ADAPTIVE LOOPS AND MEMORY IN TRANSFORMERS: THINK HARDER OR KNOW MORE?

Markus Frey[1, 2, 3], **Behzad Shomali**[1, 3], **Ali Hamza Bashir**[1, 2], **David Berghaus**[1, 2], **Joachim Koehler**[1, 2], **Mehdi Ali**[1, 2]
Lamarr Institute[1], Fraunhofer IAIS[2], University of Bonn[3]
markus.frey@iais.fraunhofer.de

ABSTRACT

Chain-of-thought (CoT) prompting enables reasoning in language models but requires explicit verbalization of intermediate steps. Looped transformers offer an alternative by iteratively refining representations within hidden states. This parameter efficiency comes at a cost, as looped models lack the storage capacity of deeper models which use unique weights per layer. In this work, we investigate transformer models that feature both adaptive per-layer looping, where each transformer block learns to iterate its hidden state via a learned halting mechanism, and gated memory banks, that provide additional learned storage. We find that looping primarily benefits mathematical reasoning, while memory banks help recover performance on commonsense tasks compared to parameter and FLOP matched models. Combining both mechanisms yields a model that outperforms an iso-FLOP baseline—with three times the number of layers—on math benchmarks. Analysis of model internals reveals layer specialization: early layers learn to loop minimally and access memory sparingly, while later layers do both more heavily.

1 INTRODUCTION

Large language models can reason explicitly via chain-of-thought (CoT) prompting (Wei et al., 2022), which uses step-by-step verbalization to produce reasoning traces, improving performance in downstream tasks. While this is effective, each reasoning step requires generating tokens (Nye et al., 2021), which has motivated interest in *implicit* reasoning, where models perform multi-step computation within their hidden representations without producing intermediate text (Saunshi et al., 2025; Bae et al., 2025).

One way of implementing implicit reasoning is by stacking transformer layers, iteratively applying the same block which refines the representations through repeated computation. This makes efficient use of parameters—a model that loops N times achieves a larger effective depth without using N times the parameters (Graves, 2016; Dehghani et al., 2018; Banino et al., 2021; Goyal et al., 2023). Recent work has shown that looped transformers can match much deeper non-looped models on reasoning tasks (Saunshi et al., 2025; Zhu et al., 2025; Raposo et al., 2024).

However, a looped model has fundamentally less capacity than a deeper model with N times the number of layers. While loops may improve reasoning, the model has fewer unique parameters in which to encode knowledge. Recent analysis suggests this trade-off is fundamental: looped models achieve their parameter efficiency not through increased knowledge storage but through *knowledge manipulation*—they are able to do multi-hop reasoning while showing similar per-parameter memorization capacity to standard transformers (Zhu et al., 2025).

Here, we investigate whether learned memory banks can restore the missing capacity. Specifically, we make the following contributions: (1) we propose an adapted looped Transformer that combines per-layer adaptive looping with gated access to local and global memory, and (2) we conduct a systematic study examining the effects of adaptive looping and the inclusion of memory banks on downstream model performance. We find that looping primarily benefits mathematical reasoning, while memory banks help recover commonsense performance compared to parameter- and FLOP-matched models. Analysis of model internals reveals layer specialization: early layers learn to loop

