# T$^2$MLR: TRANSFORMER WITH TEMPORAL MIDDLE-LAYER RECURRENCE

**Ziyang Cai**[*], **Xingyu Zhu**[*], **Yihe Dong**[†], **Yinghui He, Sanjeev Arora**
Princeton Language and Intelligence, Department of Computer Science
Princeton University
Princeton, NJ 08540, USA
{zc5794, xingyu.zhu}@princeton.edu

## ABSTRACT

We introduce Transformers with Temporal Middle-Layer Recurrence (T$^2$MLR), a generalized Transformer architecture that integrates attention and recurrence by routing a lightweight temporal pathway through the middle layers. Motivated by latent-reasoning and looped-Transformer lines of work, T$^2$MLR injects intermediate representations from deeper layers of the previous token into earlier layers of the current token via a gated recurrent pathway, enabling iterative latent computation while preserving dense, token-level supervision.

Across natural-language pretraining and multi-hop reasoning finetuning, T$^2$MLR consistently outperforms parameter-matched Transformer baselines at the same inference compute. Moreover, we find that looping only a middle-layer block (as little as 20% of all layers) often outperforms full-layer looping. This offers a new perspective on latent reasoning in Transformers: effective iterative refinement does not necessarily require full-stack recurrence. It can instead be achieved more effectively through targeted middle-layer recurrence. [1]

## 1 INTRODUCTION

Recent developments in large language models have demonstrated remarkable capabilities in reasoning, solving complex problems in mathematics, physics, and other scientific domains (Wei et al., 2022; Cobbe et al., 2021; OpenAI, 2023). Many of these tasks require multi-step reasoning in which intermediate abstract reasoning states must be iteratively refined before producing a final answer.

Despite these advances, the underlying Transformer architecture (Vaswani et al., 2017) remains fundamentally token-centric: auto-regressive generation repeatedly projects rich, high-dimensional latent representations back into a sparse one-hot vector in the token space at each decoding step, and this discrete representation is then used as the sole input for the next forward pass. This repeated projection creates an information bottleneck that limits how latent intermediate reasoning states can persist across time and influence the computation for future tokens.

To further improve the reasoning capability of Transformers, a growing line of work on *latent reasoning* seeks to relax this temporal constraint by enabling computation to persist in continuous latent space beyond explicit token-level intermediates. Such approaches either perform reasoning entirely in continuous space (Hao et al., 2024; Shen et al., 2025), or propagate uncertainty by forming linear combinations of token embeddings induced by post-softmax distributions (Zhang et al., 2025; Zhuang et al., 2025; Yue et al., 2025; Tang et al., 2026). However, these methods typically operate on representations at or beyond the final layer and feed recurrent signals into the next token through the input embedding. As a result, recurrent information is forced to live outside the middle layers, where abstract reasoning is known to occur more prominently (Tenney et al., 2019; Geva et al., 2021; Meng et al., 2022; Saunshi et al., 2024; Atanas & Liu, 2025).

[*] Equal contribution, listed in alphabetical order.
[†] Significant contribution.
[1] Code available at https://github.com/princeton-pli/T2MLR

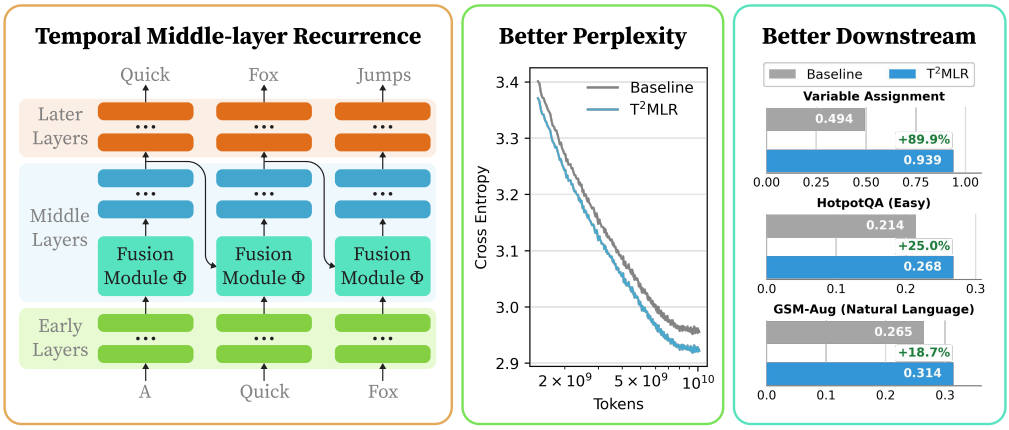


Figure 1: T$^2$MLR fuses representation from a deep layer from the previous token position into a shallow layer of the current token position (left). It gets better pretraining perplexity (middle, see Section 4.1) and reasoning downstream performance (right, see Section 4.2 and Section 4.3)

