

Improving Efficient Recurrent Depth Via Next-Latent Alignment

Daniel Tcheurekdjian Yuekun Yao Huan Sun

The Ohio State University

{tcheurekdjian.1, yao.1267, sun.397}@osu.edu

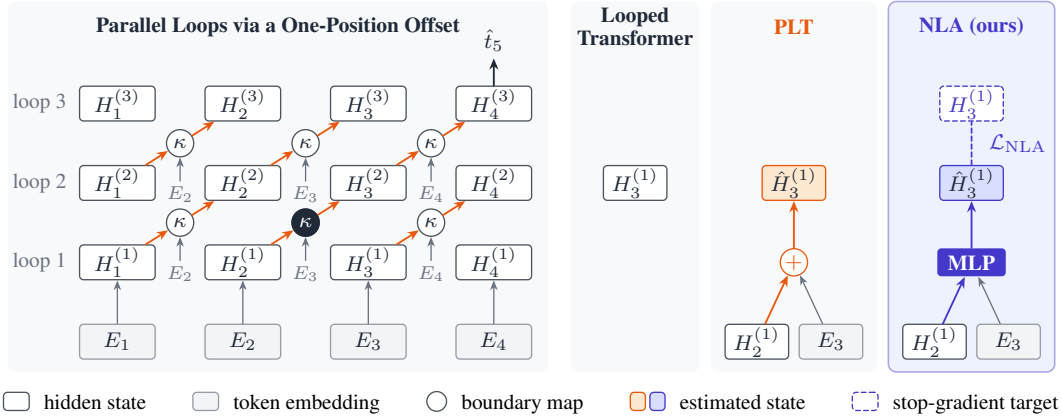


Figure 1: **Left:** the PLT. **Right:** the highlighted boundary in the looped Transformer, the PLT, and our method.

Abstract

Looped Transformers share parameters across depth, allowing for parameter-efficient language modeling at the cost of latency and KV-cache size scaling with loops. Parallel Loop Transformers (PLTs) condition each loop on the previous loop’s output, at the previous token position, removing the latency penalty incurred by looped Transformers by allowing all loops to decode simultaneously. However, PLTs face degraded absolute and per-loop performance relative to looped Transformers. This paper introduces a simple intervention that improves the performance of PLTs, while preserving decode-time parallelism and adding only a small parameter and compute overhead. By instantiating a small MLP at the loop boundary and training it jointly with the model to predict same-position hidden states at previous loops, we restore monotonic improvement with loop count, recover most of a looped Transformer’s per-loop gain, and close a substantial fraction of the remaining gap to a parameter- and FLOP-matched looped Transformer. Our results suggest that parallel recurrence works best when the state transitions available at decoding time are trained alongside the representations that depend on them.

1 Introduction

Edge and on-device deployments of large language models are often constrained not solely by available compute, but also by available memory. Under such limitations, parameter-efficient architectures become attractive. One such architecture is the looped Transformer. Looped Transformers

are Transformers which serially repeat computation via shared layer blocks (loops) allowing for parameter-efficient learning [Dehghani et al., 2019, Giannou et al., 2023, Saunshi et al., 2025]. However, in doing so, they incur test-time latency and KV-cache memory penalties scaling with loops, limiting their real-world efficacy.

Recent work has introduced the Parallel Loop Transformer (PLT), which removes the decode-time same-position dependence of looped Transformers through cross-loop parallelism (CLP) [Wu et al., 2025]. At each loop, CLP constructs the loop’s input from the previous loop’s output at the *previous* token position, summed with the embedding of the current token, allowing for the simultaneous decoding of all loops during a forward pass. Under this regime, the current token’s embedding enters every loop, but its same-position hidden state cannot pass directly from one loop to the next during a decoding step. In pretraining, PLTs exhibit degraded absolute performance and diminishing returns with loops relative to looped Transformers, and at scale, regression below unlooped baselines has been observed [Yang et al., 2026].

This paper develops a simple intervention on this positional mismatch that improves PLT performance and restores monotonic improvement with loops. We apply a small learned MLP at the loop boundary (Figure 1), and jointly train the MLP and network via a next-latent objective to approximate the model’s same-position hidden state at the previous loop [Teoh et al., 2025]. We find that our *Next-Latent Alignment (NLA)* method improves pretraining cross-entropy across loop counts and attention configurations while remaining fully parallel, and verify that this cross-entropy improvement largely transfers to benchmarks.

