How to Loop Your DiT: Retrofitting Loops on Image Generation Models

Anonymous authors
Paper under double-blind review

Abstract

Scaling inference-time compute has emerged as a prominent avenue for improving neural network performance. Looping models, where transformers are recursively applied to refine predictions, offer a particularly attractive route by allowing a user to control the effective depth of the model at inference time. In the context of image generation, looped transformers trained from scratch offer not only inference-time scaling, but can often match the performance of larger models using significantly fewer parameters. However, training looped models is expensive: an N-loop model requires roughly N times the FLOPs in both training and inference, negating much of the efficiency advantage. We tackle these limitations by retrofitting loops on top of pretrained models. We focus on the family of ImageNet-scale diffusion models and show that naively looping them often leads to limited improvement in performance. Instead, we draw on recent recursive reasoning literature and present a recipe that uses the looped model to refine a set of context tokens, rather than the main answer stream. We show that this approach can be applied on top of pre-trained models to recover performance competitive with from-scratch looping at a fraction of the training cost. Moreover, models trained in this manner exhibit favorable test-time scaling, and can be easily distilled to lower loop counts with negligible quality loss. As a concrete example, with only a third of the training budget, we retrofit a FLUX.2-style XL/2 model to a 4-loop variant that matches a $4\times$ larger feed-forward model, and distill it down to 2 loops with little degradation.

1 Introduction

Generative modeling has undergone a paradigm shift, where inference-time compute has become a key driver of performance (Wei et al., 2022; Guo et al., 2025a; Jaech et al., 2024), complementing traditional gains from larger models and datasets. One approach explored in this direction is looping, in which transformers are recursively applied to refine an intermediate prediction. Rather than increasing parameter count, looping exposes depth as a compute dial, providing a new axis through which the model can be scaled (Geiping et al., 2026; Saunshi et al., 2025; Dehghani et al., 2018).

Recently, Elastic Looped Transformers (ELT, Goyal et al., 2026), extended these ideas to image generation, showing that weight-tied transformers trained from scratch can offer inference-time flexibility and often match the quality of larger feed-forward generators with a fraction of the parameters. However, this apparent efficiency hides a substantial cost: an N-loop model requires roughly N times the FLOPs at both training and inference, since the same block is applied N times per forward pass. As such, training a $4\times$ looped model is nearly as expensive as training a $4\times$ larger feed-forward model, and the per-image cost at inference similarly grows with the loop count.

The natural way to address this bottleneck is to instead focus on retrofitting loops onto existing pretrained models. Doing so concentrates the expense of recurrent training on a shorter training phase rather than the full pretraining run, while inheriting the priors of a model that has already been trained on the target distribution. However, we find that ELT-style looping is brittle in the retrofitting setting: it often fails to improve over the pretrained model, and even where it helps, it tends to degrade at extrapolated loop counts or under distillation (see *e.g.* Figure 1, right).

Instead, we propose to modify the looping scheme in two ways: (1) We introduce a latent scratch-pad (Hao et al., 2024): a set of additional tokens that carry state across loops outside the model's answer stream, and (2) we provide the model's latest denoising target estimate as context, rather than

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

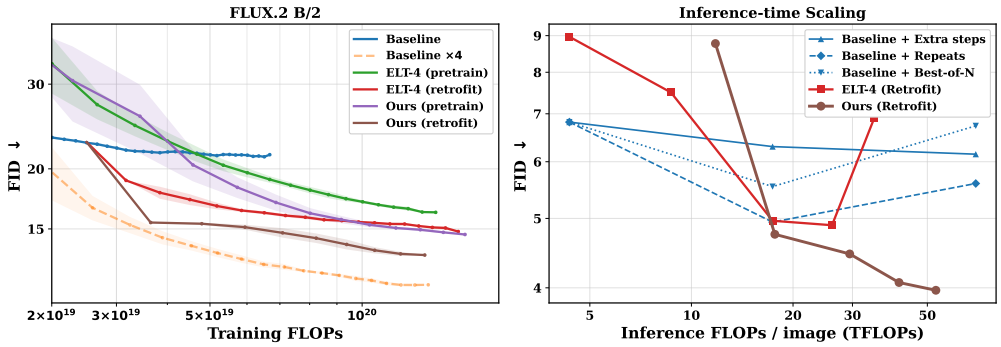


Figure 1: Looping offers a promising path for inference-time scaling in visual content generation. We present a recipe for retrofitting loops onto pre-trained diffusion transformers, and show that it provides better training efficiency and inference-time scaling compared to prior looping approaches.

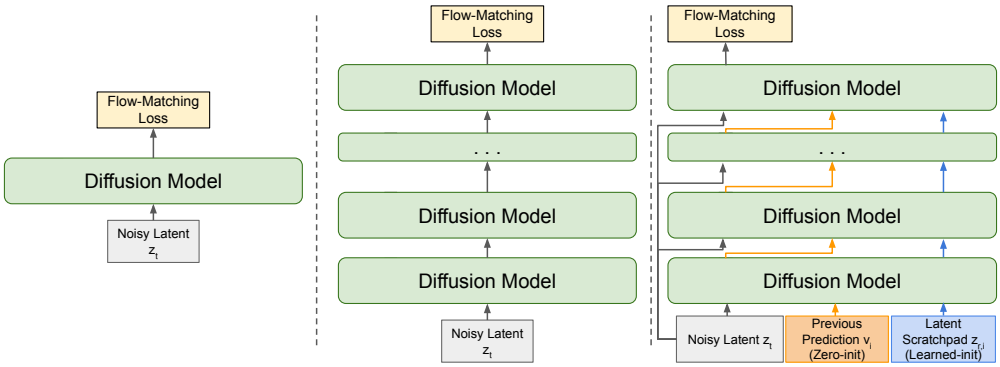
replacing its noisy latent input in the next step. We then loop through a weight-tied transformer that iteratively updates both the scratchpad and the model's working prediction of the denoising target. After a given number of iterations (e.g., 4), the latest prediction is returned as the model's output. This more closely resembles the language domain, where the auto-regressive model always has access to the original query and refines its answer by updating its context, rather than forcing additional information into the same tokens that later become the answer. It also keeps the model closer to the pre-trained baseline, reducing the initial mismatch to overcome. Indeed, in the abstract-reasoning realm, TRM (Jolicoeur-Martineau, 2025) shows that separating the answer and reasoning streams can significantly improve performance. However, this 'reasoning' framing should not be over-read: as we later show through ARC-AGI (Chollet, 2019) experiments (section C), looped diffusion models track the performance of models scaled up by the same factor, suggesting that in our setting loops are best understood as a flexible compute-and-capacity dial, rather than as a distinct reasoning mechanism.

We use our recipe to retrofit loops on ImageNet-scale (Deng et al., 2009) diffusion models (Peebles & Xie, 2023; Yu et al., 2025; Ma et al., 2024), spanning three architectures and two model sizes. We show that the resulting looped models can recover performance competitive with from-scratch looping at a fraction of the training cost (typically $1/3$), and that they scale favorably with increased inference-time compute (Figure 1). We further show that looped models trained in this manner can have their loop counts distilled down with negligible quality degradation, and that they enable a loop-skip variant of classifier guidance that further lowers inference-time costs. Finally, we compare our approach with common inference-time scaling methods and show that they typically trade recall for precision (*i.e.*, harm diversity while boosting quality). In contrast, our approach increases precision with negligible impact on recall. This allows it to better retain its performance when combined with approaches that trade recall for precision, such as CFG (Ho & Salimans, 2021). All in all, our results show that looping offers an appealing inference-time scaling axis for diffusion transformers, and that this axis can be retrofitted onto existing models at a fraction of the cost of pretraining.

2 Related Work

Inference-Time Scaling A growing body of work explores how additional compute spent at inference can improve the quality of a generative model's outputs. An early instance of this paradigm is chain-of-thought (CoT) prompting (Wei et al., 2022), which elicits intermediate reasoning steps from a model prior to producing a final answer. This was later extended through prompting strategies (Nye et al., 2022; Khot et al., 2023; Zhou et al., 2023), supervised or RL-based training of intermediate steps (Yu et al., 2024; Shao et al., 2024), searches over reasoning traces (Xie et al., 2023; Yao et al., 2023; Zhang et al., 2024), and exploration with reflection (Shinn et al., 2023; Gandhi et al., 2025).

In visual generation, immediate analogues are inference-time search methods that select or resample noises guided by verifier feedback (Ma et al., 2025a), using evolutionary (He et al., 2025) or classical tree-search (Zhang et al., 2026a). Others optimize intermediate states using a critic (Mi et al., 2026). A parallel thread fine-tunes generators for iterative reflection: Li et al. (2025) conditions the model on prior generations and VLM feedback. Zhuo et al. (2025) scale this using a large set of flawed images, reflections, and refined outputs. These methods expose inference-time compute through repeated



(a) (b) (c)

Figure 2: Single denoising step for feed-forward and looped architectures. (a) Feed-forward diffusion transformers predict the denoising target from noisy latents in a single forward pass. (b) Standard looping approaches apply the diffusion model recursively, feeding each pass the prior pass' prediction as an input. This makes inputs asymmetric across loops. (c) Our approach keeps the noisy latent fixed and instead provides the prior predictions and latent scratchpad tokens as additional context.

generations, optimization, or reflection rounds, typically guided by external verifiers. In contrast, our method exposes inference-time scaling through recurrent depth, without relying on external tools.

