustive search, but its high complexity limits scalability to complex pruning scenarios with many candidates. Recently, Fang et al. (2023) uses gradient information as a proxy for neuron importance. Compared to the occlusion method, it scales to more complex pruning cases. Nevertheless, both methods require retraining to retain the model performance. Our optimization-based EcoDiff approach formulates a differentiable pruning objective to learn a neuron mask. As a result, our method allows effective pruning without retraining.

B Preliminaries

In this section, we briefly discuss some key designs of current diffusion models relevant to our approach.

Latent diffusion model. In this study, we focus on pruning latent diffusion models (LDMs). Leveraging an efficient low-dimensional latent space, LDMs enables faster and more resource-efficient image generation compared to traditional pixel-space models Rombach et al. (2022). The training objective for LDMs is to minimize a latent noise prediction loss:

$$\mathcal{L}_{\text{LDM}} := \mathbb{E}_{z\sim\mathcal{E}(x),\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(z_t, t)\|_2^2\right], \tag{5}$$

where latent z is obtained via an encoder $\mathcal{E}$ that encodes x from image space to latent space, t is uniformly sampled from $\{1, \dots, T\}$, and $\epsilon_\theta(z, t)$ denotes a denoising model, which can be a U-Net or a transformer model. The sampling process in LDMs iteratively reduces the noise in the initial noisy latent $z_T \sim \mathcal{N}(0, I)$, until reaching the final denoised latent z_0. This latent z_0 is then decoded via a pretrained decoder $\mathcal{D}$ to reconstruct image $\hat{\mathbf{x}} = \mathcal{D}(z_0)$. One denoising step can be summarized by $f(z_t, y, t)$ as follows:

$$\begin{aligned} z_{t-1} &= \frac{1}{\sqrt{\alpha_t}}\left(z_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon(z_t, t, y)\right) + \sigma_t \eta \\ &= f(z_t, y, t), \end{aligned} \tag{6}$$

where $t \in \{T, T-1, \dots, 1\}$, $\epsilon(z_t, t, y)$ represents the latent denoising prediction of z_{t-1} conditioned on the previous latent z_t, time step t, and input condition y. The terms α_t and $\bar{\alpha}_t$ are noise schedule parameters. The term σ_t denotes a controlled level of random noise, with $\eta \sim \mathcal{N}(0, \mathbf{I})$.

Transformer blocks in diffusion models. Transformer blocks are one of the major building blocks in U-Net diffusion models Rombach et al. (2022); Podell et al. (2024). In addition, recent diffusion transformers employ fully transformer-based architectures. This makes transformer blocks a

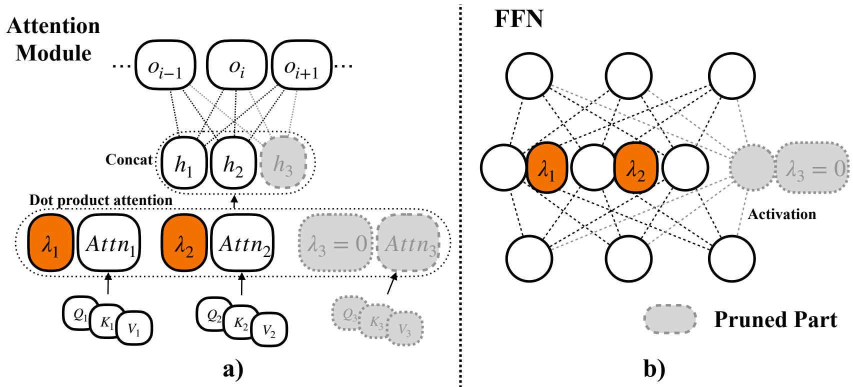


Figure 3: **Neuron masking on MHA and FFN in transformer blocks.** Here, a) illustrates masking within an attention module and b) shows masking within an FFN. λ is the mask variable.

primary target for pruning to improve efficiency. A transformer block consists of a multi-head attention (MHA) layer followed by a feed-forward network (FFN) Vaswani et al. (2017). The MHA with h attention heads is defined as,

$$\text{MHA}(Q, K, V) = (\text{attn}_1 \| \dots \| \text{attn}_h) W^o, \tag{7}$$

where each attn_i is a dot product attention for head i, which is computed as $\text{attn}_i = \text{softmax}\left(\frac{Q_i K_i^\top}{\sqrt{d_k}}\right) V_i$, with $Q_i = W_Q^i x$, $K_i = W_K^i x$, and $V_i = W_v^i x$ for an input x, $W_K^i \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_Q^i \in \mathbb{R}^{d_{\text{model}} \times d_q}$, $W_V^i \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{d_v \times d_{\text{model}}}$. Additionally, $\|$ denotes concatenation along the feature dimension across the attention heads. The FFN applied after MHA is defined as

$$\text{FFN}(x) = \sigma(xW_1 + b_1)W_2 + b_2, \tag{8}$$

where $W_1 \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$ and $W_2 \in \mathbb{R}^{d_{\text{ff}} \times d_{\text{model}}}$. $\sigma(\cdot)$ represents an activation function like GELU or GeGLU Hendrycks & Gimpel (2016); Shazeer (2020).