2 Method

2.1 (Parallel) Looped Transformers

The looped Transformer applies a shared block of layers $\mathcal{B}$ a fixed number of times K , such that effective depth grows decoupled from parameter count. Specifically, given token embeddings $E \in \mathbb{R}^{S \times d}$ for a sequence of length S , and writing $H^{(\ell)} \in \mathbb{R}^{S \times d}$ for the states after loop ℓ , the looped Transformer applies:

$$H^{(0)} = E, \quad H^{(\ell)} = \mathcal{B}\left(H^{(\ell-1)}\right), \quad \ell = 1, \dots, K, \quad (1)$$

reading next-token logits from $H_i^{(K)}$.

Due to the sequential nature of the looped Transformer, test-time latency and KV-cache size both scale as $\mathcal{O}(K)$ [Dehghani et al., 2019, Giannou et al., 2023].

Comparatively, a PLT applies a per-loop offset at each of the $K - 1$ loop boundaries

$$X_i^{(\ell)} = E_i + H_{i-1}^{(\ell-1)}, \quad \ell = 2, \dots, K, \quad (2)$$

computing $H^{(\ell)} = \mathcal{B}(X^{(\ell)})$, with $X^{(1)} = E$ and $X_1^{(\ell)} = E_1$. As loop ℓ at position i depends on the previous loop at only position $i - 1$, all K loops advance simultaneously during decoding.

We follow Wu et al. [2025] in studying three attention configurations for PLTs: full attention, first-loop KV-cache sharing (KV-Share) and KV-Share with Gated Sliding Window Attention (G-SWA). Under KV-Share, loops beyond the first attend per layer to the keys and values computed at loop 1. G-SWA attends over the loop’s own keys and values within the last 64 positions, combined with the shared cache by a per-head sigmoid gate.

2.2 Next Latent Prediction

The short horizon of next-token prediction has motivated auxiliary future-prediction signals [Gloeckle et al., 2024, DeepSeek-AI, 2024, Teoh et al., 2025]. Next-latent prediction (NextLat) [Teoh et al., 2025] augments next-token training of a standard transformer with a latent dynamics model p_ψ that predicts the next hidden state from the current state and the incoming token, trained by regression with a target-side stop-gradient:

$$\mathcal{L}_{\text{next-h}} = \text{SmoothL1}(\text{sg}[h_{t+1}], p_\psi(h_t, x_{t+1})), \quad (3)$$

added to the language-modeling loss with a scalar coefficient, alongside a distillation term aligning predicted states in token space.

2.3 Next-Latent Alignment at the Loop Boundary

The implementation of the per-loop offset can be considered a boundary map κ :

$$X_i^{(\ell)} = \kappa\left(H_{i-1}^{(\ell-1)}, E_i\right), \quad \ell = 2, \dots, K. \quad (4)$$

Composed with the shared block, κ maps a previous-position state and an incoming token to a current-position state, similar to the transition p_ψ . Our method, NLA, makes κ learnable and trains its output toward $H_i^{(\ell-1)}$.

We instantiate κ as a two-layer MLP over the concatenated pair,

$$\kappa_{\text{MLP}}\left(H_{i-1}^{(\ell-1)}, E_i\right) = \text{MLP}\left(\text{RMSNorm}\left[H_{i-1}^{(\ell-1)}; E_i\right]\right). \quad (5)$$

The MLP is additionally trained with the regression term of the objective of §2.2:

$$\mathcal{L}_{\text{NLA}} = \mathcal{L}_{\text{CE}} + \frac{1}{K-1} \sum_{\ell=2}^K \left\langle \text{SmoothL1}\left(\text{sg}\left[H_i^{(\ell-1)}\right], X_i^{(\ell)}\right) \right\rangle_i. \quad (6)$$

3 Experiments

3.1 Experimental Setup

We train 143M-parameter PLTs alongside looped Transformer baselines on FineWeb-Edu [Penedo et al., 2024] for a fixed budget of 11.2B tokens, ablating across the individual components of our method. We train across full attention, KV-Share, and G-SWA, and across loop counts $K \in \{2, 3, 4\}$ (Table 1).

