Layer- and Timestep-Adaptive Differentiable Token Compression Ratios for Efficient Diffusion Transformers

Supplementary Material

A. More Visualization of Token Routers

In Sec. 3.2, we provided an example visualization of the router predictions to evaluate the effectiveness of our DiffCR router. Here, we present additional visualization examples in Fig. 2 to further validate our findings. Our observations consistently demonstrate the following: (1) *The router effectively captures semantic information*, clearly delineating object shapes and achieving an attention-like effect while significantly reducing computational costs. (2) *The predicted token importance varies across layers and timesteps*. For example, some layers focus on object generation, while others emphasize background areas. Additionally, as timesteps progress, the router increasingly captures the semantic contours of objects, highlighting the importance of dynamic token importance estimation. (3) *The optimal compression ratio differs across layers and timesteps*. For instance, some layers assign high importance to all tokens, indicating minimal redundancy, while others selectively prune tokens from objects or backgrounds with distinct shapes, requiring different compression ratios. This variance is also observed across timesteps. In the previous MoD [8] approach, a fixed global compression rate is uniformly applied across layers and timesteps, ignoring their individual significance. Such uniform pruning risks over-pruning critical layers or timesteps while under-compressing redundant ones. This observation underscores the need for adaptive and dynamic compression ratios tailored to both layers and timesteps.

B. Ratio Trajectory Analysis for the T2I Task

In Sec. 3.3, we visualized the ratio trajectory for inpainting tasks trained with our proposed layer-wise DiffCR. Here, we also provide the training trajectory of compression ratios for all layers during fine-tuning of a PixArt-Σ model on a T2I task, as shown in Fig. 1 (a-c). The visualization consistently reveals that: (1) Each layer learns its unique compression ratio, with redundant layers achieving higher compression and critical layers remaining less or entirely uncompressed; (2) The average ratio across layers gradually converges to the target ratio. In this example, with a target of 20%, the final achieved average ratio is approximately 19%, indicating a minor gap. Notably, a trade-off exists between convergence speed and generation quality: a higher MSE loss coefficient for the ratio accelerates convergence but may degrade quality due to overly rapid compression, while a smaller coefficient promotes gradual convergence and maintains quality, albeit with slower training. In practice, we set the initial coefficient to 0.3 and dynamically adjust it during training to balance speed and quality effectively; (3) The middle layers exhibit greater redundancy, while the later layers generally have lower redundancy and often cannot be compressed. The early layers show variable redundancy levels.

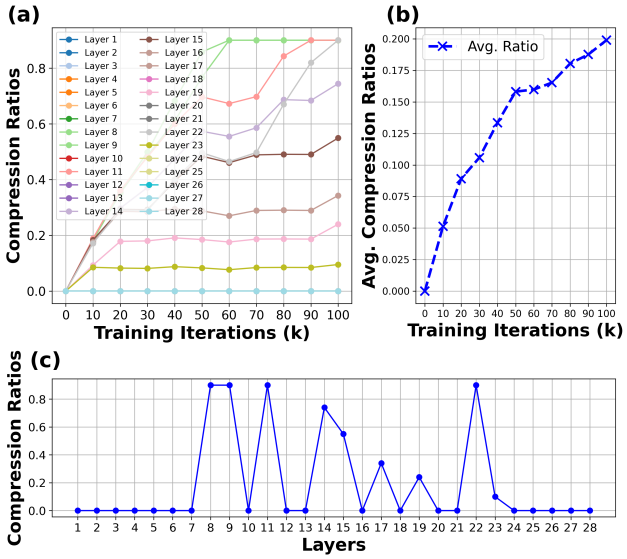

Figure 1. Visualization of the compression ratio trajectory during fine-tuning for a T2I task: (a) Trajectories for each of the 28 layers in the PixArt-Σ model; (b) Average ratio trajectory across all layers; and (c) The final learned ratio distribution across 28 layers.