Concurrently, another line of work has attempted to scale reasoning capacity by extending computation along the depth dimension. These approaches loop over the transformer layers multiple times during the forward pass for a single token without increasing parameter count (Saunshi et al., 2025; Geiping et al., 2025; Zhu et al., 2025b). While successful at amplifying reasoning capabilities, such architectures incur increased inference cost which scales with the number of loops.

In this paper, we introduce Transformers with Temporal Middle-Layer Recurrence (T$^2$MLR), a novel Transformer-based architecture addressing these limitations through middle-layer temporal recurrence (see Figure 1, left). Rather than confining recurrence to the token or embedding space, or increasing inference-time depth via looping, T$^2$MLR allows abstract intermediate representations computed in the middle layers of the network to persist and evolve across decoding steps.

Concretely, T$^2$MLR injects representations from a deeper layer at the previous token directly into an earlier layer of the current token via a gated recurrent pathway. This design enables temporally extended latent reasoning while preserving standard auto-regressive decoding, dense token-level supervision during training, and the inference-time computational profile of a standard transformer.

Empirically, we show that on data and parameter matched settings, T$^2$MLR yields substantial gains across tasks that stress different aspects of reasoning. The main findings are summarized as follows:

- Shallow T$^2$MLR solves S5-RETRIEVAL, a challenging synthetic benchmark requiring both non-solvable group state tracking and in-context retrieval, where standard Transformers and recurrent models fail in isolation (Section 3.2).
- In pretraining settings, T$^2$MLR achieves lower perplexity against parameter-matched Transformers and demonstrates clear improvement on zero-shot NLP benchmarks (Section 4.1). The best improvement is seen when only looping over 20% of the middle layers.
- Finetuning on downstream reasoning datasets such as GSM-Aug (Deng et al., 2023), ProsQA (Hao et al., 2024), HotPotQA (Yang et al., 2018), and Variable Assignment (Saunshi et al., 2024), T$^2$MLR consistently outperforms parameter- and data-matched baselines (Sections 4.2 and 4.3), and notably, middle layer recurrence variants consistently outperform all-layer recurrence ones.

## 2 TRANSFORMER WITH TEMPORAL MIDDLE-LAYER RECURRENCE

In this section, we formally introduce T$^2$MLR. We will go over the design motivation, characterize the recurrence and representation fusion module, and describe how we are training the model.

### 2.1 MOTIVATION: EFFECT OF MIDDLE LAYERS AND THE INFORMATION BOTTLENECK

Recent mechanistic analyses reveal that intermediate layers serve as the primary locus of abstract reasoning in Transformers, whereas early layers focus on lexical and syntactic processing and late

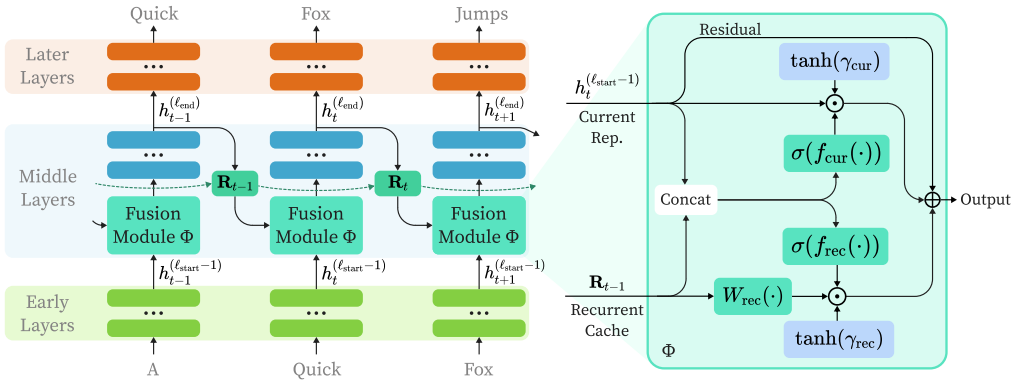


Figure 2: Illustration of the T$^2$MLR architecture (left) and the representation fusion module $\Phi$ (right). When doing forward pass for the $t$-th token, we use the representation after layer $\ell_{\text{end}}$ to update the recurrent cache $\mathbf{R}_t$. During the forward pass for the $t+1$-th token, $\mathbf{R}_t$ is then merged with the representation before layer $\ell_{\text{start}}$ via the representation fusion module.

layers specialize in projecting representations onto the output vocabulary (Tenney et al., 2019; Geva et al., 2021; Meng et al., 2022; Saunshi et al., 2024; Atanas & Liu, 2025). This suggests that the most valuable reasoning-related computation occurs mainly in the middle layers, and that representations produced after these layers encode rich information about the model's current reasoning state.