Table 1: Constructions. In looped Transformer+ $\mathcal{L}_{\text{NLA}}$ the MLP is an auxiliary predictor. Looped Transformer+in-block places the MLP inside the shared block. In + $\mathcal{L}_{\text{NLA}}$ the model is trained with Eq. 6 without the boundary MLP in the forward pass. Full details can be found in Appendix A.

construction	boundary κ	objective	Δparams	ΔFLOPs
PLT	$H + E$	CE	0	0
PLT+MLP	$\text{MLP}(\text{RMSNorm}[H; E])$	CE	+7.1M	+12.5%
NLA (ours)	$\text{MLP}(\text{RMSNorm}[H; E])$	Eq. 6	+7.1M	+12.5%
PLT+ $\mathcal{L}_{\text{NLA}}$	$H + E$	Eq. 6	0	0
Looped Transformer	$X^{(\ell)} = H^{(\ell-1)}$	CE	0	0
Looped Transformer+MLP	$\text{MLP}(\text{RMSNorm}[H_i^{(\ell-1)}; E_i])$	CE	+7.1M	+12.5%
Looped Transformer+ $\mathcal{L}_{\text{NLA}}$	$X^{(\ell)} = H^{(\ell-1)}$	Eq. 6 (aux. head)	+7.1M	+12.5%
Looped Transformer+in-block	$X^{(\ell)} = H^{(\ell-1)}$ (MLP in block, $\ell \geq 2$)	CE	+7.1M	+12.5%

We use a BPE tokenizer and AdamW with linear warmup and cosine decay as our optimizer (Appendix A).

3.2 Main Results

During pretraining, NLA restores improvement with loop count and closes about half of the gap to a parameter- and FLOP-matched looped Transformer. To measure the effect of our method during pretraining, we report validation loss on a held-out set of 2.1M tokens from the FineWeb-Edu dataset across our trained models, shown in Table 2. Our results show that our method improves on the PLT in all cells, and at four loops, our method closes 56%, 50%, and 38% of the difference between the PLT and the parameter- and FLOP-matched looped Transformer+in-block (Table 2 caption) under full attention, KV-Share, and G-SWA respectively.

Table 2: Main results of our method and baselines across configurations. Serial passes: block applications per decoded token. Per loop: the change in loss from $K=2$ to $K=4$, halved. Bold: best parallel construction. §: mean of four seeds. ‡: collapsed during training. Looped Transformer+in-block (Table 1), trained at $K=4$ only: 2.7948, 2.8452, and 2.7989 under full attention, KV-Share, and G-SWA.

Attention	KV cache	Construction	Serial Passes	Validation Loss			$K=2 \rightarrow 4$
				$K=2$	$K=3$	$K=4$	Per Loop
Full Attention	$K S d$	Looped Transformer	K	2.8755	2.8440	2.8243	-0.0256
		Looped Transformer+MLP	K	2.8735	2.8730	3.0492	+0.0879
		Looped Transformer+ $\mathcal{L}_{NLA}$	K	2.8694	2.8436	2.8204	-0.0245
		PLT	1	2.8921	2.8768§	2.8785§	-0.0068
		PLT+ $\mathcal{L}_{NLA}$	1	2.8966	3.1968‡	3.1599‡	+0.1316‡
		PLT+MLP	1	2.8763	2.8383	2.8368	-0.0197
		NLA (ours)	1	2.8652	2.8396	2.8318	-0.0167
KV-Share	$S d$	Looped Transformer	K	2.8973	2.8775	2.8695	-0.0139
		Looped Transformer+MLP	K	2.8849	2.8763	2.8667	-0.0091
		Looped Transformer+ $\mathcal{L}_{NLA}$	K	2.8891	2.8710	2.8618	-0.0136
		PLT	1	2.9174	2.9032	2.9053	-0.0061
		PLT+ $\mathcal{L}_{NLA}$	1	2.9305	2.9253	3.1889‡	+0.1292‡
		PLT+MLP	1	2.8984	2.8977	2.8854	-0.0065
		NLA (ours)	1	2.8969	2.8879	2.8750	-0.0109
G-SWA	$S d + 64(K-1)d$	Looped Transformer	K	2.8708	2.8418	2.8232	-0.0238
		Looped Transformer+MLP	K	2.8675	2.8710	2.8684	+0.0004
		Looped Transformer+ $\mathcal{L}_{NLA}$	K	2.8650	2.8344	2.8205	-0.0222
		PLT	1	2.9012	2.8752	2.8662	-0.0175
		PLT+ $\mathcal{L}_{NLA}$	1	2.9316	2.9010	2.8899	-0.0208
		PLT+MLP	1	2.8874	2.8707	2.8515	-0.0179
		NLA (ours)	1	2.8832	2.8523	2.8404	-0.0214

Our method improves monotonically with loop count in every configuration, and recovers 65%, 78%, and 90% of a looped Transformer’s per-loop improvement under full attention, KV-Share, and G-SWA respectively, as shown in Figure 2.

