NA-LR: Noise-Adaptive Low-Rank Parameterisation for Efficient Diffusion Models

Jingyuan Wang
Department of Computing,
Imperial College London,
London, UK
`jingyuan.wang124@imperial.ac.uk`

Federico Ottomano
Department of Chemistry,
Imperial College London,
London, UK
`f.ottomano@imperial.ac.uk`

Yingzhen Li
Department of Computing,
Imperial College London,
London, UK
`yingzhen.li@imperial.ac.uk`

Abstract

Scaling diffusion models delivers state-of-the-art image quality but at substantial training and inference cost. Low rank parameterisation effectively attacks both of these bottlenecks. We empirically find that the effective ranks of weights and gradients in diffusion models are *noise-dependent*: as noise increases, updates concentrate in lower-dimensional subspaces. This makes static low-rank parameterisations inefficient, underallocating capacity to low-noise (hard) steps while overallocating to high-noise (easy) ones. We propose ***Noise-Adaptive Low-Rank parameterisation*** (NA-LR) that uses a nested rank schedule: harder timesteps with less noise are allocated more rank slices, while easier ones with high noise require fewer. This is implemented via per-sample masking at training time and structured rank slicing at inference time. Coupled with a timestep curriculum learning, which initially restricting the timestep sampling range, we reduce both inference and training compute by achieving training-time slicing. On DiT-S/2 across CIFAR-10 and CelebA, our method improves FID by 15–26% over static low-rank at matched compute, cuts inference FLOPs by up to 25% at matched parameters with negligible FID change, and lowers training FLOPs by up to 12%.

1 Introduction

Diffusion models have rapidly advanced image generation quality [1–4], aided by the general lesson that models with more parameters and computing improve generation performance [5]. However, the iterative denoising process and high-capacity backbones make both pretraining and deployment expensive. Low rank parameterisation, which factorised the original dense weight matrix into two low rank matrices, addresses both of these two bottlenecks. However, training low-rank networks from scratch normally compromises quality [6–9].

Traditional low rank parametersiation for diffusion models treats all timesteps equally, assigns the same rank to all timesteps. However, our analysis reveals that this uniform treatment is inefficient: the effective rank of both weights and gradients is **noise-dependent**: as noise levels increase, they concentrate in lower-dimensional subspaces (Figure 1). This imply that high-noise (early) steps require less expressive capacity, while low-noise (late) steps require more (see Appendix B for

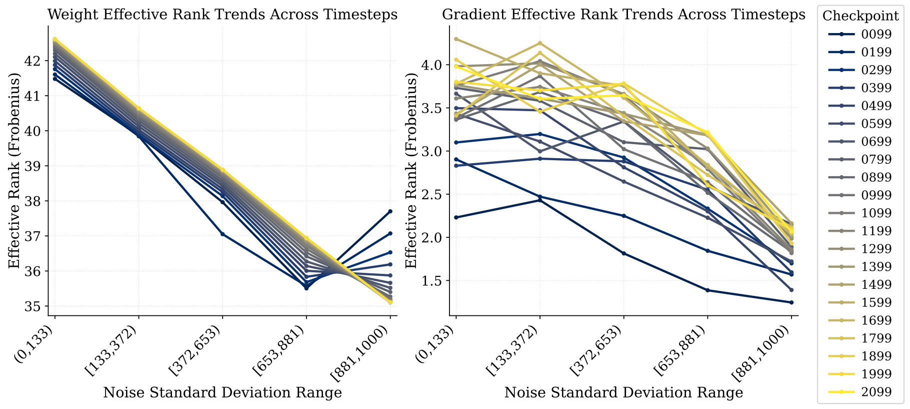


Figure 1: Effective rank is noise dependent: as noise increases, updates concentrate in lower-dimensional subspaces, making uniform rank inefficient.

details). This reveals that traditional uniform-rank parameterisations **over-parameterise** early steps and **under-parameterise** late ones.

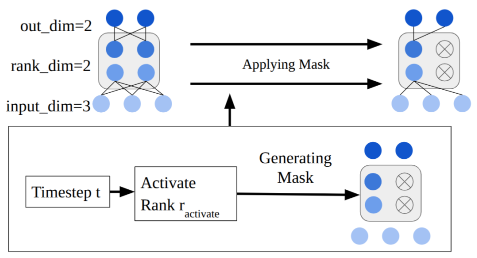
To exploit the noise-dependent low-rank structure, we propose ***Noise-Adaptive Low-Rank parameterisation*** (NA-LR), an architecture-agnostic approach that requires no changes to the training pipeline or optimisation procedure. NA-LR activates a variable number of rank slices per input timestep via a nested schedule. During training, we use per-sample masking to gate slices while preserving a static computation graph; at inference, we slice the low-rank factors directly, reducing FLOPs without modifying the sampler.

To further reduce training compute, we adopt a timestep curriculum learning technique that initially restricts sampling to high-noise (easier) ranges and progressively expands to low-noise (harder) ranges, thereby limits the range of activated parameters within each curriculum, enabling training-time weight matrices slicing.

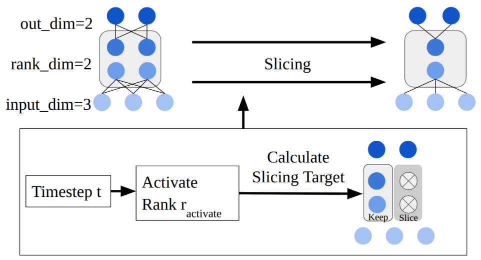
On CIFAR-10 (32×32) and CelebA (64×64) with DiT-S/2 [4], our approach consistently strengthens the efficiency–quality Pareto front relative to static low-rank training. At matched inference compute (ISO-Compute), FID improves by 15–26%; at matched parameters (ISO-Param), inference FLOPs drop by up to 25% with negligible FID change. The curriculum learning enables up to 12% total training FLOP reduction without degrading final quality.

2 Method

2.1 Noise Adaptive Low Rank Parameterisation



(a) Training (timestep-dependent masking).



(b) Inference (structured rank slicing).

Figure 2: Illustration of Noise-adaptive low-rank method during training and inference.

Low-rank parameterisation. For any linear layer with weight $W \in \mathbb{R}^{m\times n}$ we use a rank–r factorisation

$$W = UV^\top, \quad U \in \mathbb{R}^{m\times r},\ V \in \mathbb{R}^{n\times r}. \tag{1}$$

We replace the Q/K/V/Out projections and MLP in/out in DiT blocks with (1). To stabilise training, we adopt spectral initialisation [10] and light orthogonality regularisation [11] on factorised layers.

Nested noise-adaptive rank schedule. Let $t \in \{1, \dots, T\}$ be the diffusion timestep and r_{max} the maximum activatable rank for a layer. We activate only the first r_{act} channels of the rank space, where

$$r_{\text{act}} \;=\; \Phi(t) \;=\; r_{\min} \;+\; \left\lceil \big(r_{\text{max}} - r_{\min}\big)\, f(t) \right\rceil, \quad r_{\min} = \lceil \rho_{\min}\, r_{\text{max}} \rceil, \;\; f(t) \in [0,1], \tag{2}$$

$\Phi(t)$ is the rank schedule and its *monotonically decreasing* with t (high noise $\Rightarrow$ smaller k, with the boundary conditions $\Phi(0) = r_{\text{min}}$ and $\Phi(T) = r_{\text{max}}$. The schedule induces a *nested* property:$t_1 < t_2 \Rightarrow \Phi(t_1) \geq \Phi(t_2)$, so slices required at higher noise are a subset of those required at lower noise. We use a logistic profile for $f(t)$ in practice; linear alternatives are reported in the Appendix C.3.

Here $\rho_{\min} \in [0,1]$ is a hyperparameter controlling the minimum active rank at the noisiest timesteps, defined as $r_{\min} = \lfloor \rho_{\min} r \rfloor$. Unless stated, we set $\rho_{\min} = 0.4$ based on a small validation sweep; see Appendix C.1.1 for selection criteria and sensitivity.

Training-time masking. In standard diffusion training, each sample's timestep t_i is drawn uniformly from $[1, T]$, so NA-LR would activate different ranks per sample. To keep a static computation graph and efficient batching, we apply a per-sample *binary mask* over rank channels that zeros out inactive rank slices. For a low-rank layer $W = UV^\top$ with rank r,we define a *binary mask* $S \in \{0,1\}^{r \times B}$ for a minibatch of size B as

$$S_{j,i} = \mathbf{1}[\, j \leq k(t_i)\,], \tag{3}$$

where $k(t_i)$ is denotes the active rank for timestep t_i. Given inputs $X \in \mathbb{R}^{n \times B}$, the output is computed as

$$Y \;=\; U\big((V^\top X) \odot S\big) \;\in\; \mathbb{R}^{m \times B} \tag{4}$$

where $\odot$ indicates elementwise multiplication along the rank dimension. This formulation enables per-sample rank gating while preserving a single static forward and backward computation graph.

Inference-time slicing. At sampling time, all examples in a minibatch share the same timestep t and thus the same active rank $r_{\text{act}} = \Phi(t)$. We therefore *slice off* the inactive rank dimention and compute only with the corresponding active dimention, without modifying the sampling procedure.

Let $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{n \times r}$. For input $X \in \mathbb{R}^{n \times B}$,

$$Y = U_{:,\,1:r_{\text{act}}} \left(V^\top_{:,\,1:r_{\text{act}}} X \right) \in \mathbb{R}^{m \times B}. \tag{5}$$

The per-sample computational cost is then given by

$$\text{FLOPs}_{\text{LR}}(r_{\text{act}}) = 2r_{\text{act}}(m+n) \quad \text{vs.} \quad 2r_{\text{max}}(m+n) \text{ (static low-rank).} \tag{6}$$

This gives a linear reduction with theoretical speedup $r_{\text{max}}/r_{\text{act}}$, and is independent of batch size B.

2.2 Curriculum for Training-time Slicing

Typical diffusion model training samples timesteps uniformly, so no subset of parameters can be safely sliced within a batch. To enable training-time savings, we adopt a short *timestep curriculum* inspired by Kim et al. [12]. Timesteps are grouped by log-SNR into N clusters $G_0, \dots, G_{N-1}$ (low $\rightarrow$ high noise). Training starts with the highest-noise group G_{N-1} and progressively adds lower-noise groups, ensuring early phases activate only high-noise slices and allowing true training time slicing.

Progression is triggered when the exponential moving average (EMA) of batch loss plateaus, extending the curriculum adaptively, as longer curriculum phases yield greater FLOPs savings. At each stage, we apply *progressive uniform sampling*—previous groups retain full probability mass while the new group receives the remainder—to avoid forgetting. A brief learning-rate boost and optimiser reset stabilise transitions. Full implementation details are provided in Appendix C.3.

