Fine-Tuning Aligns Preferred Test-Time Compute Depth in Looped Transformers

Muhammad Aaliyan
Independent Researcher, Pakistan

May 3, 2026

Abstract

Adaptive inference asks when additional test-time compute improves model quality and when it merely adds latency. Looped Transformers make this question concrete because they expose inference depth as a runtime knob: parameters stay fixed while the number of loop iterations changes. We study whether supervised fine-tuning (SFT) shapes the *preferred* test-time compute depth of a looped language model. Starting from LoopFormer-3block-8iterations, we fine-tune one model with 8 training loops and another with 4, then sweep inference loops in $\{2, 4, 6, 8, 10, 12, 16\}$. Our primary metric is gold-answer negative log-likelihood (NLL) on GSM8K, paired with exact generation cost in seconds per sample. Across both a discovery run and a seeded confirmation run, the 8-loop-trained model achieves its best test NLL at 8 loops, while the 4-loop-trained model achieves its best test NLL at 4 loops. Additional loops beyond the training-aligned optimum degrade quality while sharply increasing latency. The base model does not show a stable optimum across reruns, but the post-SFT models do. We interpret this as evidence that preferred inference budget is a learned property of post-training, not a free monotonic scaling knob.

1 Introduction

Resource-adaptive inference is increasingly important for foundation models. In deployment, practitioners care not only about peak quality but also about how quality changes with compute, latency, and cost. Yet many models still expose a rigid inference path: every example pays the same computation budget regardless of difficulty.

Looped Transformers offer a useful alternative. By reapplying the same block multiple times, they expose a simple test-time compute control: keep parameters fixed and vary the number of loop iterations. This makes them a natural setting for studying adaptive inference and quality-resource tradeoffs [4, 5, 6].

The question we ask is practical and, to our knowledge, underexplored: after supervised fine-tuning on a reasoning task, what test-time compute depth does a looped model actually prefer? If a model is trained with 8 loops, can one improve it by simply running 12 or 16 loops at inference? Or does post-training lock in a task-aligned compute budget?

We study this question on GSM8K [2] using LoopFormer [4]. Because small-scale exact-match generation is noisy in our current setup, we rely primarily on gold-answer negative log-likelihood (NLL), which provides a denser signal about whether extra compute makes the correct answer more probable. We pair this with measured seconds per sample to expose the quality-latency frontier.

The main result is simple and stable across two runs. A model trained with 8 loops prefers 8 loops at inference; a model trained with 4 loops prefers 4 loops. Past that point, quality worsens

while cost rises steeply. We therefore argue that for looped language models, useful test-time compute is not a free monotonic bonus. It is a learned, post-training-aligned budget.

2 Related Work

Recent work has revived looped and recurrent Transformer-style architectures as a route to inference-time scaling. LoopFormer studies budget-aware latent reasoning with elastic depth [4]. Parcae investigates stable looped language models and scaling behavior under test-time looping [5]. Saunshi et al. [6] frame looping as a mechanism for latent reasoning, while other work studies where recursive depth helps and where it fails [3, 1].

Our contribution is narrower than a new architecture or systems paper. We do not introduce a new adaptive runtime mechanism. Instead, we isolate a post-training question: does supervised fine-tuning determine the compute depth at which adaptive inference is most useful? This places the work at the intersection of adaptive test-time compute and post-training behavior.

3 Experimental Setup

3.1 Models and Data

We start from `armenjeddi/LoopFormer-3block-8iterations` and use GSM8K [2] as the reasoning task. To keep runtime bounded on Kaggle T4 $\times 2$, we fine-tune on a 512-example subset of the GSM8K train split. Exact-match generation uses 40 held-out test examples. Gold-answer NLL uses 200 examples in the discovery run (Version 16) and 300 examples in the seeded confirmation run (Version 17), plus matching train-split diagnostics.

3.2 Training

We fine-tune two looped variants:

- **Train4**: trained with 4 loop iterations
- **Train8**: trained with 8 loop iterations

Both use two epochs, batch size 1, gradient accumulation 8, learning rate 5×10^{-6}, maximum sequence length 256, and fp32 optimization for stability. The confirmation run uses explicit seed 2026 and deterministic cuDNN settings.