Structural Pruning via Neuron Masking. Structural pruning has one advantage over nonstructural pruning, as no special hardware support is required. This work focuses on pruning neurons in transformer blocks. To incorporate sparsity, we apply a learnable discrete pruning mask $\boldsymbol{\mathcal{M}} \in \{0, 1\}^n$ on certain neurons of MHA and FFN, which is inspired by LLM-pruner and related methods Ma et al. (2023); Kwon et al. (2022b); Sun et al. (2023). For MHA, we apply the pruning mask on each attention head as shown in Figure 3a, resulting in a masked multi-head attention (MHA) defined as:

$$\text{MHA}_{\text{mask}}(Q, K, V, \boldsymbol{\mathcal{M}}) = (\mathcal{M}_1 \cdot \text{attn}_1 \| \dots \| \mathcal{M}_h \cdot \text{attn}_h) W^o, \tag{9}$$

where $\mathcal{M}_i \in \{0, 1\}$ controls the output of each head, allowing for head-wise pruning based on learned values. The masked FFN, as shown in Figure 3b, applies a mask $\boldsymbol{\mathcal{M}}_{\text{ffn}} \in \{0, 1\}^{d_{\text{ff}}}$ on the neurons after the activation layer, resulting in:

$$\text{FFN}_{\text{mask}}(x, \boldsymbol{\mathcal{M}}_{\text{ffn}}) = (\sigma(xW_1 + b_1) \odot \boldsymbol{\mathcal{M}}_{\text{ffn}})W_2 + b_2, \tag{10}$$

where $\odot$ denotes as the Hadamard product. This masking design of MHA and FFN will not change the input and output dimensions of a pruned module, resulting in easier deployment with fewer modifications on a pruned model.

C Details of Algorithms

We present algorithms for end-to-end mask learning and time step gradient checkpointing.

D Derivation of Regularization Loss

In this section, we present the derivation of the approximated regularization loss with a simpler form in our mask optimization. The sigmoid function is defined as:

$$Sigmoid(x) = \frac{1}{1 + e^{-z}} \tag{11}$$

Algorithm 1 End-to-End Diffusion Model Pruning

Input: Pre-trained denoise model ϵ_θ and masked pre-trained denoise model $\epsilon_\theta^{\text{mask}}$ with masking parameters $\boldsymbol{\mathcal{M}}$ with initial value $\mathcal{M}_{\text{init}}$, text prompts $x \sim \mathcal{X} = \left\{x^i\right\}_{i=1}^N$, regularization coefficient β, learning rate η
Output: Learned pruning mask $\boldsymbol{\mathcal{M}}$

$z_T \sim \mathcal{N}(0, \mathbf{I})$ ▷ Initialize with latent random noise
for x **in** $\mathcal{X}$ **do** ▷ x as text prompt for training
  $z_0 \leftarrow \mathcal{F}_{\epsilon_\theta}(z_t, x)$ ▷ Original latent z_0 via ϵ_θ
  $\hat{z}_0 \leftarrow \mathcal{F}_{\epsilon_\theta^{\text{mask}}}(z_t, x, \boldsymbol{\mathcal{M}})$ ▷ Masked latent $\hat{z}_0$ via $\epsilon_\theta^{\text{mask}}$
  $\mathcal{L} = \sum_x \|\hat{z}_0 - z_0\|_2 + \beta \|\boldsymbol{\mathcal{M}}\|_0$ ▷ Loss with regularization
  $\boldsymbol{\mathcal{M}} \leftarrow \boldsymbol{\mathcal{M}} + \eta \frac{d\mathcal{L}}{d\boldsymbol{\mathcal{M}}}$
end for
return $\boldsymbol{\mathcal{M}}$

Algorithm 2 Time Step Gradient Checkpointing

Require: masked diffusion model $\epsilon_\theta^{\text{mask}}$, target latent z_0, diffusion time steps $t = 1, 2, \dots, T$, masking parameters $\boldsymbol{\lambda}$, loss function $\mathcal{L}$, learning rate η.

$\frac{d\mathcal{L}}{d\boldsymbol{\lambda}} = 0, \mathcal{H} \leftarrow \emptyset$ ▷ Memory initialization for intermediate latent
for $t = T, T-1, \dots, 1$ **do**
  $\hat{z}_{t-1} \leftarrow \epsilon_\theta^{\text{mask}}(z_t, \boldsymbol{\lambda})$ ▷ Calculate latent
  $\mathcal{H} \leftarrow \mathcal{H} \cup \{\hat{z}_{t-1}\}$ ▷ Only store denoised latent at this step
end for
$\mathcal{L} = \mathcal{L}_E(\hat{z}_0, z_0) + \beta \mathcal{L}_0(\boldsymbol{\lambda})$ ▷ Calculate loss
$\frac{d\mathcal{L}}{d\boldsymbol{\lambda}} += \frac{d\mathcal{L}}{d\hat{z}_0} \frac{d\hat{z}_0}{\boldsymbol{\lambda}}$
for $t = 1, 2, \dots, T-1$ **do**
  $\hat{z}_{t-1} \leftarrow \epsilon_\theta^{\text{mask}}(z_t, \boldsymbol{\lambda})$ ▷ Recompute to save all intermediates
  $\frac{d\mathcal{L}}{d\hat{z}_t} = \frac{d\mathcal{L}}{d\hat{z}_{t-1}} \frac{d\hat{z}_{t-1}}{d\hat{z}_t}$ ▷ Get loss gradient w.r.t. the step before
  $\frac{d\mathcal{L}}{d\boldsymbol{\lambda}} += \frac{d\mathcal{L}}{d\hat{z}_t} \frac{d\hat{z}_t}{d\boldsymbol{\lambda}}$ ▷ Accumulate gradient
end for
return $\frac{d\mathcal{L}}{d\boldsymbol{\lambda}}$