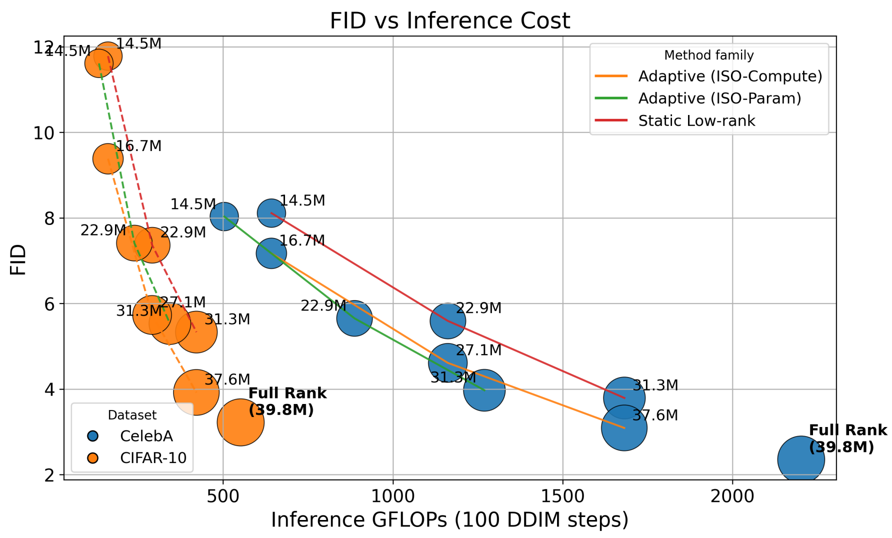


Figure 3: Inference cost vs. FID for full-rank, fixed low-rank, and adaptive low-rank (ISO-Compute/ISO-Param) models on CIFAR-10 and CelebA. Markers show parameter counts.

3 Experiments and Results

3.1 Settings

We evaluate our approach on CIFAR-10 (32×32, class-conditional) and CelebA (64×64, unconditional) using the DiT-S/2 architecture [4]. For each low-rank layer, we set the rank such that the number of trainable parameters is reduced to 25%, 50%, or 75% of the original size, and then apply NA-LR to these layers. Detailed training configurations are provided in Appendix D.1.

To enable a fair comparison with a static low-rank baseline of rank r_{base}, we define two NA-LR presets: **ISO-Compute:** We select the maximum activatable rank $r_{\max}$ such that the schedule-averaged active rank over sampling steps $\mathcal{S}$ satisfies $\frac{1}{|\mathcal{S}|}\sum_{t\in\mathcal{S}}\Phi(t) = r_{\text{base}}$ (100 DDIM steps unless otherwise noted). This configuration matches inference FLOPs on average. Since the adaptive schedule activates fewer channels at high noise and more at low noise, the total parameter count may differ slightly. **ISO-Param:** Set $r_{\max} = r_{\text{base}}$ to match the parameter count of the baseline. In this case, realised FLOPs are typically lower because $\Phi(t) \leq r_{\text{base}}$ and decreases with higher noise levels. Details for each preset model are reported in Appendix C.2.

3.2 Efficiency and Generation Quality Performance

Table 1 (CIFAR-10) reports the headline comparison under a representative low-rank configuration. At matched inference compute (ISO-Compute), the adaptive model improves FID by **26%** (5.33 $\rightarrow$ 3.92) with only a modest increase in parameters. At matched parameters (ISO-Param), it reduces inference FLOPs by approximately **19%** (421 $\rightarrow$ 343 GFLOPs) with only a marginal change in FID (5.33 $\rightarrow$ 5.53). Precision and recall follow similar trends, with recall (diversity) being more sensitive to aggressive rank reduction.

Figure 3 illustrates the efficiency–quality trade-off across presets and datasets: NA-LR consistently forms a stronger Pareto front than static low-rank baselines.

Method	Inference FLOPs (10^9)	Params (M)	FID ↓	Precision ↑	Recall ↑	Train FLOPs (10^{18})
Full Rank	551.674	39.80	3.22	0.82	0.79	1.677
Static Low-rank (75%)	421.256	31.34	5.33	0.76	0.67	1.281
Adaptive (75%, ISO-Compute)	421.159	37.60	**3.92**	**0.80**	**0.73**	1.369
Adaptive (75%, ISO-Param)	**343.214**	31.34	5.53	0.75	0.66	**1.149**

Table 1: CIFAR-10 (32×32), DiT-S/2. Inference GFLOPs measured for 100 DDIM steps.

3.3 Curriculum Learning Efficiency Analysis

The proposed curriculum enables training-time slicing, effectively reducing end-to-end training FLOPs. Across presets and datasets, total training compute decreases by **5–12%** with no degradation in final FID (see Appendix E.1 for full tables and schedules).

4 Conclusion

In this paper, we uncover a clear *noise-dependent* low-rank structure in diffusion models and exploit it with a *Noise-Adaptive Low-Rank* (NA-LR) scheme that preserves a static computational graph, enables inference-time rank slicing, and—via a brief timestep curriculum—achieves training-time slicing. NA-LR consistently strengthens the FID–compute Pareto front for low-rank parameterisation, improving both training and inference efficiency. Overall, NA-LR provides a simple yet effective route toward more compute-efficient generative diffusion systems.

References

[1] Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models. In: Advances in Neural Information Processing Systems; 2020. .

[2] Dhariwal P, Nichol AQ. Diffusion Models Beat GANs on Image Synthesis. In: Advances in Neural Information Processing Systems; 2021. Available from: `https://arxiv.org/abs/2105.05233`.

[3] Rombach R, Blattmann A, Lorenz D, Esser P, Ommer B. High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2022. .

[4] Peebles W, Xie S. Scalable diffusion models with transformers. In: Proceedings of the IEEE/CVF international conference on computer vision; 2023. p. 4195-205.

[5] Kaplan J, McCandlish S, Henighan T, Brown TB, Chess B, Child R, et al. Scaling laws for neural language models. arXiv preprint arXiv:200108361. 2020.

[6] Han A, Li J, Huang W, Hong M, Takeda A, Jawanpuria PK, et al. SLTrain: a sparse plus low rank approach for parameter and memory efficient pretraining. Advances in Neural Information Processing Systems. 2024;37:118267-95.

[7] He X, Fang JZ, Zheng J, Piramuthu R, Bansal M, Ordonez V, et al. Efficient Low-Rank Diffusion Model Training for Text-to-Image Generation.

[8] Huh M, Cheung B, Bernstein J, Isola P, Agrawal P. Training neural networks from scratch with parallel low-rank adapters. arXiv preprint arXiv:240216828. 2024.

[9] Rao Kamalakara S, Locatelli A, Venkitesh B, Ba J, Gal Y, Gomez AN. Exploring Low Rank Training of Deep Neural Networks. arXiv preprint arXiv:220913569. 2022.

[10] Khodak M, Tenenholtz N, Mackey L, Fusi N. Initialization and Regularization of Factorized Neural Layers. In: Proceedings of the 9th International Conference on Learning Representations (ICLR); 2021. Available from: `https://openreview.net/forum?id=FH1oFoNzS1h`.

[11] Po R, Yang G, Aberman K, Wetzstein G. Orthogonal adaptation for modular customization of diffusion models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition; 2024. p. 7964-73.

[12] Kim JY, Go H, Kwon S, Kim HG. Denoising task difficulty-based curriculum for training diffusion models. arXiv preprint arXiv:240310348. 2024.

[13] Sohl-Dickstein J, Weiss E, Maheswaranathan N, Ganguli S. Deep unsupervised learning using nonequilibrium thermodynamics. In: International Conference on Machine Learning; 2015. .

[14] Nichol AQ, Dhariwal P. Improved denoising diffusion probabilistic models. arXiv preprint arXiv:210209672. 2021.

[15] Saharia C, Ho J, Chan W, et al. Image super-resolution via iterative refinement. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2022.

[16] Yi X, Wu Z, Xu Q, Zhou P, Lim JH, Zhang H. Diffusion time-step curriculum for one image to 3d generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2024. p. 9948-58.

[17] Balaji Y, Nah S, Huang X, Vahdat A, Song J, Zhang Q, et al. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. arXiv preprint arXiv:221101324. 2022.

[18] Kim M, Ki D, Shim SW, Lee BJ. Adaptive non-uniform timestep sampling for diffusion model training. arXiv preprint arXiv:241109998. 2024.

[19] Go H, Lee Y, Lee S, Oh S, Moon H, Choi S. Addressing negative transfer in diffusion models. Advances in Neural Information Processing Systems. 2023;36:27199-222.

[20] Hang T, Gu S, Li C, Bao J, Chen D, Hu H, et al. Efficient diffusion training via min-snr weighting strategy. In: Proceedings of the IEEE/CVF international conference on computer vision; 2023. p. 7441-51.

[21] Ma Q, Ning X, Liu D, Niu L, Zhang L. Decouple-Then-Merge: Finetune Diffusion Models as Multi-Task Learning. In: Proceedings of the Computer Vision and Pattern Recognition Conference; 2025. p. 23281-91.

[22] Nguyen Q, Hein M. The loss surface of deep and wide neural networks. In: International conference on machine learning. PMLR; 2017. p. 2603-12.

[23] Li D, Ding T, Sun R. Over-parameterized deep neural networks have no strict local minima for any continuous activations. arXiv preprint arXiv:181211039. 2018.

[24] Du S, Lee J, Li H, Wang L, Zhai X. Gradient descent finds global minima of deep neural networks. In: International conference on machine learning. PMLR; 2019. p. 1675-85.

[25] Sui Y, Yin M, Gong Y, Xiao J, Phan H, Yuan B. ELRT: Efficient Low-Rank Training for Compact Convolutional Neural Networks. arXiv preprint arXiv:240110341. 2024.

[26] Idelbayev Y, Carreira-Perpinán MA. Low-rank compression of neural nets: Learning the rank of each layer. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition; 2020. p. 8049-59.

[27] Wu Y, Shi Y, Wei J, Sun C, Yang Y, Shen HT. DiffLoRA: Generating Personalized Low-Rank Adaptation Weights with Diffusion. arXiv preprint arXiv:240806740. 2024. Available from: `https://arxiv.org/abs/2408.06740`.

[28] Rigo A, Stornaiuolo L, Martino M, Lepri B, Sebe N. ESPLoRA: Enhanced Spatial Precision with Low-Rank Adaption in Text-to-Image Diffusion Models for High-Definition Synthesis. arXiv preprint arXiv:250413745. 2025. Available from: `https://arxiv.org/abs/2504.13745`.

[29] Staniszewski Ł, Zaleska K, Deja K. Low-Rank Continual Personalization of Diffusion Models. arXiv preprint arXiv:241004891. 2024. Available from: `https://arxiv.org/abs/2410.04891`.

[30] He X, Fang JZ, Zheng J, Piramuthu R, Bansal M, Ordonez V, et al.. Efficient Low-Rank Diffusion Model Training for Text-to-Image Generation; 2023. ICLR 2024 Conference Withdrawn Submission. Available from: `https://openreview.net/forum?id=edx7LTufJF`.

[31] Wang X, Chen Y, Zhu W. A survey on curriculum learning. IEEE transactions on pattern analysis and machine intelligence. 2021;44(9):4555-76.

[32] Croitoru FA, Hondru V, Ionescu RT, Sebe N, Shah M. Curriculum direct preference optimization for diffusion and consistency models. In: Proceedings of the Computer Vision and Pattern Recognition Conference; 2025. p. 2824-34.

[33] Kim J, Lee J. Strategic data ordering: Enhancing large language model performance through curriculum learning. arXiv preprint arXiv:240507490. 2024.

[34] Chen X, Lu J, Kim M, Zhang D, Tang J, Piché A, et al. Self-Evolving Curriculum for LLM Reasoning. arXiv preprint arXiv:250514970. 2025.

[35] Han Y, Huang G, Song S, Yang L, Wang H, Wang Y. Dynamic neural networks: A survey. IEEE transactions on pattern analysis and machine intelligence. 2021;44(11):7436-56.