Note that to prevent the model from learning 0% compression ratios across all layers, we balance diffusion loss (favoring lower ratios for higher quality) and MSE loss (driving the target average ratio) using a coefficient, without additional regularization or penalties. A higher coefficient speeds up convergence but may compromise quality, while a smaller one ensures gradual convergence and preserves quality. Some layers naturally learn 0% ratios, underscoring their importance.

C. Correlation Between Learned Compression Ratios and Router Predictions

We select three representative layers with high, medium, and low learned compression ratios to visualize the corresponding predictions of the DiffCR router and analyze potential correlations. As shown in Fig. 3, where "C.R." denotes the compression ratios, we observe a strong correla-

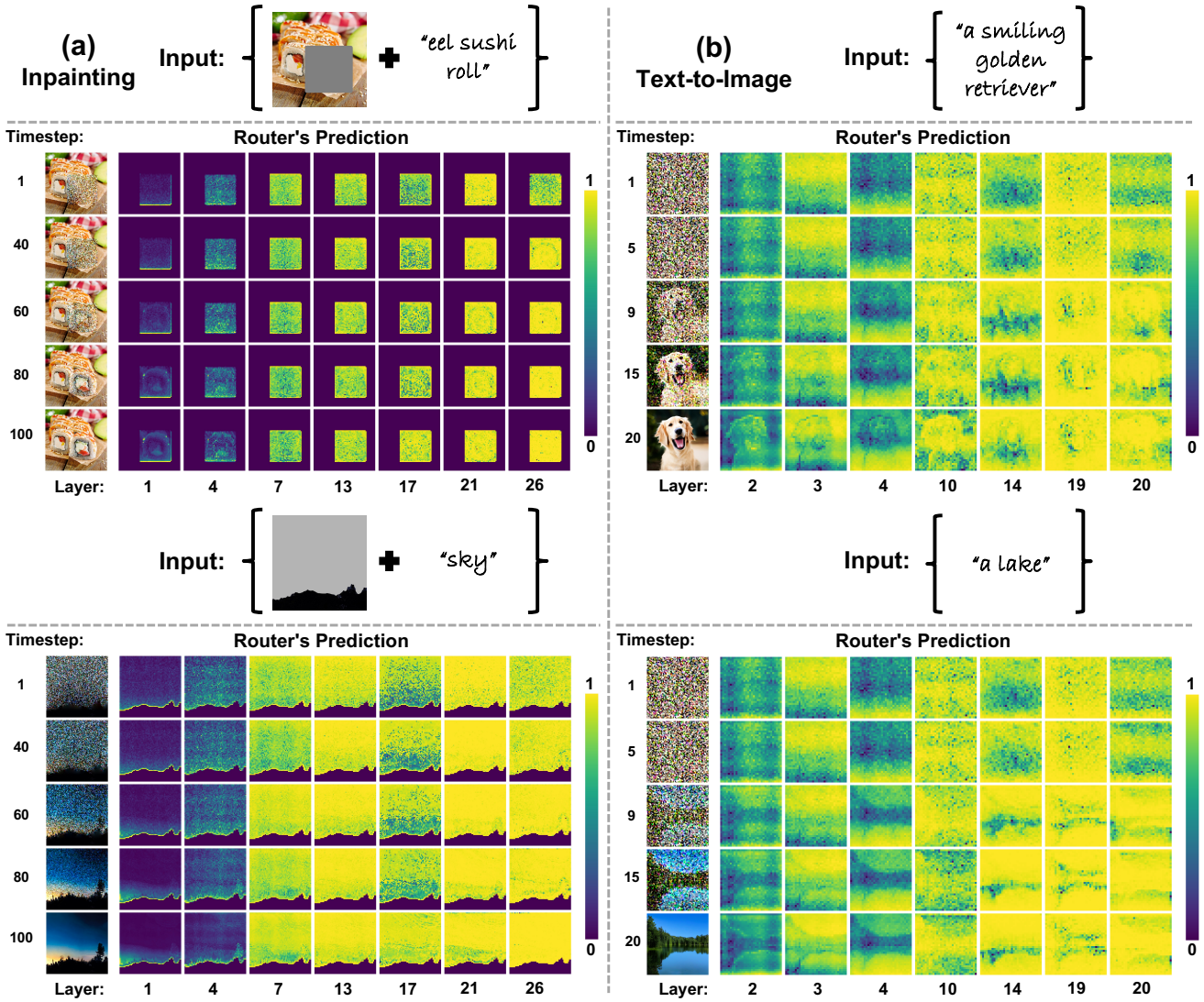

Figure 2. More visualizations of the router's predictions: (a) For inpainting tasks, where inputs are masked images with text prompts, we follow the previous SOTA method Lazy-Diffusion [7] to generate only the masked area rather than the entire image; (b) For text-to-image (T2I) tasks, where inputs are noise and text prompts, we follow PixArt-Σ [2] for generation. Each visualization includes the router's prediction map with values ranging from 0 to 1. The generated image at each corresponding timestep is shown on the left, while the router's prediction maps across various layers and timesteps are displayed on the right.

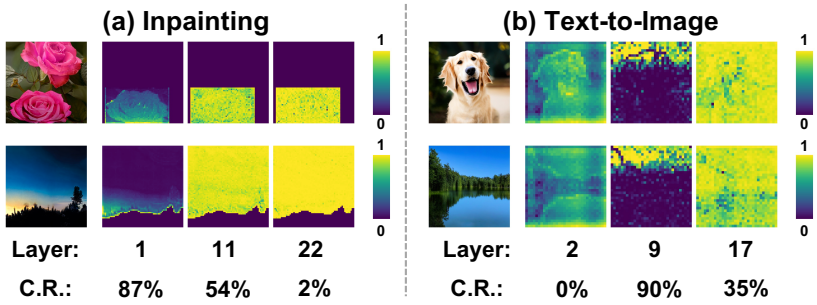

Figure 3. Visualization and analysis of the correlation between the learned compression ratios and the DiffCR router's predictions.