Now we expand one summation term in Equation 4 by consider $C = -\beta \log \frac{-\gamma}{\zeta}$:

$$Sigmoid(\log \lambda_j + C) = \frac{1}{1 + e^{-(\log \lambda_j + C)}} \tag{12}$$
$$= \frac{1}{1 + e^{-\log \lambda_j} e^{-C}} \tag{13}$$
$$= \frac{1}{1 + \lambda_j^{-1} e^{-C}} \tag{14}$$
$$= \frac{\lambda_j}{\lambda_j + e^{-C}} \tag{15}$$
$$= \lambda_j * \frac{1}{\lambda_j + e^{-C}} \tag{16}$$
$$= \lambda_j * g(\lambda_j), \tag{17}$$

where $g(\lambda_j) = \frac{1}{\lambda_j + e^{-C}}$. We further simplify the result by considering the new constant to be $C := e^{-C}$. The gradient of $g(\lambda_j)$ is:

$$g'(\lambda_j) = -\frac{1}{(\lambda_j + C)^2}. \tag{18}$$

Note that $\lambda_i > 0$ due to $\log \lambda_j$. Hence, $-\frac{1}{C^2} < g'(\lambda_j) < 0$, and for larger C, we can approximately have $g'(\lambda_j) \approx 0$. Therefore, we can treat $g(\lambda_j)$ as constant and reformulate Equation 4 as

$$\mathcal{L}_0(\boldsymbol{\lambda}) = \sum_{j=1}^{\boldsymbol{\lambda}} Sigmoid(log\lambda_j - \beta \log \frac{-\gamma}{\zeta}) \approx ||\lambda||_1. \tag{19}$$

The approximation $\|\boldsymbol{\lambda}\|_1$ reflects the relationship between the regularization term and the $L1$ norm, as the Sigmoid function applied to each λ_j induces sparsity by encouraging values closer to zero, similar to $L1$ regularization. Therefore, we use the $L1$ norm as a regularization loss. This approximation has the benefit of reducing calculation and providing a simpler form of regularization.

E Detailed Experiment Setting and Evaluation Setting

By default, we set the masking value λ to 5, ensuring that the effective $\mathcal{M}$ is approximately 1 with a high probability to facilitate smoother training. Table 2 and Table 3 are the default sampling training configurations.

Table 2: Training Configuration for FLUX

Parameter	**Value**
Batch size	4
lr_{attn}	0.05
lr_{ffn}	1
lr_{norm}	0.5
β	0.1
δ	0.1
Optimizer	Adam
Training Steps	400
Weight decay	1×10^{-2}
Scheduler	constant
Diffusion pretrained weight	FLUX.1-schnell
Hardware used	$1 \times$ NVIDIA H100

Table 3: Training Configuration for SDXL

Parameter	**Value**
Batch size	4
lr_{attn}	0.15
lr_{ffn}	0.15
lr_{norm}	0
β	0.5
δ	0.5
Optimizer	Adam
Training Steps	400
Weight decay	1×10^{-2}
Scheduler	constant
Diffusion pretrained weight	stable-diffusion-2-base
Hardware used	$1 \times$ NVIDIA H100

E.1 Evaluation Configuration

We conduct extensive experiments to evaluate the performance of **EcoDiff** using both FID and CLIP scores, as summarized in Table 1. For the CLIP score, we use a pretrained **ViT-B/16** model as the backbone. For the FID score, we utilize the FID function from **torchmetrics** with its default settings (e.g., the number of features set to 2048). To accelerate the evaluation process, all images are resized to a uniform resolution of 512×512 using **bicubic interpolation**.

F Hard Discrete Distribution Visualization

Figure 5 illustrates the hard discrete distributions with varying values of δ. With higher δ values, such as 1 or 2, after 100 runs, the estimated hard discrete distribution is closer to a step function than

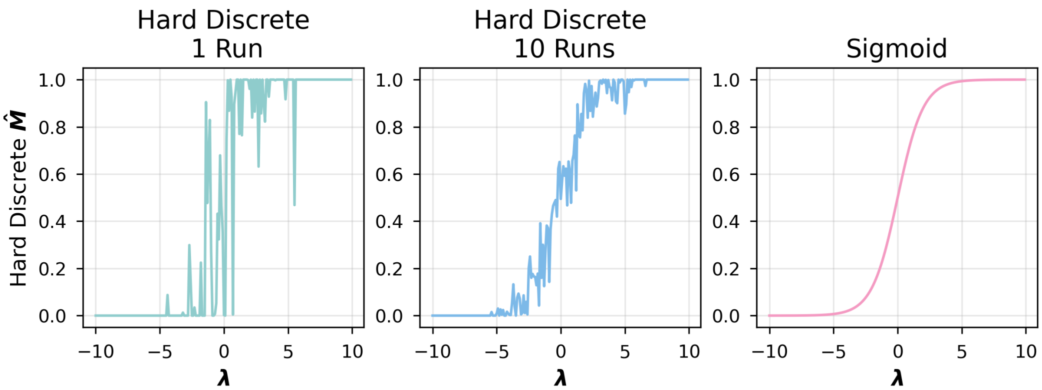


Figure 4: **Comparison of hard discrete and sigmoid distributions.** The hard discrete distribution yields a discrete-like, sparse output by clamping values between 0 and 1, making it well-suited for masking tasks where controlled sparsity is beneficial. Parameters are set to $\beta = 0.83$, $\delta = 1 \times 10^{-8}$, $\zeta = 1.1$, and $\gamma = -0.1$. Additional samples of the hard discrete distribution with varying δ values are provided in Appendix F.

the original sigmoid function. Conversely, with lower δ, the distribution aligns more closely with the sigmoid function. By adjusting δ, the training process can be tuned to either enhance robustness against randomness or increase randomness to escape local minima. Further detailed ablation studies on δ are necessary and will be addressed in future work.