[36] Han Y, Liu Z, Yuan Z, Pu Y, Wang C, Song S, et al. Latency-aware unified dynamic networks for efficient image recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2024;46(12):7760-74.

[37] Han Y, Pu Y, Lai Z, Wang C, Song S, Cao J, et al. Learning to weight samples for dynamic early-exiting networks. In: European conference on computer vision. Springer; 2022. p. 362-78.

[38] Campos V, Jou B, Giró-i Nieto X, Torres J, Chang SF. Skip rnn: Learning to skip state updates in recurrent neural networks. arXiv preprint arXiv:170806834. 2017.

[39] Jacobs RA, Jordan MI, Nowlan SJ, Hinton GE. Adaptive mixtures of local experts. Neural computation. 1991;3(1):79-87.

[40] Cai W, Jiang J, Wang F, Tang J, Kim S, Huang J. A survey on mixture of experts in large language models. IEEE Transactions on Knowledge and Data Engineering. 2025.

[41] Sun H, Lei T, Zhang B, Li Y, Huang H, Pang R, et al. Ec-dit: Scaling diffusion transformers with adaptive expert-choice routing. arXiv preprint arXiv:241002098. 2024.

[42] Jia W, Huang M, Chen N, Zhang L, Mao Z. Dˆ 2iT: Dynamic Diffusion Transformer for Accurate Image Generation. In: Proceedings of the Computer Vision and Pattern Recognition Conference; 2025. p. 12860-70.

[43] Lin B, Tang Z, Ye Y, Cui J, Zhu B, Jin P, et al. Moe-llava: Mixture of experts for large vision-language models. arXiv preprint arXiv:240115947. 2024.

[44] Krizhevsky A, Hinton G, et al. Learning multiple layers of features from tiny images. 2009.

[45] Loshchilov I, Hutter F. Fixing Weight Decay Regularization in Adam. arXiv preprint arXiv:171105101. 2017.

[46] facebookresearch. fvcore: A lightweight core library for FAIR computer vision research; 2025. Accessed: 2025-09-07. `https://github.com/facebookresearch/fvcore`.

[47] Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09; 2009. .

– Supplementary Materials –

Appendix Contents

A Related Work

Diffusion models. Denoising diffusion models (DDMs) learn data distributions by reversing a fixed noising process [1, 13]. Subsequent work improved training and sampling with refined noise schedules and objectives [14], and established state-of-the-art image generation compared to GANs on standard benchmarks [2, 15]. Latent Diffusion Models compress images with a VAE and run diffusion in the latent space, improving efficiency without sacrificing fidelity [3]. Architecturally, Diffusion Transformers (DiT) substitute U-Nets with ViT-style backbones and conditional layer norm, showing strong scaling and achieving competitive FID on ImageNet [4].

Timesteps as tasks. A distinctive feature of diffusion models is the explicit timestep index: early (high-noise, high-t) steps primarily establish global structure, whereas late (low-noise, low-t) steps refine fine details [16, 17]. The different timestep task focus result in distinct training dynamics for each diffusion model timestep, including gradient variance and convergence speed [12, 18] . Treating timesteps as distinct subproblems reveals multi-task interference during training—gradients from different timesteps can conflict and slow convergence [19]. Prior work mitigates this with timestep-aware loss reweighting or sampling [20], gradient projection/decorrelation across timestep-defined tasks [19], and architectural decoupling into experts specialised for disjoint timestep groups [21]. These findings motivate allocating model capacity and scheduling optimisation by timestep.

While this literature characterises conflict and specialisation across timesteps, the relationship between *intrinsic rank* and timestep remains largely unexplored. In this work, we provide an empirical study of how intrinsic rank interacts with diffusion timesteps by (i) analysing per-timestep group intrinsic rank for gradient and weight matrices, (ii) analysing loss sensitivity to rank across the denoising trajectory, and (iii) measuring subspace overlap between per-timestep gradients. Our results inform the design of noise-aware, rank-adaptive training strategies used in our method.

Low-rank parameterisation and pretraining. Low-rank parameterisations factorise weight matrices to reduce parameters and FLOPs while providing implicit regularisation. However, directly training low-rank models often underperforms full-rank counterparts due to optimisation challenges and rank collapse [6–9]. Explanations include the loss-smoothing benefits of overparameterisation [22–24]. To mitigate these issues, prior work proposed spectral initialisation to match full-rank statistics [10], orthogonality and tensor decompositions (e.g., Tucker-2 for convolutions) to better preserve structure [25], and hybrid schedules that warm start or intermittently train in full rank before projecting to low rank [9]. Another line of work obtains low-rank factorised networks via post-hoc compression: train a full-rank model, then apply rank truncation (e.g., SVD) or knowledge distillation to produce a low-rank proxy, which serves as initialisation for low-rank training process. This approach typically trades training-time efficiency for stronger final accuracy [26].

Low-rank diffusion models. Most low-rank efforts in diffusion focus on fine-tuning pretrained full-rank models with low-rank adapters or compression, reporting substantial memory savings and competitive downstream quality [27–29]. ELR-Diffusion follows a two-stage pipeline—distilling a low-rank U-Net from a large base model and then fine-tuning—achieving notable parameter and memory reductions but incurring FID regressions relative to full rank, especially without distillation [30]. In contrast, low-rank *pretraining from scratch* for diffusion remains underexplored and is the focus of our study, where we address optimisation difficulties without relying on a full-rank teacher.

Curriculum learning for diffusion. Curriculum learning exposes models to progressively harder data or tasks to smooth optimisation and accelerate convergence [31]. Recent applications to diffusion include timestep-ordered curricula for 3D generation [16], curricula based on denoising difficulty or convergence [12], and preference-optimisation curricula ordered by preference-gap difficulty [32]. Broader adaptive curricula in language tasks (e.g., ordering by prompt difficulty) similarly report convergence and performance gains [33, 34]. Our approach leverages a timestep-aware curriculum tailored to noise adaptive low-rank training: we deliberately extend the curriculum horizon while safeguarding final performance, yielding larger training FLOP reductions than curricula designed primarily for accuracy gains.

Dynamic neural networks with diffusion. Dynamic architectures adjust computation per input to trade off efficiency and quality [35], using mechanisms such as width/depth adaptation [36, 37] or temporal/skipped computation [38]. Mixture-of-Experts (MoE) routing scales parameters at near-constant per-token compute [39, 40] and has been adapted to diffusion backbones. For instance, MoE-augmented DiTs activate more experts for semantically detailed regions [41], while dynamic patching varies patch size spatially to reduce FLOPs without degrading image quality [42, 43]. Our approach is orthogonal: we exploit *timestep-dependent* low-rank structure and allocate capacity monotonically with timestep via nested rank slices. Unlike MoE or spatial gating, NA-LR introduces no routers, keeps the sampler/optimiser unchanged, and preserves a static batch-wise computation graph (per-sample masks only), enabling structured slicing at inference and during a brief curriculum.

Positioning. In sum, our work sits at the intersection of (i) timestep-aware training and specialisation [19–21], (ii) low-rank parameterisation and its optimisation remedies [9, 10, 25], (iii) curriculum learning tailored to diffusion [12, 16, 32], and (iv) dynamic computation for generative models [41, 42]. We differ from prior low-rank diffusion work by *pretraining* low-rank models from scratch and by coupling a timestep-dependent dynamic rank with a diffusion-aware curriculum to improve the efficiency–performance trade-off.

B Empirical Observation Detail

We examine timestep-dependent low-rank structure in diffusion models, analysing both weight space and gradient space. Our study focuses on three questions:

1. How do the effective ranks of weights and gradients evolve over the training process for different timestep groups?
2. How much overlap is there between the principal gradient subspaces of different timestep groups?
3. How sensitive is each timestep group's loss to low-rank parameterisation?

B.1 Empirical Observation Experiment Setting

All empirical observation experiments in this chapter use the DiT-S/2 architecture [4] with a cosine noise schedule [14] on the CIFAR-10 dataset [44].

To analyse behaviour across different regions of the diffusion trajectory, we cluster timesteps into five groups according to their signal-to-noise ratio (SNR), where we use dynamic programming and set the maximum inner class log SNR difference as the clustering cost. In the context of diffusion models, the SNR at timestep t quantifies the relative strength of the underlying image signal to the added noise:

$$\mathrm{SNR}(t) = \frac{\alpha_t}{1-\alpha_t}.$$

where

$$\alpha_t = \prod_{s=1}^{t}(1-\beta_s).$$

As t increases, more noise is injected and α_t decreases; consequently, $\mathrm{SNR}(t)$ monotonically decreases. The SNR for different timestep groups under the cosine noise schedule is shown in Figure 4.

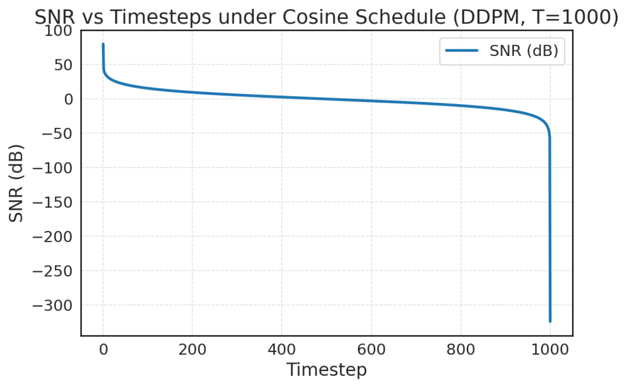


Figure 4: Signal-to-noise ratio across timesteps under the cosine schedule.

By clustering similar timesteps, we can reduce variance within groups and facilitate clearer comparisons across distinct SNR regimes. We form five timestep groups via dynamic programming, the resulting groups are shown in Figure 5. This same timestep clustering is used throughout all empirical observation experiments in this section.

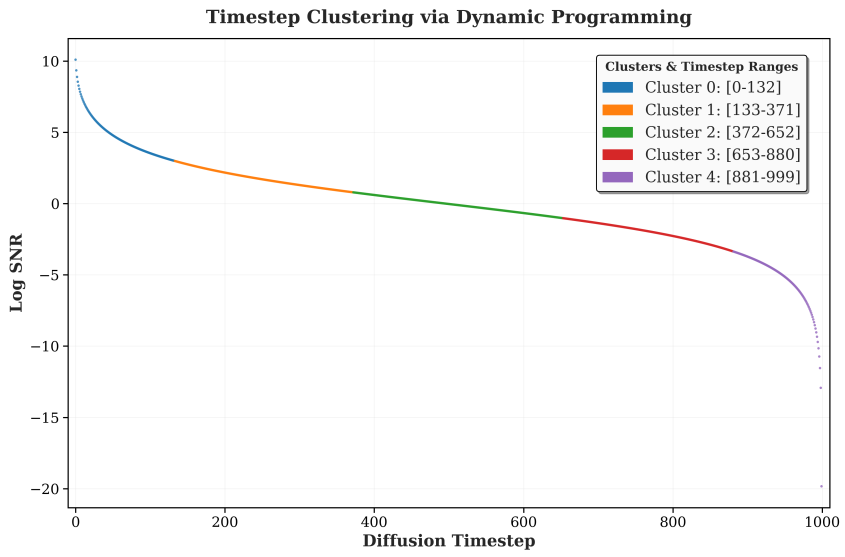


Figure 5: Timestep clustering via dynamic programming based on log-SNR.