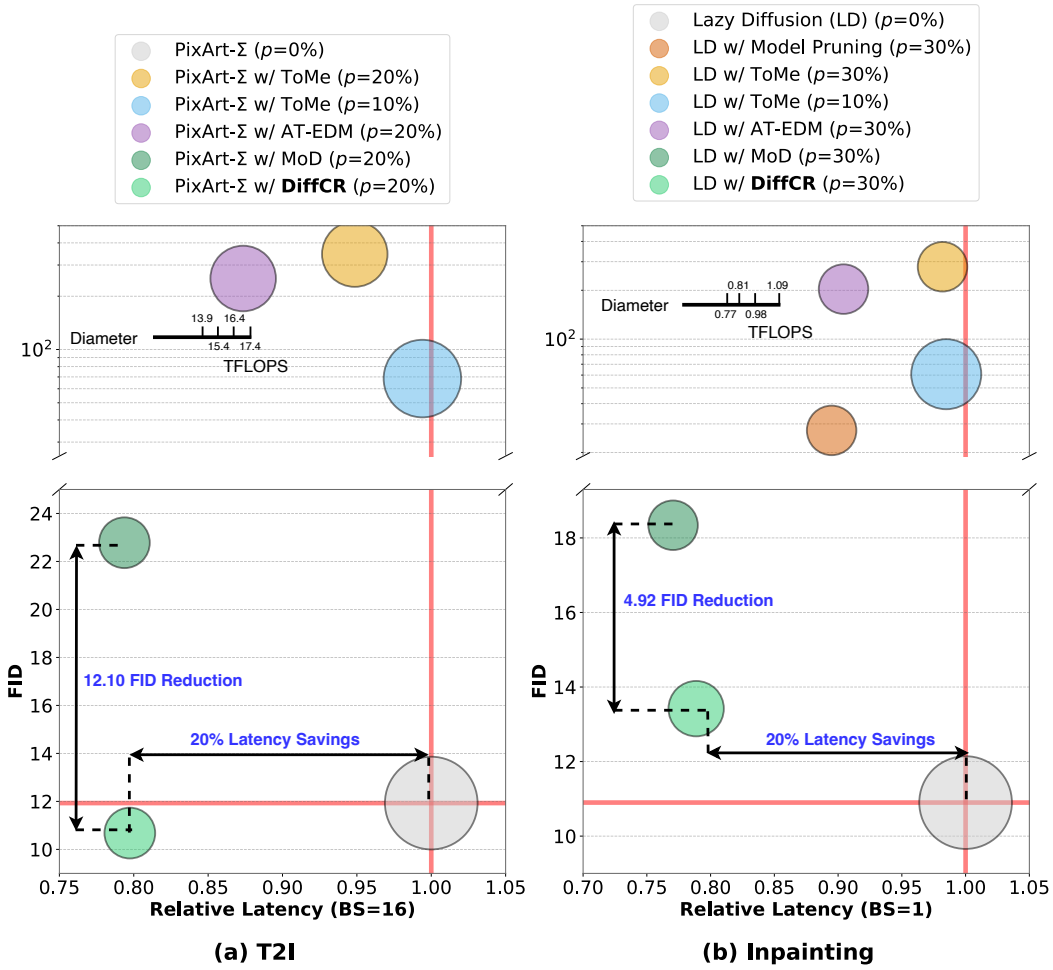

Figure 4. Overall comparison of DiffCR with baselines in terms of latency, FID, and TFLOPS for both T2I and inpainting tasks.

tion between the learned ratios and the router's predictions. For layers with high compression ratios, such as layer 1 in inpainting or layer 9 in T2I, the router consistently predicts lower importance scores for many semantic areas, adopting an extremely "lazy behavior" to save computations. Conversely, for layers with low compression ratios, the router assigns higher importance scores to most areas. This visualization validates the joint learning effect between our token-level routers and the differentiable ratios.

D. Trade-offs for Choosing Timestep Regions

In Sec. 3.4, we introduced the timestep-wise DiffCR, where the timestep regions are evenly divided into 10 regions for inpainting tasks with a total of 100 sampling timesteps, and 4 regions for T2I tasks with 20 sampling timesteps. Here, we provide additional guidance on selecting the number of timestep regions and the associated trade-offs. A larger number of timestep regions allows for learning finer-grained and more precise compression ratios across all timesteps. However, too many regions can make training unstable and challenging. To reduce training complexity and enhance stability, we select a smaller number of regions, such as 4 for T2I tasks. Conversely, using too few regions risks oversimplifying the method, reducing it to heuristic approaches like SpeeD [10], which manually defines three timestep regions. In practice, we choose between 4 and 10 timestep regions to balance granularity and stability. While our approach aligns with the general insights of SpeeD, it is more systematic and adaptive. Unlike manual exploration of a large design space, our method efficiently handles a significantly greater number of regions in a principled manner, balancing granularity and training stability.

E. Overall Comparison Figure

In Sec. 4.2, we presented a comprehensive comparison of our DiffCR method against baseline approaches for both inpainting and T2I tasks. Here, we provide the overall comparison figures to better illustrate the achieved im-