However, autoregressive inference in a standard decoder-only Transformer provides no mechanism for such intermediate representations computed for the previous token to be directly leveraged during the forward pass of the current token: shallow layers processing the current token cannot directly access deeper-layer representations computed at the previous step, even though these representations are already available in memory. To exploit this information, the model must either reconstruct it by traversing the same depth again and retrieving it indirectly via attention, or rely on the input embedding, a highly compressed representation that must pass through the unembed–decode–embed bottleneck before reaching deeper layers.

### 2.2 Temporal Middle-Layer Recurrence

To relax the bottleneck without changing the autoregressive interface, we introduce a lightweight recurrent pathway that exposes a cached intermediate representation from the previous step as an explicit input at the current step. We now make our architectural change precise in this section.

We start from a standard $L$-layer decoder-only Transformer with hidden dimension $d$, and work at the level of Transformer blocks with KV caches (Ott et al., 2019). We abstract out the key and value caches, and assume each token at each layer corresponds to a single cache vector in $\mathbb{R}^{d_{\text{KV}}}$.

Let $\boldsymbol{h}_t^{(0)} \in \mathbb{R}^d$ be the input embedding of the $t$-th token $x_t$. During autoregressive generation, each layer $\ell \in [L]$ maintains a KV cache of past tokens up to step $t-1$, denoted by $\mathbf{KV}_{1:t-1}^{(\ell)} \in \mathbb{R}^{(t-1)\times d_{\text{KV}}}$. We view the $\ell$-th Transformer block as a map that takes the current token representation together with the past cache and returns an updated representation and an updated cache:

$$\left(\boldsymbol{h}_t^{(\ell)}, \mathbf{KV}_{1:t}^{(\ell)}\right) = \mathcal{F}_\ell\left(\boldsymbol{h}_t^{(\ell-1)}, \mathbf{KV}_{1:t-1}^{(\ell)}\right), \qquad \ell = 1, 2, \ldots, L. \tag{2.1}$$

To break the depth-time barrier, we introduce *temporal middle-layer recurrence*, parameterized by two layer indices $1 \le \ell_{\text{start}} \le \ell_{\text{end}} \le L$ and a learnable representation fusion module $\Phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$. We additionally maintain a constant-size *recurrent cache* $\mathbf{R}_t \in \mathbb{R}^d$ for each decoding step $t$. $\mathbf{R}_t$ will store information for the representation after $\ell_{\text{end}}$ at the current step and be used by $\ell_{\text{start}}$ in the next.

Formally, for any step $t \ge 1$, we modify the $\ell_{\text{start}}$-th transformer block computation from (2.1) into:

$$\left(\boldsymbol{h}_t^{(\ell_{\text{start}})}, \mathbf{KV}_{1:t}^{(\ell_{\text{start}})}\right) = \mathcal{F}_{\ell_{\text{start}}}\left(\Phi\left(\boldsymbol{h}_t^{(\ell_{\text{start}}-1)}, \mathbf{R}_{t-1}\right), \mathbf{KV}_{1:t-1}^{(\ell_{\text{start}})}\right). \tag{2.2}$$

Before layer $\ell_{\text{start}}$, we fuse the recurrent cache $\mathbf{R}_{t-1}$ with the pre-$\ell_{\text{start}}$ representation $\boldsymbol{h}_t^{(\ell_{\text{start}}-1)}$. This creates a direct pathway, allowing representations computed via the middle layers in the previous step

to be utilized early on in the current step. The computation for the other layers remains the same as in a standard decoder-only transformer. After passing through layer $\ell_{\text{end}}$, we will compute the recurrent cache $\mathbf{R}_t$ based on $\boldsymbol{h}_t^{(\ell_{\text{end}})}$ and $\mathbf{R}_{t-1}$, preparing for the next recurrence.