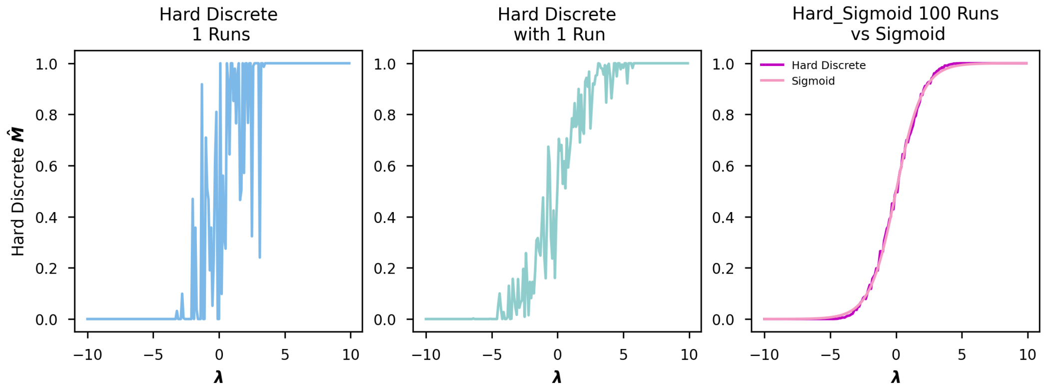
G Additional Results

G.1 Complexity Assessments

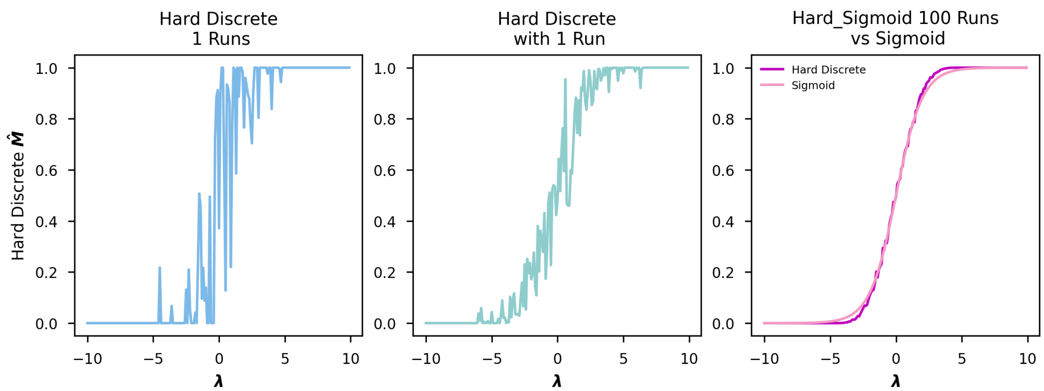
We evaluate our pruning framework's runtime and memory consumption with SD2, as shown in Figure 8. The experiment results demonstrate the consistency with the theorem stated in Section 2.3. By using time-step gradient checkpointing, **we significantly reduce memory usage from $O(T)$ to $O(1)$**, as illustrated in Figure 8a). Additionally, Figure 8b) demonstrates that, despite the significantly reduced memory usage, the runtime complexity remains $O(T)$ with only an $2\times$ increase in runtime. The time-step gradient checkpoint enables end-to-end training on resource-constrained devices, including the largest DiT model, FLUX, with just a single 80GB GPU. We further provide the carbon footprint analysis of performing pruning using EcoDiff in Appendix N.