Looped Transformers Recursive application of a shared block of parameters has a long history as a mechanism for trading depth against parameter count, particularly in the language domain (Dehghani et al., 2018; Lan et al., 2019; Bai et al., 2019) or for discriminative vision tasks (Jaegle et al., 2021; Yang et al., 2022). More recent work recasts looping as programmable computation (Giannou et al., 2023), in-context gradient descent (Gatmiry et al., 2024), length generalization (Fan et al., 2025), and test-time compute scaling for latent reasoning (Geiping et al., 2026; Saunshi et al., 2025; Chen et al., 2025). These have been scaled to full LLM training (Zhu et al., 2025) and extended with budget-conditioned elastic depth (Jeddi et al., 2026). Recurrence can also be retrofitted into pretrained language models, either by iterating a subset of layers during mid-training (Koishekenov et al., 2025) or through a curriculum over recurrence depth (McLeish et al., 2026). Mixture-of-Recursions (Bae et al., 2026) and its ViT variant (Li, 2025) route tokens through variable numbers of loop applications, while Loop-ViT (Shu et al., 2026) adds a parameter-free entropy-based exit.

Looping was also recently applied to visual generation models. Thinking Pixel (Sun et al., 2026) adds a recursive mixture-of-experts module to the joint attention of a multimodal DiT, iteratively refining the visual tokens. Elastic Looped Transformers (ELT, Goyal et al. (2026)) train weight-tied models from scratch, using intra-loop self-distillation to ensure elasticity across loop counts. We argue that the natural setup for adding looping is on top of a pre-trained model, a setting in which ELT struggles. Our work overcomes this hurdle by leveraging the recurrent block to update a set of context tokens, rather than intervening in the image stream itself.

Recursive Reasoning and Latent Scratchpads A growing line of work performs reasoning in hidden states rather than explicit CoT (Goyal et al., 2024; Cheng & Van Durme, 2024; Pfau et al., 2024; Hao et al., 2024; Wang et al., 2024; Jin et al., 2025), with continuous intermediate representations shown to outperform discrete token-level reasoning across several settings (Shen et al., 2025; Tack et al., 2025; Zhang et al., 2026b). Theoretical analyses further ties these to reasoning capacity (Merrill & Sabharwal, 2024; Chen & Zou, 2024; Saunshi et al., 2024). In diffusion, EndoCoT (Dai et al., 2026) iteratively refines latent states in the text encoder and feeds them to the generative model.

Most directly relevant to our work, HRM (Wang et al., 2025) introduced a recurrent architecture with a dedicated set of reasoning scratchpad tokens that are iteratively updated alongside a working prediction of the answer. TRM (Jolicoeur-Martineau, 2025) simplified the architecture and demonstrated that small recursive networks rival much larger feed-forward models on ARC-AGI (Chollet, 2019). Our recipe adopts the scratchpad design of HRM and TRM, introducing a set of scratchpad tokens alongside the model's image tokens and refining both in tandem via a weight-tied transformer block.

Additional related work is covered in Appendix Section G.

3 METHOD

Recursive Looping Architecture Our goal is to retrofit recursive loops onto a pretrained non-looping diffusion model and recover the quality benefits of looped training at a reduced computational cost. A naive approach is to apply ELT-style (Goyal et al., 2026) looping, in which the output of one pass through the model is chained as input to the next (Figure 2b). This procedure can be applied on top of a pretrained network, and the resulting loop can be fine-tuned end-to-end with the standard flow-matching loss while back-propagating through the entire chain. As we show in Section 4, this approach struggles to move the pretrained model away from its non-looped behaviour, and the additional passes yield little improvement in most settings.

Instead, we draw inspiration from the abstract-reasoning literature, where Tiny Recursive Models (TRM, Jolicoeur-Martineau (2025)) demonstrate the benefit of a multi-stream recursive design. In TRM, each application of the recurrent network takes in three inputs: a fixed query X that specifies the problem to be solved, the model's most recent answer Y, and a *reasoning state* Z_r that acts as a latent scratchpad. The block updates the answer and the scratchpad jointly,

$$Y',\ Z'_r \ =\ f_\theta(X,\, Y,\, Z_r), \tag{1}$$

and this update is applied repeatedly, with X held fixed throughout, grounding the recursion (Figure 2c). This setup gives the model two ingredients that a single feed-forward pass lacks: a slot in which it can reconsider and correct its latest answer, and a latent channel that is decoupled from that answer, letting it plan in a less constrained space. Jolicoeur-Martineau (2025) show this decoupling is important: removing the scratchpad and folding it into the answer slot degrades performance.

Adapting looping to diffusion. We adapt a TRM-style approach to diffusion transformers by augmenting each denoising step with a recursive inner loop. Let z_t denote the noisy latent at flow time t, c the condition (e.g., class-index), and let v_i and $z_{r,i}$ denote the v-prediction and the "reasoning" scratchpad at the i-th internal loop, respectively. Each loop applies the same shared backbone:

$$v_{i+1},\ z_{r,\,i+1} \ =\ f_\theta\big(z_t,\, c,\, t,\, v_i,\, z_{r,i}\big). \tag{2}$$

We zero-initialize v_0 and set $z_{r,0}$ to a learned embedding. The additional tokens are processed using the same transformer layers as the baseline model, with a few ($\sim 0.5\%$) additional parameters used for new input / output projection layers for each stream. In contrast to TRM, we do not split each cycle into separate reasoning-only and answer-only sub-steps: both v and z_r are updated jointly in every loop. We found in early experiments that explicit reasoning/answer splits performed worse in this setting. Algorithm 1 in the Appendix summarizes the resulting per-step procedure.

Training objective. We train the looped models using the same losses as the underlying diffusion or flow matching model. For example, in the flow matching case, we draw a sample x_0, noise $\epsilon \sim \mathcal{N}(0, I)$, and a flow time $t \sim p(t)$. We form $z_t = (1-t)\, x_0 + t\, \epsilon$ with target velocity $v^\star = \epsilon - x_0$. The loss is the standard flow-matching objective applied to the final-iteration output v_N:

$$\mathcal{L}_{\text{flow}}(\theta) \ =\ \mathbb{E}_{x_0,\,\epsilon,\,t}\Big[\ \|v_N(z_t,\, c,\, t) - v^\star\|_2^2\ \Big]. \tag{3}$$

Retrofitting Initialization When retrofitting a pretrained model with the recursive structure, one has to take care with how the new components are initialized. On the one hand, bolting on the looping machinery with arbitrary (*e.g.*, standard random) initializations produces a large deviation in the model's outputs at the first training step, where the model has to concurrently learn to handle to both loop and new context tokens. We find that in such a setup the model can struggle to recover. On the other hand, a purely zero-initialized configuration, in which the new tokens have no influence on the output, leaves the model with very little gradient pressure to use them, and they remain effectively dead throughout fine-tuning. We find that a mixed initialization scheme can balance these extremes: The new output projections are zero-initialized, such that each of the loops initially sees the same input. On the other hand, we do not modify the attention layers to discard the influence of the new tokens, forcing the model to learn to contend with them even at a single forward pass. This gives a more controlled start, from which we see better and more stable convergence (see Section 4).

Stochastic loop counts. The procedure outlined so far trains with a single fixed loop count N. As also reported by ELT (Goyal et al., 2026), models trained at a single loop budget are inelastic: their quality drops sharply when evaluated at any other loop count. ELT addresses this with an internal

self-distillation objective in which lower-loop variants of the model are trained to mimic the full-loop variant. We find that a simpler scheme often suffices: during training, we sample a per-local-batch loop count $N' \leq N$, which we sample from a distribution skewed toward the full value N. As we show in Section 4, this not only reduces training costs (by no longer requiring a full N-step loop at every training step), but also gives better inference-time scaling performance.

Loop-count distillation. Stochastic loop training already produces a model that operates acceptably at a range of loop counts, but performance at low budgets can be further improved with a short distillation phase. Concretely, we take the trained looped model as a teacher, freeze it, and fine-tune a student copy that is constrained to operate at a fixed reduced budget N'. The student is trained to match the teacher's full-budget output:

$$\mathcal{L}_{\text{loop-distill}}(\theta) \;=\; \mathbb{E}_{x_0,\,\epsilon,\,t}\Big[\,\big\|\, v_{N'}^{\text{student}}(z_t, c, t) \;-\; \text{sg}\big[v_{N}^{\text{teacher}}(z_t, c, t)\big]\,\big\|_2^2\Big], \tag{4}$$

where $\text{sg}[\cdot]$ denotes stop-gradient. In Section 4 we show that this simple approach can typically recover the quality of the full-budget configuration, while further reducing inference costs.

Loop-skip guidance. Stochastic loop training also exposes a new CFG (Ho & Salimans, 2021) axis. In standard CFG, the guided v-prediction is formed by extrapolating along the conditional axis,

$$\tilde{v}_c \;=\; v(\varnothing, N) \;+\; w_c\,\big(\,v(c, N) - v(\varnothing, N)\,\big), \tag{5}$$

where $v(c, N)$ and $v(\varnothing, N)$ are the conditional and unconditional v-predictions, both evaluated at the full loop budget N. Because our model also produces sensible predictions at smaller loop budgets, we can analogously extrapolate along a *loop-effort* axis between high-loop and low-loop evaluations. This is reminiscent of the work of Karras et al. (2024), which guides a model with a worse version of itself. Let $v(\varnothing, N_{\text{lo}})$ denote the unconditional v-prediction at a reduced budget $N_{\text{lo}} < N$. We apply the following guidance direction:

$$v_{\text{guided}} \;=\; v(\varnothing, N_{\text{lo}}) \;+\; w_g\,\Big(\,v(c, N) - v(\varnothing, N_{\text{lo}})\,\Big). \tag{6}$$

As we show in Section 4, mixing the loop-axis into the guidance term has two benefits: (1) It generally matches the performance of the baseline guidance of Equation (5), and (2) it reduces the compute overhead of looping, since the unconditional path requires fewer (or even no) loops.