3.3 Evaluation

We sweep inference loops in $\{2, 4, 6, 8, 10, 12, 16\}$ for teacher-forced gold-answer NLL, and in $\{2, 8, 16\}$ for exact-match generation. Lower NLL is better. We also measure generation latency in seconds per sample. Exact-match uses Wilson 95% confidence intervals because the evaluation slice is intentionally small.

4 Results

4.1 Preferred Compute Depth Tracks Training Depth

Table 1 summarizes the core result. Across both runs, the Train8 model achieves its best test NLL at 8 loops, while the Train4 model achieves its best test NLL at 4 loops. The base model does not

Table 1: Best loop counts and best NLL values across the discovery run (V16) and seeded confirmation run (V17). The stable effect is the alignment between SFT depth and preferred inference depth.

Run	Setting	Base test	Train8 test	Train4 test	Train/test match
V16	200-sample NLL	16 / 11.61	8 / 6.43	4 / 6.79	yes
V17	300-sample NLL, seed 2026	2 / 12.84	8 / 6.53	4 / 7.03	yes

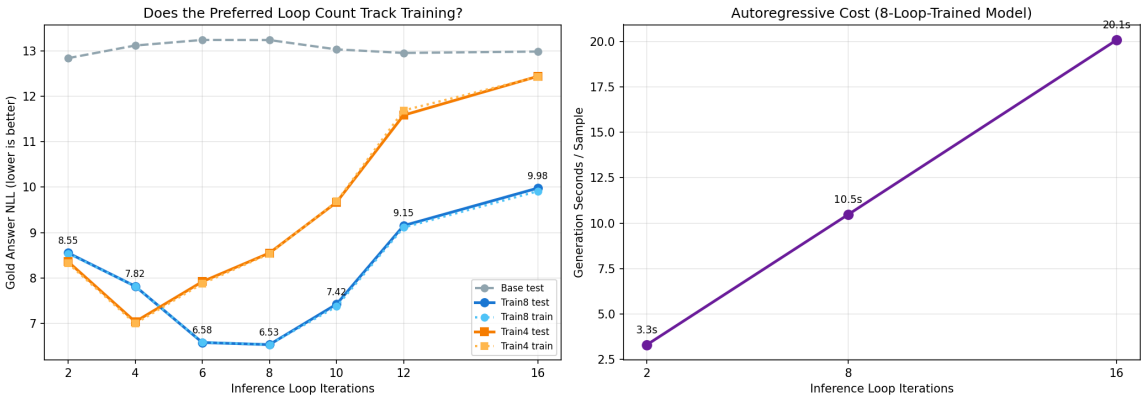

Figure 1: Seeded confirmation run (V17). Left: gold-answer NLL over inference loops. Right: measured generation latency for the 8-loop-trained model.

show a stable optimum across reruns.

Figure 1 shows the seeded confirmation run. The Train8 model improves up to 8 loops and then degrades. The Train4 model reaches its minimum at 4 loops and worsens thereafter. This is exactly the opposite of a "more loops always help" story.

4.2 Quality-Latency Tradeoff is Sharp

The latency curve in Figure 1 makes the systems implication plain. In Version 17, the Train8 model takes 3.30s/sample at 2 loops, 10.48s/sample at 8 loops, and 20.10s/sample at 16 loops. Past the preferred compute depth, we pay substantially more runtime for worse answer likelihood. This gives a concrete quality-resource frontier with a nontrivial optimum.

4.3 Exact-Match Remains Weak

Exact-match is much weaker than the answer-likelihood signal. In Version 16, the Train8 model scored 0/40 at loops 2, 8, and 16. In Version 17, it scored 1/40 at 2 loops and 0/40 at 8 and 16 loops. Figure 2 therefore serves as a cautionary diagnostic: extra compute can shape latent answer preference before it translates into robust free-form decoding quality.