G.2 Ablation Study

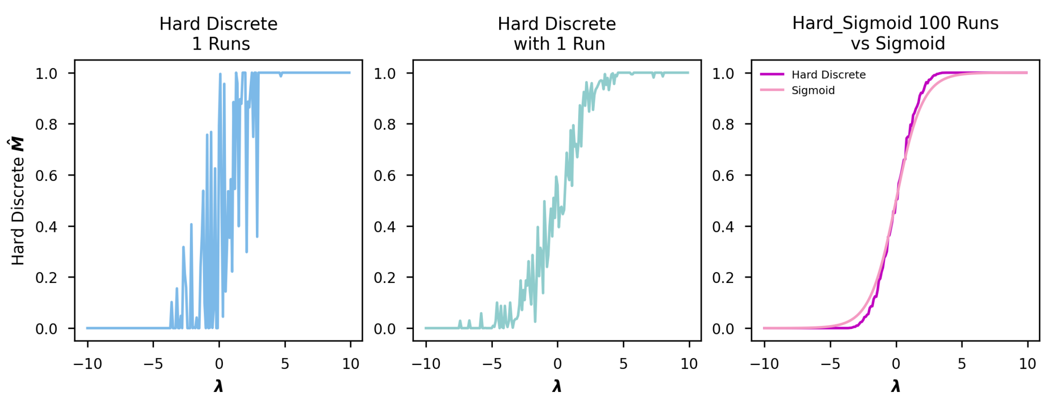
Evaluation with different pruning ratios. With our proposed framework, the learned pruning mask $\mathcal{M}$ acts as a score-based mask, indicating the redundancy of each network component. The pruning ratio can be flexibly adjusted by applying different thresholds τ in Equation **??**. Figure 7 demonstrates how the quality of the generated image changes with different pruning ratios. The quality of generated images can be preserved up to a pruning ratio of 20%. At pruning ratios of 10% and 15%, for the prompt, "A cat and a dog are playing chess," the generated image even exhibits enhanced semantic meaning compared to the original image, further validating the decreased FID score in Table 1 and Figure 9. With higher pruning ratio, model performance starts to degrade at 20% pruning rate for FLUX and 25% for SDXL. **Hyperparameter ablations.** We perform an ablation study on the regularization coefficient β in Equation B as well as the training dataset size. An optimal β value, such as 0.5, is necessary to promote sparsity while maintaining high semantic fidelity. However, as illustrated in Table 4, excessively high values of β can easily lead to training divergence. Additionally, we conduct experiments across various training dataset sizes, ranging from 1 to 512, as shown in Table 5. Notably, even when training with a single randomly selected text prompt, the performance degradation relative to a dataset size of 100 remains minimal, suggesting that the pruned or masked components of the network are largely independent of the dataset size.



(a) Hard Discrete with $\delta = 0.05$



(b) Hard Discrete with $\delta = 0.1$



(c) Hard Discrete with $\delta = 0.2$

Figure 5: **Hard discrete distribution with varying δ values.** (a) $\delta = 0.05$, (b) $\delta = 0.1$, (c) $\delta = 0.2$. We use the default hard discrete setting as it states in Figure 4.

Other ablations. More ablation studies in Appendix K. We show the compatibility of EcoDiff with time step distillation and feature reuse methods in Appendix I.

H RESULTS ON STABLE DIFFUSION 2

EcoDiff is designed to be versatile, adaptable and model-agnostic. For Stable Diffusion 2 (SD2), we specifically focus our pruning approach on the attention (**Attn**) and feed-forward network (**FFN**)

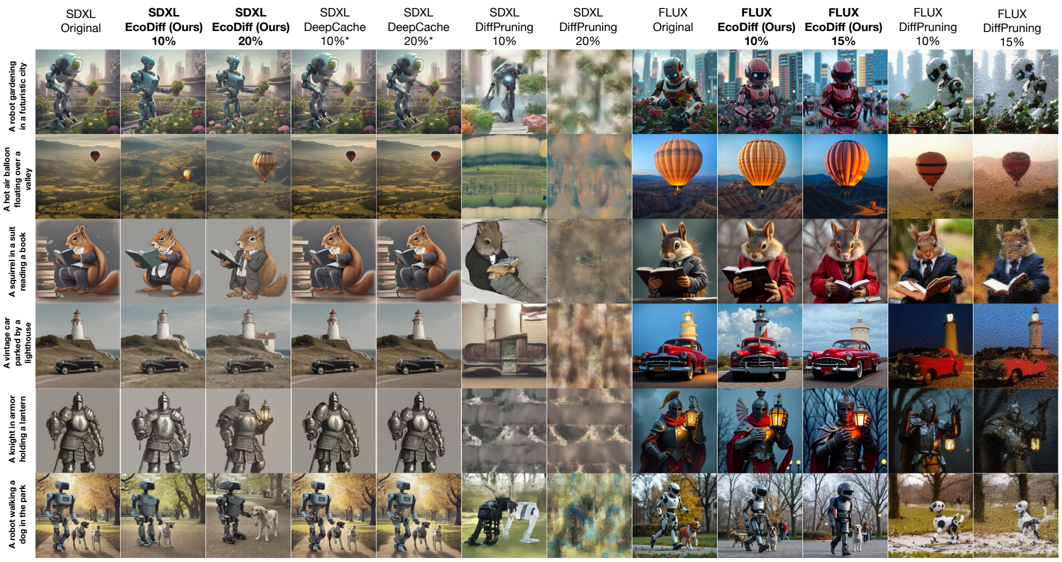


Figure 6: **Example images for comparison.** We intentionally use ChatGPT to generate prompts with rich semantics. No retraining is performed for all methods. DeepCache does not reduce the model size. More examples are provided in Appendix L.

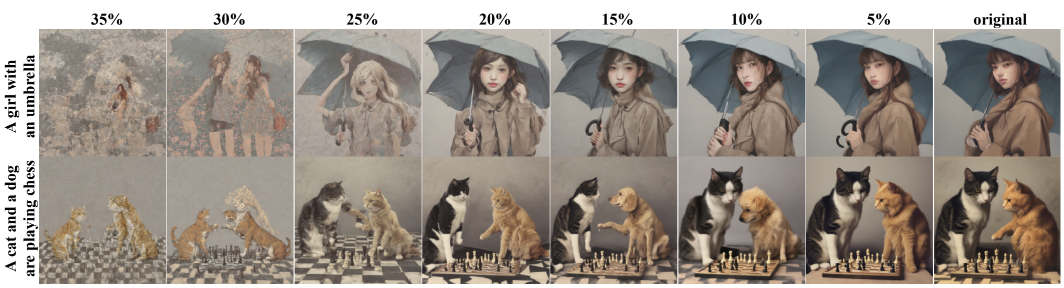


Figure 7: **Masking diffusion model, *SDXL-base*, with different pruning ratios.** These examples highlight the versatility of our framework in applying different pruning ratios to mask the diffusion model effectively. In addition, **pruning redundant parts can sometimes fix the incorrect semantics**, as shown in the second row (10% and 15% pruning ratio cases). More examples are provided in Appendix M.

blocks. Table 6 highlights the model-agnostic feature of EcoDiff, demonstrating its ability to optimize diverse architectures without requiring model-specific adjustments or design changes.