4 Experiments

We perform our comparisons on ImageNet (Deng et al., 2009) models, on three different models and two different training scales. Specifically, we consider DiT (Peebles & Xie, 2023), SiT (Ma et al., 2024) and FLUX.2-style models (Labs, 2025) across two different size scales: B/2 and XL/2. These choices cover a range of common architectural and training options, including both diffusion (DiT), flow-matching (SiT) and REPA-style training (FLUX.2). They also differ in their latent spaces (SD VAE (Rombach et al., 2022) vs FLUX.2), positional encoding schemes (additive vs RoPE) and more.

For each setting, we consider three parameter matched baselines: (1) The vanilla, unmodified base model. (2) ELT (Goyal et al., 2026), a looping approach that chains the output of one model pass as the input for the next pass. (3) Our approach, which moves the previous loop's prediction to its own stream, and provides a set of additional context tokens that serve as a latent scratchpad. Unless otherwise noted, both looped approaches use 4 loops. For each model, we further consider the impact of existing inference-time scaling methods: (a) scaling the number of steps, (b) re-noising and repeating blocks of steps with varying noise (following RePaint, Lugmayr et al. (2022), where N10R4 denotes 10-step blocks each repeated 4 times), and (c) best-of-N selection with an off-the-shelf reward model (PickScore, Kirstain et al. (2023)). Finally, we also compare to a wider, scaled-up model that matches the naively-looped baselines in training and inference FLOPs. Unless otherwise noted, we compare all models after 1M training steps and report the best result up to that point. Retrofitted loop models start from the 750k step checkpoint of the baseline, and train for 250k additional steps. We compare all approaches using a standard metric set: FID (Heusel et al., 2017), DINOv2-FD (Oquab et al., 2023), precision and recall (Kynkäänniemi et al., 2019). Additional implementation details are in Appendix Section A. FLOPs-matched comparison tables are in Appendix Section B, along with CFG results (Section F).

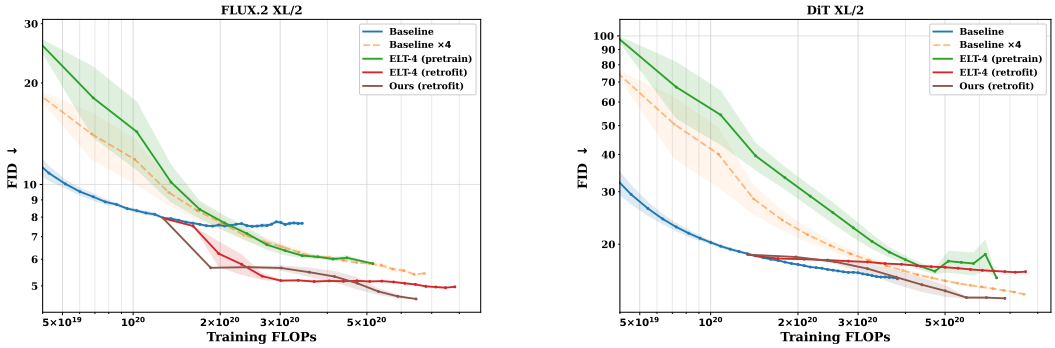


Figure 3: FID versus training FLOPs for FLUX.2 XL/2 (left) and DiT XL/2 (right). Results are smoothed over 5 neighboring points with 1σ confidence bounds.

We further follow SiT (Ma et al., 2024) and compare the baselines both with and without CFG. In the CFG case, we test the optimal scales reported in both RePA and SiT ($w = 1.35$ and $w = 1.5$ respectively) for every model, including the baselines, and report results for the value that achieved lower FID. Unless otherwise noted, we use 50 sampling steps with DDIM (Song et al., 2021) for the DiT baseline and Euler for the flow-matching models. For CFG experiments, we apply guidance across all steps. The results are shown in Tables 1 and 2, with additional results in the Appendix (Tables 7 to 11). Additionally, we show FID-to-FLOPs training curves for select models in Figure 3, including comparisons to naive-looping models trained from scratch.

As can be observed from Figure 3, our approach is competitive with looping methods trained from scratch, in some cases achieving similar FID with a third of the training FLOPs. Analyzing the precision and recall metrics, we can see that the looped model increases precision with limited impact on recall, in contrast with best-of-N or RePaint-style looping which trade one for the other. Indeed, we observe that when combined with CFG (Table 7), the best-of-N approaches can reduce recall to the point where increasing the number of samples can ironically even harm FID. As a consequence, our approach can also better compound with these inference-time scaling methods, such that a 4x looped model combined with best-of-4 can outperform the baseline at best-of-16.

From Table 1 we also observe that the basic ELT looping approach struggles when retrofit on top of existing models, where for many of the cases it fails to improve over the baseline checkpoint. We hypothesize this to be tied to the greater initial shift from the baseline model behaviour, which creates a situation where the shortest path to improving the loss is learning to discard the loops.

Interestingly, when using our approach we often observe grokking, where the retrofitted loops lead to no appreciable change in model performance for most of the initial fine-tuning steps, before suddenly transitioning to a sharp gain. This behavior strongly hints that the loop-retrofitting process can be further refined, possibly through improved initialization or better scheduling.

In Figure 4 we show visual examples of samples drawn from the best performing baselines. We hold the seed fixed across each row. As can be observed, our approach leads to significant increase in visual quality, with results almost indistinguishable from the x4 model outputs. In particular, we observe the largest improvements in overall structure and layout. This aligns with our expectations. After all, diffusion models are themselves are already looped models, and they learn to correct and improve the denoising trajectory across time steps. However, low frequency image content like structure is determined early on, leaving little time for the diffusion to correct them. It is there that the extra looping steps give the greatest boost. In section D we further verify this by examining the effect of looping on the model's ability to denoise images at different noise levels.

Inference-time scaling and loop-distillation Figure 5 compares our approach to retrofitted ELT (@1M steps) across inference-time loop budgets. Our approach extends more gracefully to different loop counts, and in some cases (FLUX.2) continues to improve beyond the maximal training loop count, where ELT often diverges. This suggests that the separation of streams provides the model with better stability. We hypothesize that the gap in deeper loops between FLUX.2 and DiT/SiT is due to its use of RoPE positional encoding applied at each attention layer, rather than additive positional encodings applied only at the model input. This better preserves positional information in deeper loops. However, we leave a detailed investigation to future work.

Table 1: ImageNet FID without CFG. **Bold** / underline = best / 2nd best over all; blue / purple = best / 2nd best excluding inference-time scaling. Entries within 2σ are treated as tied.

	DiT		SiT		Flux2	
Method	B/2	XL/2	B/2	XL/2	B/2	XL/2
Baseline	34.80	17.38	26.17	12.82	21.74	7.44
w/ 200 steps	34.71	17.27	24.95	11.73	20.64	6.31
w/ 800 steps	34.97	17.48	24.74	11.48	20.29	6.06
w/ RePaint-style loops (N10R4)	27.86	12.34	22.49	9.94	15.77	5.26
w/ RePaint-style loops (N10R16)	47.79	25.11	18.82	7.95	13.86	5.63
w/ Best-of-4	18.96	6.26	13.18	4.92	11.54	3.67
w/ Best-of-16	**12.53**	5.36	**9.04**	5.60	8.57	5.56
ELT-4	34.92	17.43	26.35	12.86	16.15	5.06
w/ 200 steps	34.93	17.23	25.16	11.71	15.43	4.23
w/ RePaint-style loops (N10R4)	28.74	13.05	22.47	9.76	12.01	3.72
w/ Best-of-4	18.96	6.25	13.30	4.87	8.75	3.72
Ours (4 Loops)	32.67	13.67	22.35	9.81	14.36	5.25
w/ 200 steps	32.48	13.38	21.20	8.83	13.27	4.22
w/ RePaint-style loops (N10R4)	27.01	9.99	18.85	7.05	10.77	3.99
w/ Best-of-4	17.52	**4.93**	10.91	**4.19**	**7.16**	**3.41**
$\times 4$ model	21.50	13.86	17.81	11.72	11.28	5.35

Table 2: ImageNet Precision / Recall without CFG. **Bold** / underline = best / 2nd best over all; blue / purple = best / 2nd best excluding inference-time scaling. Entries within 2σ are treated as tied.

	DiT		SiT		Flux2	
Method	B/2	XL/2	B/2	XL/2	B/2	XL/2
Baseline	0.654 / 0.743	0.739 / **0.754**	0.690 / **0.735**	0.766 / **0.747**	0.708 / 0.748	0.800 / 0.750
w/ 200 steps	0.652 / 0.746	0.739 / **0.759**	0.689 / **0.741**	0.770 / **0.752**	0.720 / 0.747	0.813 / 0.747
w/ 800 steps	0.651 / 0.744	0.737 / **0.758**	0.688 / **0.742**	0.769 / **0.747**	0.722 / 0.747	0.817 / 0.744
w/ RePaint-style loops (N10R4)	0.693 / 0.661	0.786 / 0.668	0.714 / 0.722	0.796 / 0.726	0.738 / **0.751**	0.823 / 0.737
w/ RePaint-style loops (N10R16)	0.568 / 0.552	0.699 / 0.564	0.733 / 0.712	0.816 / 0.697	0.729 / 0.738	0.820 / 0.713
w/ Best-of-4	0.769 / 0.676	0.866 / 0.656	0.807 / 0.654	0.885 / 0.638	0.820 / 0.676	0.908 / 0.637
w/ Best-of-16	**0.847** / 0.597	**0.926** / 0.554	**0.873** / 0.574	**0.937** / 0.530	**0.882** / 0.600	**0.950** / 0.526
ELT-4	0.652 / 0.741	0.737 / 0.750	0.690 / **0.735**	0.766 / **0.750**	0.756 / 0.736	0.823 / 0.741
w/ 200 steps	0.652 / 0.742	0.738 / 0.752	0.688 / **0.740**	0.768 / **0.754**	0.765 / 0.736	0.838 / 0.731
w/ RePaint-style loops (N10R4)	0.681 / 0.664	0.783 / 0.665	0.712 / 0.725	0.797 / 0.728	0.785 / 0.726	0.851 / 0.724
w/ Best-of-4	0.769 / 0.679	0.865 / 0.656	0.806 / 0.659	0.887 / 0.633	0.862 / 0.647	0.926 / 0.614
Ours (4 Loops)	0.670 / 0.738	0.765 / 0.750	0.716 / 0.732	0.790 / 0.738	0.758 / 0.744	0.821 / 0.746
w/ 200 steps	0.669 / 0.742	0.764 / 0.750	0.717 / **0.737**	0.793 / 0.741	0.770 / 0.741	0.837 / 0.737
w/ RePaint-style loops (N10R4)	0.700 / 0.654	0.818 / 0.644	0.741 / 0.725	0.823 / 0.718	0.779 / 0.732	0.839 / 0.733
w/ Best-of-4	0.784 / 0.673	0.885 / 0.638	0.834 / 0.646	0.906 / 0.623	0.870 / 0.649	0.922 / 0.625
$\times 4$ model	0.720 / **0.752**	0.761 / **0.757**	0.735 / **0.739**	0.785 / 0.734	0.769 / **0.756**	0.811 / **0.758**