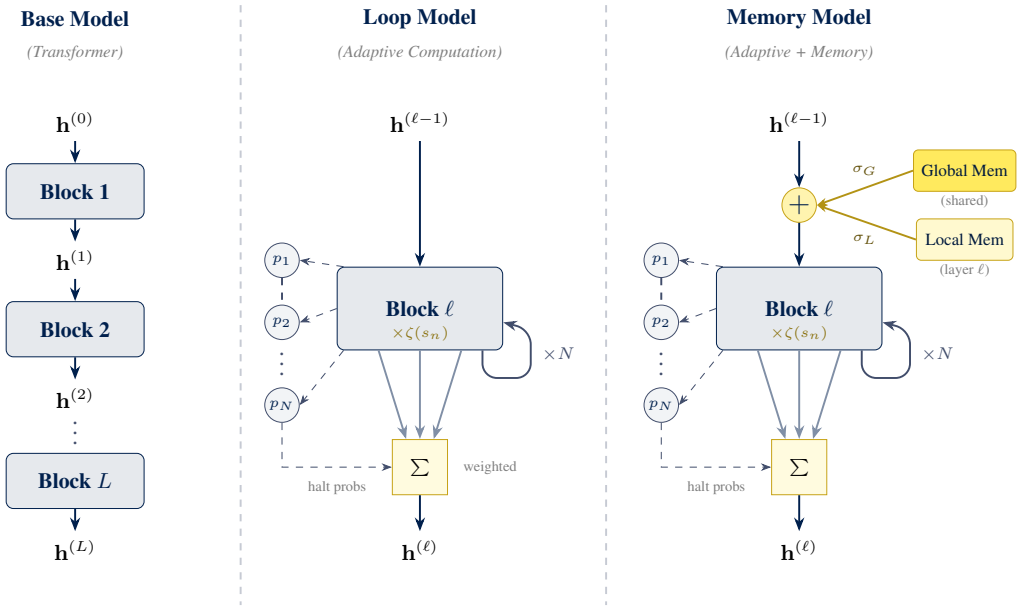

Figure 1: **Architecture overview.** *Left:* A standard transformer passes hidden states through L unique blocks. *Center:* Our loop model allows each block to iterate up to N times, with a learned halting mechanism that produces a weighted combination of intermediate states. Per-step scales $\zeta(s_n)$ are initialized near zero for training stability. *Right:* The combined model additionally retrieves from local (per-layer) and global (shared) memory banks, gated by learned input-dependent scalars.

minimally and access memory sparingly, while later layers do both more heavily. This specialization means the model learns to choose between thinking harder and knowing more and where to do each.

2 Methods

We augment a standard decoder-only transformer (Vaswani et al., 2017) with two mechanisms: adaptive looping for repeating computation and memory banks for retrieving learned knowledge. Figure 1 illustrates the architecture and Appendix A.1 provides additional details.

2.1 Adaptive Looping

A standard transformer block applies multi-head self-attention followed by a feed-forward network using residual connections and layer normalization:

$$\mathbf{h}' = \mathbf{h} + \text{Attn}(\text{LN}(\mathbf{h})) \tag{1}$$

$$\mathbf{h}'' = \mathbf{h}' + \text{FFN}(\text{LN}(\mathbf{h}')) \tag{2}$$

where $\mathbf{h} \in \mathbb{R}^{B\times T\times D}$ is the hidden state with batch size B, sequence length T, and embedding dimension D. We allow each transformer block to be applied multiple times with a learned halting mechanism, inspired by PonderNet (Banino et al., 2021). At each iteration $t \in \{1, \ldots, N_{\max}\}$, a halting router predicts the probability of stopping:

$$p_t = \sigma\left(\mathbf{W}_h \left[\mathbf{h}^{(t)}; t/N_{\max}\right] + b_h\right) \tag{3}$$

where $[\cdot;\cdot]$ denotes concatenation and $t/N_{\max}$ provides a normalized step embedding. The final output is computed as a weighted combination over all iterations:

$$\mathbf{h}_{\text{out}} = \sum_{t=1}^{N_{\max}} p_{\text{halt}}^{(t)} \cdot \mathbf{h}^{(t)} \tag{4}$$

where $p_{\text{halt}}^{(t)} = p_t \prod_{i=1}^{t-1}(1-p_i)$ is the probability of halting at exactly step t.

Learnable Loop Scales. To stabilize model training, we introduce per-step learnable scale parameters. Each iteration applies:

$$\mathbf{h}^{(t)} = \mathbf{h}^{(t-1)} + \text{softplus}(\alpha_t) \cdot f_\theta(\text{LN}(\mathbf{h}^{(t-1)})) \tag{5}$$

where f_θ denotes the transformer block and α_t is initialized to -7.0, which ensures the loop begins as an approximate identity mapping, and the model gradually learns when and how much to intervene.

2.2 Memory Banks

We introduce two types of learned memory, a local and a global one. For the *Local (Per-Layer) Memory* each layer ℓ maintains its own memory bank $(\mathbf{K}_\ell, \mathbf{V}_\ell) \in \mathbb{R}^{M_L \times D}$ with M_L slots. This enables layer-specific storage of intermediate computations or specialized knowledge appropriate to that depth. The *Global (Shared) Memory* uses a single memory bank $(\mathbf{K}_G, \mathbf{V}_G) \in \mathbb{R}^{M_G \times D}$ that is shared across all layers, allowing storage of information that might be beneficial for all layers.

Memory retrieval uses scaled dot-product attention with QK-normalization (Dehghani et al., 2023):

$$\mathbf{m}_{\text{local}} = \text{softmax}\left(\frac{\text{LN}_q(\mathbf{h}) \cdot \text{LN}_k(\mathbf{K}_\ell)^\top}{\sqrt{D}}\right)\mathbf{V}_\ell \tag{6}$$

$$\mathbf{m}_{\text{global}} = \text{softmax}\left(\frac{\text{LN}_q(\mathbf{h}) \cdot \text{LN}_k(\mathbf{K}_G)^\top}{\sqrt{D}}\right)\mathbf{V}_G \tag{7}$$

Unlike the KV-cache in standard attention, which stores activation history during inference, our memory banks are *static learnable parameters* that are optimized via backpropagation during training but fixed during inference. Our memory implementation draws inspiration from memory-augmented architectures (Lample et al., 2019; Sukhbaatar et al., 2019; Wu et al., 2022) and neural Turing machines (Graves et al., 2014).