### 2.3 Gated Fusion and Recurrent Cache Update

Now we are ready to describe how the recurrent cache is fused with the current stream of information from shallower layers. Recall that $\boldsymbol{h}_t^{(\ell_{\text{start}}-1)} \in \mathbb{R}^d$ is the representation of the current token before layer $\ell_{\text{start}}$ and $\mathbf{R}_{t-1} \in \mathbb{R}^d$ is the recurrent cache from the previous decoding step. The gated fusion module computes

$$\begin{aligned}\Phi\left(\boldsymbol{h}_t^{(\ell_{\text{start}}-1)}, \mathbf{R}_{t-1}\right) = \boldsymbol{h}_t^{(\ell_{\text{start}}-1)} &+ \tanh\left(\gamma_{\text{cur}}\right)\, \sigma\left(f_{\text{cur}}\left(\left[\boldsymbol{h}_t^{(\ell_{\text{start}}-1)}, \mathbf{R}_{t-1}\right]\right)\right) \odot \boldsymbol{h}_t^{(\ell_{\text{start}}-1)} \\ &+ \tanh\left(\gamma_{\text{rec}}\right)\, \sigma\left(f_{\text{rec}}\left(\left[\boldsymbol{h}_t^{(\ell_{\text{start}}-1)}, \mathbf{R}_{t-1}\right]\right)\right) \odot \boldsymbol{W}_{\text{rec}}\mathbf{R}_{t-1},\end{aligned} \tag{2.3}$$

where $f_{\text{cur}}, f_{\text{rec}} : \mathbb{R}^{2d} \to \mathbb{R}^d$ are learnable linear layers applied to the concatenation of the current representation and the recurrent cache, $\boldsymbol{W}_{\text{rec}} : \mathbb{R}^d \to \mathbb{R}^d$ is a learnable linear projection, and $\sigma(\cdot)$ denotes the element-wise sigmoid function. The scalar factors $\tanh\left(\gamma_{\text{cur}}\right)$ and $\tanh\left(\gamma_{\text{rec}}\right)$ act as learnable input-independent gates, while the element-wise $\sigma$ terms provide local, input-dependent modulation. We find this design particularly helpful for early training stability as we could initialize $\gamma$'s to zero and the gates randomly.

The fused representation $\Phi(\boldsymbol{h}_t^{(\ell_{\text{start}}-1)}, \mathbf{R}_{t-1})$ is then passed through the transformer blocks from layer $\ell_{\text{start}}$ to layer $\ell_{\text{end}}$. After the computation at layer $\ell_{\text{end}}$, the recurrent cache is updated as

$$\mathbf{R}_t = \text{RMSNorm}\left(\boldsymbol{h}_t^{(\ell_{\text{end}})} + \mathbf{R}_{t-1}\right). \tag{2.4}$$

The updated cache $\mathbf{R}_t$ is used in the next decoding step through the fusion operation in (2.2).

### 2.4 Approximated Training of T$^2$MLR with Temporal Parallelism

Due to the recurrent dependence of the deep cache $\mathbf{R}_t$ across decoding steps, T$^2$MLR cannot directly adopt the standard token-parallel training procedure used by vanilla Transformers. To retain scalability, we approximate $\mathbf{R}$ for all tokens in a sequence in parallel using a constant number of Jacobi fixed-point iterations. Due to space constraints, we defer the full training characterization to Algorithm 1 in Appendix Section A, and provide a simplified description below.

We first run a standard forward pass assuming no recurrent cache is available, and take the resulting representation after layer $\ell_{\text{end}}$ (shifted appropriately) as an initial cache $\mathbf{R}^{\langle 0\rangle}$. We then fuse this cache with the representation before layer $\ell_{\text{start}}$, run the network forward again through the middle layers up to layer $\ell_{\text{end}}$, and use the new layer-$\ell_{\text{end}}$ representations to get a refined cache $\mathbf{R}^{\langle 1\rangle}$. During training, we repeat this procedure for a fixed number of iterations $d_{\text{forward}}$, yielding a refined $\mathbf{R}^{\langle d_{\text{forward}}\rangle}$ that will be used in the final forward pass all the way up to the last layer.