Given the mild gap between performance at reduced loop counts in Figure 5, we ask whether performance gaps can be further reduced via a short distillation phase. In Table 3 we compare looped models with different loop counts and their distilled versions. In general, we observe that loop counts can indeed be distilled to 2 steps without considerable degradation, and in some cases with a minor boost in performance (which we attribute to the extra training steps involved in distillation). Notably, this is in contrast to the performance of ELT-style training in which fewer-step sampling often leads to inferior results, despite their use of a similar distillation loss during the training phase.

Loop-skip guidance As highlighted in Section 3, we can further reduce inference costs by modifying the unconditional path such that it uses only a single forward pass through the model. In Figure 6 we show the effect of using this approach across a range of training steps and different models. We observe a negligible FID gap, while saving roughly 37.5% of inference-time compute.

Additional ablations Having shown that retrofitting loops can enable inference-time scaling and improved performance at a reduced cost, we next ablate the contribution of each component of our approach. In particular, we consider: (1) Replacing the previous-answer tokens with registers (Darcet et al., 2024). (2) TRM-like additive states instead of split streams. (3) TRM-like separate update

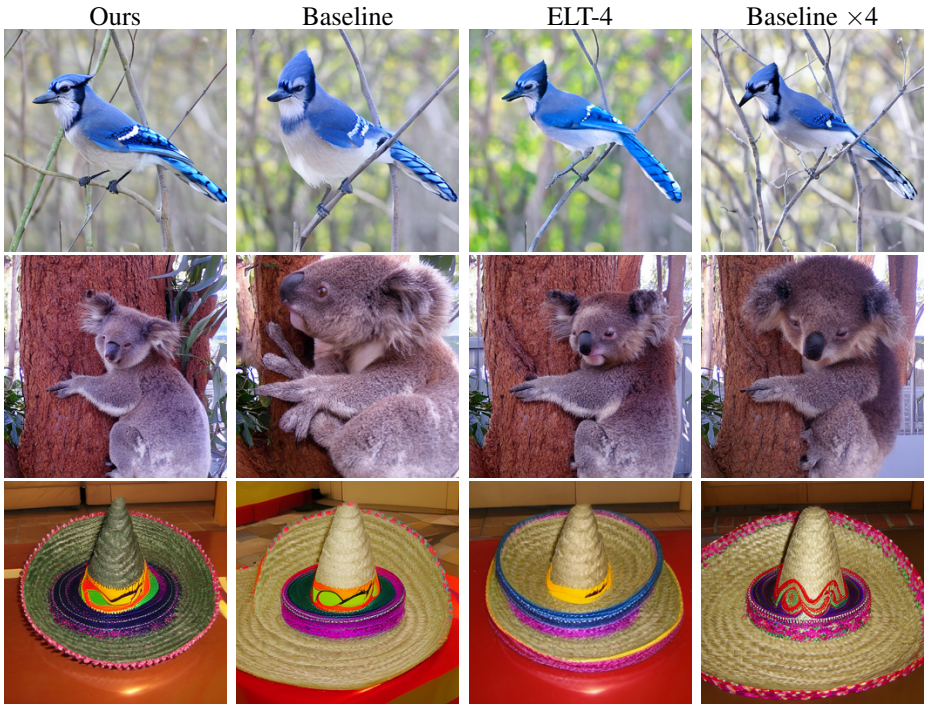


Figure 4: Qualitative comparison across methods, using CFG=4.5. Each row shows the same sample generated by our 4-loop retrofitted model, a Flux2-XL/2 baseline, retrofitted ELT-4, and the baseline scaled up to ×4 parameters. Our approach outperforms the parameter-matched baseline and retrofitted ELT, and remains competitive with the scaled-up baseline at a fraction of the training cost.

Table 3: FID without CFG for teachers, loop-reduced and distilled models.

Model	DiT B/2	DiT XL/2	SiT B/2	SiT XL/2	Flux2 B/2	Flux2 XL/2
Ours (4 loops)	32.67	13.67	22.35	9.81	**14.36**	5.25
Ours (2 loops)	32.54	**13.53**	23.18	9.90	16.38	6.34
Ours (Distilled 4→2)	**32.12**	13.72	**21.63**	**9.74**	15.98	5.69
ELT (4 loops)	34.92	17.43	26.35	12.86	16.15	**5.06**
ELT (Distilled 4→2)	34.59	17.35	26.19	12.76	23.95	9.05

loops for scratchpad and previous answer tokens. (4) ELT's in-training distillation (ILSD) instead of our stochastic loop depth, (5) Removing stochastic loop depth entirely, and (6) randomly initializing the new token output projections instead of zero-initializing them. These are tested on a FLUX.2-B/2 retrofitted-loop setup, at $1M$ steps. The results are shown in Table 4. Notably, these results show that providing the previous answer as a context gives greater benefit than simply adding more register tokens, showing the importance of letting the model reflect on its prior output.

Removing stochastic depths or replacing them with ILSD leads to comparable performance. However, both require a full N-loop pass at every training iteration, making them considerably more expensive to train. Moreover, as ELT showed, models trained at a fixed depth lose their inference time elasticity.

Table 4: Flux2 B/2 ablations. All models are retrofitted with 250k steps on top of the same baseline.

Method	FID (w/o CFG)	FID (w/ CFG)
Ours	**14.36**	**4.55**
w/o Previous Answer	17.39	5.13
w/ TRM Additive States	18.55	5.37
w/ TRM Reasoning/Answer Update Split	16.74	5.06
w/ ILSD	14.67	4.62
w/o Stochastic Depth	15.06	4.71
w/ Random Initialization	15.31	4.83

Reasoning or capacity? In Appendix Section C we conduct ARC-AGI (Chollet, 2019) experiments with looping-based variants, where our goal is to see whether looping induces increased performance on reasoning tasks. Full details and results are provided in the appendix, but they suggest that in our context, looping acts as an effective parameter capacity increase rather than a reasoning mechanism.

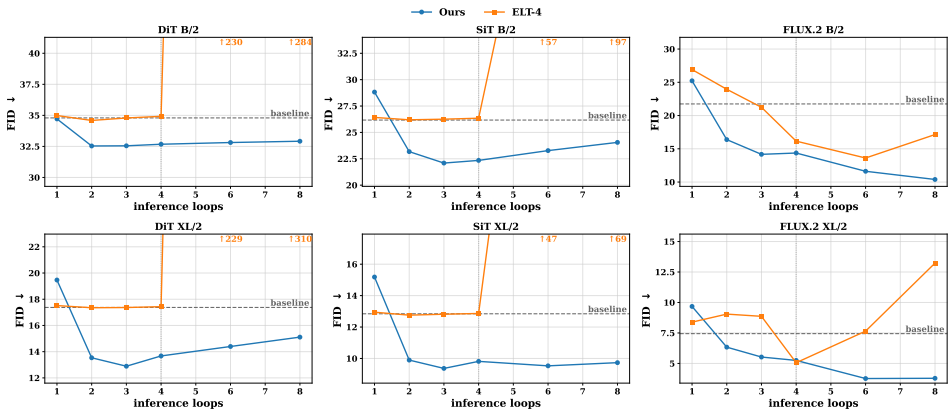


Figure 5: Inference time scaling performance for retrofitted ELT and our method. All models were trained with 4 loops. The dashed line shows the performance of unlooped baseline at $1M$ steps.

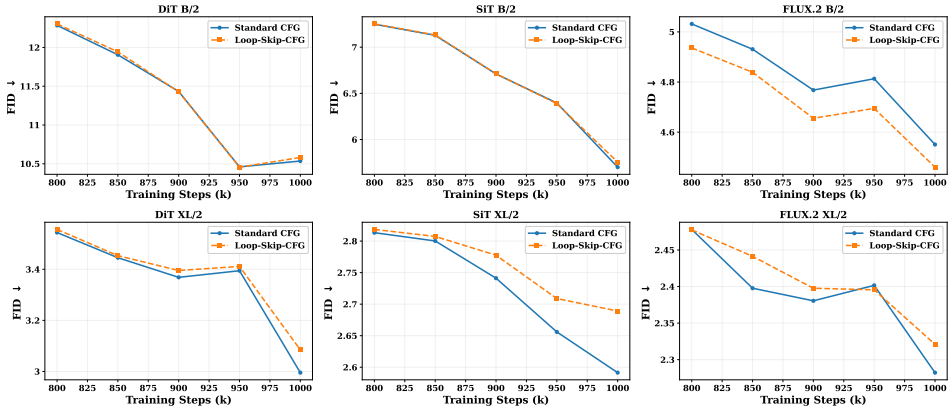


Figure 6: Loop-skip CFG cuts inference costs by 37.5% while keeping comparable quality.