Gated Memory Integration A critical design choice is how to integrate retrieved memory into the residual stream. Naive addition would force the model to always use memory, potentially harming performance on tasks where loops alone suffice. We therefore employ input-dependent gating:

$$\mathbf{g} = \sigma(\mathbf{W}_g\mathbf{h} + b_g) \tag{8}$$

$$\mathbf{h}_{\text{enriched}} = \mathbf{h} + \mathbf{g} \odot \mathbf{W}_m\mathbf{m} \tag{9}$$

where $\odot$ denotes element-wise multiplication. Separate gates control local and global memory contributions:

$$\mathbf{h}_{\text{memory}} = \mathbf{h} + \mathbf{g}_L \odot \mathbf{W}_L\mathbf{m}_{\text{local}} + \mathbf{g}_G \odot \mathbf{W}_G\mathbf{m}_{\text{global}} \tag{10}$$

We study the effect of gate bias initialization b_g, comparing $b_g \in \{-3, 0, 3\}$ corresponding to initial gate activations of approximately $\sigma(-3) \approx 0.05$ (nearly closed), $\sigma(0) = 0.5$ (balanced) and $\sigma(3) \approx 0.95$ (nearly open).

3 Results

3.1 Experimental Setup

Model. Our base architecture is a decoder-only transformer with $L = 12$ layers and a total of $\sim$200M parameters (see Appendix A.1 for full details). For looped models, we use the same 12-layer architecture and allow each layer to iterate up to $N_{\max} \in \{3, 5, 7\}$ times. For memory-augmented models, we add $M_L = 1024$ local memory slots per layer and $M_G = 512$ global memory slots, which in total adds approximately 10M parameters. We adapt our iso-param and iso-FLOP models to compensate for the additional parameters from the halting router and per-step scales. We pretrain all models on deduplicated FineWeb-Edu (Penedo et al., 2024) for 14B tokens and use a peak learning rate of 0.003.

Table 1: **Summary of results**, averaged across benchmarks within each group. CS = commonsense; BPB = bits per byte (lower is better, see Appendix A.2 for details). Best result per column within each model group is **bolded**. Full per-benchmark breakdowns are in Appendix A.4.

Model	Type	CS Acc ↑	CS BPB ↓	Math BPB ↓
Without Memory				
Base	IsoPar baseline	0.477	0.859	2.163
Loop-3	Ours ($N_{\max}$=3)	0.501	0.813	1.687
Loop-5	Ours ($N_{\max}$=5)	0.503	0.823	1.737
Loop-7	Ours ($N_{\max}$=7)	0.498	0.832	**1.659**
IsoFLOP	36-layer	**0.523**	**0.780**	1.801
With Memory (all use Loop-3)				
IsoPar-M	Wider FFN	0.459	0.823	2.108
Mem (g_0=−3)	Ours (closed init)	0.472	0.810	1.619
Mem (g_0=0)	Ours (balanced init)	0.481	0.810	1.662
Mem (g_0=3)	Ours (open init)	0.511	0.794	**1.616**
IsoFLOP-M	36-layer wider	**0.535**	**0.749**	1.761

Baselines. We compare against two types of baselines, first a **Iso-Parameter** model, where the FFN width is increased so that the total parameter count matches the target model. This controls for the possibility that any improvements come simply from having more parameters. And second a **Iso-FLOP (IsoFLOP)** model, which uses 3× the layers (36 layers), matching the forward-pass cost of a model with $N_{\max} = 3$ loops. Table 2 summarizes all configurations. We evaluate on common-sense and math tasks using the OLMES framework (Gu et al., 2025) (see Appendix A.2 for details).

3.2 Adaptive Loops and Memory

Looping Improves Mathematical Reasoning We first compare averaged benchmark results for looped models without memory (see Table 1, full per-benchmark results are given in Appendix A.4). Introducing adaptive looping with $N_{\max} = 3$ yields improvements in math BPB (1.687 vs. 2.163 for the base model, a 22% reduction) alongside moderate gains in commonsense accuracy (0.501 vs. 0.477) and commonsense BPB (0.813 vs. 0.859). Improvements in math are consistent across subcategories with the largest gains on Precalculus (-31%) and Intermediate Algebra (-26%).

When we further increase the number of loops, the performance increase is modest relative to the initial improvement from the base model (Loop-7 improves by 1.7% over Loop-3). Interestingly, commonsense performance shows a slight downward trend with more loops. These results suggest that additional iterations aid algorithmic computation (math) but do not help tasks that depend on stored knowledge (commonsense). Intriguingly the improvement on math benchmarks remains when we compare against the IsoFLOP model (1.687 vs. 1.801, a 6.4% advantage) despite only having one-third the number of layers. This suggest that looping is a more parameter-efficient way to improve on math benchmarks than simply adding layers, in line with Saunshi et al. (2025).