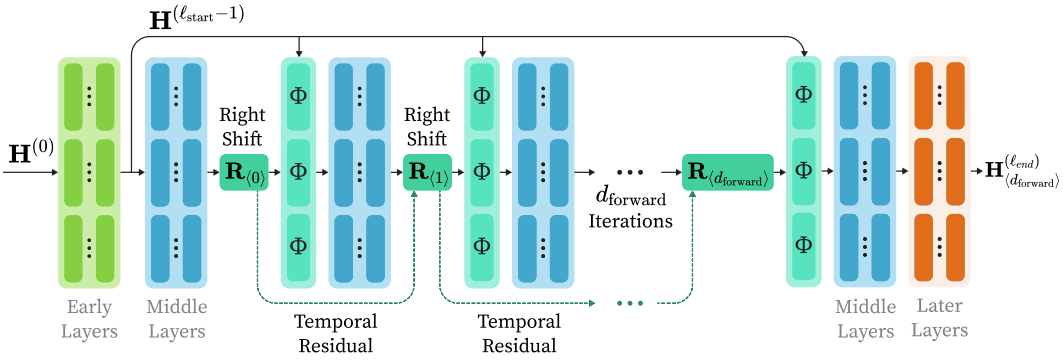


Figure 3: Illustration of the approximated training scheme. It approximated the ground truth residual cache $\mathbf{R}^*$ by a Jacobi iteration passing through the middle layers for $d_{\text{forward}}$ times.

Note that since the above calculation is fully differentiable, gradient computation can be back-propagated with recurrence depth of $d_{\text{forward}}$ as well. In order to attain better training efficiency, we allow a separate hyperparameter $d_{\text{backward}}$ controlling the backward depth analogous to truncated back propagation through time (TBPTT) in classical RNN training (Williams & Zipser, 2013).

Throughout the paper, we use $d_{\text{forward}} = 16$ and $d_{\text{backward}} = 4$ for experiments unless explicitly stated otherwise. We justify the empirical choices of such parameters as balance of approximation quality and training efficiency (see more analysis in Section A.3). We would also want to highlight that while the middle layer recurrence does increase the compute budget necessary for training, the inference cost remains nearly identical to that of the standard Transformer during auto-regressive generation. The only additional overhead comes from the constant compute per step added by the fusion module.

## 3 Best of Both Worlds: State Tracking and Retrieval

Before testing T$^2$MLR on general natural language modeling, we first isolate its core inductive bias on a synthetic benchmark. We use $S_5$-Retrieval, a task constructed to sharply separate T$^2$MLR with (i) standard Transformers, which excel at in-context retrieval but struggle with long-horizon state tracking at small depth, and (ii) recurrent models (e.g., RNNs/LSTMs), which naturally support state tracking but bottleneck retrieval through a fixed-width state.

### 3.1 The $S_5$-Retrieval Task

The $S_5$ state-tracking task, first described by Liu et al. (2023), is a sequence to sequence modeling task where the input is an ordered random sequence of $N$ elements $(a_1, a_2, \ldots, a_N)$ drawn from $S_5$ (the permutation group on 5 elements, with a cardinality of 120) and the output is the cumulative composition of the input sequence $(a_1, a_1 \circ a_2, \ldots, \Pi_{i=1}^{N} a_i)$. Here the $i$-th element is considered as the $i$-th state, which transits into the $i+1$-th state by applying $a_{i+1}$ to itself.

Leveraging circuit complexity arguments, Merrill et al. (2024) showed that Transformers require $\Omega(\log N)$ layers to exactly solve the $S_5$ state-tracking, and the known shallowest learnable solution in transformers is via parallel associative scan which requires $\lceil \log_2 N \rceil$ layers (Li et al., 2025). On the other hand, RNNs can solve the task with just constant number of layers (Merrill et al., 2024), since its circuit depth grows along the temporal dimension. However, classical recurrent architectures fail in key-value retrieval tasks as the associations must be compressed into a fixed-dimensional state.

To stress test a model's capability of composing both state tracking and in-context retrieval, we define the $S_5$-Retrieval Task as follows (see sample data in Table 6):

Let $\aleph$ be an alphabet, and let $\mathcal{D} : S_5 \to \aleph^K$ be a random mapping from the set $S_5$ to length-$K$ strings from the alphabet. The input is a serialization of the dictionary $\{(a, \mathcal{D}(a)) : a \in S_5\}$ followed by a delimiter and the action sequence $(a_1, a_2, \ldots, a_N)$ (appropriately padded to match token count). The target output is the step-wise state-tracking results interleaved with retrieval results based on the state:

$$\big(a_1, \quad \mathcal{D}(a_1), \quad a_1 \circ a_2, \quad \mathcal{D}(a_1 \circ a_2), \quad \ldots, \quad \Pi_{i=1}^{N} a_i, \quad \mathcal{D}\left(\Pi_{i=1}^{N} a_i\right)\big).$$

### 3.2 Experiments: Empirical Separation between T$^2$MLR vs Transformers / RNNs

We train the $S_5$-Retrieval task on (i) a 4-layer LSTM model, (ii) a 4-layer, 6-head Llama (Touvron et al., 2023) transformer model, and (iii) a T$^2$MLR model built on top of the small Llama model with $\ell_{\text{start}} = 0$ and $\ell_{\text{end}} = 4$, all with around 8 million parameters.