5 Conclusion

In this work we presented a method for retrofitting loops onto pre-trained diffusion transformers. We showed that its performance can match or exceed that of looped and scaled up models trained from scratch, at significantly reduced compute cost. Moreover, our approach demonstrates better inference time scaling elasticity and is more amenable to distillation than existing, naive looping approaches.

While this direction for scaling shows promise, more work remains to be done. The compute required for retrofitting is still beyond typical post-training work, being on par with pre-training itself. However, the grokking behavior that we currently observe strongly hints that these requirements can be further reduced. Moreover, our approach increases the number of tokens used by the model, leading to higher inference-time costs without distillation or loop-skip guidance. Future work could explore directions for reducing this, such as by encoding the prior-loop prediction in a compressed space. Lastly, while our focus was on post-training, the ability to adapt models to looped variants as training progresses hints at possible training strategies that progressively increase the model's depth during training by simply increasing the number of loops. Such strategies could potentially further reduce the remaining gap between retrofitted loops and pretrained models with increased parameter counts, while still reducing computational costs. We hope to explore such directions in the future.

AI USE STATEMENT

In this work, we used generative AI tools to assist in implementing our method on top of existing DiT, SiT and FLUX.2 implementations. We reviewed the generated code, including through the use of sanity tests and comparison with results published by prior work where applicable. We further used generative models to orchestrate experiments on our cluster and to modify paths and hyper-parameters for evaluation sampling. We used pre-existing (non-generated) evaluation code to calculate metrics over the generated images, and manually inspected the results. We have not used generative AI tools to: Generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, clean and reformat datasets, support qualitative and thematic data analysis, or interpret results. Additionally, we used generative AI tools to write the code for the figures in the paper, to edit parts of the research paper to improve readability or shorten phrases, and to find relevant related work that we may have missed with our initial search. We have reviewed all AI-assisted work. For example, we did not directly use any AI-provided citations but instead opened the papers, ensured their relevancy, and manually extracted their bibtex references to avoid hallucinations. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Our work closely follows prior art, using only publicly available datasets (ImageNet, ARC-AGI) and cheaply accessible metrics. We follow the settings laid out by prior published work and repeat them along with any new hyper-parameters in Appendix Section A. We further lay out a simple, explicit algorithmic implementation in Algorithm 1. We will release code upon publication.

REFERENCES

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, et al. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. *Advances in Neural Information Processing Systems*, 38:96572–96617, 2026.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in neural information processing systems*, 32, 2019.

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8, 2023.

Xingwu Chen and Difan Zou. What can transformer learn with varying depth? case studies on sequence learning tasks. *arXiv preprint arXiv:2404.01601*, 2024.

Yilong Chen, Junyuan Shang, Zhenyu Zhang, Yanxi Xie, Jiawei Sheng, Tingwen Liu, Shuohuan Wang, Yu Sun, Hua Wu, and Haifeng Wang. Inner thinking transformer: Leveraging dynamic depth scaling to foster adaptive internal thinking. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28241–28259, 2025.

Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*, 2024.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

Xuanlang Dai, Yujie Zhou, Long Xing, Jiazi Bu, Xilin Wei, Yuhong Liu, Beichen Zhang, Kai Chen, and Yuhang Zang. Endocot: Scaling endogenous chain-of-thought reasoning in diffusion models. *arXiv preprint arXiv:2603.12252*, 2026.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=2dnO3LLiJ1.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

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

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. In *International Conference on Learning Representations*, volume 2025, pp. 14502–14520, 2025.

Kanishk Gandhi, Ayush K Chakravarthy, Anikait Singh, Nathan Lile, and Noah Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective STars. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=QGJ9ttXLTy.

Khashayar Gatmiry, Nikunj Saunshi, Sashank J Reddi, Stefanie Jegelka, and Sanjiv Kumar. Can looped transformers learn to implement multi-step gradient descent for in-context learning? *arXiv preprint arXiv:2410.08292*, 2024.

Jonas Geiping, Sean Michael McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=S3GhJooWIC.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning*, pp. 11398–11442. PMLR, 2023.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ph04CRkPdC.

Sahil Goyal, Swayam Agrawal, Gautham Govind Anil, Prateek Jain, Sujoy Paul, and Aditya Kusupati. Elt: Elastic looped transformers for visual generation. *arXiv preprint arXiv:2604.09168*, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025a.

Ziyu Guo, Renrui Zhang, Chengzhuo Tong, Zhizheng Zhao, Peng Gao, Hongsheng Li, and Pheng-Ann Heng. Can we generate images with cot? let's verify and reinforce image generation step by step. *arXiv preprint arXiv:2501.13926*, 2025b.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

William Harvey and Frank Wood. Visual chain-of-thought diffusion models. *arXiv preprint arXiv:2303.16187*, 2023.

Haoran He, Jiajun Liang, Xintao Wang, Pengfei Wan, Di Zhang, Kun Gai, and Ling Pan. Scaling image and video generation via test-time evolutionary search, 2025.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pp. 4651–4664. PMLR, 2021.

Ahmadreza Jeddi, Marco Ciccone, and Babak Taati. Loopformer: Elastic-depth looped transformers for latent reasoning via shortcut modulation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RzYXb5YWBs.

Dongzhi Jiang, Ziyu Guo, Renrui Zhang, Zhuofan Zong, Hao Li, Le Zhuo, Shilin Yan, Pheng-Ann Heng, and Hongsheng Li. T2i-r1: Reinforcing image generation with collaborative semantic-level and token-level cot. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=O7311P8YgP.

Mingyu Jin, Weidi Luo, Sitao Cheng, Xinyi Wang, Wenyue Hua, Ruixiang Tang, William Yang Wang, and Yongfeng Zhang. Disentangling memory and reasoning ability in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1681–1701, Vienna, Austria, July 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.acl-long.84/.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025.

Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. Guiding a diffusion model with a bad version of itself. *Advances in Neural Information Processing Systems*, 37:52996–53021, 2024.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=_nGgzQjzaRy.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. *Advances in neural information processing systems*, 36:36652–36663, 2023.

Yeskendir Koishekenov, Aldo Lipani, and Nicola Cancedda. Encode, think, decode: Scaling test-time reasoning with recursive latent thoughts. *arXiv preprint arXiv:2510.07358*, 2025.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in neural information processing systems*, 32, 2019.

Black Forest Labs. FLUX.2: Frontier Visual Intelligence. https://bfl.ai/blog/flux-2, 2025.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Shufan Li, Konstantinos Kallidromitis, Akash Gokul, Arsh Koneru, Yusuke Kato, Kazuki Kozuka, and Aditya Grover. Reflect-dit: Inference-time scaling for text-to-image diffusion transformers via in-context reflection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15657–15668, 2025.

YiZhou Li. Imc-net: A lightweight content-conditioned encoder with multi-pass processing for image classification, 2025. URL https://arxiv.org/abs/2507.21761.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di ZHANG, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=oCBKGw5HNf.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11461–11471, 2022.

Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pp. 23–40. Springer, 2024.

Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, et al. Inference-time scaling for diffusion models beyond scaling denoising steps. *arXiv preprint arXiv:2501.09732*, 2025a.

Xiaoxiao Ma, Haibo Qiu, Guohui Zhang, Zhixiong Zeng, Siqi Yang, Lin Ma, and Feng Zhao. Stage: Stable and generalizable grpo for autoregressive image generation, 2025b. URL https://arxiv.org/abs/2509.25027.

Xiaoxiao Ma, Jiachen Lei, Tianfei Ren, Jie Huang, Siming Fu, Aiming Hao, Jiahong Wu, Xiangxiang Chu, and Feng Zhao. Mar-grpo: Stabilized grpo for ar-diffusion hybrid image generation. *arXiv preprint arXiv:2604.06966*, 2026.

Sean Michael McLeish, Ang Li, John Kirchenbauer, Dayal Singh Kalra, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Jonas Geiping, Tom Goldstein, and Micah Goldblum. Teaching pretrained language models to think deeper with retrofitted recurrence. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=PXVQTHYwgt.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *International Conference on Learning Representations*, volume 2024, pp. 7690–7706, 2024.

Yapeng Mi, Yanpeng Zhao, Hengli Li, Chenxi Li, Huimin Wu, Xiaojian Ma, Song-Chun Zhu, Ying Nian Wu, and Qing Li. MILR: Improving multimodal image generation via test-time latent reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=dw0pORtnKI.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models, 2022. URL https://openreview.net/forum?id=iedYJm92o0a.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

Jacob Pfau, William Merrill, and Samuel R. Bowman. Let's think dot by dot: Hidden computation in transformer language models. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=NikbrdtYvG.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Nikunj Saunshi, Stefani Karp, Shankar Krishnan, Sobhan Miryoosefi, Sashank J Reddi, and Sanjiv Kumar. On the inductive bias of stacking towards improving reasoning. *Advances in Neural Information Processing Systems*, 37:71437–71464, 2024.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=din0lGfZFd.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

Wen-Jie Shu, Xuerui Qiu, Rui-Jie Zhu, Harold Haodong Chen, Yexin Liu, and Harry Yang. Loopvit: Scaling visual arc with looped transformers. *arXiv preprint arXiv:2602.02156*, 2026.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=St1giarCHLP.

Yuwei Sun, Yuxuan Yao, Hui Li, and Siyu Zhu. The thinking pixel: Recursive sparse reasoning in multimodal diffusion latents. *arXiv preprint arXiv:2604.25299*, 2026.

Jihoon Tack, Jack Lanchantin, Jane Yu, Andrew Cohen, Ilia Kulikov, Janice Lan, Shibo Hao, Yuandong Tian, Jason Weston, and Xian Li. Llm pretraining with continuous concepts. *arXiv preprint arXiv:2502.08524*, 2025.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025.