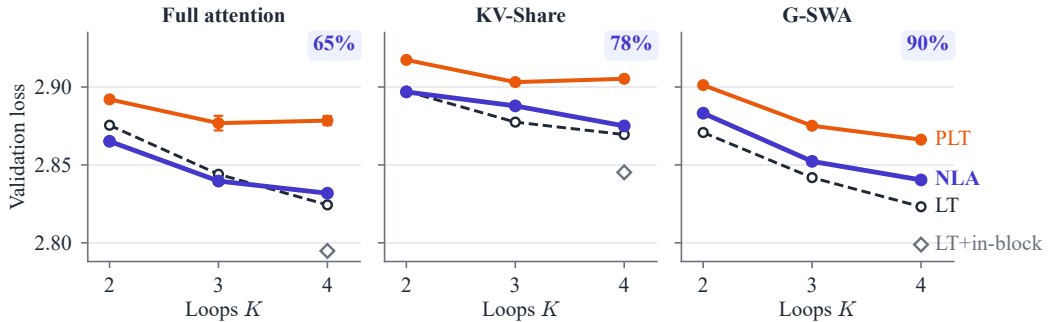


Figure 2: **Validation loss against loop count.** Percentages: NLA’s improvement from $K=2$ to $K=4$ relative to the looped Transformer’s (LT). Diamond: LT+in-block, parameter- and FLOP-matched to NLA, trained at $K=4$ only. Error bars: ± 1 standard deviation over four PLT seeds.

Our improvement holds on downstream evaluation under full attention and G-SWA. We report zero-shot accuracy at $K=4$ with the LM Evaluation Harness on ARC-Challenge and ARC-Easy [Clark et al., 2018], COPA [Roemmele et al., 2011], HellaSwag [Zellers et al., 2019], LAMBADA in its OpenAI and standard forms [Paperno et al., 2016], OpenBookQA [Mihaylov et al., 2018], PIQA [Bisk et al., 2020], RACE [Lai et al., 2017], SciQ [Welbl et al., 2017], and WinoGrande [Sakaguchi et al., 2020], shown in Table 3. In both configurations our method is the best parallel construction on average, trailing the looped Transformer by under one point of mean accuracy. The four full-attention PLT seeds at $K=4$ have a standard deviation of 0.15 points in mean accuracy, so our 0.28-point

margin over PLT+MLP under full attention is within seed variation, while the 1.03-point margin under G-SWA is not. Under KV-Share, the loss improvement does not carry over to benchmarks: the means are 43.45%, 43.43%, 43.42%, and 43.84% for PLT, PLT+MLP, our method, and the looped Transformer, as seen in Appendix D.

Table 3: Zero-shot accuracy (%) with LM Evaluation Harness [Gao et al., 2023] on 11 tasks at $K=4$ under full attention and G-SWA. *: token-length-normalized loglikelihood. §: mean of four seeds; every other column is a single seed. Bold: best parallel construction.

Task	Full Attention				G-SWA			
	PLT§	+MLP	NLA	Looped Transformer	PLT	+MLP	NLA	Looped Transformer
ARC-Challenge	23.14	23.55	22.18	23.04	23.55	23.38	22.78	25.09
ARC-Easy	55.16	55.13	55.72	56.10	54.80	55.51	55.47	56.31
COPA	63.75	61.00	63.00	64.00	62.00	62.00	67.00	65.00
HellaSwag*	34.32	35.34	35.71	36.05	34.47	34.84	34.92	36.04
LAMBADA (OpenAI)	25.76	26.18	26.33	29.36	25.36	25.25	26.92	28.10
LAMBADA (Standard)	19.45	21.77	21.54	24.65	18.22	19.39	19.37	19.95
OpenBookQA*	32.25	32.40	33.60	32.00	32.60	31.40	32.60	33.20
PIQA*	63.98	63.71	65.40	65.34	62.68	64.20	64.25	65.02
RACE	30.33	30.43	29.38	29.95	29.76	28.42	29.67	29.00
SciQ	82.25	82.00	82.20	83.50	81.90	81.50	82.20	82.40
WinoGrande	50.75	51.30	50.75	50.51	52.09	49.72	51.78	49.72
<i>Average</i>	43.74	43.89	44.17	44.95	43.40	43.24	44.27	44.53

Improvement is conditional on boundary placement and parameters. We utilize our ablations to examine the impact of several characteristics of interest for our method, shown in Figure 3. When we apply $\mathcal{L}_{NLA}$ to a PLT at $K=4$ without learnable capacity at the boundary, our training run collapses under full attention and KV-Share, and loss rises under G-SWA. Adding the boundary MLP to the PLT lowers loss in every configuration, while adding the same MLP to the looped Transformer, whose loops already receive the same-position state, raises loss under full attention and G-SWA. Adding $\mathcal{L}_{NLA}$ to PLT+MLP lowers loss further, accounting for 11%, 34%, and 43% of NLA’s improvement over the PLT.