B.2 Gradient Analysis

In this section, we analyse how the low-rank property of the gradient changes as training progresses across different timestep groups. We train the DiT-S/2 network until convergence. We sample the gradient of each timestep group every 100 epochs. To reduce mini-batch noise, we estimate the gradient by accumulating the loss over 10 batches, each containing 512 samples; we then backpropagate the loss and evaluate the gradient's effective rank and Frobenius norm. The results are shown in Fig. 6.

As shown in Figure 6, we find a clear trend: as timestep increases, both the effective rank and Frobenius norm of the gradient decrease. Additionally, as training progresses, this trend becomes more pronounced. The low-rankness of the gradient suggests that the update subspace of higher timesteps lies in a low-rank subspace. This motivates us to test whether the solutions for higher timestep groups also lie in progressively lower-rank subspaces.

B.3 Model Weight Analysis

Building upon the observation that the updates for higher timesteps live in progressively lower-rank subspaces, we proceed to confirm whether the converged solutions of higher timestep groups also lie in increasingly lower-rank spaces.

To test this, we train five models, each trained exclusively on timesteps from one timestep group defined in Section B.1, using the same model size and dataset, until convergence. We evaluate the average Frobenius norm and effective rank of all the weight matrices in the model. The results are shown in Figure 7.

From Figure 7, we observe that, despite an initial noisy training stage, models trained on higher-timestep groups have weight matrices that gradually converge to lower-rank solutions compared with models trained on lower-timestep groups. The same trend is observed for the Frobenius norms of the weight matrices, indicating a simpler solution found by models trained on higher timesteps. This shows that, despite using the same model and dataset, models trained on higher timesteps converge to lower-rank solutions than those trained on lower-timestep groups.

B.4 Timestep Loss Sensitivity Analysis

Based on the observations from previous sections, we directly compare the per-timestep loss between a full-rank model and a low-rank parameterised model that reduces 90% of the parameters in each

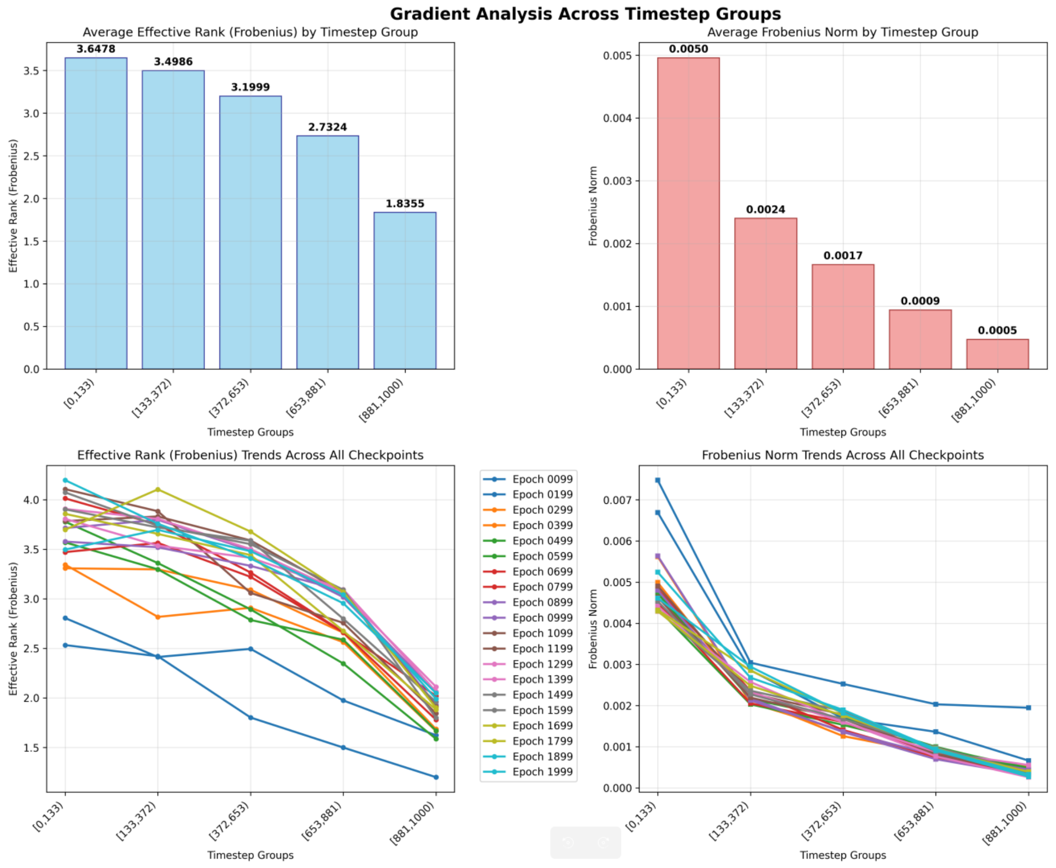


Figure 6: Effective rank and Frobenius norm of the gradients of different timestep groups across checkpoints during DiT-S/2 model training.

linear layer, which reduces the total parameter count by 76%. We train both to convergence and compare the loss for each timestep; the results are shown in Figure 8.

From Figure 8, we find that, despite having only 24% of the parameters and 15% of the FLOPs, the loss differences are concentrated in the low-timestep groups, with the high-timestep groups exhibiting a negligible amount of loss difference. This indicates that higher timesteps correspond to easier

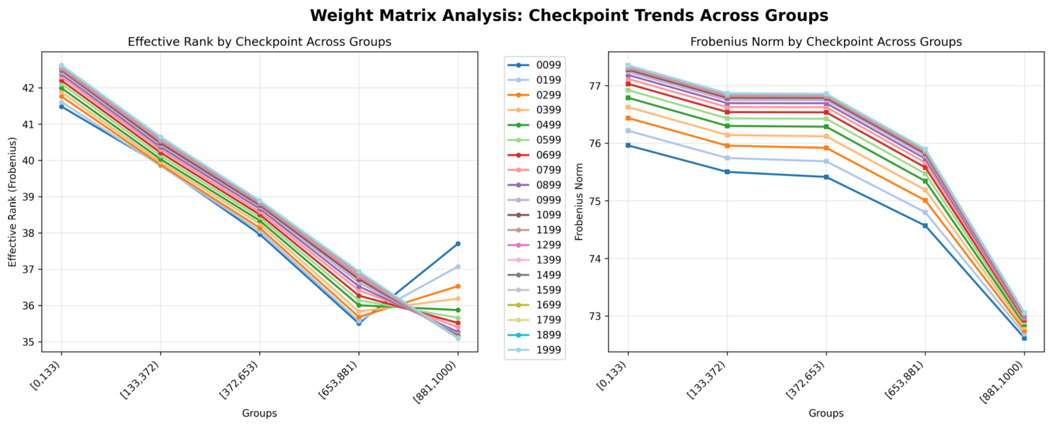


Figure 7: Effective rank and Frobenius norm of the weight matrices of DiT-S/2 models trained on different timestep groups.

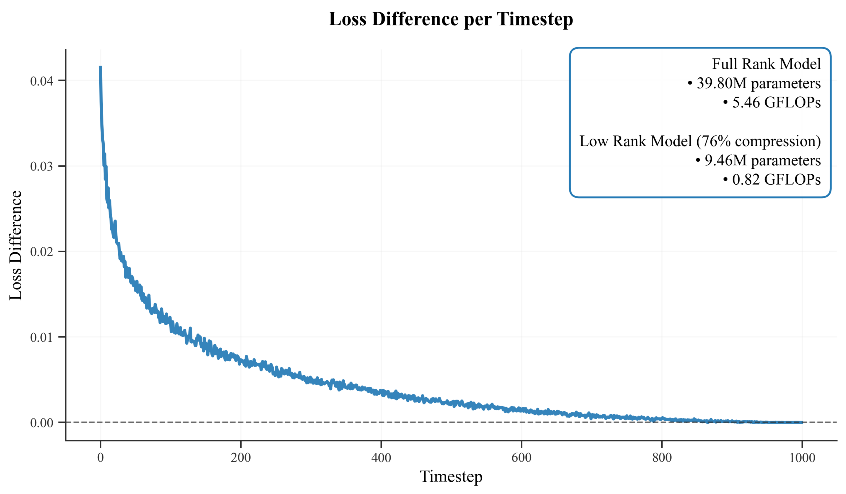


Figure 8: Loss difference between a full-rank model and a low-rank parameterised model (76% compression).

prediction tasks that can be approximated very well with very low-rank parameterisation, whereas lower timesteps require higher rank to approximate. This directly motivates noise-adaptive low-rank parameterisation.

B.5 Similarity Analysis

Based on the previous results, we take a step further to analyse the similarities of gradients between each timestep group throughout the training process.

B.6 Similarity Measurement

We focus primarily on directional similarity. As both gradient and weight matrices are high-dimensional, directly evaluating the cosine similarity between flattened matrices yields values near 0 or 1 due to the high-dimensional nature of the spaces.

Thus, we evaluate the average principal angles between the subspaces spanned by the column spaces of the two gradient matrices. This captures how each layer's output-side directions want to move, and is invariant to linear reparameterisation of the input, helping us compare gradients across different timesteps.

To calculate the similarity between a given pair of matrices G_1 and G_2, we first perform singular value decomposition of both matrices:

$$G_1 = U_1 \Sigma_1 V_1^\top, \qquad G_2 = U_2 \Sigma_2 V_2^\top.$$

We then choose the top k singular values s_i such that

$$\frac{\sum_{i=1}^{k} s_i^2}{\sum_{i=1}^{\min(m,n)} s_i^2} \geq \texttt{energy\_threshold}.$$

This helps us filter out noise and keep only the dominant directions.

We then take the top-k singular vectors from U_1 and U_2, which gives us $U_{1,k}$ and $U_{2,k}$. These singular vectors form orthonormal bases for the subspaces spanned by the gradients G_1 and G_2, i.e., the subspaces where the gradients "live".

We then apply SVD to the product of these bases; the singular values of this product are the cosines of the principal angles between the two subspaces:

$$M = U_{1,k}^\top U_{2,k}.$$

We compute the mean of these singular values, which represents the average cosine:

$$\text{similarity} = \frac{1}{k}\sum_{i=1}^{k}\cos(\theta_i).$$

This average cosine represents the degree of overlap between the subspaces in which G_1 and G_2 lie, providing a measure of gradient similarity.

B.6.1 Gradient Similarity Analysis

We quantify gradient similarity on a per–weight-matrix basis. To isolate the effect of timestep and remove batch-induced stochasticity, we fix the input mini-batch and the sampled noise realisations across all measurements. For each timestep group G_i, we sample timesteps exclusively from G_i, run a single forward/backward pass (without updating parameters), and record gradients for every weight matrix.

We measure gradient similarity using the method explained in Section B.6, comparing the subspaces that explain each matrix's 95% of energy. We compute the similarity for each layer's gradient matrix and average across layers to obtain a model-level similarity. The results are reported in Figure 9.

From Figure 9, we observe three consistent trends:

1. **Temporal locality.** Neighbouring timestep groups exhibit higher gradient similarity than temporally distant groups, indicating that nearby denoising tasks induce more aligned update directions.
2. **Progressive specialisation.** As training progresses, inter-group similarity systematically decreases, suggesting that groups specialise and their update subspaces become more distinct.
3. **Asymmetry across denoising tasks.** High-timestep groups display stronger mutual alignment than low-timestep groups; the latter exhibit less aligned gradient subspaces, consistent with their higher intrinsic difficulty and variability.