We use $K = 4$ and uniformly sample the number of states $N$ from $\{1, 2, \ldots, 32\}$ within the training set. We randomly resample the dictionary $\mathcal{D}$ for each sequence to ensure the necessity of doing purely in-context retrieval. We train the models with learning rate $1 \times 10^{-3}$, batchsize of 64 for 250,000 steps, and evaluate on a held-out set of input-output pairs with $N$ ranging from 1 to 48.

Figure 4 compares T$^2$MLR against recurrent and Transformer baselines on the S5-Retrieval task. T$^2$MLR substantially outperforms both baselines across input lengths, achieving high accuracies over a wide range of sequence lengths. In contrast, the RNN-LSTM baseline only succeeds at the shortest length and fails catastrophically as length increases, while the Transformer baseline fails to solve the task at all, remaining at 0 accuracy for every length tested.

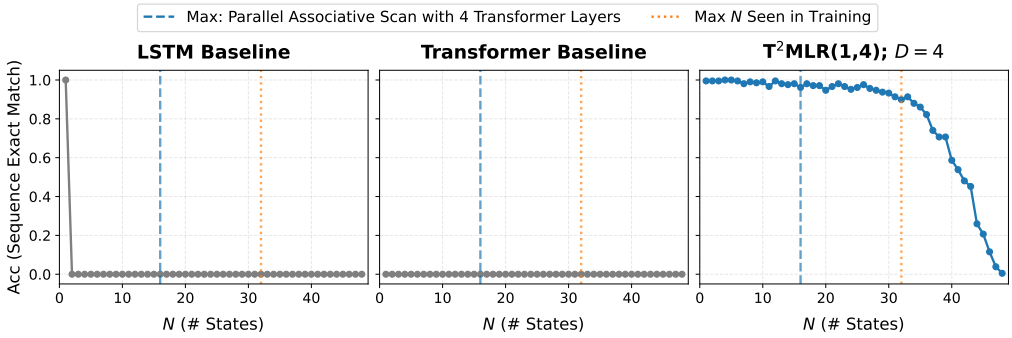


Figure 4: Performance comparison between LSTM, Transformer, and T$^2$MLR on the $S_5$-Retrieval task. $x$-axis denotes the test length, and $y$-axis is the accuracy. After extensive training, the LSTM and Transformer baselines do not learn the task at all, while T$^2$MLR shows notably nontrivial accuracy and partial length generalization.

Overall, these results demonstrate that T$^2$MLR combines trainability and expressivity on $S_5$-Retrieval, whereas standard recurrent and Transformer architectures fail despite extensive training.

## 4 Experiments

In this section, we provide empirical results of pretraining / finetuning a small T$^2$MLR model on natural language data as well as synthetic reasoning tasks. Across a diverse set of tasks we considered, T$^2$MLR achieves notable gains over the parameter-matched transformer baseline, and middle-layer recurrence generally yields larger gains compared to full-model recurrence. Section B contains further mechanistic experiments on T$^2$MLR, in particular future token planning.

### 4.1 Pretraining T$^2$MLR

We first test T$^2$MLR in standard autoregressive language modeling. We set the baseline as SmolLM2-135M (Bakouch et al., 2025), a lightweight LLaMA-like decoder-only Transformer architecture. We construct T$^2$MLR variants on top of the same backbone, parameterized by recurrence boundaries $(\ell_{\text{start}}, \ell_{\text{end}})$. We denote the recurrence depth by $D = \ell_{\text{end}} - \ell_{\text{start}} + 1$, and refer to each model as T$^2$MLR $(\ell_{\text{start}}, \ell_{\text{end}})$ (optionally annotated with $D$).

The representation fusion module introduces additional parameters. To ensure fair comparison, we adjust the baseline hidden size from 576 to 584 so that all models have essentially identical parameter counts (136.4M), with the baseline being slightly larger and thus conservative. We train all models for one epoch on the official 10B-token FineWeb-Edu subset (Penedo et al., 2024) using identical optimization hyperparameters (Appendix Section D.3), and evaluate zero-shot downstream performance using lm-eval-harness (Gao et al., 2024).