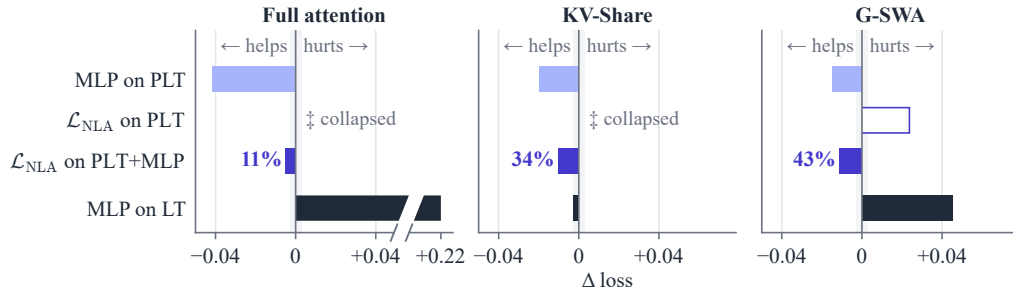


Figure 3: Change in validation loss from each addition. Percentages: the objective’s share of NLA’s improvement over the PLT. †: collapsed during training. Gray band: ± 1 standard deviation over four PLT seeds.

4 Related Work

Looped and parallel-loop Transformers. Shared-depth Transformers support parameter-efficient computation [Dehghani et al., 2019, Lan et al., 2019, Giannou et al., 2023]. Studies of language modeling and latent reasoning investigate the value of recurrent depth [Saunshi et al., 2025, Geiping et al., 2025, Zhu et al., 2025, Kohli et al., 2026]; other work retrofits recurrence into pretrained models [Bae et al., 2024, McLeish et al., 2025], varies recurrence at inference [Bae et al., 2025, Jeddi et al., 2026], or replaces softmax attention in looped Transformers with linear-time variants [Deng

et al., 2026]. The PLT replaces the same-position recurrence with delayed cross-loop inputs and shared attention [Wu et al., 2025]. LoopCoder-v2 reports diminishing and sometimes negative returns from additional parallel loops at 7B parameters, and relates them to positional mismatch [Yang et al., 2026]. Our experiments address the learned transition at this boundary.

Latent prediction. Multi-token prediction supplies additional token-level supervision [Gloeckle et al., 2024, DeepSeek-AI, 2024]; NextLat trains a latent predictor with target-side stop-gradient and source-side representation learning [Teoh et al., 2025]. EAGLE uses next-state prediction for speculative drafting [Li et al., 2024]. Our boundary MLP instead supplies the recurrent model’s next-loop input, making the learned prediction part of its ordinary language-model computation.

5 Discussion and Limitations

Our results suggest that effective parallel recurrence depends on learning representations that can be updated through the information available at each loop boundary. The form of these representations can vary: a boundary MLP gives later loops better-adapted inputs, which need not approximate hidden states. $\mathcal{L}_{\text{NLA}}$ trains the MLP’s output toward the same-position state $H_i^{(\ell-1)}$, but its gradient also reaches the network through the MLP’s inputs, so the network may learn source representations from which that state is easier to predict. Consistent with this, a linear map fit to predict $H_i^{(\ell-1)}$ from $[H_{i-1}^{(\ell-1)}; E_i]$ is more accurate on NLA’s states than on those of PLT+MLP or the PLT, while NLA’s own MLP is less accurate than that map (Table 5). Without an MLP, the objective can only be satisfied by changing the states themselves, and PLT+ $\mathcal{L}_{\text{NLA}}$ is worse than the PLT in all nine cells and collapses during training in three (Appendix B).

The application of $\mathcal{L}_{\text{NLA}}$ raises the per-loop improvement over PLT+MLP under KV-Share and G-SWA but not under full attention. With full attention, later loops retain access to their own historical computations; KV-Share replaces these caches with the first loop’s, making the boundary a more important route for information produced by additional loops, which may explain why $\mathcal{L}_{\text{NLA}}$ adds more per loop there.