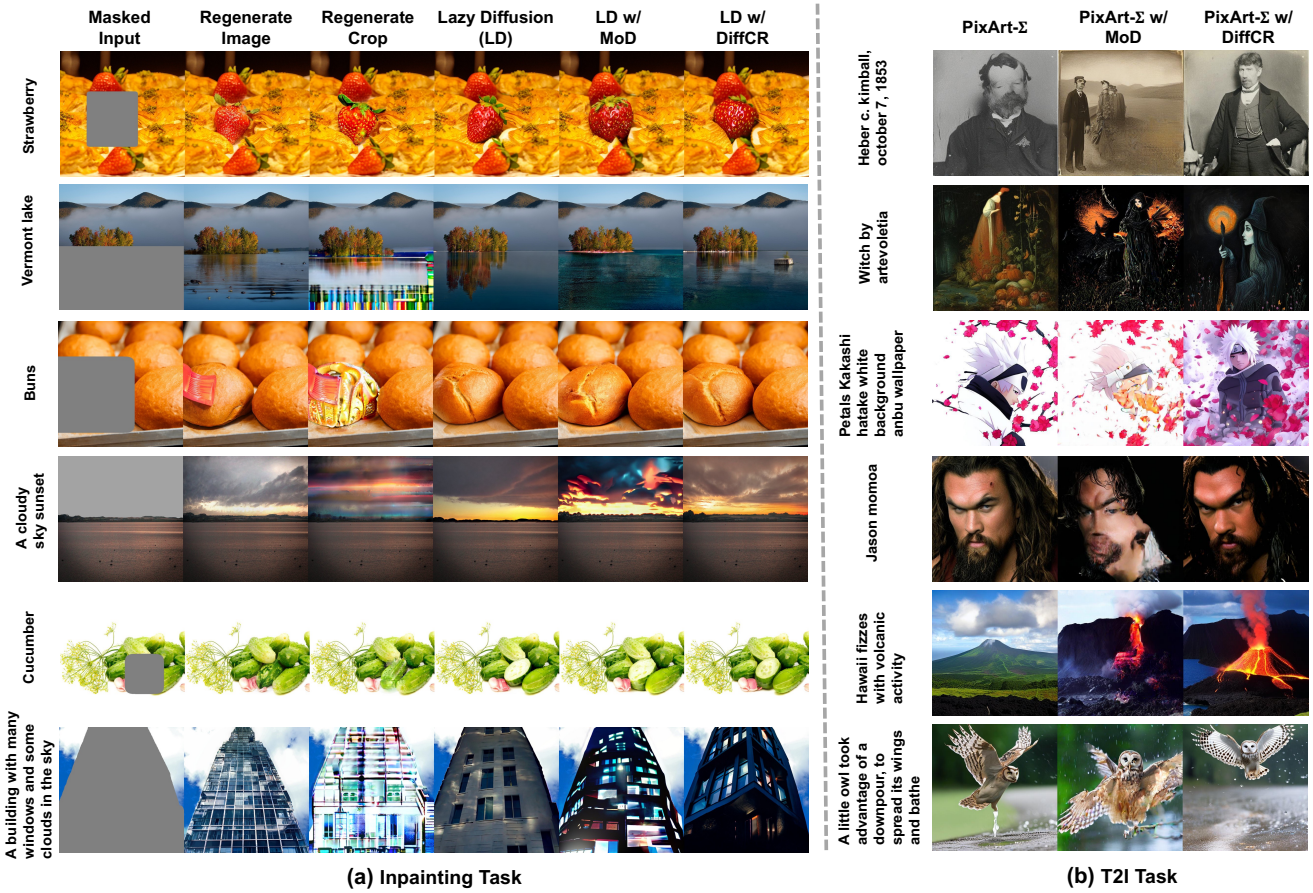

Figure 5. Additional visual comparisons of our DiffCR with previous uncompressed models and SOTA compression methods: (a) Inpainting tasks, where DiffCR is applied to LD models [7], and (b) T2I tasks, where DiffCR is applied to PixArt-Σ [2].

provements in FID and latency reductions. As shown in Fig. 4, our DiffCR consistently delivers superior trade-offs between FID and latency, achieving FID reductions of 12.10 and 4.92 for T2I and inpainting tasks, respectively, at comparable GPU latency when compared to the most competitive baseline.