Table 1: Language Modeling Results for pretrained T$^2$MLR with different $(\ell_{\text{start}}, \ell_{\text{end}})$ configurations. All results are attained from LM-eval-harness (Gao et al., 2024) and we report normalized accuracy (acc_n) whenever available. We use the following abbreviations for the NLP benchmarks: ARC-C/E: ARC-Challenge/Easy, HS: HellaSwag, OBQA: OpenBookQA, WG: Winogrande.

| Model/Config<br>Metric | ARC-C<br>acc_n ↑ | ARC-E<br>acc_n ↑ | HS<br>acc_n ↑ | OBQA<br>acc_n ↑ | PIQA<br>acc_n ↑ | SciQ<br>acc_n ↑ | WG<br>acc ↑ | Average<br>- ↑ |
|---|---|---|---|---|---|---|---|---|
| Baseline | **24.74** | 44.28 | 29.81 | 30.20 | 61.53 | 60.80 | 48.46 | 42.83 |
| T$^2$MLR (1,30); $D$=30 | 24.49 | 43.48 | 29.98 | 30.00 | 60.72 | 62.60 | **52.25** | 43.36 |
| T$^2$MLR (5,26); $D$=22 | 23.98 | **46.13** | **30.75** | 29.80 | 60.77 | 62.80 | 51.85 | 43.73 |
| T$^2$MLR (9,22); $D$=14 | 24.15 | 45.24 | 30.40 | 29.20 | 61.15 | **66.40** | 51.54 | 44.01 |
| T$^2$MLR (13,18); $D$=6 | 24.23 | 45.50 | 29.95 | **31.20** | **61.70** | 64.20 | 52.17 | **44.14** |
| T$^2$MLR (15,16); $D$=2 | 24.40 | 45.83 | 30.11 | 29.20 | 59.63 | 60.20 | 50.83 | 42.89 |

As shown in Table 1, T$^2$MLR consistently matches or improves upon the parameter-matched Transformer baseline on most downstream benchmarks. Across recurrence configurations, T$^2$MLR (13,18) ($D$=6) achieves the highest average score while only looping over 20% of the layers. However, full recurrence ($D$=30), as used by all the previous latent reasoning works, generally performs weaker than middle layer recurrence.

In Figure 5, we provide the training loss for different variants of T$^2$MLR. Most configurations attain lower evaluation loss compared to the baseline except for $\ell_{\text{start}} = 1$ (full recurrence with $D = 30$) and $\ell_{\text{start}} = 14$ (only recurring on $D = 2$ layers). Except for these two exceptions, we note that the evaluation loss monotonically decreases with increasing number of recurrent layers, yielding distinct trend as in downstream. This suggest *distinct implicit bias of middle layers* on reasoning that might not be captured by perplexity metrics and resembles the observation in Saunshi et al. (2024) for middle layer stacking.

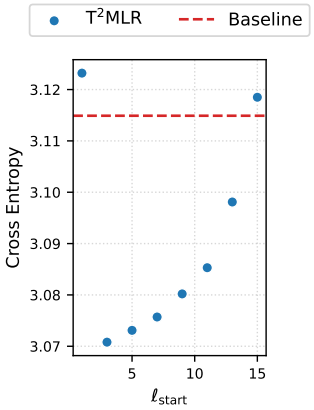


Figure 5: Loss vs $\ell_{\text{start}}$ for pretraining runs at 16k steps.

### 4.2 Multi-Hop Reasoning

While perplexity reflects average next-token prediction, it can understate changes in the model's internal computation. To directly test whether temporal middle-layer recurrence improves latent reasoning, we finetune the pretrained checkpoints from Section 4.1 on a suite of multi-hop reasoning tasks:

**Variable assignment** We take the variable assignment task from the set of "reasoning-primitives" task proposed by Saunshi et al. (2024). In this task, the input is a series of variable assignments, and the model needs to output the value of a query variable. Since the baseline models already saturates the original dataset, we increase the difficulty of the task from depth$=2$ to depth$=5$.

**ProsQA** We evaluate on the ProsQA task from Hao et al. (2024). The input is a directed-acyclic-graph, and a query asks a binary question for the connectivity between a starting and two choices of ending nodes. The output steps through the true path that connects the starting and end nodes. Since we observe the original ProsQA data is saturated by the baseline model, we increase the average number of nodes to 60 and average length of the path to 8.