I Compatibility with Other Methods

I.1 Compatibility With Step Distillation

I.2 Compatibility With Feature Reuse

EcoDiff demonstrates high versatility by efficiently **integrating with feature reuse methods like DeepCache** Ma et al. (2024b) as shown in Figure 10. Additionally, combining EcoDiff with DeepCache enables significant reductions in effective **GFLOPs** and runtime, highlighting its compatibility and effectiveness in optimizing resource usage.

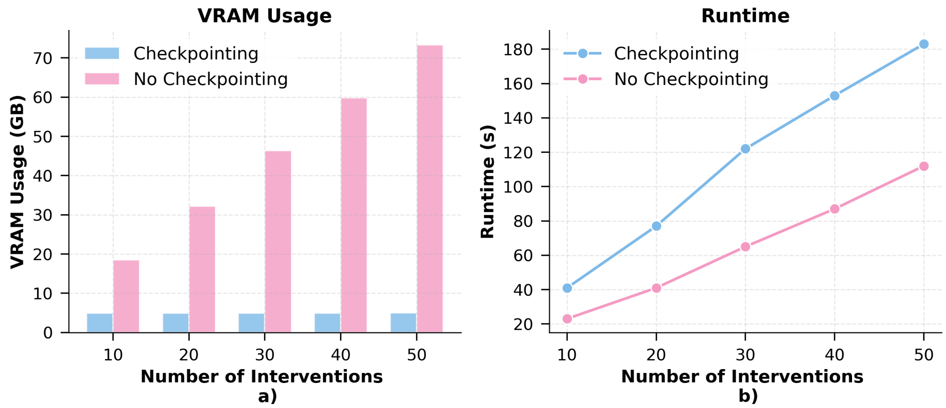


Figure 8: **VRAM usage and runtime per training step comparison with and without gradient checkpointing on SD2.** Measurements are averaged over five runs.

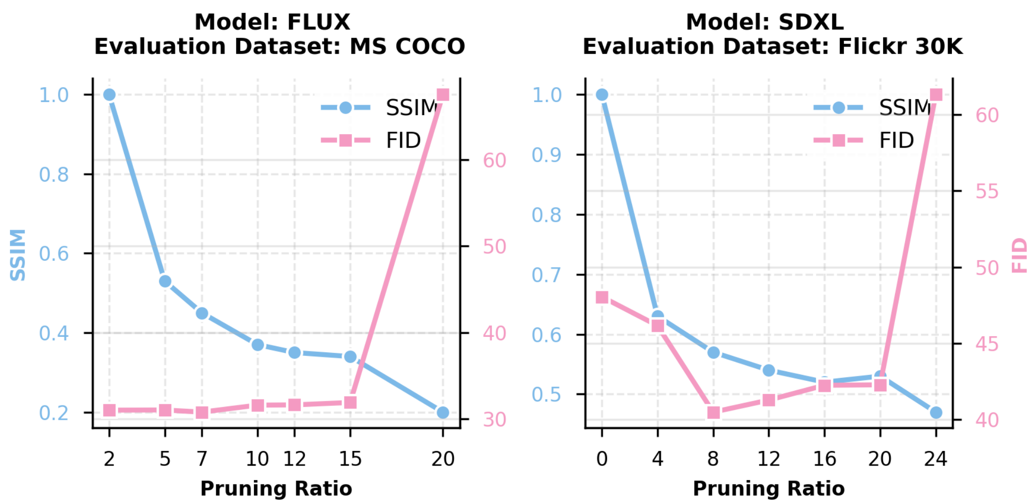


Figure 9: **Evaluation on different pruning ratios**. Models start to degrade at 20% pruning for FLUX and 25% for SDXL.

J More Results on Light Retraining

Figure 11 and Table 7 illustrate that the pruned model's performance with EcoDiff drops significantly beyond a 20% pruning ratio. To mitigate this, we adopt a lightweight retraining strategy using LoRA fine-tuning with only 100 image-text pairs. The images, generated using the same prompts from GCC3M Sharma et al. (2018), are synthesized by the original diffusion model rather than the original dataset. The retraining, conducted around 12 hours and 10,000 steps, demonstrates the compatibility of our EcoDiff framework. As seen in Figure 11, prompts such as *A robot dog exploring an abandoned spaceship*, *A mystical wolf howling under a glowing aurora*, and *A cozy library with a roaring fireplace* highlight that while semantic fidelity is largely preserved, fine-grained features degrade significantly in the pruned model. After LoRA retraining, these fine-grained details are substantially recovered while maintaining semantic fidelity with some exceptions with a significant reduction in FID score and a noticeable increase in CLIP score as is shown in Table 8.