Local and Global Memory complements Loops We augment the Loop-3 model with local and global memory banks (see Section 2.2) and compare three gate initializations. All memory models share the same architecture and parameter count, only the initial gate bias differs.

All three memory variants outperform their iso-parameter baseline (IsoPar-M) on both tasks, confirming that the gains are not simply due to having more parameters. Compared to our Loop-3 model without memory we further improve on math benchmarks by 4.2% and on commonsense accuracy by 2%, indicating that the memory provides complementary value beyond what looping alone achieves. The comparison to the iso-FLOP baseline shows a similar pattern to above: IsoFLOP-M is better on commonsense but the memory augmented model is better on math benchmarks. Taken together, we observe that memory is able to close some of the commonsense gap that loops alone cannot bridge.

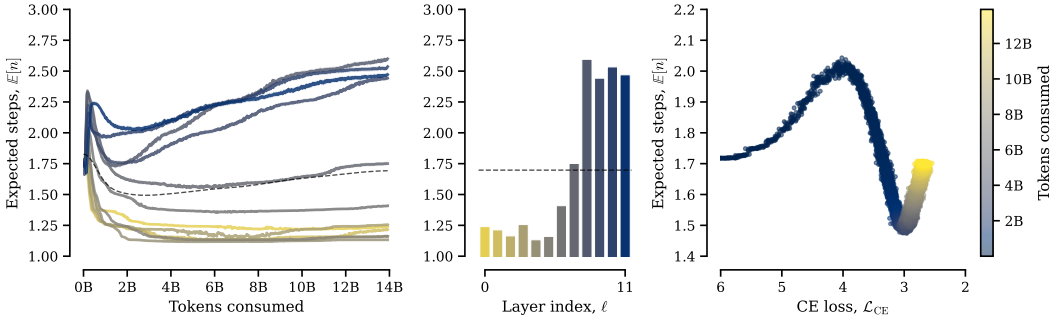

Figure 2: **Expected number of loop iterations per layer over training.** *Left:* Each curve represents one layer. Early layers (lighter colors) consistently use fewer iterations than later layers (darker colors). *Middle:* Expected steps at the end of training. *Right:* All models show a characteristic transition which occurs at approximately the same cross-entropy value across configurations (see Figure 3 in the Appendix for all configurations).

3.3 Training Dynamics of looped memory Models

We further investigated the training dynamics during the training of our models. Since we set $\lambda = 0$ (no ponder penalty), the patterns that emerge are driving entirely by the language modeling objective, i.e. next-token prediction. Figure 2 shows the expected number of iterations $\mathbb{E}[n_\ell]$ for each layer ℓ over the course of training (see Appendix A.1 for details on $\mathbb{E}[n_\ell]$).

We first observe that not all layers start to loop more over the course of training. We see later layers consistently use more iterations than earlier layers. This seems consistent with studies showing that early transformer layers encode local syntactic patterns while later layers handle more complex semantic and reasoning operations (Tenney et al., 2019; Rogers et al., 2020). This means the simpler computations performed by early layers do not benefit from iterations while the more complex operations in deeper layers do.

We also observe that the expected number of loops does not increase monotonically from the start of training (Right side of Figure 2). The onset of the increase in the number of loops occurs at approximately the same validation cross-entropy value across all loop configurations, around 3.27 ± 0.59 (see Figure 3 for comparison across models). This suggests the model only begins using additional iterations once it has acquired sufficient language competence to benefit from iterative refinement.

4 Discussion

Our preliminary results point to a functional dissociation between iterative computation and capacity in transformer models. Adaptive looping improves mathematical reasoning but does little for commonsense tasks where additional world knowledge needs to be encoded in the parameters. This aligns with previous work suggesting that transformer feed-forward layers act as key-value memories that store factual associations (Geva et al., 2021; Meng et al., 2022), while attention layers route and manipulate information. While looping seems to improve the routing of information, it cannot compensate for insufficient storage capacities. Put differently, the core tradeoff is between knowledge manipulation, which looping enhances as it repeatedly refines the representations, and knowledge capacity, which requires additional unique parameters.

Memory banks are one way of addressing this capacity bottleneck, and when combined with looping show promises in decreasing the gap on commonsense benchmarks. Notably, these dynamics emerge without any ponder penalty. The model is under no explicit pressure to minimize or maximize its loops, therefore the layer-wise specialization we see and the phase transition in the utilization of loops are all consequences of optimizing the language modeling loss alone.

There are several limitations and open questions which constrain some of the conclusions we can draw. First, our experiments are at a relatively small scale (~200M parameters, 12 layers, 14B tokens). Whether our conclusions hold at multi-billion parameter scale, where base models already

have substantial capacity, is an open question. Secondly, our math evaluation uses BPB instead of accuracy, which limits our ability to make strong claims about reasoning capabilities. Additionally, while we compare against iso-parameter and iso-FLOP baselines, we do not yet provide a full characterization of the efficiency tradeoff between adding loops or memory slots versus simply increasing depth or width under a continuous compute budget. These limitations will be addressed in follow-up work.