Xinyi Wang, Lucas Caccia, Oleksiy Ostapenko, Xingdi Yuan, William Yang Wang, and Alessandro Sordoni. Guiding language model reasoning with planning tokens. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=wi9IffRhVM.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. Self-evaluation guided beam search for reasoning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Bw82hwg5Q3.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, et al. Dancegrpo: Unleashing grpo on visual generation. *arXiv preprint arXiv:2505.07818*, 2025.

Jiewen Yang, Xingbo Dong, Liujun Liu, Chao Zhang, Jiajun Shen, and Dahai Yu. Recurring the transformer for video action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14063–14073, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=5Xc1ecxO1h.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=N8N0hgNDRt.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=DJSZGGZYVi.

Shihao Yuan, Yahui Liu, Yang Yue, Jingyuan Zhang, Wangmeng Zuo, Qi Wang, Fuzheng Zhang, and Guorui Zhou. Ar-grpo: Training autoregressive image generation models via reinforcement learning. *arXiv preprint arXiv:2508.06924*, 2025.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772, 2024.

XiangCheng Zhang, Haowei Lin, Haotian Ye, James Zou, Jianzhu Ma, Yitao Liang, and Yilun Du. Inference-time scaling of diffusion models through classical search. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=b7Ftp6U78i.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, and Xin Wang. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. *Advances in Neural Information Processing Systems*, 38:168990–169012, 2026b.

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. Diffusionnft: Online diffusion reinforcement with forward process. *arXiv preprint arXiv:2509.16117*, 2025.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WZH7099tgfM.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

Le Zhuo, Liangbing Zhao, Sayak Paul, Yue Liao, Renrui Zhang, Yi Xin, Peng Gao, Mohamed Elhoseiny, and Hongsheng Li. From reflection to perfection: Scaling inference-time optimization for text-to-image diffusion models via reflection tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15329–15339, 2025.

A Implementation Details

Here we provide additional details on hyperparameter choices involved in our work. A forward pass of our method involves 384 additional tokens per loop iteration: 256 previous-answer tokens plus 128 learned reasoning tokens. During training, we assign a probability of 0.3 to cut the loop chain off at any intermediate point. Since these odds compound, we have an effective probability of 0.300, 0.210, 0.147, and 0.343 for performing 1, 2, 3 or 4 iterations respectively ($\mathbb{E}[\text{length}] \approx 2.53$). Unless otherwise noted, all looped models in the paper are evaluated with 4 answer-update iterations per denoising step, with no separate scratchpad-only update iterations. Training hyper-parameters for our model and all baselines follow REPA (Yu et al., 2025): AdamW with a constant learning rate of $1{\times}10^{-4}$, $(\beta_1, \beta_2) = (0.9, 0.999)$, no weight decay, a global batch size of 256, gradient-norm clipping at 1.0, and an EMA of the weights with decay 0.9999. All reported results use the EMA weights. Retrofitted models are initialized from the corresponding baseline's EMA checkpoint at 750k steps, with newly introduced projections zero-initialized so training starts close to the baseline's function. Unless noted otherwise, all models train on ImageNet $256{\times}256$ and sample with 50 solver steps. For sampling, SiT and FLUX.2 use an Euler ODE solver. DiT uses a deterministic DDIM ($\eta = 0$) except for the RePaint case which uses stochastic DDIM ($\eta = 1$). We saved and evaluated checkpoints at a $50k$ step granularity. Distilled models were evaluated at the first checkpoint ($50k$ steps).

Since ELT Goyal et al. (2026) has no official implementation, we re-implement it following the descriptions in the paper. The pretrained ELT curves in Figure 3 show that in the training-from-scratch scenario investigated in their paper, our implementation reproduces results consistent with their report (ELT performs comparably to a scaled up model). For the retrofit case, we use the same code but initialize the model weights from the pretrained baseline weights at $750k$ steps.

In Table 5 we further report the number of parameters and FLOPs used by each model during training and inference.

Table 5: Per model parameter and flops counts. *Train* accounts for the full training FLOPs for the model, including the baseline and any distillation. Values are reported at a total of $1M$ training steps before distillation. *Retrofit* is the compute spent after the inherited 750k-step baseline pretrain, except for distillation rows where it reports the extra compute for distillation. Inference is measured for 50 solver steps. With CFG, baselines and ELT-4 double the model forwards, while our models uses Loop-Skip-CFG that only requires one extra loop. REPA feature extraction and VAE costs excluded. *T* indicates the number of tokens per batch element. *TF* and *EF* indicate tera- and exa-flops respectively.

	Params (M)		T	Loops	GFLOPs/loop		Inf. (TF/img)		Inf. CFG (TF/img)		Train (EF)		Retrofit (EF)	
Method	B/2	XL/2			B/2	XL/2	B/2	XL/2	B/2	XL/2	B/2	XL/2	B/2	XL/2
DiT / SiT														
Baseline	130.5	675.1	256	1	45.9	236.8	2.3	11.8	4.6	23.7	35.3	181.8	–	–
$\times 4$ model	519.3	2695.9	256	1	178.8	930.1	8.9	46.5	17.9	93.0	137.3	714.3	–	–
ELT-4 (retrofit)	130.5	675.1	256	4	45.9	236.8	9.2	47.4	18.4	94.7	61.7	318.2	35.3	181.8
Ours (retrofit)	130.7	675.4	640	4	123.8	623.6	24.8	124.7	31.0	155.9	86.7	439.7	60.2	303.3
Ours distilled (4→2)	130.7	675.4	640	2	123.8	623.6	12.4	62.4	18.6	93.5	102.5	519.5	15.9	79.8
FLUX.2														
Baseline	137.8	683.2	256	1	43.5	224.1	2.2	11.2	4.3	22.4	33.4	172.1	–	–
$\times 4$ model	528.2	2706.3	256	1	169.1	879.4	8.5	44.0	16.9	87.9	129.9	675.4	–	–
ELT-4 (retrofit)	137.8	683.2	256	4	43.5	224.1	8.7	44.8	17.4	89.6	58.4	301.2	33.4	172.1
Ours (retrofit)	139.3	686.3	640	4	117.8	591.9	23.6	118.4	29.4	148.0	82.3	416.9	57.3	287.9
Ours distilled (4→2)	139.3	686.3	640	2	117.8	591.9	11.8	59.2	17.7	88.8	97.4	492.7	15.1	75.8

B FLOPs-matched Comparisons

While our approach shows a benefit even in the typical fixed-step comparison setup, we highlight that one of its core benefits is in reduced compute budget. To better highlight the performance gap in this scenario, Table 6 provides FID results in a FLOPs-matched setup. As can be seen, under this scenario the gaps between our method and the baselines becomes even more pronounced, highlighting the efficiency of our method, particularly when compared with ELT. Note in particular that our performance compared to the scaled up ($\times 4$) model improves as we move to bigger models. We hypothesize that larger models may be under-utilizing their parameters, and looping has more

room to improve performance on that front. However, we leave more detailed investigation for the future.

Table 6: ImageNet FID without classifier-free guidance at matched training FLOPs within each column. Cells report the best FID over checkpoints up to the matched step. **Bold** / underline = best / 2nd best over all; blue / purple = best / 2nd best excluding inference-time scaling. Entries within 2σ are treated as tied.

	DiT		SiT		Flux2	
Method	B/2	XL/2	B/2	XL/2	B/2	XL/2
Matched training FLOPs	9.0×10^{19}	6.8×10^{20}	9.0×10^{19}	4.6×10^{20}	1.3×10^{20}	6.5×10^{20}
ELT-4	33.34	15.96	25.43	12.24	15.08	5.01
w/ 200 steps	33.28	15.78	24.19	11.05	13.82	3.92
w/ RePaint-style loops (N10R4)	27.19	11.43	21.67	9.31	11.47	3.54
w/ Best-of-4	**17.74**	5.34	12.56	4.65	7.85	**3.50**
Ours (4 Loops)	32.67	12.82	22.35	9.81	13.06	4.57
w/ 200 steps	32.48	12.79	21.20	8.83	12.26	3.59
w/ RePaint-style loops (N10R4)	27.01	9.45	18.85	7.05	10.25	3.60
w/ Best-of-4	**17.52**	**4.49**	**10.91**	**4.19**	**6.61**	**3.41**
$\times 4$ model	24.29	14.02	20.50	11.72	11.28	5.35

C ARC-AGI Experiments

Loops and latent scratchpads are common approaches for tackling abstract reasoning tasks, and indeed TRM (Jolicoeur-Martineau, 2025) served as our motivation for applying them to image generation. An intriguing question is then whether diffusion models imbued with such components show greater gains on reasoning tasks, or whether the loops and latent scratchpad largely serve to provide greater capacity. To investigate this question we train a simple diffusion transformer on ARC-AGI-1 and compare three variants: a tiny 13.7M parameter model, a scaled-up version with 4x the parameter count, and a looped version with a total loop factor of 4. We use an architecture similar to the Flux.2 ImageNet experiments, but replace the class condition with a task vector. This task vector is learned for each ARC-AGI task during training, and optimized for novel tasks at inference time using their provided in-context examples. We do not use a VAE, and instead encode each example as a 30x30 image with 3 color channels and an opacity channel that designates whether a specific cell is active or blank. The model is trained to denoise such a grid into the correct solution for a given task, with the opacity channel allowing the model to control the 'size' of the solution.

The looped variant follows TRM more closely: we use additive states rather than concatenating them along the token dimension, and further split into reasoning-update and answer-update steps. We found this TRM-like setup to work better for ARC-AGI. We use 2 answer-update steps with 1 reasoning-update for each, leading to a total loop factor of 4.

In Figure 7 we report the accuracy on the public ARC-AGI validation set. As can be seen, the 4-looped models quickly converge to the same performance as the model with 4x the capacity, strongly indicating that in this scenario, the gains are largely a function of better effective capacity rather than strictly better reasoning capabilities.