Our method does not grow the KV cache relative to a PLT (Table 2), but adds 7.1M parameters, about 5% of the model, and 12.5% more FLOPs per token at $K=4$ (Table 1). On-device decoding usually runs at small batch sizes, where each step is limited by reading the weights and the KV cache from memory rather than by arithmetic. In that regime we expect the added FLOPs to matter less than the added weights, which are read once per decoding step for all $K-1$ boundaries.

Limitations. Our work is limited in scale: we train 143M-parameter models on 11.2B tokens with at most four loops, and only in pretraining, so it is unclear whether our method holds at larger scales, at higher loop counts, or beyond pretraining. Our experiments also do not fully identify the mechanism through which we improve PLTs. We did not measure wall-clock latency.

6 Conclusion

We introduce a simple technique that improves the performance of parallel looped Transformers while maintaining their latency and memory advantages, at the cost of minimal additional compute at train and test.

References

- Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise LoRA. *arXiv preprint arXiv:2410.20672*, 2024.
- 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. *arXiv preprint arXiv:2507.10524*, 2025.
- Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.
- DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.
- Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=HyzdRiR9Y7>.
- Chunyuan Deng, Yizhe Zhang, Rui-Jie Zhu, Yuanyuan Xu, Jiarui Liu, T. S. Eugene Ng, and Hanjie Chen. LT2: Linear-time looped transformers. *arXiv preprint arXiv:2605.20670*, 2026.
- Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation. Zenodo, 2023.
- Jonas Geiping, Sean 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. *arXiv preprint arXiv:2502.05171*, 2025.
- 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*, 2023. arXiv:2301.13196.
- Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In *International Conference on Machine Learning*, 2024. arXiv:2404.19737.
- Ahmadreza Jeddi, Marco Ciccone, and Babak Taati. LoopFormer: Elastic-depth looped transformers for latent reasoning via shortcut modulation. In *International Conference on Learning Representations*, 2026.
- Harsh Kohli, Srinivasan Parthasarathy, Huan Sun, and Yuekun Yao. Loop, think, & generalize: Implicit reasoning in recurrent-depth transformers. *arXiv preprint arXiv:2604.07822*, 2026.
- Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale Reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.
- 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.
- Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning*, 2024. arXiv:2401.15077.

- Sean McLeish, Ang Li, John Kirchenbauer, Dayal Singh Kalra, Brian R. Bartoldson, Bhavya Kaikhura, Avi Schwarzschild, Jonas Geiping, Tom Goldstein, and Micah Goldblum. Teaching pretrained language models to think deeper with retrofitted recurrence. *arXiv preprint arXiv:2511.07384*, 2025.
- Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.
- Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.
- Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024. *arXiv:2406.17557*.
- Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*, 2011.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.
- Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. *arXiv preprint arXiv:2502.17416*, 2025.
- Jayden Teoh, Manan Tomar, Kwangjun Ahn, Edward S. Hu, Tim Pearce, Pratyusha Sharma, Akshay Krishnamurthy, Riashat Islam, Alex Lamb, and John Langford. Next-latent prediction transformers learn compact world models. *arXiv preprint arXiv:2511.05963*, 2025.
- Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text (W-NUT)*, 2017.
- Bohong Wu, Mengzhao Chen, Xiang Luo, Shen Yan, Qifan Yu, Fan Xia, Tianqi Zhang, Hongrui Zhan, Zheng Zhong, Xun Zhou, Siyuan Qiao, and Xingyan Bin. Parallel loop transformer for efficient test-time computation scaling. *arXiv preprint arXiv:2510.24824*, 2025.
- Jian Yang, Shawn Guo, Wei Zhang, Tianyu Zheng, Yaxin Du, Haau-Sing Li, Jiajun Wu, Yue Song, Yan Xing, Qingsong Cai, Zelong Huang, Chuan Hao, Ran Tao, Xianglong Liu, Wayne Xin Zhao, Mingjie Tang, Weifeng Lv, Ming Zhou, and Bryan Dai. LoopCoder-v2: Only loop once for efficient test-time computation scaling. *arXiv preprint arXiv:2606.18023*, 2026.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.
- 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.