5 ACKNOWLEDGMENTS

We want to thank Max Lübbering, Timm Heine Ruland, David Fitzek and Richard Rutmann for helpful discussions and technical expertise regarding the Modalities framework (Lübbering et al., 2026). This work was funded by the Federal Ministry of Research, Technology & Space Germany (BMFTR) and the state of North Rhine-Westphalia as part of the Lamarr Institute for Machine Learning and Artificial Intelligence (LAMARR22B).

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. *arXiv preprint arXiv:2507.10524*, 2025.

Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 2021.

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

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *International conference on machine learning*, pp. 7480–7512. PMLR, 2023.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, 2021.

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. *arXiv preprint arXiv:2310.02226*, 2023.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. Olmes: A standard for language model evaluations. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 5005–5033, 2025.

Guillaume Lample, Alexandre Sablayrolles, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Large memory layers with product keys. *Advances in Neural Information Processing Systems*, 32, 2019.

Max Lübbering, Timm Ruland, Richard Rutmann, Felix Stollenwerk, David Fitzek, Michael Fromm, Alexander Weber, Rafet Sifa, Nicolas Flores-Herr, Joachim Köhler, et al. Modalities, a pytorch-native framework for large-scale llm training and research. *arXiv preprint arXiv:2602.08387*, 2026.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. 2021.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

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

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020.

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.

Sainbayar Sukhbaatar, Edouard Grave, Guillaume Lample, Herve Jegou, and Armand Joulin. Augmenting self-attention with persistent memory. *arXiv preprint arXiv:1907.01470*, 2019.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. *arXiv preprint arXiv:1905.05950*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Yuhuai Wu, Markus N Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. *arXiv preprint arXiv:2203.08913*, 2022.

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 Appendix

A.1 Implementation Details

Our base models utilize a standard 12-layer transformer architecture with an embedding dimension of $D = 768$, $H = 12$ attention heads, and an FFN hidden dimension of 3072. With a vocabulary size of 50,304, the total parameter count is approximately 200M. For adaptive looping models, we vary the maximum loop depth $N_{\max} \in \{3, 5, 7\}$ and initialize the loop scale parameter to $\alpha_t = -7.0$. Memory-augmented variants are configured with $M_L = 1024$ local slots and $M_G = 512$ global slots; we ablate gate bias initializations over $b_g \in \{-3.0, 0.0, 3.0\}$.

All models are trained on approximately 13.9B tokens ($\sim$38, 620 steps) using the AdamW optimizer with a batch size of $\sim$360K tokens. We employ a cosine learning rate schedule with a peak learning rate of 3.0×10^{-3}.

For the model loss we combine the next-token prediction loss with an optional ponder penalty:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda \cdot \tilde{n} \tag{11}$$

where $\mathcal{L}_{\text{CE}}$ is the categorical cross-entropy and $\tilde{n}$ is the normalized expected number of loop iterations, averaged across all layers:

$$\tilde{n} = \frac{\bar{n} - 1}{N_{\max} - 1}, \qquad \bar{n} = \frac{1}{L} \sum_{\ell=1}^{L} \mathbb{E}[n_\ell] \tag{12}$$

Here $\mathbb{E}[n_\ell]$ denotes the expected step count at layer ℓ and $N_{\max}$ is the maximum allowed iterations. This normalization maps the ponder cost to $[0, 1]$, making λ interpretable independently of $N_{\max}$.

We set $\lambda = 0$ for the majority of our experiments, meaning the model receives no explicit incentive to minimize loop iterations. Any loop utilization patterns that emerge are driven entirely by the language modeling loss.

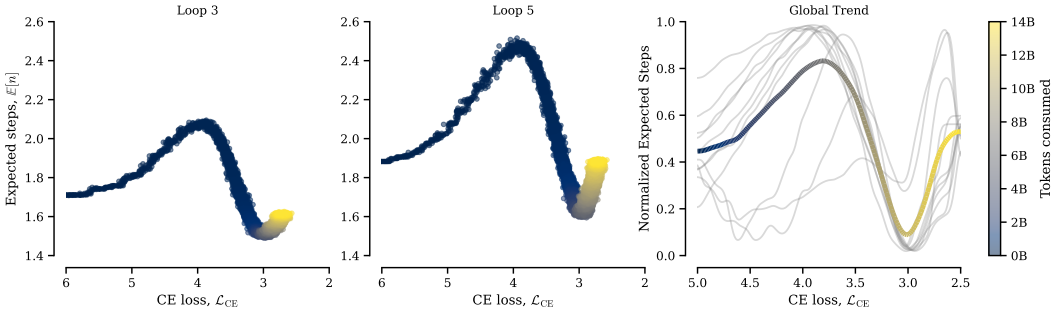

Figure 3: **Expected loop iterations vs. validation cross-entropy for all configurations.** Each point represents one evaluation during training; curves are colored by model configuration. Across all looped models, the expected number of iterations begins to increase rapidly once the cross-entropy drops below approximately 3.27 ± 0.59. This phase transition is consistent across Loop-3, Loop-5, and Loop-7 configurations, suggesting it depends on the model's language competence rather than the maximum number of allowed iterations.