F. Model Trajectories of DiffCR

In Sec. 4.2, we visualized the model trajectories during the training of DiffCR-L for both T2I and inpainting tasks. This revealed a key benefit: during fine-tuning, the averaged compression ratios across all layers gradually converge to the target ratio, producing a series of "by-product" models with varying compression ratios. Here, we also supply the model trajectories of DiffCR-LT ("-LT" denotes layer- and timestep-wise DiffCR). As shown in Fig. 6, we visualize the FID scores and corresponding compression ratios during the fine-tuning of DiffCR-LT. The observations consistently validate the benefits of this approach, showing that it enables the generation of a series of models with diverse compression ratios. Also, we observe that inpainting tasks

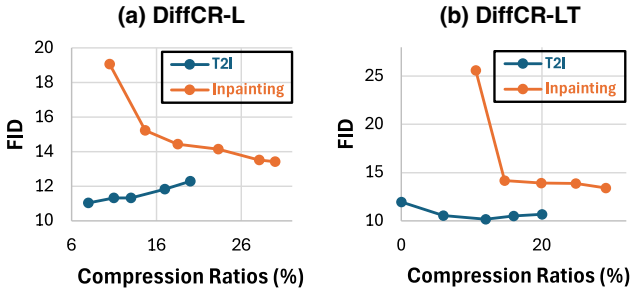

Figure 6. Model trajectories of DiffCR.

and Latent Diffusion (LD) models [7] are more sensitive to pruning and require longer fine-tuning to improve generation quality effectively, compared to T2I tasks. Moreover, for T2I tasks, DiffCR-LT demonstrates slightly greater stability in model trajectory compared to DiffCR-L.

G. More Visualization of Visual Examples

In Sec. 4.4, we selected challenging input prompts to evaluate the qualitative performance of our proposed DiffCR.

Table 1. Characteristics of our method and caching-based baselines.

Method	Model	Skip / Cache	Granulariy	Learnable	Token Pruning	Timestep-wise Feature Cache
DeepCache [12]	U-Net	Block	Block	✗	✗	✓
CMYC [6]	U-Net	Block	Block	✗	✗	✓
L2C [5]	DiT	Attn. & MLP	Layer	✓	✗	✓
TGATE [4]	DiT	Attn.	Layer	✗	✗	✓
DiffCR **(Ours)**	DiT	Attn. & MLP	Token	✓	✓	✗ but compatible

Here, we provide additional visual examples, as shown in Fig. 5. The examples consistently demonstrate that DiffCR achieves comparable or even superior generation quality compared to the RegenerateCrop baseline and even uncompressed LD or PixArt-Σ for inpainting and T2I tasks, respectively. Note that ToMe [1] and AT-EDM [9] are omitted here due to their poor generation quality when applied to DiTs, even at a modest compression ratio of 10%.

H. Comparison with Caching-based Baselines

We summarize the characteristics of our method and caching-based baselines in Tab. 1. DeepCache [12] and CMYC [6] are designed for U-Net-based models, making direct comparison challenging, while L2C [5] and TGATE [4] target DiTs by caching layer features to reduce recomputation in future timesteps. Unlike these approaches, our method focuses on token pruning with learnable layer- and timestep-dependent compression ratios, and while it does not employ temporal caching, it remains compatible with such techniques. To directly compare, we evaluate all methods using PixArt-Σ on the MS-COCO-30K dataset (T2I task) under approximately 25% latency savings, where L2C achieves an FID of 28.6 (with our trained routers reproducing a similar caching pattern as reported), TGATE yields 43.6 FID, and our DiffCR achieves 28.6 FID. These results show that our method performs comparably to or better than caching-based baselines, and it can be further combined with them to achieve an additional 15 $\sim$ 30% latency reduction.

I. Human Preference Score for Inpainting

In Sec. 4.4, we utilized a computer vision model to estimate likely human preferences and evaluate the ability of models to generate high-quality, contextually relevant images for the T2I task. Here, we also provide the evaluation for inpainting tasks. Specifically, we generated 2K samples for the inpainting task and used HPSv2 [11] to assess human preferences for images produced by different methods. As shown in Tab. 2, for inpainting tasks, we applied all compression methods to Lazy Diffusion (LD) [7]. DiffCR achieves a higher human preference score of 2.181/0.263 compared to previous compression methods, ToMe [1] and vanilla MoD [8], respectively.

Table 2. Human Preference Score (HPS) (↑) comparison of the proposed DiffCR with baselines for the inpainting task.