A Experimental details

We use a model dimension of 768 and six transformer layers per loop pass for our 143M-parameter models, with six attention heads of dimension 128, a ReLU^2 feed-forward dimension of 3072, RMSNorm in a post-residual-norm layout, QK-norm, no biases, and a RoPE base of 50,000. Input and output embeddings are tied, and value embeddings on alternating layers are re-indexed from the token ids at every loop. The vocabulary has 32,768 entries and the context length is 2048. All weights other than those named next are initialized with the scaled-zero scheme with orthogonal matrices; the boundary MLP and the in-block MLP are initialized with Gaussian weights of standard deviation $\sqrt{2/5d} \approx 0.023$ in both layers, and the G-SWA gate is zero-initialized. The boundary MLP has dimensions $1536 \rightarrow 3072 \rightarrow 768$ with RMSNorm before it and adds 7.1M parameters. Looped Transformer+in-block uses a matched $768 \rightarrow 4608 \rightarrow 768$ MLP with RMSNorm, applied residually after layer 3 of 6 in loops $\ell \geq 2$, and is trained at $K=4$ only. Every construction other than looped Transformer+in-block spans $K \in \{2, 3, 4\}$ and all three attention configurations. G-SWA uses a 64-token local window.

Our models are trained on batches of 256 sequences of length 2048, corresponding to 524K tokens per batch, for 11.2B tokens (21,362 updates). Across all runs we use AdamW with a max learning rate of 2×10^{-3} , 1,000 warmup updates (0.52B tokens), and cosine decay down to 2×10^{-5} , with $(\beta_1, \beta_2) = (0.9, 0.95)$, a weight decay of 0.1, and gradient clipping at 1.0. Hyperparameters are shared across the sweep without per-cell tuning. The regression term of Eq. 6 is SmoothL1 with $\beta = 1$, averaged over the 768 coordinates and over positions 2 to S ; the regressed tensor is the next loop’s input $X^{(\ell)}$ itself. Its coefficient is 1.

We train on 11.2B tokens of FineWeb-Edu [Penedo et al., 2024], tokenized with a 32,768-entry byte-fallback BPE tokenizer and packed into BOS-aligned rows of 2049 tokens. We report validation cross-entropy on a fixed held-out set of 1,024 rows (2.1M tokens), computed from the logits and excluding the auxiliary loss. All runs at a given loop count and attention configuration share seed and data order.

All constructions at a given seed start from the same stored initialization of the shared block and embeddings, so adding the boundary MLP never changes their initial weights. The KV-Share and G-SWA runs of the PLT constructions predate this protocol; PLT+MLP and NLA rerun under both protocols differ by at most 0.003 in validation loss. We use one seed throughout, except for the full-attention PLT at $K=3$ and $K=4$, where we report the mean of four seeds.

We train in bf16 with `torch.compile` on four NVIDIA H100 GPUs per run; a run takes 2.2 to 4.8 hours depending on loop count. The three looped Transformer+in-block runs were trained on a rented four-H100 node with the same code, data order, and initialization. The reported runs total about 235 node-hours (940 H100-hours).

B Component and latent analyses

Table 4 shows final-checkpoint diagnostics for the objective without the MLP ($\text{PLT} + \mathcal{L}_{\text{NLA}}$).

Table 4: $\text{PLT} + \mathcal{L}_{\text{NLA}}$ across configurations. D : norm of the last loop’s update. N : norm of $H^{(K)}$. The auxiliary loss is the regression term of Eq. 6. ‡: collapsed during training.

Attention	K	D		N		Auxiliary loss	
		PLT	$+\mathcal{L}_{\text{NLA}}$	PLT	$+\mathcal{L}_{\text{NLA}}$	$\text{PLT} + \mathcal{L}_{\text{NLA}}$	NLA
full attention	2	4.1	3.9	3.1	2.5	0.0023	0.0032
	3‡	3.1	0.003	4.2	0.48	0.0019	0.0029
	4‡	1.8	0.004	3.3	0.49	0.0019	0.0029
KV-Share	2	5.3	220.8	5.7	8.5	0.0018	0.0022
	3	3.9	239.9	5.8	6.8	0.0023	0.0031
	4‡	3.9	0.004	5.7	0.66	0.0021	0.0031
G-SWA	2	5.1	181.7	5.5	11.1	0.0018	0.0028
	3	3.3	6.2	5.5	5.4	0.0027	0.0032
	4	3.3	25.9	5.5	5.5	0.0028	0.0032

Table 5 shows how far each construction’s loop input is from the same-position state it stands in for.