A.2 Evaluation Protocol

We evaluate on two groups of downstream tasks using the OLMes framework (Gu et al., 2025): commonsense benchmarks (ARC-Challenge, ARC-Easy, HellaSwag, LAMBADA, PIQA, QASPER, SocialIQA, Winogrande) and math benchmarks (Algebra, Counting & Probability, Geometry, Intermediate Algebra, Number Theory, Prealgebra, Precalculus). For commonsense tasks, we report both accuracy and bits-per-byte (BPB). For math tasks, we report BPB only. BPB is computed as the negative log-likelihood of the gold answer divided by the number of UTF-8 bytes in the answer string. Formally, given a negative log-likelihood loss ℓ, Olmes computes BPB $= \ell / \ln(2) \cdot (L_T / L_B)$, where L_T is the length in tokens and L_B is the length in UTF-8 bytes (Gao et al., 2020). Lower BPB

Table 2: **Model configurations.** All models use the same base transformer architecture. Loop parameters include per-step scales and halting router weights. Memory parameters include local/global key-value banks and gating networks. Iso-parameter baselines add extra FFN capacity to match the corresponding model's parameter count. Iso-FLOP baselines use 36 layers to approximate the forward-pass cost of 3-loop models.

Configuration	Loops	Memory	Parameters	Description
IsoPar	×	×	200M	12 Layers
Loop-N	✓	×	200M	Adaptive looping, $N_{\max} \in \{3, 5, 7\}$
IsoFLOP	×	×	332M	36 Layers
IsoPar-M	×	×	210M	Wider FFN
Mem (g_0)	✓	✓	210M	Loop-3 + Memory
IsoFLOP-M	×	×	480M	36 Layers + Wider FFN

indicates better modeling of the target domain. We use BPB as it provides a continuous signal that reveals performance differences throughout pre-training, in contrast to GSM8k, which can remain at or near zero throughout training.

A.3 Additional Analysis

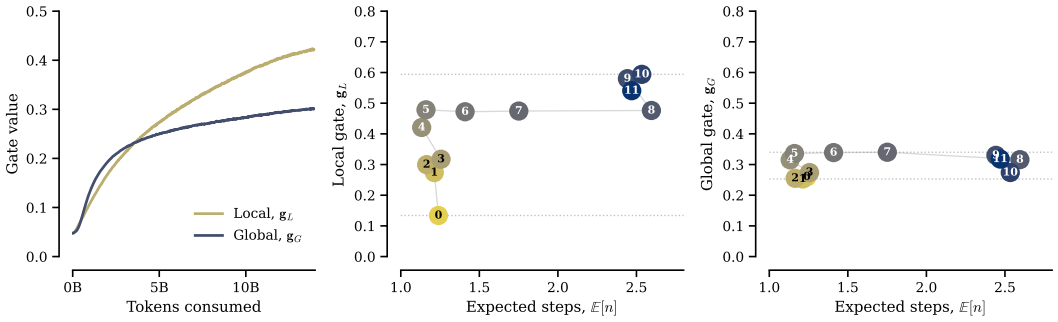

Figure 4: **Memory gate activations across layers and training.** *Left:* Local memory gate values show high variance across layers while later layers tend to have higher gate activations, and the spread increases over training. *Right:* Global memory gate values increase during training but converge to a more uniform profile across layers, with activations rising up to approximately layer 5 and then plateauing.

Layer-wise specialization of Memory Gates In Figure 4 we show the dynamics of the local and global memory gate during model training. We observe that local memory gates are used more strongly at the end of training and that the variance across layers is higher than for global gates (local: 0.42 ± 0.13, global: 0.30 ± 0.03). Since each layers local memory stores distinct key-value pairs, this variance likely reflects differences in the type and amount of information needed at each depth.

Global memory gates, by contrast, converge to a more uniform activation profile, indicating that the global memory acts as a shared knowledge base, which seems useful at all depths but not requiring layer-specific adaptation. Lastly, we observe that layers that loop more tend to have higher memory gate value, suggesting that the model does not treat loops and memory as substitutes but rather as complements, i.e. layers that need more computation also need more external information.

A.4 FULL RESULTS

Table 3: **Full results at the final checkpoint.** Best result within each group is bolded. Base = standard transformer; LN = Loop-N; L3_{IF} = iso-FLOP for Loop-3; M-B_{IP} = memory iso-parameter baseline; M_{g_0} = memory model with gate init g_0; M-B_{IF} = memory iso-FLOP baseline.