These observations motivate a nested-rank schedule—allocating more capacity where subspaces diverge—and inform our curriculum and sampling designs to mitigate interference across timestep regimes.

C Method Detail

In this section, we explain the details of our rank schedule, model Flops and parameter count for each low rank preset, and curriculum learning design.

C.1 NA-LR Rank Scheduler

Our rank scheduler has two components: (1) a *minimum activated proportion* $\rho_{\min} \in (0, 1]$, which specifies the fraction of the minimum rank used at the highest timesteps (i.e., the easiest denoising task), and (2) a *decreasing schedule* $f(t) \in [0, 1]$ that controls how the active-rank proportion decays from 1 (low timesteps) down to $\rho_{\min}$ (high timesteps). Let $r_{\max}$ denote the maximum rank of a low-rank layer and T the total number of timesteps. Defining $r_{\min} = \lceil \rho_{\min}\, r_{\max} \rceil$, the active rank at timestep t is

$$\Phi(t) \;=\; r_{\min} \;+\; \Big\lceil \big(r_{\max} - r_{\min}\big)\, f(t) \Big\rceil, \qquad f(t) \in [0, 1], \tag{7}$$

C.1.1 Ablation on $\rho_{\min}$

We ablate $\rho_{\min} \in \{0.3,\, 0.4,\, 0.5,\, 0.6\}$ and track the training loss on the highest-timestep group to detect underfitting caused by overly aggressive rank reduction, results shown in Figure 10. For $\rho_{\min} \geq 0.4$, learning trajectories closely match the non–noise-adaptive baseline; below 0.4 we found a persistent performance gap. Because larger $\rho_{\min}$ increases inference cost, we set $\rho_{\min} = 0.4$ for all experiments.

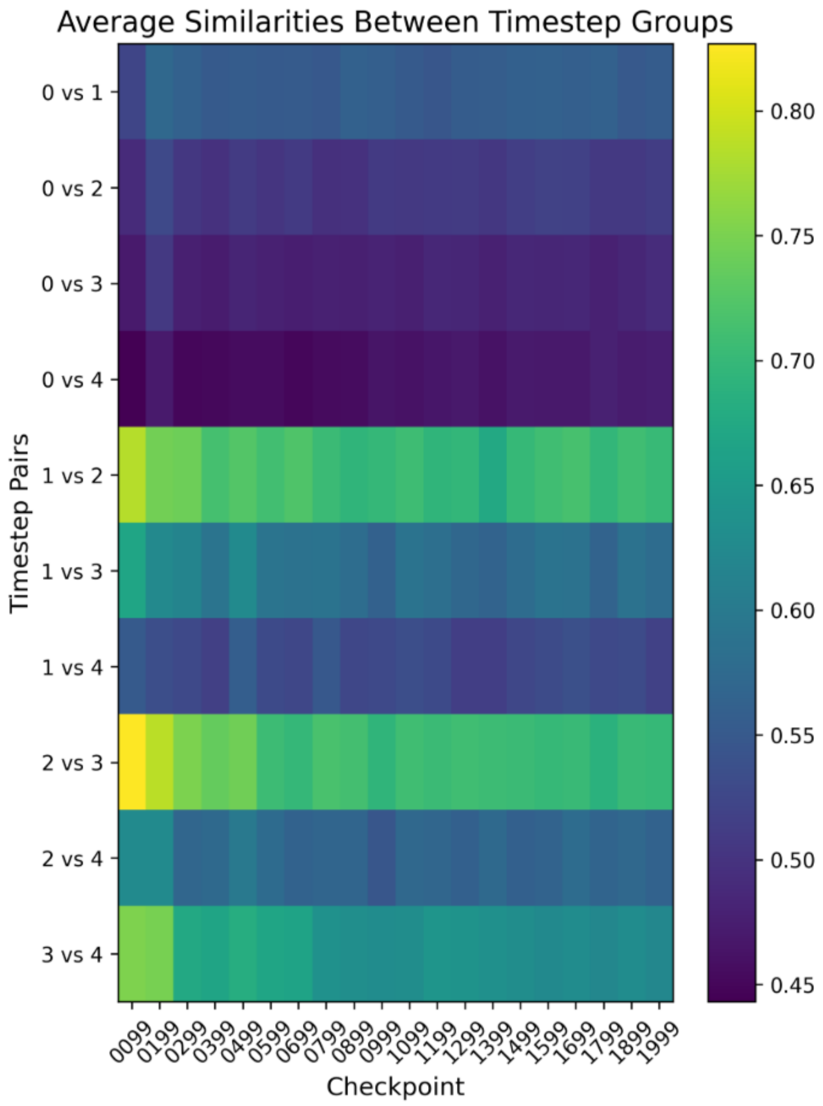


Figure 9: Subspace overlap of gradients between timestep groups over the course of training. Higher values indicate greater alignment of the top-k gradient subspaces.

C.1.2 Decreasing schedules

We tested two decreasing schedule linear schedule, which is inspired by the trend of the effective rank of weight matrices of the model trained with different timestep groups shown in Section B.3, we also add logistic schedule, which is inspired by the trend of the effective rank of gradient of different timestep groups shown in Section B.2:

1. Linear schedule
$$f_{\text{lin}}(t) \;=\; 1 - \frac{t}{T},$$
which decays the active-rank proportion linearly with timestep.
2. Logistic schedule
$$f_{\text{log}}(t) \;=\; \sigma\Big(-k\Big(\frac{t}{T} - m\Big)\Big), \qquad \sigma(x) = \frac{1}{1+e^{-x}},$$
where m controls the midpoint and k the sharpness of the curve.

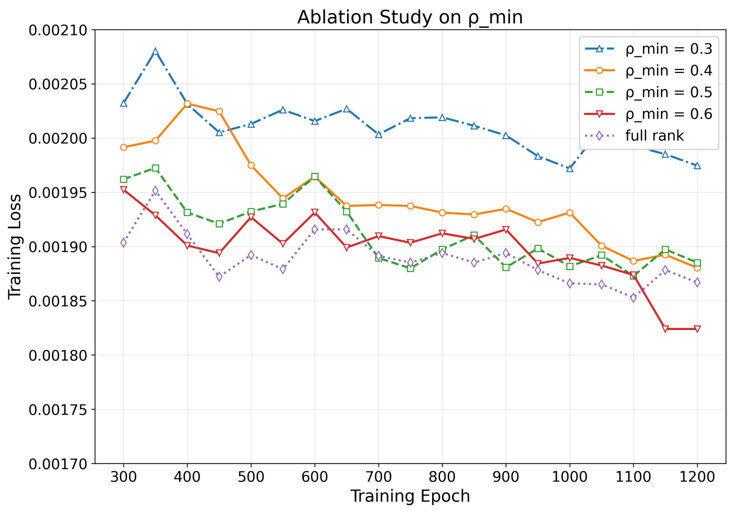


Figure 10: Ablation of the minimum activated proportion $\rho_{\min}$: training loss on the highest-timestep group.

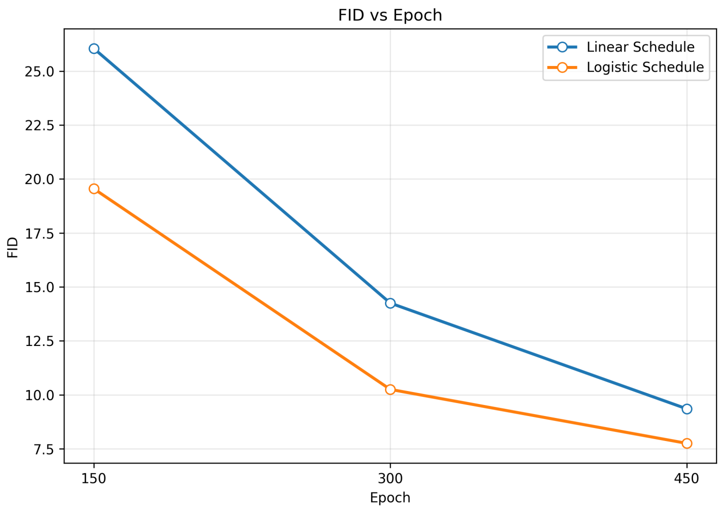


Figure 11: Linear vs. logistic decreasing schedules on FID.

We ablated study the two proposed rank schedule using the setting in Section B.1. For comparability, we set $m = 0.5$ and $k = 8$ so that the expected active rank under $f_{\log}$ matches that of f_{lin} while mirroring the gradient-rank trend observed in Section B.2. As shown in Figure 11, the logistic schedule consistently yields lower FID than the linear schedule; we therefore adopt the logistic schedule in all experiments.

C.1.3 Max-rank warm start

Because $f(t)$ approaches 1 only for very low timesteps, the top rank slices could otherwise be activated too infrequently. We therefore apply a short warm start with ratio $\gamma = 0.1$: for $t \leq \lfloor \gamma T \rfloor$, we set $\Phi(t) = r_{\max}$ in (7). This prevents underfitting of the highest slices without affecting overall compute.

C.2 Low rank Model preset information

Here we report per-image inference cost (GFLOPs over 100 DDIM steps) and parameter counts for full-rank DiT-S/2 and our low-rank variants on $3 \times 32 \times 32$ output size, shown in Table 2.

Table 2: CIFAR-10 (32×32). Per-image inference GFLOPs (100 DDIM steps) and parameter counts for DiT-S/2, fixed low-rank variants, and noise-adaptive low-rank variants under ISO-Parameter (Param-matched) and ISO-Compute (FLOPs-matched).

Method (CIFAR-10, 32×32)	GFLOPs	Params (M)
DiT-S/2	551.674	39.80
DiT-S/2 low rank 75%	421.256	31.34
DiT-S/2 adaptive low rank 75% (ISO-Compute)	421.159	37.60
DiT-S/2 adaptive low rank 75% (ISO-Param)	343.214	31.34
DiT-S/2 low rank 50%	291.060	22.94
DiT-S/2 adaptive low rank 50% (ISO-Compute)	290.754	27.12
DiT-S/2 adaptive low rank 50% (ISO-Param)	238.572	22.94
DiT-S/2 low rank 25%	160.865	14.53
DiT-S/2 adaptive low rank 25% (ISO-Compute)	160.974	16.69
DiT-S/2 adaptive low rank 25% (ISO-Param)	134.335	14.53

C.3 Curriculum Learning Design

In this section, we introduce a curriculum learning strategy that, during a dedicated curriculum phase, restricts the range of timesteps from which each batch is sampled. By confining batches to progressively shifting subranges and activating only the corresponding rank slices, we realise training-time slicing and reduce training FLOPs. After the curriculum phase, sampling gradually shift to uniform over the entire timestep range to avoid bias and preserve final performance.

C.3.1 Initial Curriculum Learning Design

Our initial curriculum design and scheduling therefore largely follow Kim et al. [12].

We cluster timesteps into ten groups $G_0, G_1, \dots, G_9$ using the log-SNR, which aligns grouping with the underlying noise-level progression. The resulting clusters are shown in Figure 12.