D Distribution of Improvement Across Timesteps

To investigate the impact of looping across different denoising steps, we take a look at the model's prediction error across the diffusion process. Specifically, we compute the flow matching loss on a fixed validation set which re-uses the same images and noise seeds across all different steps. In fig. 8 we show the resulting losses for the baseline model compared with our retrofitted $4\times$ loop model. As can be observed, the loss improves mainly across the early steps, with the boost declining towards the end of the process. This indicates that the model can best use the additional passes to improve layout and structure mistakes, and suggests that looping could be made even more efficient by using a different number of loops at different denoising steps. However, we leave such investigation to future work.

E ALGORITHMIC FORM OF OUR METHOD

While Section 3 describes our approach, in Algorithm 1 we additionally provide a simple summary in algorithm form, highlighting the relative simplicity of the implementation. In practice, we train our models using 4 loops and back propagate through all of them. If using higher loop counts, one can follow TRM and truncate the number of loops through which back-propagation is done.

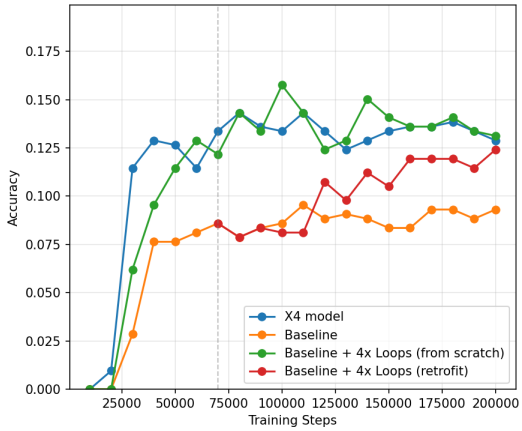


Figure 7: ARC-AGI 1 (Chollet, 2019) accuracy for a baseline diffusion transformer, a parameter scaled version, and two loop-scaled versions. The looped versions roughly align with simple parameter scaling.

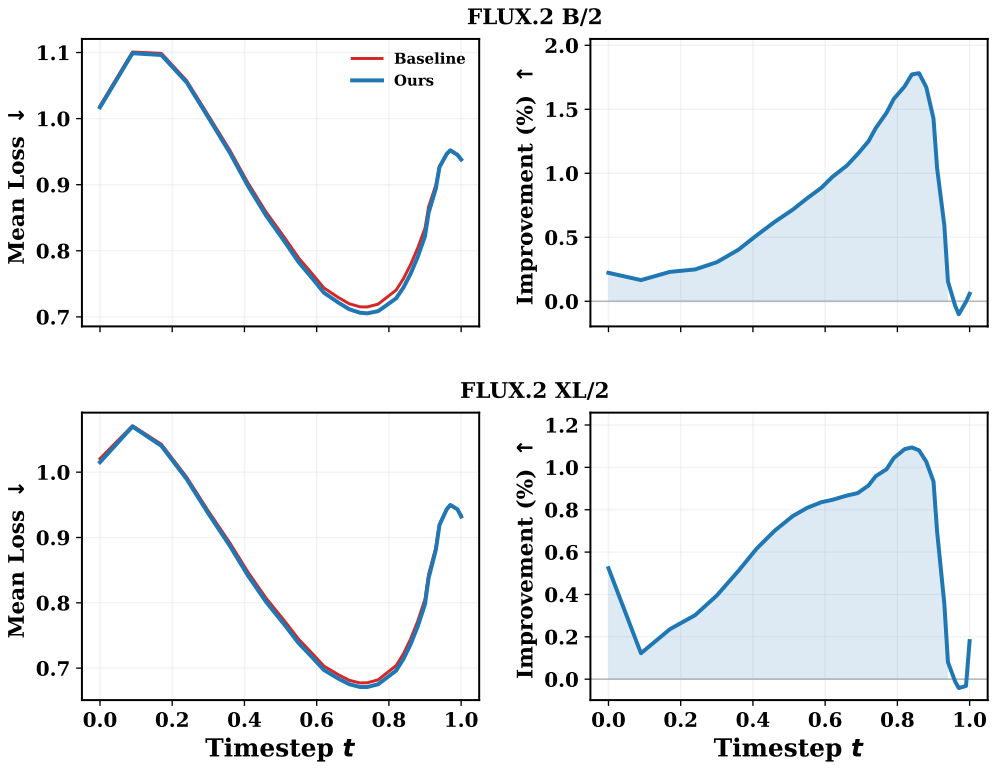


Figure 8: Mean diffusion loss on the fixed held-out validation set across different denoising steps. $t = 1$ indicates pure noise. The looped model shows the greatest benefits in early time steps, with improvement declining towards the cleaner image.

Algorithm 1 Looped diffusion step at flow time t.

Require: Noisy latent z_t, condition c, timestep t, total loops N, backprop window N_b

$v_0 \leftarrow \mathbf{0}, \quad z_{r,0} \leftarrow z_{r,\text{init}}$ ▷ learned scratchpad initialization

for $i = 0, \ldots, N-1$ **do**

  with `torch.set_grad_enabled`$(i \geq N - N_b)$**:**

    $v_{i+1},\ z_{r,\,i+1} \leftarrow f_\theta(z_t,\ c,\ t,\ v_i,\ z_{r,i})$

end for

return v_N

Table 7: ImageNet FID with classifier-free guidance. **Bold** / underline = best / 2nd best over all; blue / purple = best / 2nd best excluding inference-time scaling.

	DiT		SiT		Flux2	
Method	B/2	XL/2	B/2	XL/2	B/2	XL/2
Baseline	10.89	3.25	6.38	2.66	6.82	2.51
w/ 200 steps	10.83	3.16	6.04	2.57	6.30	2.32
w/ 800 steps	10.94	3.22	6.00	2.57	6.15	2.30
w/ RePaint-style loops (N10R4)	9.42	5.50	4.80	2.67	4.89	2.42
w/ RePaint-style loops (N10R16)	20.05	11.25	5.26	3.93	5.38	3.53
w/ Best-of-4	6.91	4.34	6.00	5.37	5.47	5.25
w/ Best-of-16	7.47	7.63	7.34	7.89	6.69	8.72
ELT-4	11.01	3.23	6.50	2.66	4.96	2.38
w/ 200 steps	10.91	3.14	6.17	2.58	4.71	2.35
w/ RePaint-style loops (N10R4)	9.70	5.76	4.77	2.61	4.01	2.56
w/ Best-of-4	6.93	4.36	6.03	5.40	5.63	6.10
Ours (4 Loops)	10.46	3.00	5.70	2.59	4.55	2.28
w/ 200 steps	10.39	2.88	5.37	**2.51**	4.26	2.21
w/ RePaint-style loops (N10R4)	9.28	5.78	4.24	2.60	3.66	2.41
w/ Best-of-4	6.69	4.72	5.82	5.47	4.90	5.39
$\times 4$ model	**4.64**	**2.55**	**3.61**	2.98	**2.95**	**1.98**

F Additional Results

In Tables 7 to 11 we report additional comparison results, including both additional metrics (DINOv2-FD) and results with CFG. In Figures 9 and 10 we show additional qualitative comparisons between our model and the baselines.

G Additional Related Work

Post-Training Methods for Visual Generation A complementary line of work improves visual generation through post-training procedures rather than architectural changes. The dominant approach applies RL-based preference optimization directly to the autoregressive tokens or flow matching trajectories that produce the image. For autoregressive image models, AR-GRPO (Yuan et al., 2025), STAGE (Ma et al., 2025b), and the token-level component of T2I-R1 (Jiang et al., 2026) apply GRPO-style (Shao et al., 2024) preference optimization to image-patch tokens. For flow matching and diffusion models, Flow-GRPO (Liu et al., 2026) and DanceGRPO (Xue et al., 2025) adapt GRPO to the continuous setting by reformulating the ODE sampling process as an SDE for exploration, with subsequent work (Zheng et al., 2025; Ma et al., 2026) tackling the resulting training dynamics. A related thread uses VLMs as external judges to refine prompts or to score generated images for selection, both as a training signal and at inference time (Betker et al., 2023; Harvey & Wood, 2023; Guo et al., 2025b). These methods improve the denoising trajectory's alignment with external preferences, but they do not surface an inference-time scaling knob: the trained model still generates each image in a fixed number of forward passes, with no analogue to the compute-for-thinking trade-off that motivates our work.



Figure 9: Additional qualitative comparison across methods, using CFG=4.5. Each row shows the same sample generated by our 4-loop retrofitted model, a Flux2-XL/2 baseline, retrofitted ELT-4, and the baseline scaled up to ×4 parameters.