	Without Memory					With Memory				
Bench	Base	L3	L5	L7	L3_{IF}	M-B_{IP}	M_{-3}	M_{0}	M_{3}	M-B_{IF}
Commonsense Accuracy ↑										
ARC-C	0.367	0.375	**0.492**	0.398	0.430	0.398	0.414	0.359	0.398	**0.438**
ARC-E	0.609	0.672	0.609	0.586	**0.688**	0.602	0.648	0.672	0.641	**0.680**
HellaSwag	0.445	0.469	0.445	0.438	**0.508**	0.453	0.461	0.453	0.461	**0.508**
Lambada	0.211	0.266	**0.289**	0.281	0.211	0.227	0.250	**0.281**	0.242	0.273
PIQA	0.625	0.602	0.664	**0.680**	0.672	0.672	0.656	0.680	0.664	**0.688**
Qasper	0.625	**0.703**	0.641	0.688	**0.703**	0.422	0.531	0.484	0.680	**0.703**
SocialIQA	0.398	**0.430**	0.398	0.391	0.422	**0.445**	0.398	0.430	0.406	0.438
Winogrande	0.531	0.492	0.484	0.523	**0.555**	0.453	0.414	0.484	**0.594**	0.555
AVG	0.477	0.501	0.503	0.498	**0.523**	0.459	0.472	0.481	0.511	**0.535**
Commonsense BPB ↓										
ARC-C	0.913	0.840	0.854	0.860	**0.784**	0.833	0.846	0.851	0.813	**0.754**
ARC-E	0.846	0.758	0.772	0.762	**0.706**	0.789	0.740	0.733	0.721	**0.685**
HellaSwag	0.921	0.898	0.897	0.900	**0.866**	0.901	0.898	0.895	0.895	**0.849**
Lambada	1.002	0.935	0.952	0.979	**0.917**	0.964	0.939	0.924	0.917	**0.847**
PIQA	1.163	1.157	1.149	1.141	**1.097**	1.133	1.131	1.120	1.126	**1.064**
Qasper	0.305	**0.289**	0.314	0.350	0.310	0.320	0.305	0.336	**0.294**	0.295
AVG	0.859	0.813	0.823	0.832	**0.780**	0.823	0.810	0.810	0.794	**0.749**
Math BPB ↓										
Algebra	2.267	1.792	1.860	**1.766**	1.895	2.244	1.718	1.773	**1.717**	1.867
Count&Prob	1.960	1.565	1.634	**1.530**	1.641	1.946	1.491	1.524	**1.488**	1.626
Geometry	1.987	1.638	1.679	**1.618**	1.717	1.930	1.577	1.613	**1.566**	1.651
IntAlgebra	2.540	1.892	1.914	**1.839**	2.067	2.481	**1.799**	1.855	1.815	2.005
NumTheory	1.855	1.560	1.588	**1.538**	1.641	1.837	1.508	1.532	**1.498**	1.584
PreAlgebra	1.755	1.437	1.482	**1.423**	1.483	1.728	1.389	1.418	**1.372**	1.449
PreCalc	2.778	1.924	2.004	**1.901**	2.165	2.587	**1.847**	1.917	1.854	2.147
AVG	2.163	1.687	1.737	**1.659**	1.801	2.108	1.619	1.662	**1.616**	1.761