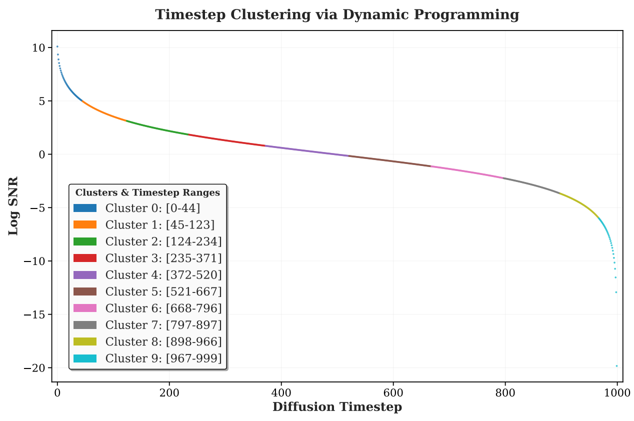


Figure 12: Timestep clustering used for curriculum learning (log-SNR based).

Training starts with the highest-timestep group G_9 (easiest task). After convergence on the current task, we add the next group G_8 and train jointly on $G_9 \cup G_8$ until convergence, and so on, until the final group G_0 is included. Thereafter the model trains on the full range $\bigcup_{i=0}^{N-1} G_i$, matching standard diffusion training. Within each curriculum stage, timesteps are sampled uniformly from the union of the groups included in that stage.

We schedule curriculum transitions by monitoring the per-batch training loss. If the loss does not improve for τ iterations, we introduce the next timestep group. Kim et al. [12] report that $\tau \in [100, 800]$ yields similar final performance with a sweet spot around $\tau = 200$. Accordingly, we adopt $\tau = 200$ in all experiments.

C.3.2 Improvement Over Curriculum Learning Design

Kim et al. [12] use a relatively short curriculum (e.g., 7k–12k steps within 2M total steps). In our DiT-S/2 + CIFAR-10 setting, the curriculum phase ends around 60k steps, with total training lasting 800k steps. Because a longer curriculum period yields greater compute savings in our framework, we deliberately extend the curriculum via stricter convergence monitoring using an exponential moving average (EMA) of the loss:

$$\mathrm{EMA}_t \;=\; \alpha L_t \;+\; (1-\alpha)\,\mathrm{EMA}_{t-1},$$

where L_t is the current mini-batch loss, EMA_{t-1} is the previous EMA, and $\alpha = \frac{2}{N+1}$ sets the effective window size N. We declare convergence for a stage if the EMA does not improve for 200 iterations.

We ablate $\alpha \in \{0.1, 0.05, 0.01, 0.005\}$. The number of iterations required for curriculum completion under each α is shown in Figure 13. For $\alpha \leq 0.05$, the curriculum reliably extends to $\sim$320k steps in our setting; we therefore use $\alpha = 0.05$ by default.

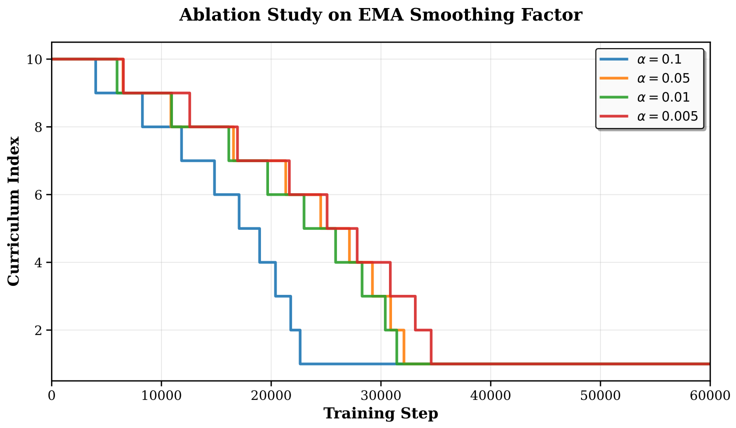


Figure 13: Effect of the EMA smoothing factor α on curriculum learning phase length (iterations until all groups are introduced).

C.3.3 Mitigating Negative Transfer Between Curriculum Stages

With the EMA-based schedule, we initially observed that the curriculum-trained model underperformed the non-curriculum baseline in terms of FID score after the curriculum phase, which is contrary to [12]. Inspecting per-group losses revealed *negative transfer*: upon introducing a new group, losses for already-included groups ceased improving or even worsened, which is shown in Figure 14.

We hypothesise this arises from reduced sampling probability for the already-included groups. As the curriculum expands and sampling remains uniform over the *current* range, each existing group's probability decreases (Figure 15a), leading to gradual forgetting.

To counteract this, we propose *progressive uniform sampling*. After introducing a new group, each previously introduced group retains the same per-sample probability it would have under full-range uniform sampling; the remaining budget in each mini-batch is allocated to the newly introduced group. Thus, the occurrence probability of earlier groups remains constant throughout training, yielding unbiased gradients for them. As more groups are added, the scheme smoothly converges to true uniform sampling over the full range, easing the transition out of the curriculum phase.

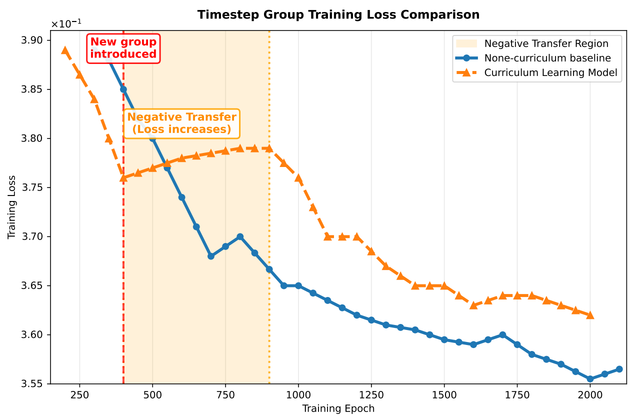
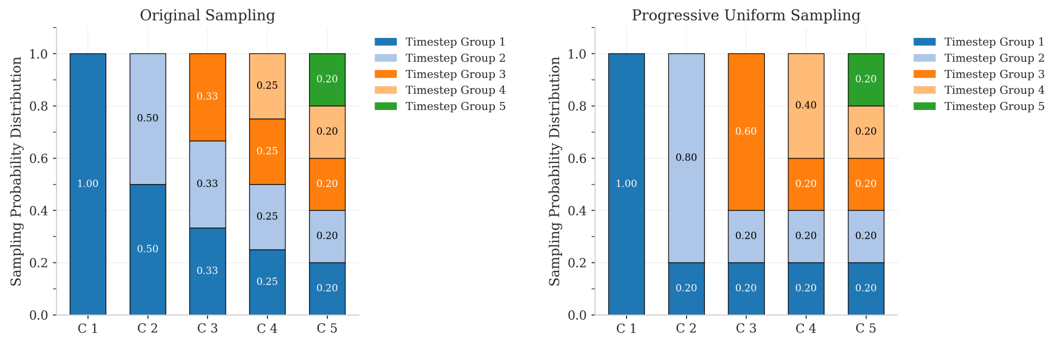


Figure 14: Illustration of negative transfer: after adding a new group, previously introduced groups exhibit increased losses.



(a) Uniform-over-current-range sampling (as in [12]) reduces the sampling probability of previously introduced groups as the curriculum widens.

(b) Progressive uniform sampling: after a group is learned, its sampling probability is fixed to its full-range uniform value; additional samples are drawn from newly added groups.

Figure 15: Side-by-side comparison of curriculum learning sampling strategies

With progressive uniform sampling, losses for previously introduced groups continue to decrease after new groups are added, resolving the negative-transfer issue, as shown in Figure 16.

C.3.4 Mitigating the Late-Start Problem

After addressing negative transfer, we observed that low-timestep groups (introduced later) lagged behind the non-curriculum baseline, exhibiting a persistent loss gap (Figure 17). We attribute this to two factors: (1) when late groups are introduced, the base learning rate has already decayed due to the learning rate schedule; and (2) newly activated parameters have accumulated near-zero gradients for a long period, so the first and second moments in Adam induce an effectively small learning rate upon activation.

We remedy this with a **learning-rate boost** and **optimiser reset** at each group introduction: we reset the base learning rate to the initial value, reduced by 2% per already-introduced group, and clear the optimiser state. To smooth the abrupt change in dynamics (objective, learning rate, and optimiser moments), we apply a short warm-up of 20% of the initial warm-up length.

As shown in Figure 17, after applying the learning-rate boost and optimiser reset, the late-introduced timestep groups achieve performance comparable to the non-curriculum baseline.

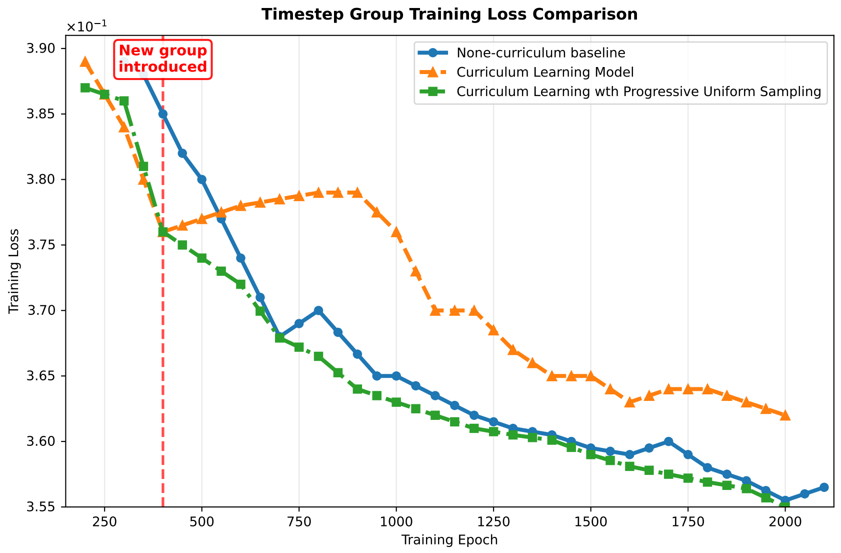


Figure 16: Progressive uniform sampling mitigates negative transfer: previously introduced groups continue improving after new groups are added.

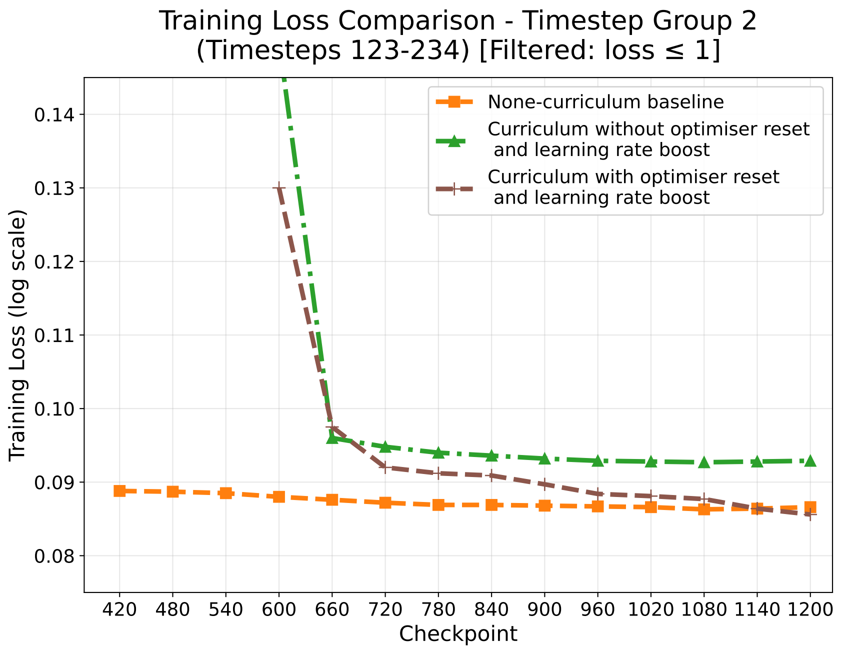


Figure 17: Late-start groups (low timesteps) initially underperform the non-curriculum baseline. Applying a learning-rate boost and optimiser reset closes the gap.