Table 5: Relative L2 distance of the loop input from the same-position state $H_i^{(\ell-1)}$ at boundary b , $K=4$, 256 held-out rows. Probes: linear maps from $[H_{i-1}^{(\ell-1)}; E_i]$ fitted on 1,024 rows of each construction. Single seed.

attention	b	PLT states		trained boundary MLP		fresh linear probe on		
		stale	sum	PLT+MLP	NLA	PLT	PLT+MLP	NLA
G-SWA	0	1.00	1.05	1.84	0.54	0.58	0.55	0.44
	1	0.81	0.90	2.01	0.58	0.61	0.58	0.51
	2	0.77	0.86	2.15	0.59	0.58	0.57	0.52
KV-Share	0	—	—	3.65	0.45	—	0.62	0.39
	1	—	—	3.73	0.46	—	0.57	0.44
	2	—	—	2.84	0.54	—	0.51	0.45

C Substituting the same-position state

Table 6 shows the CE increase derived from replacing the boundary output with $H_i^{(\ell-1)}$ or a norm-matched control.

Table 6: CE increase at $K=4$ (96 held-out rows) when the boundary output κ_i is replaced by the same-position reference state $H_i^{(\ell-1)}$, at one boundary or all three. Position: the all-boundaries reference-state increase over the first and last quarter of the sequence.

Attention	Construction	reference state at boundary				control, all boundaries			reference, all, by position	
		1	2	3	all	shuf. residual	Gaussian	shuf. reference	first quarter	last quarter
Full attention	PLT+MLP	0.541	1.178	2.156	5.288	2.554	0.397	5.459	4.717	5.642
	NLA	0.052	0.143	0.675	2.327	2.143	1.534	5.322	2.088	2.510
KV-Share	PLT+MLP	0.975	0.273	1.181	2.052	1.098	0.168	2.914	2.112	1.985
	NLA	0.004	0.019	0.118	0.290	0.149	0.122	1.167	0.277	0.305
G-SWA	PLT+MLP	0.165	0.332	1.253	2.700	0.900	0.189	3.800	2.792	2.614
	NLA	0.012	0.044	0.182	0.653	0.373	0.291	1.929	0.667	0.645

D Evaluation details

We use LM Evaluation Harness [Gao et al., 2023] version 0.4.12 with identical batch composition across models. All contexts are BOS-prepended to match pretraining.

Table 7: Mean zero-shot accuracy (%) over the 11 tasks. Bold: best parallel construction within each attention and loop-count cell. §: mean of four seeds. ‡: collapsed during training.

Attention	Construction	$K = 2$	$K = 3$	$K = 4$
full attention	PLT	43.14	43.46 [§]	43.74 [§]
	PLT+MLP	42.86	44.24	43.89
	NLA	44.01	43.93	44.17
	PLT+ $\mathcal{L}_{\text{NLA}}$	43.48	40.27 [‡]	40.94 [‡]
	Looped Transformer	43.72	44.28	44.95
KV-Share	PLT	43.06	43.20	43.45
	PLT+MLP	43.32	43.05	43.43
	NLA	43.16	43.57	43.42
	PLT+ $\mathcal{L}_{\text{NLA}}$	42.76	43.25	39.12 [‡]
	Looped Transformer	43.59	43.72	43.84
G-SWA	PLT	42.90	43.33	43.40
	PLT+MLP	43.49	43.19	43.24
	NLA	43.39	44.34	44.27
	PLT+ $\mathcal{L}_{\text{NLA}}$	42.68	43.43	43.41
	Looped Transformer	43.69	43.92	44.53

Table 8: KV-Share zero-shot accuracy (%) at $K = 4$. *: token-length-normalized loglikelihood. Bold: best parallel construction per task. The PLT+ $\mathcal{L}_{\text{NLA}}$ run collapsed during training (Table 2).

Task	PLT	+MLP	NLA	+ $\mathcal{L}_{\text{NLA}}$	Looped Transformer
ARC-Challenge	22.61	22.10	23.04	20.48	23.46
ARC-Easy	55.51	55.05	55.26	49.92	55.85
COPA	65.00	65.00	64.00	56.00	65.00
HellaSwag*	33.55	33.68	34.22	29.54	34.45
LAMBADA (OpenAI)	24.57	24.43	25.52	17.08	26.12
LAMBADA (Standard)	18.55	17.87	18.57	13.60	17.17
OpenBookQA*	33.20	33.00	32.00	27.00	34.20
PIQA*	64.85	64.31	62.95	60.45	64.74
RACE	28.80	29.19	28.71	27.66	30.33
SciQ	81.20	80.80	81.70	75.80	82.60
WinoGrande	50.12	52.33	51.62	52.80	48.30
<i>Mean</i>	43.45	43.43	43.42	39.12	43.84