Table 4: **Results at the early checkpoint (step 5000).**

Bench	Base	L3	L5	L7	$L3_{IF}$	$M\text{-}B_{IP}$	M_{-3}	M_0	M_3	$M\text{-}B_{IF}$
Commonsense Accuracy ↑										
ARC-C	0.320	0.281	0.336	0.336	0.359	0.359	0.312	0.305	**0.367**	0.344
ARC-E	0.547	0.469	0.477	0.523	**0.602**	0.531	0.508	0.508	0.555	0.586
HellaSwag	**0.430**	0.391	0.398	0.422	0.414	0.398	0.383	0.391	0.359	0.414
Lambada	0.164	0.156	0.188	0.164	0.148	0.133	**0.219**	0.203	0.164	0.172
PIQA	0.555	0.609	0.633	0.617	0.539	0.594	**0.641**	0.609	0.609	0.578
Qasper	0.664	**0.695**	0.680	0.672	**0.695**	0.672	0.633	0.672	0.508	0.688
SocialIQA	**0.461**	0.430	0.398	0.414	0.398	0.422	0.406	0.422	0.414	0.445
Winogrande	0.484	0.539	0.453	0.492	**0.586**	0.531	0.531	0.531	0.477	0.531
AVG	0.453	0.446	0.445	0.455	0.468	0.455	0.454	0.455	0.432	**0.470**
Commonsense BPB ↓										
ARC-C	1.133	1.050	1.044	1.028	1.004	1.058	1.076	1.059	1.029	**0.980**
ARC-E	1.075	0.992	0.992	0.947	0.929	1.017	1.044	0.955	0.965	**0.891**
HellaSwag	1.022	1.017	1.004	1.005	0.986	1.014	1.007	0.996	1.000	**0.974**
Lambada	1.303	1.388	1.368	1.323	1.283	1.383	1.305	1.247	**1.232**	1.270
PIQA	1.315	1.269	1.278	1.274	1.240	1.282	1.237	1.256	1.252	**1.236**
Qasper	0.357	**0.309**	0.407	0.358	0.385	0.424	0.361	0.384	0.389	0.494
AVG	1.034	1.004	1.015	0.989	**0.971**	1.030	1.005	0.983	0.978	0.974
Math BPB ↓										
Algebra	2.461	2.179	2.398	2.342	2.284	2.411	**2.038**	2.067	2.093	2.295
Count&Prob	2.084	1.857	1.963	1.971	1.936	2.040	**1.751**	1.766	1.765	1.920
Geometry	2.186	1.970	2.068	2.090	2.032	2.135	**1.855**	1.890	1.926	2.022
IntAlgebra	2.655	2.308	2.530	2.560	2.456	2.584	**2.168**	2.174	2.231	2.457
NumTheory	2.028	1.862	1.980	1.953	1.903	2.013	1.782	**1.770**	1.782	1.890
PreAlgebra	1.937	1.747	1.864	1.838	1.804	1.913	**1.653**	1.668	1.671	1.783
PreCalc	2.789	2.314	2.658	2.632	2.575	2.703	**2.160**	2.189	2.258	2.641
AVG	2.306	2.034	2.209	2.198	2.142	2.257	**1.915**	1.932	1.961	2.144

Table 5: **Results at the mid checkpoint (step 20000).**

Bench	Base	L3	L5	L7	$L3_{IF}$	$M\text{-}B_{IP}$	M_{-3}	M_0	M_3	$M\text{-}B_{IF}$
Commonsense Accuracy ↑										
ARC-C	0.398	0.414	**0.453**	0.375	0.438	0.312	0.367	0.352	0.359	**0.453**
ARC-E	0.547	0.617	0.523	0.562	0.625	0.594	0.625	0.578	**0.648**	0.602
HellaSwag	0.406	0.375	0.406	0.430	**0.461**	0.391	0.422	0.438	0.430	0.453
Lambada	0.234	**0.328**	0.297	0.273	0.289	0.250	0.297	0.273	0.289	0.289
PIQA	0.594	0.625	0.609	0.656	0.664	0.633	0.641	0.641	0.656	**0.672**
Qasper	0.578	**0.688**	0.375	**0.688**	0.617	0.312	0.469	0.359	0.500	0.492
SocialIQA	0.391	0.414	**0.430**	0.398	0.406	0.391	0.406	0.422	0.406	0.375
Winogrande	0.523	0.484	0.469	0.508	0.531	0.539	0.516	0.484	0.508	**0.562**
AVG	0.459	0.493	0.445	0.486	**0.504**	0.428	0.468	0.443	0.475	0.487
Commonsense BPB ↓										
ARC-C	0.987	0.917	0.929	0.937	0.871	0.944	0.937	0.925	0.894	**0.848**
ARC-E	0.927	0.847	0.879	0.861	0.803	0.873	0.851	0.831	0.789	**0.768**
HellaSwag	0.959	0.950	0.946	0.938	0.919	0.948	0.940	0.935	0.939	**0.906**
Lambada	1.050	0.951	0.967	1.011	0.951	1.000	0.981	0.941	0.931	**0.912**
PIQA	1.217	1.216	1.195	1.204	1.152	1.203	1.186	1.187	1.173	**1.139**
Qasper	0.315	**0.300**	0.347	0.324	0.332	0.453	0.355	0.352	0.320	0.333
AVG	0.909	0.863	0.877	0.879	0.838	0.904	0.875	0.862	0.841	**0.818**
Math BPB ↓										
Algebra	2.296	1.950	2.075	1.892	2.151	2.154	1.900	**1.870**	1.919	2.007
Count&Prob	1.983	1.677	1.808	1.622	1.822	1.874	1.626	**1.612**	1.640	1.737
Geometry	2.014	1.775	1.857	1.712	1.880	1.914	**1.701**	1.712	1.719	1.778
IntAlgebra	2.471	2.014	2.213	1.971	2.350	2.288	1.961	**1.946**	2.003	2.128
NumTheory	1.906	1.701	1.755	1.653	1.789	1.821	1.638	**1.615**	1.661	1.683
PreAlgebra	1.820	1.567	1.643	**1.514**	1.662	1.727	1.521	1.520	1.526	1.579
PreCalc	2.597	2.045	2.343	1.990	2.492	2.375	**1.979**	1.984	2.043	2.252
AVG	2.155	1.819	1.956	1.765	2.021	2.022	1.761	**1.751**	1.787	1.880