D Experiment Details

In this section we explain the detail setup for all the experiments in this paper.

D.1 Training Hyperparameters

We train with the AdamW optimiser [45] using a base learning rate of 1×10^{-4} with a warmup of 3000 steps and no weight decay. The global batch size is 128. During training we apply random horizontal flips as data augmentation.

We use a DDPM cosine noise schedule [14] with $T = 1000$ timesteps, during inference time we utilise DDIM scheduler with $T = 100$. For classifier-free guidance, the guidance scale is set to 2 at sampling time, and an unconditional conditioning token is used with probability 0.1 during training. We maintain an exponential moving average (EMA) of model weights with decay 0.9999, updated every training step. These settings follow common practice in the diffusion literature.

For the noise-adaptive low-rank scheduler, the base activated rank proportion is set to 0.4, with a logistic schedule (midpoint 0.5, steepness $k = 8$). For curriculum learning, we use 10 timestep clusters and an EMA monitor with decay 0.95. The rationale for these hyperparameters is detailed in Section C.

D.2 Experimental Environment

All experiments are carried out on a machine with Ubuntu 24.04 LTS and uses an AMD Ryzen 7 5700X (8 cores), 16 GB RAM, and an NVIDIA A40 GPU. We implement all experiments using the open-source `PyTorch` and `diffusers` libraries, model FLOPs is measured by `fvcore` [46] library, and accompanying scripts is provided for reproducibility.

D.3 Full Results on CIFAR-10 and CelebA dataset

Here we present the full results across all low-rank presets, including both the static low-rank baselines and our noise-adaptive low-rank variants, both the results on the CelebA dataset and the CIFAR-10 dataset are shown.

Table 3: Inference FLOPs, model parameter size, and generation quality (FID, Precision, Recall) for baseline, fixed low-rank variants, and noise-adaptive low-rank variants of DiT-S/2 model on CIFAR 10 dataset. The Inference FLOPs shown is the FLOPs require to carry out 100 DDIM steps.

Method (CIFAR10 32*32)	GFLOPs	Params (M)	FID ↓	Precision ↑	Recall ↑
Full Rank	551.674	39.80	3.22	0.82	0.79
Low rank 75%	421.256	31.34	5.33	0.76	0.67
Adaptive low rank 75% (ISO-Compute)	421.159	37.60	3.92	0.80	0.73
Adaptive low rank 75% (ISO-Param)	343.214	31.34	5.53	0.75	0.66
Low rank 50%	291.060	22.94	7.36	0.70	0.60
Adaptive low rank 50% (ISO-Compute)	290.754	27.12	5.73	0.74	0.64
Adaptive low rank 50% (ISO-Param)	238.572	22.94	7.41	0.69	0.61
Low rank 25%	160.865	14.53	11.78	0.61	0.51
Adaptive low rank 25% (ISO-Compute)	160.974	16.69	9.38	0.66	0.55
Adaptive low rank 25% (ISO-Param)	134.335	14.53	11.64	0.62	0.53

Table 4: Inference FLOPs, model parameter size, and generation quality (FID, Precision, Recall) for baseline, fixed low-rank variants, and noise-adaptive low-rank variants of DiT-S/2 model on CelebA 64x64 dataset. The Inference FLOPs shown is the FLOPs required to carry out 100 DDIM steps.

Method (CelebA 64*64)	GFLOPs	Params (M)	FID ↓	Precision ↑	Recall ↑
Full Rank	2202.420	39.80	2.35	0.82	0.75
Low rank 75%	1681.805	31.34	3.79	0.75	0.67
Adaptive low rank 75% (ISO-Compute)	1681.451	37.60	3.09	0.79	0.71
Adaptive low rank 75% (ISO-Param)	1269.420	31.34	3.97	0.76	0.68
Low rank 50%	1162.066	22.94	5.59	0.68	0.59
Adaptive low rank 50% (ISO-Compute)	1161.756	27.12	4.61	0.71	0.63
Adaptive low rank 50% (ISO-Param)	887.041	22.94	5.65	0.68	0.58
Low rank 25%	642.328	14.53	8.11	0.59	0.49
Adaptive low rank 25% (ISO-Compute)	642.108	16.69	7.17	0.63	0.53
Adaptive low rank 25% (ISO-Param)	503.332	14.53	8.03	0.60	0.50

E Additional Experiments

In this section, we present additional experiments that further characterise our noise-adaptive low-rank parameterisation: (i) an ablation of the curriculum design, (ii) a quality–efficiency comparison against post-hoc low-rank compression, and (iii) a scalability study on DiT-B/2 with ImageNet-128.

E.1 Ablation Study on Curriculum Learning

We ablate the impact of curriculum learning on final performance on CIFAR-10 dataset. Our goal is to test whether deliberately extending the curriculum phase harms generation quality. We therefore train the three low-rank presets with the noise-adaptive parameterisation *with* and *without* curriculum learning for 800k steps, and compare FID, inference GFLOPs, and parameter counts. Results are shown in Table 5.

Table 5: Ablation of curriculum learning on CIFAR-10. NC refers no curriculum.

Method	FID $\downarrow$	Inference GFLOPs $\downarrow$	Parameters (M) $\downarrow$
Low-rank adaptive 75%	5.53	421.256	31.34
Low-rank adaptive 75%(NC)	5.58	421.159	37.60
Low-rank adaptive 50%	7.41	291.060	22.94
Low-rank adaptive 50%(NC)	7.35	290.754	27.12
Low-rank adaptive 25%	11.64	160.865	14.53
Low-rank adaptive 25%(NC)	11.71	160.974	16.69

The results indicate that curriculum learning does not degrade final image quality: models trained with curriculum achieve comparable FID to their no-curriculum counterparts. In fact, for the 75% and 25% presets, the curriculum-trained models slightly outperform the no-curriculum versions, while also providing training-compute savings as quantified above.

E.2 Comparing with Post-hoc Compression Method

The results above show that our noise adaptive low rank parameterisation improves the efficiency–quality trade-off; when combined with curriculum learning it further reduces training FLOPs without sacrificing performance. To position our approach within the broader landscape of low-rank diffusion, we compare against the common pretrain, compress then fine-tune pipeline reviewed in Section A. Concretely, we evaluate the ELR-diffusion pipeline of He et al. [7], which differs from standard compression in that the fine-tuning stage performs knowledge distillation (KD) from the full-rank teacher rather than directly fine-tuning on data.

We train a full-rank DiT-S/2 on CelebA for 800k steps, apply low-rank compression, and then KD fine-tune the low-rank student for an additional 200k steps. We compare this post-hoc method against our adaptive low-rank 50% presets under two fair settings: (i) **ISO-Compute** (match inference GFLOPs) and (ii) **ISO-Parameter** (match parameter count). Metrics are reported for 100 DDIM sampling steps.

The results are shown in Table 6. At matched compute (ELR compression 50% vs. adaptive 50% ISO-Compute), the compression method attains a lower FID (3.89 vs. 4.61), a 0.72 absolute (approximately 15.6%) improvement, with slightly higher precision (0.72 vs. 0.71) and notably higher recall (0.70 vs. 0.63). This indicates that the FID advantage is driven primarily by improved coverage (diversity). At matched size (ELR compression 50% vs. adaptive 50% ISO-Parameter), the compression method again yields a lower FID (3.89 vs. 5.65, 31.2% relative), but does so at substantially higher inference compute (1162 GFLOPs vs. 887 GFLOPs).

The advantage of post-hoc compression in FID comes with a significant higher training cost because it requires full-rank pretraining followed by low-rank KD fine-tuning. Table 7 shows that ELR compression consumes $7.097 * 10^{18}$ FLOPs, which is 6.08% higher than training a full-rank model end-to-end ($6.696 * 10^{18}$ PFLOPs), 90.62% higher than our adaptive 50% ISO-Compute preset ($3.723 * 10^{18}$ FLOPs), and about 128.20% higher than our adaptive 50% ISO-Parameter preset ($3.110 * 10^{18}$ FLOPs). In other words, it doubles the training compute compared with our adaptive method to achieve a 0.72 absolute FID improvement at matched inference compute.

Table 6: Comparative Results in FID, precision and recall between Adaptive low rank method with post-hoc low rank compression method.

Method	Inference GFLOPs	Parameters (M)	FID	Precision	Recall
Full Rank	2202.42	39.8	2.35	0.82	0.75
Low rank compression 50%	1162.066	22.94	3.89	0.72	0.70
Low rank adaptive 50% (ISO-Compute)	1161.756	27.12	4.61	0.71	0.63
Low rank adaptive 50% (ISO-Param)	887.041	22.94	5.65	0.68	0.58

Table 7: Total Training cost comparison between full rank and low rank methods.

Method	Total Training FLOPSs (10^{18})
Full Rank	6.696
Low rank compression 50%	7.097
Low rank adaptive 50% (ISO-Compute)	3.723
Low rank adaptive 50% (ISO-Param)	3.110

Post-hoc compression with KD can achieves lower FID at the same inference copmutational cost, the lower FID is largely caused by improving recall (sample diversity). However, this benefit is offset by substantially higher training cost. In settings where training compute is the bottleneck, or where one seeks a favourable end-to-end efficiency profile, our noise adaptive low rank approach with curriculum learning offers a more compute efficient alternative, while retaining competitive precision/recall and enabling further reductions under ISO-Parameter configurations.

E.3 Scalability Analysis

To assess scalability, we train DiT-B/2 on ImageNet at 128×128 resolution [47]. Following the original DiT setup [4], we use the Stable Diffusion VAE as the image autoencoder. We compare the full-rank DiT-B/2 against a noise-adaptive low-rank DiT-B/2 with a 50% ISO-Compute preset.

Table 8: Model comparison on FID, Recall, and Precision on ImageNet 128*128 dataset.

Model	Parameter (M)	Inference GFLOPs	FID	Recall	Precision
DiT-B/2 Full Rank	147.42	2203.974	30.13	0.53	0.33
DiT-B/2 Adaptive Low Rank 50% ISO Compute	105.00	1162.366	35.59	0.47	0.25

As shown in Table 8, the adaptive low-rank model reduces inference compute by approximately 47.3% (2203.974 to 1162.366 GFLOPs) and the parameter count by about 28.8% (147.42M to 105.00M), while incurring an 17.1% increase in FID (30.13 to 35.59). Recall and precision also decrease, reflecting the greater difficulty of preserving coverage and fidelity under aggressive rank reduction at this scale. Overall, these results indicate that noise-adaptive low-rank parameterisation scales to larger backbones and datasets, delivering substantial compute savings with competitive generation quality.

F Limitation and Future Works

The main limitation of our work is the method currently relies on a manually tuned rank schedule and exhibits sensitivity in recall performance under aggressive compression Future work could investigate learnable rank schedulers that adapt online under an explicit FLOPs budget, potentially framed as constrained optimisation or reinforcement learning. Additional directions include joint training with distillation to retain diversity under aggressive rank reduction and evaluations on higher-resolution, text-to-image, or video diffusion could further clarify the scalability of our method.