4.4 Training is Stable

Both trained variants optimize stably. In Version 17, Train8 loss decreases from 10.57 to 7.94 and Train4 decreases from 10.64 to 8.04. This helps rule out the possibility that the preferred-depth result is just an artifact of unstable optimization.

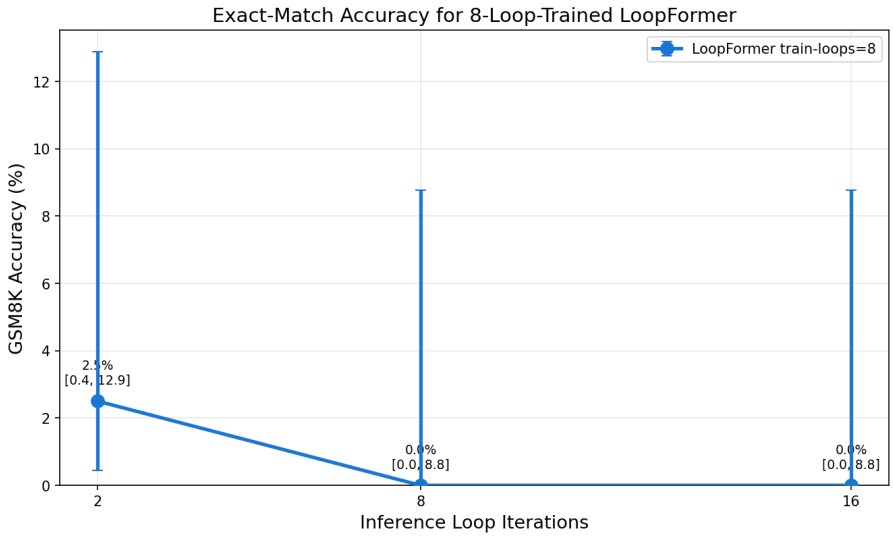

Figure 2: Exact-match accuracy for the 8-loop-trained model in the seeded confirmation run.

5 Discussion

The main lesson for adaptive inference is that test-time compute in a looped model behaves like a learned budget, not like a universally helpful dial. Once the model is fine-tuned on a task with a particular loop count, that count becomes a strong predictor of where quality peaks.

This result matters because many adaptive-inference stories implicitly assume that extra computation can be applied greedily at deployment time. Our evidence pushes against that assumption. For looped language models, post-training appears to organize the dynamics around a task-aligned amount of iterative refinement. After that point, additional iterations may overshoot and reduce quality while still increasing latency.

The result is also notable because it is visible in teacher-forced answer likelihood even when exact-match decoding is weak. That suggests a useful methodological point for adaptive-inference research: when free-form generation is too noisy to expose a compute-quality trend, denser latent metrics can reveal whether extra compute is helping internally or not.

6 Limitations

This is still a narrow pilot study. We use a 512-example train subset, a small exact-match evaluation slice, and one main architecture family. The strongest evidence is on gold-answer NLL rather than end-to-end decoded reasoning accuracy. A stronger next version would add a second looped backbone, larger evaluation, and decoding interventions that test whether latent gains can be converted into better output quality.

7 Conclusion

We study adaptive inference in looped Transformers through the lens of preferred test-time compute depth. Across both a discovery run and a seeded confirmation run, a model trained with 8 loops prefers 8 loops at inference, and a model trained with 4 loops prefers 4 loops. Beyond these task-aligned optima, answer likelihood worsens while latency rises sharply. We therefore conclude

that supervised fine-tuning can align the useful inference budget of a looped model with the compute depth used during training.