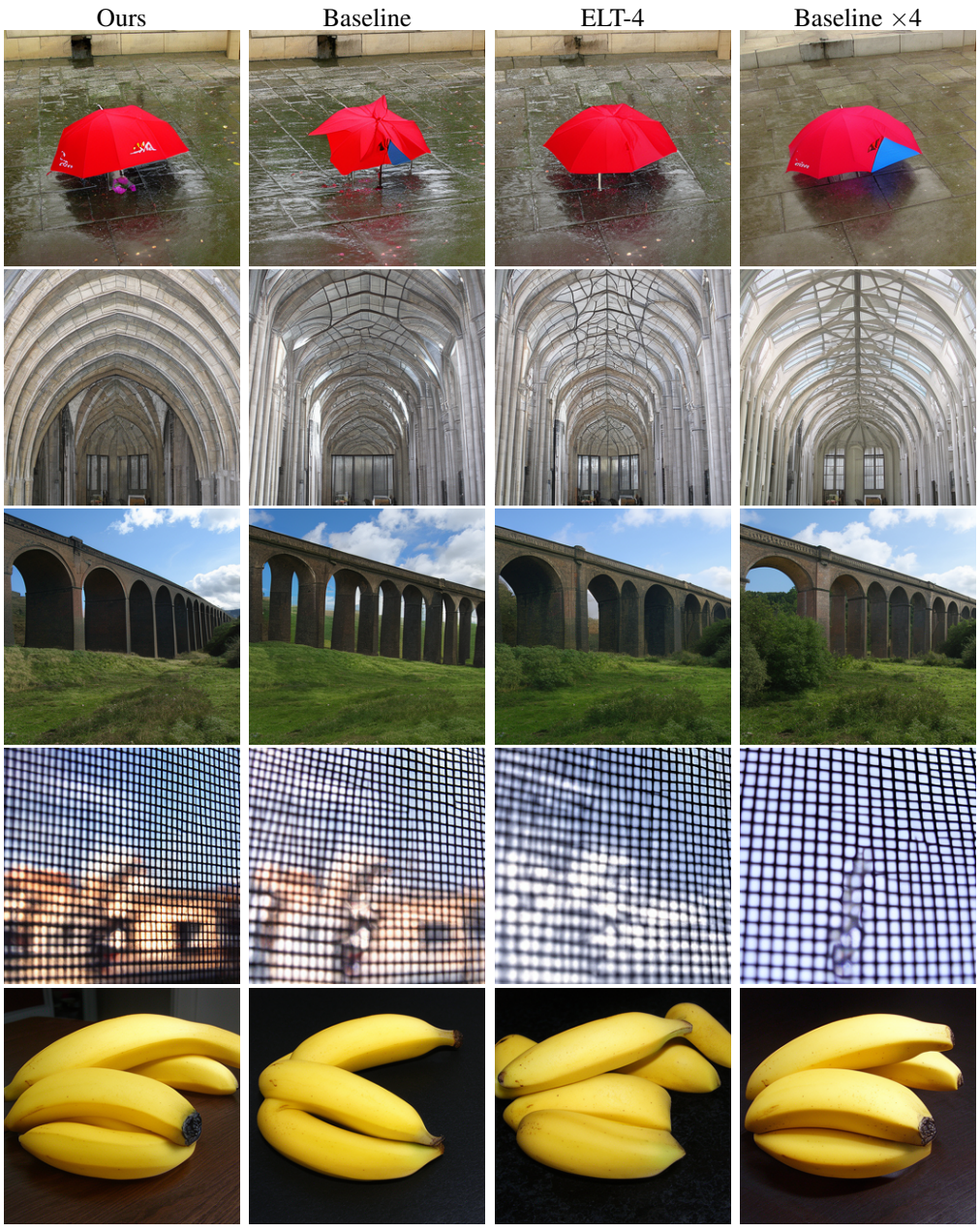


Figure 10: Additional qualitative comparison across methods, using CFG=4.5. Each row shows the same sample generated by our 4-loop retrofitted model, a Flux2-XL/2 baseline, retrofitted ELT-4, and the baseline scaled up to ×4 parameters.

Table 8: ImageNet DINO-FD without classifier-free guidance. **Bold** / underline = best / 2nd best over all; blue / purple = best / 2nd best excluding inference-time scaling.

	DiT		SiT		Flux2	
Method	B/2	XL/2	B/2	XL/2	B/2	XL/2
Baseline	689.3	376.7	634.8	363.1	453.1	171.7
w/ 200 steps	694.7	379.8	619.5	347.8	430.7	153.2
w/ 800 steps	698.9	383.3	616.2	344.2	424.3	148.9
w/ RePaint-style loops (N10R4)	543.7	254.6	575.5	308.0	353.4	138.0
w/ RePaint-style loops (N10R16)	654.8	332.9	484.5	246.3	291.7	128.1
w/ Best-of-4	541.0	242.0	484.9	237.1	345.9	112.0
w/ Best-of-16	**444.1**	**185.0**	**393.0**	**186.7**	283.6	101.3
ELT-4	689.9	376.3	636.0	363.7	379.3	135.2
w/ 200 steps	696.7	378.6	620.6	348.7	360.7	120.5
w/ RePaint-style loops (N10R4)	550.7	259.7	574.3	309.7	309.1	110.0
w/ Best-of-4	543.8	242.2	485.5	236.8	284.6	95.6
Ours (4 Loops)	660.0	306.0	570.6	299.1	330.2	126.3
w/ 200 steps	665.2	306.2	555.8	284.8	309.6	111.4
w/ RePaint-style loops (N10R4)	525.5	208.9	512.0	248.2	266.5	108.4
w/ Best-of-4	514.6	190.0	428.1	190.7	**239.1**	**87.6**
$\times 4$ model	474.0	321.9	488.1	354.4	263.2	122.8

Table 9: ImageNet DINO-FD with classifier-free guidance. **Bold** / underline = best / 2nd best over all; blue / purple = best / 2nd best excluding inference-time scaling.

	DiT		SiT		Flux2	
Method	B/2	XL/2	B/2	XL/2	B/2	XL/2
Baseline	430.7	185.1	371.3	175.2	280.5	96.2
w/ 200 steps	435.4	186.6	362.3	168.2	264.7	88.8
w/ 800 steps	438.7	188.7	360.4	166.9	260.2	87.2
w/ RePaint-style loops (N10R4)	292.5	125.7	298.2	133.7	215.2	84.5
w/ RePaint-style loops (N10R16)	358.6	164.3	241.2	**114.5**	187.2	89.5
w/ Best-of-4	324.9	133.1	275.4	133.5	215.3	90.3
w/ Best-of-16	271.4	127.8	**236.2**	133.8	188.3	99.6
ELT-4	434.1	185.3	372.9	175.6	226.7	84.5
w/ 200 steps	439.3	186.5	363.6	169.1	214.8	79.6
w/ RePaint-style loops (N10R4)	297.0	127.8	297.5	135.2	186.0	77.0
w/ Best-of-4	327.0	133.3	277.1	133.8	177.2	86.4
Ours (4 Loops)	418.5	158.4	337.0	150.7	202.6	80.8
w/ 200 steps	423.2	158.6	328.8	145.1	190.3	**76.4**
w/ RePaint-style loops (N10R4)	288.0	119.4	271.2	119.8	164.7	**76.3**
w/ Best-of-4	313.2	**118.3**	251.2	121.1	156.4	82.0
$\times 4$ model	**248.4**	153.7	252.9	174.9	**142.5**	**75.9**

Table 10: ImageNet Precision with classifier-free guidance. **Bold** / underline = best / 2nd best over all; blue / purple = best / 2nd best excluding inference-time scaling.

	DiT		SiT		Flux2	
Method	B/2	XL/2	B/2	XL/2	B/2	XL/2
Baseline	0.823	0.882	0.862	0.902	0.841	0.906
w/ 200 steps	0.821	0.880	0.858	0.902	0.848	0.914
w/ 800 steps	0.820	0.879	0.857	0.903	0.850	0.916
w/ RePaint-style loops (N10R4)	0.881	0.934	0.890	0.929	0.854	0.913
w/ RePaint-style loops (N10R16)	0.797	0.905	0.904	0.943	0.828	0.899
w/ Best-of-4	0.909	0.954	0.934	0.964	0.920	0.962
w/ Best-of-16	**0.941**	**0.974**	**0.958**	**0.979**	**0.949**	**0.976**
ELT-4	0.824	0.882	0.858	0.900	0.876	0.917
w/ 200 steps	0.820	0.881	0.855	0.901	0.882	0.925
w/ RePaint-style loops (N10R4)	0.878	0.933	0.889	0.931	0.888	0.922
w/ Best-of-4	0.910	0.954	0.933	0.964	**0.942**	0.965
Ours (4 Loops)	0.828	0.882	0.867	0.909	0.865	0.911
w/ 200 steps	0.825	0.884	0.866	0.908	0.872	0.917
w/ RePaint-style loops (N10R4)	0.881	0.939	0.895	0.935	0.870	0.912
w/ Best-of-4	0.911	0.957	0.938	0.964	0.939	0.963
$\times 4$ model	0.873	0.891	0.890	0.912	0.888	0.911

Table 11: ImageNet Recall with classifier-free guidance. **Bold** / underline = best / 2nd best over all; blue / purple = best / 2nd best excluding inference-time scaling.

	DiT		SiT		Flux2	
Method	B/2	XL/2	B/2	XL/2	B/2	XL/2
Baseline	0.683	0.694	**0.666**	0.677	**0.698**	0.689
w/ 200 steps	**0.687**	**0.697**	**0.667**	**0.678**	0.694	0.679
w/ 800 steps	**0.689**	**0.699**	**0.666**	0.677	**0.695**	0.680
w/ RePaint-style loops (N10R4)	0.547	0.547	0.641	0.650	**0.702**	0.678
w/ RePaint-style loops (N10R16)	0.454	0.451	0.596	0.586	**0.695**	0.654
w/ Best-of-4	0.582	0.550	0.546	0.524	0.588	0.538
w/ Best-of-16	0.493	0.438	0.444	0.406	0.503	0.427
ELT-4	0.678	**0.696**	**0.664**	**0.678**	0.670	0.672
w/ 200 steps	**0.686**	**0.701**	**0.666**	**0.683**	0.674	0.668
w/ RePaint-style loops (N10R4)	0.549	0.546	0.644	0.647	0.677	0.660
w/ Best-of-4	0.579	0.567	0.548	0.523	0.547	0.523
Ours (4 Loops)	0.674	0.692	0.659	0.671	0.690	0.684
w/ 200 steps	0.681	**0.696**	**0.665**	0.675	**0.695**	0.677
w/ RePaint-style loops (N10R4)	0.551	0.545	0.634	0.639	**0.696**	0.675
w/ Best-of-4	0.575	0.545	0.540	0.507	0.573	0.537
$\times 4$ model	0.680	**0.697**	**0.668**	0.661	0.695	**0.699**