G Visual Comparison of the Generated Image

Here we present the non-curated generated images for across different dataset and low rank present.

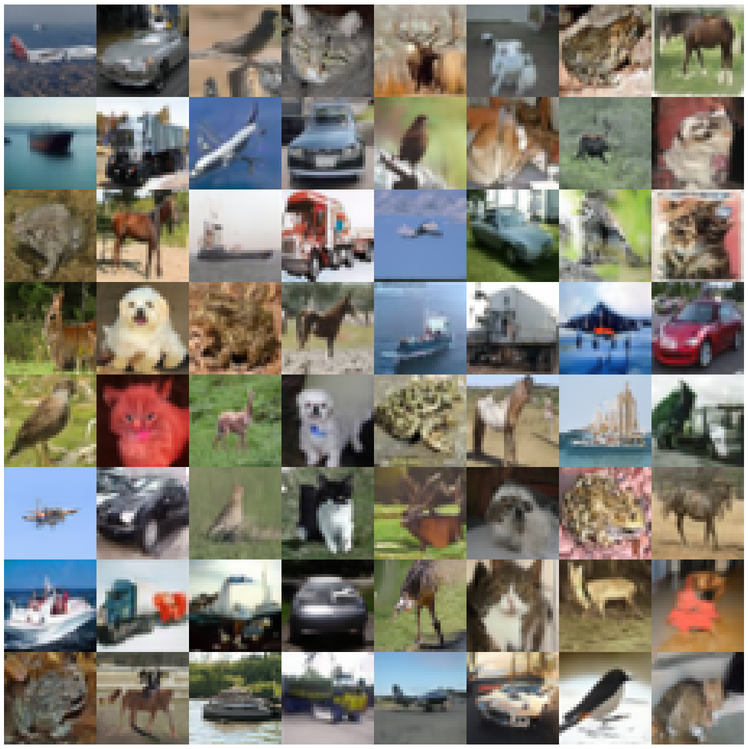


Figure 18: CIFAR-10 image generated by full rank DiT-S/2.

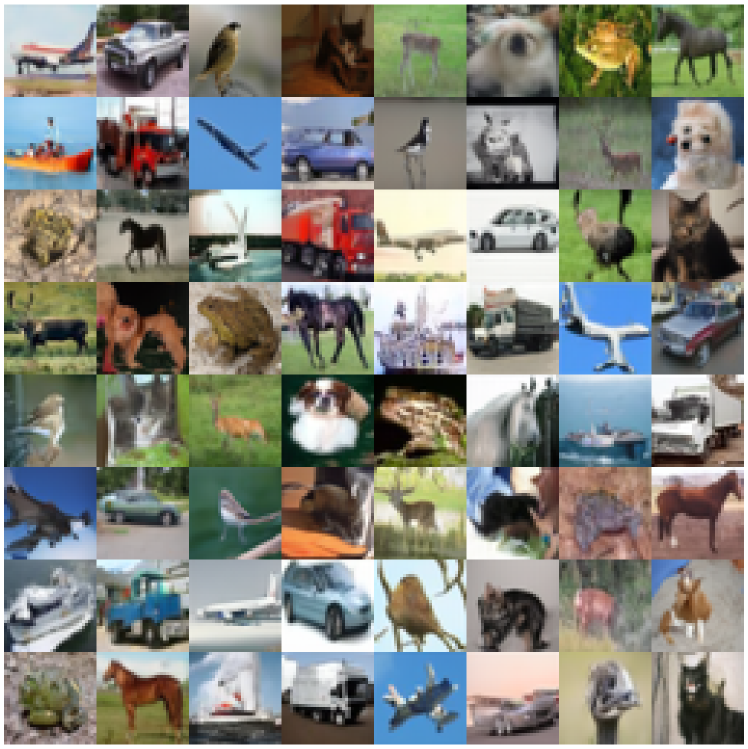


Figure 19: CIFAR-10 image generated by low rank DiT-S/2 adaptive low rank 50% ISO compute preset.

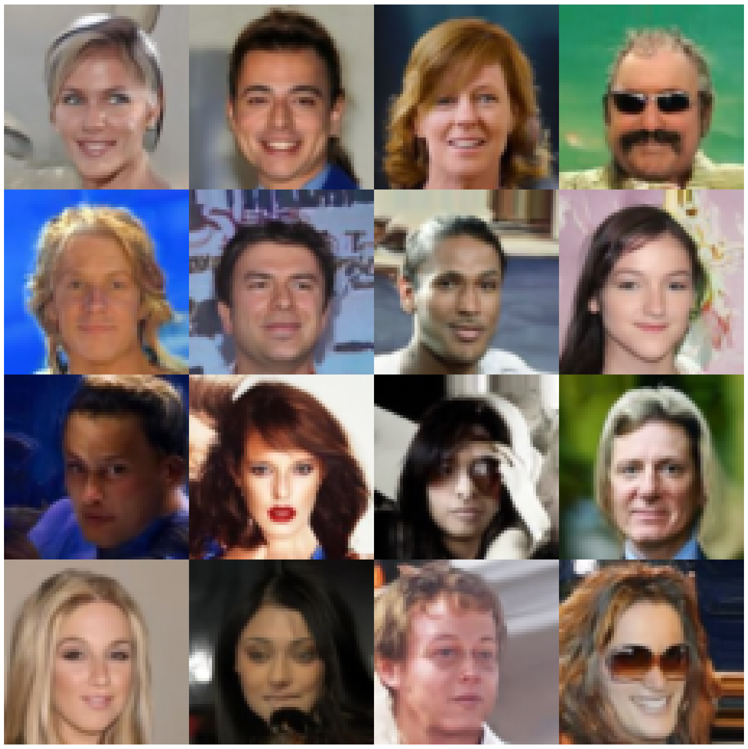


Figure 20: CelebA image generated by full rank DiT-S/2.

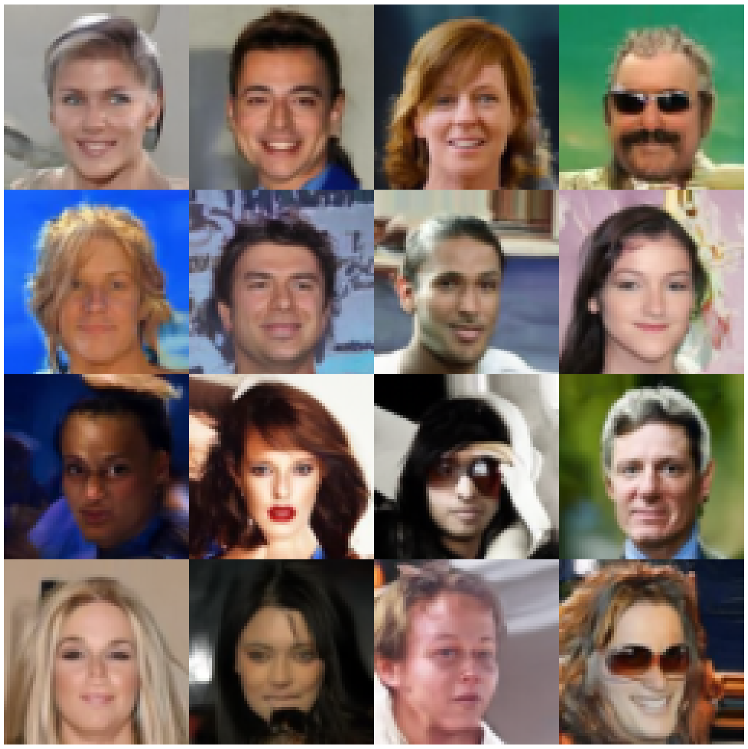


Figure 21: CelebA image generated by low rank DiT-S/2 adaptive low rank 50% ISO compute preset.

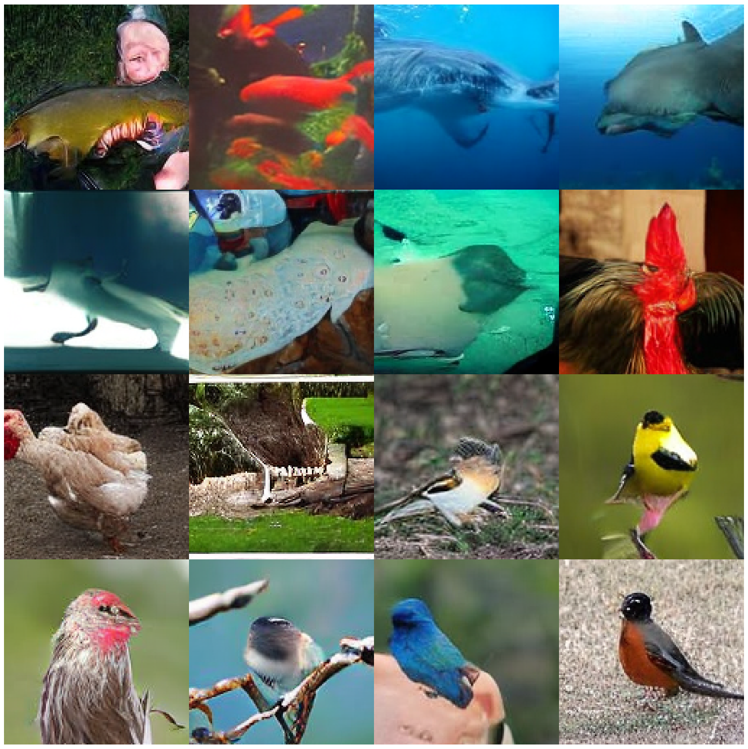


Figure 22: ImageNet image generated by full rank DiT-B/2.

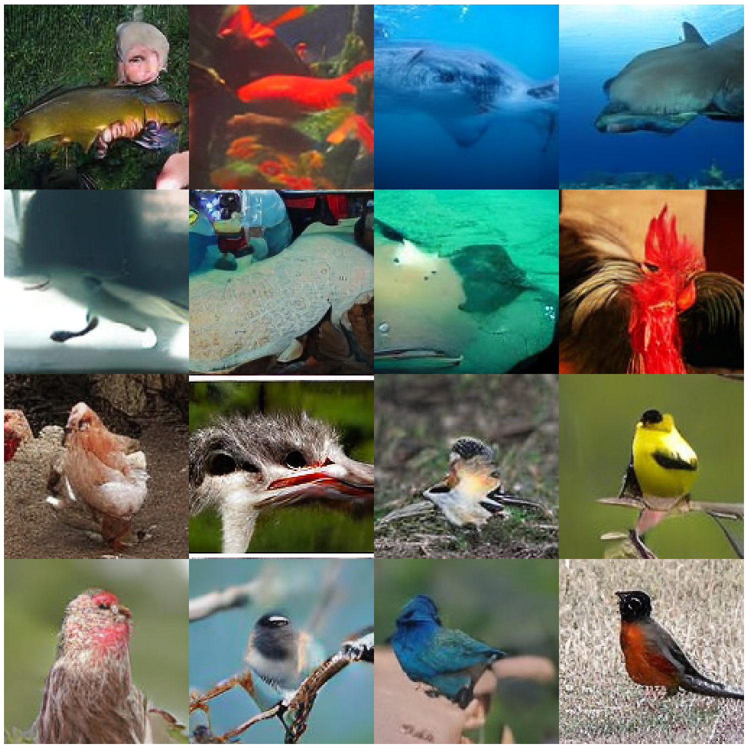


Figure 23: ImageNet image generated by low rank DiT-B/2 adaptive low rank 50% ISO compute preset.