Methods	DiT C.R.	HPS Score
RegenerateImage	0%	21.056
RegenerateCrop	0%	19.466
Lazy Diffusion (LD)	0%	20.464
LD w/ ToMe	30%	18.187
LD w/ MoD	30%	20.105
LD w/ DiffCR	30%	20.368

Table 3. Ablation study on the impact of different compression ratios with a batch size of 16.

Metrics\Ratios	0%	10%	30%	50%	70%	90%
FID Score (↓)	27.80	27.53	28.64	28.57	28.44	29.21
CLIP Score (↑)	16.23	16.28	16.44	16.37	16.37	16.37
T2I Latency (s)	11.90	11.16	10.31	9.23	8.19	7.12

J. Ablation Analysis on Compression Ratios

In this work, we target lower latency as a step toward edge deployment. To analyze the effect of varying compression ratios, we conducted an ablation study using the PixArt-Σ model on the MS-COCO-30K dataset [3]. Notably, $1/3$ of the timesteps were allocated to full-model inference to preserve accuracy. The results in the table below show that our method scales effectively to larger compression ratios, with only a slight increase in FID (< 1). A 30% compression ratio was previously selected for challenging generation tasks to maintain accuracy while building upon existing state-of-the-art efficient methods.

K. Is MSE Loss Alone Sufficient?

We found that simply using the MSE loss effectively guides ratios toward the target without additional regularization, so we fixed it to MSE loss, but other loss functions may also work well. In addition, although we did not enforce binary prediction, the routers tend to learn a polarized distribution in some layers, separating important tokens from unimportant ones, with the learned ratios aligning accordingly, as shown in Fig. 3.

References

[1] Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4599–4603, 2023. 5

[2] Junsong Chen, Chongjian Ge, Enze Xie, Yue Wu, Lewei Yao, Xiaozhe Ren, Zhongdao Wang, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-sigma: Weak-to-strong training of diffusion transformer for 4k text-to-image generation. *arXiv preprint arXiv:2403.04692*, 2024. 2, 4

[3] 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. 5

[4] Haozhe Liu, Wentian Zhang, Jinheng Xie, Francesco Faccio, Mengmeng Xu, Tao Xiang, Mike Zheng Shou, Juan-Manuel Perez-Rua, and Jürgen Schmidhuber. Faster diffusion via temporal attention decomposition. *arXiv preprint arXiv:2404.02747*, 2024. 5

[5] Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-cache: Accelerating diffusion transformer via layer caching. *arXiv preprint arXiv:2406.01733*, 2024. 5

[6] Giovane CM Moura, John Heidemann, Ricardo de O Schmidt, and Wes Hardaker. Cache me if you can: Effects of dns time-to-live. In *Proceedings of the Internet Measurement Conference*, pages 101–115, 2019. 5

[7] Yotam Nitzan, Zongze Wu, Richard Zhang, Eli Shechtman, Daniel Cohen-Or, Taesung Park, and Michaël Gharbi. Lazy diffusion transformer for interactive image editing. *arXiv preprint arXiv:2404.12382*, 2024. 2, 4, 5

[8] David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024. 1, 5

[9] Hongjie Wang, Difan Liu, Yan Kang, Yijun Li, Zhe Lin, Niraj K Jha, and Yuchen Liu. Attention-driven training-free efficiency enhancement of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16080–16089, 2024. 5

[10] Kai Wang, Yukun Zhou, Mingjia Shi, Zhihang Yuan, Yuzhang Shang, Xiaojiang Peng, Hanwang Zhang, and Yang You. A closer look at time steps is worthy of triple speed-up for diffusion model training. *arXiv preprint arXiv:2405.17403*, 2024. 3

[11] Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023. 5

[12] Mengwei Xu, Mengze Zhu, Yunxin Liu, Felix Xiaozhu Lin, and Xuanzhe Liu. Deepcache: Principled cache for mobile deep vision. In *Proceedings of the 24th annual international conference on mobile computing and networking*, pages 129–144, 2018. 5