A Additional Figures

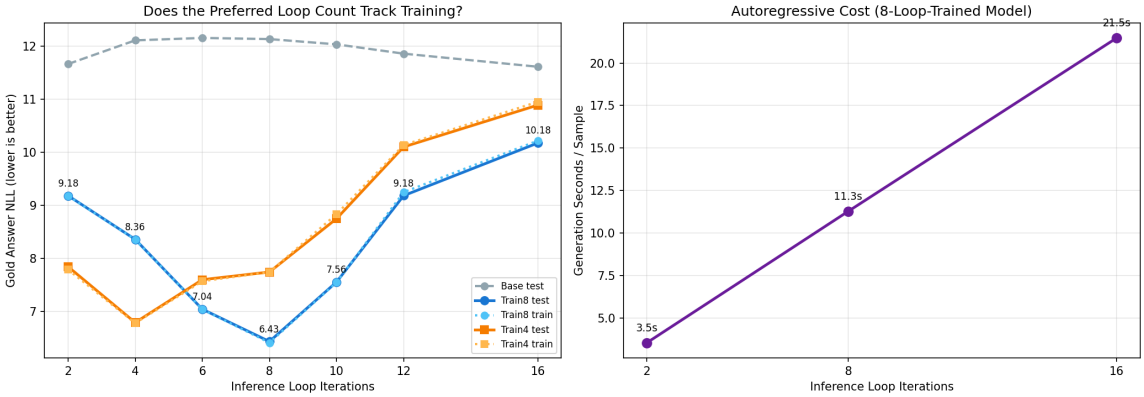

Figure 3: Discovery run (V16). The same qualitative quality-resource pattern appears before the seeded rerun.

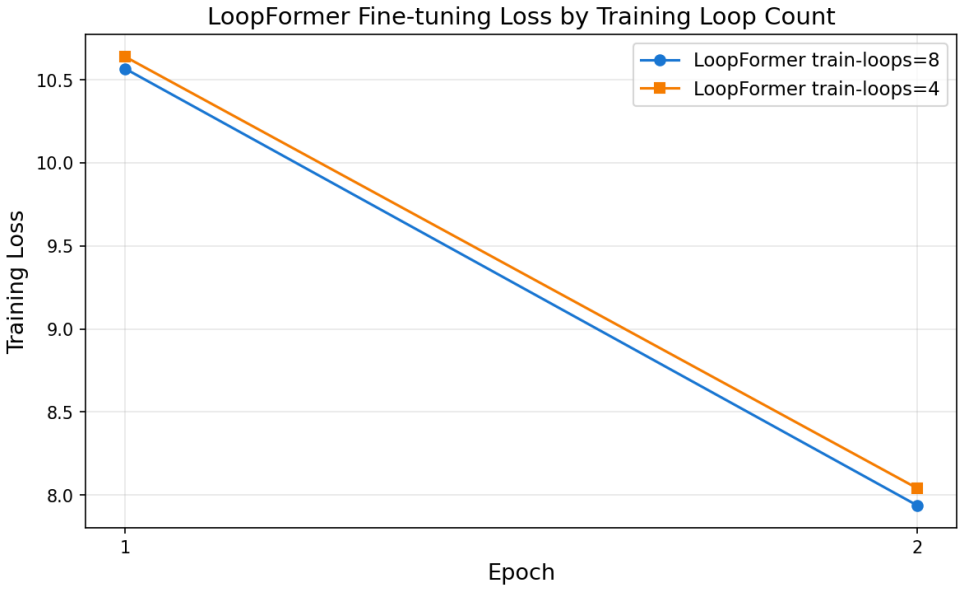

Figure 4: Training-loss curves for the seeded confirmation run (V17).

References

[1] Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise LoRA, 2025. URL https://arxiv.org/abs/2410.20672.

[2] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

[3] Markus Frey, Behzad Shomali, Ali Hamza Bashir, David Berghaus, Joachim Koehler, and Mehdi Ali. Adaptive loops and memory in transformers: Think harder or know more?, 2026. URL https://arxiv.org/abs/2603.08391.

[4] Ahmadreza Jeddi, Marco Ciccone, and Babak Taati. Loopformer: Elastic-depth looped transformers for latent reasoning via shortcut modulation, 2026. URL https://arxiv.org/abs/2602.11451.

[5] Hayden Prairie, Zachary Novack, Taylor Berg-Kirkpatrick, and Daniel Y. Fu. Parcae: Scaling laws for stable looped language models, 2026. URL https://arxiv.org/abs/2604.12946.

[6] Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers, 2025. URL https://arxiv.org/abs/2502.17416.