β	0.01	0.1	0.5	1	2
FID ↓	41.23	34.23	**33.74**	67.78	121.2
CLIP ↑	0.29	0.30	**0.30**	0.221	0.19

Table 4: **Ablation study on β.** Evaluated on MS COCO dataset with a fixed pruning ratio of 20% on SDXL

Size	1	8	64	100	256	512
FID ↓	35.18	36.17	34.13	**33.74**	33.76	33.68
CLIP ↑	0.30	0.31	0.31	0.30	0.31	**0.31**

Table 5: **Ablation study on training data size.** Evaluated on MS COCO dataset with a fixed pruning ratio of 20% on SDXL. We can achieve good pruning result with even one sample.

K More Ablation Study

K.1 Ablation on Pruning Single Modules

Figure 13 demonstrates the flexibility of our pruning mask, allowing for adjustable thresholding across different blocks, such as pruning exclusively in the **Attn** or **FFN** blocks. Notably, image quality retains high fidelity regardless of the targeted pruning module. However, minor variations in fine-grained details and semantic meaning are observed, and we will investigate them further in future work.

K.2 Ablation on Pruning Setting

We apply a threshold τ to mask the head and FFN layers. Thresholding can be applied in two ways: **globally** across all blocks or **locally** within each block (e.g., FFN, Norm, Attn). Global thresholding uses a single threshold τ applied to the entire mask $\mathcal{M}$. In contrast, local thresholding applies τ individually within each block, treating them independently. To evaluate these approaches, we conduct an ablation study, and the results are shown in Figure 12. The global thresholding approach demonstrates superior results, accounting for the interactions and trade-offs between layers and blocks, leading to more effective masking. In contrast, local thresholding results in a deterioration in quality.

K.3 Ablation Study on FLUX

Table 9 and Table 10 present the results of our ablation studies on FLUX. Similar results are observed with the SDXL model. Notably, even with just a single training data point, FLUX can produce relatively competitive results, highlighting its robustness and efficiency in resource-constrained scenarios.

L Additional Sample Results

Figure **??** demonstrate the additional samples with baseline approaches, DeepCache and DiffPruning Ma et al. (2024b); Fang et al. (2023).

M Semantic Changing with Different pruning ratios

Figure 15 and Figure 14 show the superior performance of EcoDiff with up to 20% of pruning ratio for SDXL and 15% for FLUX.

Model	MS COCO		Flickr30K	
	FID	**CLIP**	**FID**	**CLIP**
SD2	30.15	0.33	36.10	0.35
EcoDiff Pruning 10%*	30.61	0.32	38.12	0.34
EcoDiff Pruning 20%*	31.16	0.32	42.67	0.34

Table 6: Ablation study of SD2 models. Metrics include FID and CLIP scores on MS COCO and Flickr30K datasets. * denotes as the model pruning ratio on the **Attn** and **FFN** only.

Model	MS COCO		Flickr30K	
	FID	**CLIP**	**FID**	**CLIP**
FLUX Dev	35.59	0.32	45.68	0.34
FLUX-Schnell Step-Distilled	30.99	0.33	39.70	0.35
FLUX EcoDiff Pruning 10%	32.16	0.32	42.56	0.34
FLUX EcoDiff Pruning 15%	31.76	0.30	43.25	0.33

Table 7: Ablation study of Flux models, *Flux-dev*, *Flux-schnell*. Metrics include FID and CLIP scores on MS COCO and Flickr30K datasets.

N CARBON FOOTPRINT ANALYSIS

We conduct a carbon footprint analysis for EcoDiff training. The analysis uses training configurations with only 200 training steps, as most configurations converge within 400 steps. The carbon footprint calculations are based on the default configurations outlined in Appendix E. All calculations follow the methodology used in the SD2 carbon footprint analysis Rombach et al. (2022).

Figure 10: **EcoDiff with DeepCache**

Figure 11: **EcoDiff Pruning with light retraining.**

Model	**MS COCO**		**Flickr30K**	
	FID	**CLIP**	**FID**	**CLIP**
SDXL original	35.50	0.31	49.34	0.34
EcoDiff with retraining Pruning 25%	35.65	0.31	46.47	0.34
EcoDiff without retraining Pruning 25%	36.24	0.30	47.49	0.31
EcoDiff with retraining Pruning 30%	40.21	0.30	48.98	0.32
EcoDiff without retraining Pruning 30%	79.84	0.25	75.07	0.27

Table 8: Performance comparison with and without light retraining on SDXL

Figure 12: **EcoDiff Pruning with global thresholding and local thresholding**

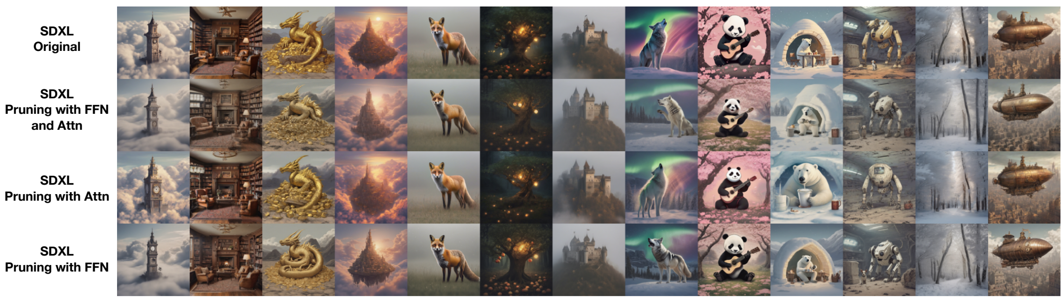


Figure 13: **EcoDiff Prunig with masking on single Module, FFN module or Attn module.**

β	0.001	0.005	0.01	0.1	0.5
FID ↓	43.21	44.50	43.24	87.5	323.1
CLIP ↑	0.32	0.34	0.33	0.21	0.03

Table 9: **Ablation study on β.** Evaluated on Flickr30K dataset with a fixed pruning ratio of 15% on FLUX.

Size	1	8	100	256	512
FID ↓	29.79	31.51	31.76	77.48	33.25
CLIP ↑	0.31	0.30	0.30	0.26	0.33

Table 10: **Ablation study on training data size.** Evaluated on MS COCO dataset with a fixed pruning ratio of 15% on FLUX.

Model	**MS COCO**		**Flickr30K**	
	FID	**CLIP**	**FID**	**CLIP**
SDXL Original	35.50	0.31	49.34	0.34
EcoDiff 20% Global	34.41	0.31	42.84	0.33
EcoDiff 20% Local	92.84	0.26	117.18	0.27

Table 11: Ablation study of SDXL models with different pruning setting

Model	**Training Time(hours)**	**VRAM Usage**	**GPU hours**	**Carbon Footprint**
SD2	3	4.6G	0.16	12.0g
SDXL	4.4	22.9G	1.24	93.0g
FLUX	0.54	64.2G	0.42	31.50g

Table 12: Carbon Footprint and Running hours.

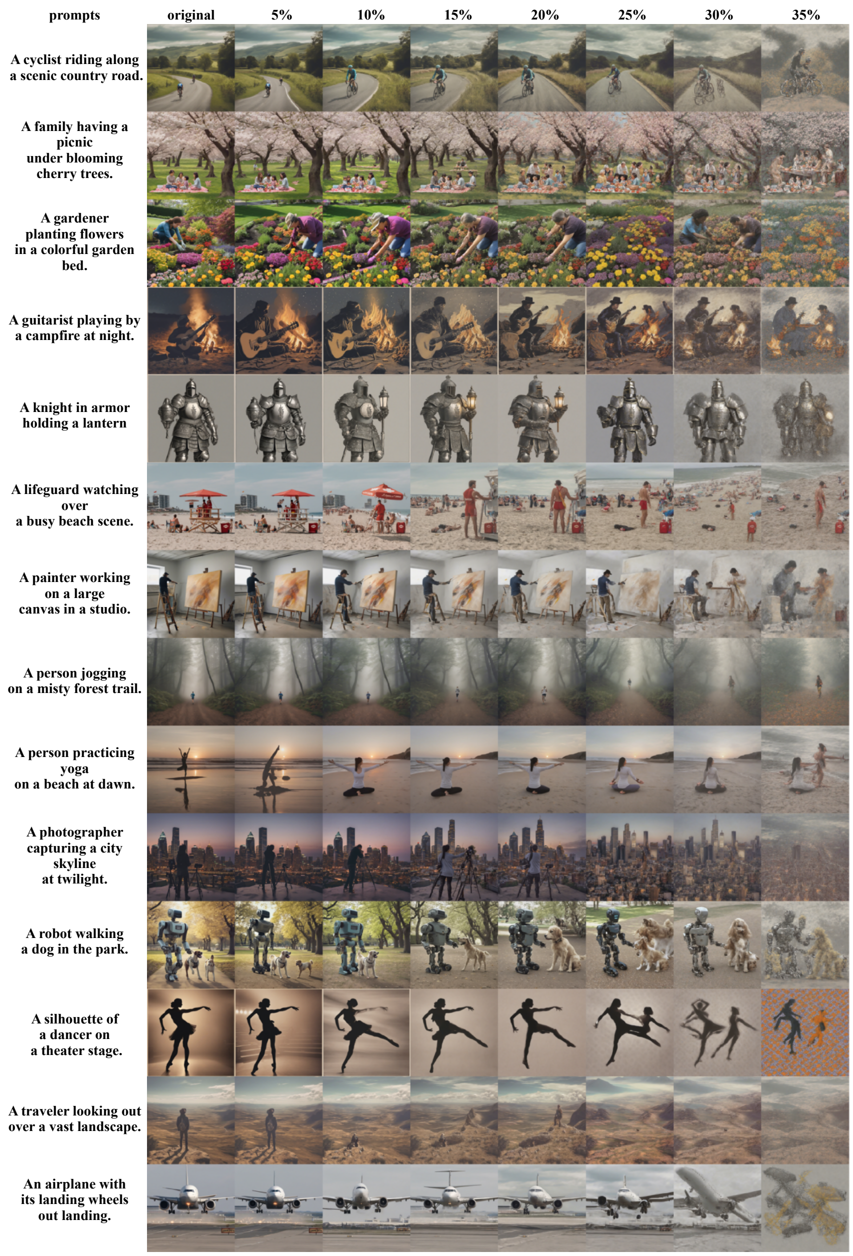


Figure 14: **More Samples with masking diffusion model, *SDXL-base*, with different pruning ratios.**

Figure 15: **More Samples with masking diffusion model, *FLUX-schnell*, with different pruning ratios.**