**HotPotQA-Simple** HotpotQA is a question answering dataset featuring natural, multi-hop questions, with annotated supporting facts (Yang et al., 2018). We use a teacher model (GPT-4o-mini (OpenAI et al., 2024)) to generate natural language chain-of-thought SFT data. Since the medium and hard level subset are too challenging at the model scale we are testing, we present the results for the easy subset.

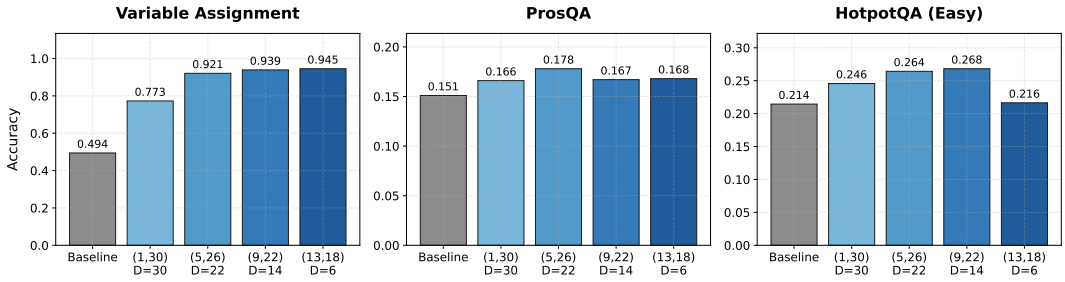


Figure 6: Fine-tuning performance comparison between T$^2$MLR and baseline models on multi-hop reasoning tasks. We report pass@1 accuracy for all experiments. In all three tasks, T$^2$MLR outperforms parameter and data-matched baseline. Notably, the middle-layer looping variants ($D = 22, 14, 6$) achieves higher performance than full looping ($D = 30$).

### 4.3 SIMPLE MATH REASONING

The multi-hop tasks above emphasize compositional structure (variable updates, path tracing, and evidence aggregation). We next test whether the same inductive bias improves arithmetic-heavy reasoning by finetuning on GSM-8K-style grade-school math problems.

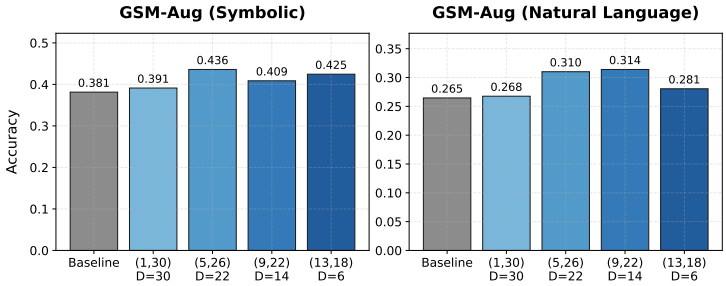


Figure 7: Fine-tuning performance comparison between T$^2$MLR and baseline models on GSM-8k. We report pass@1 accuracy for all experiments. T$^2$MLR outperforms parameter and data-matched baseline. The best performance is achieved at $D = 22$ and $D = 14$.

On both the symbolic and the natural language variants of the datasets, T$^2$MLR with middle-layer recurrence shows superior performance compared to the parameter and data-matched baseline. Meanwhile the full layer recurrence model show nearly negligible improvement. This further highlights the potential of middle layer recurrence to boost latent reasoning capabilities.

## 5 RELATED WORKS

**Latent Reasoning** Many recent works study or exploit *latent computation* in transformers to improve reasoning without relying on long explicit chain-of-thought (CoT).

On the modeling side, recent work has explored increasing reasoning capacity without relying on long explicit chain-of-thought traces, by shifting computation into latent or continuous representations. One line of work focuses on *internalizing* chain-of-thought into hidden states, via distillation or progressively removing explicit intermediate steps while preserving final-answer performance (Deng et al., 2023; 2024; Hao et al., 2024; Shen et al., 2025). A second line explicitly allocates additional *inference-time computation* to latent processes, such as iterative latent refinement, token-wise branch-and-merge reasoning, or latent communication in multi-agent settings (Kong et al., 2025; Fu et al., 2025; Tang et al., 2026; Zou et al., 2025). Finally, several approaches replace discrete reasoning tokens with *soft or continuous* alternatives, including mixed latent/text token training and continuous

| Model | Recurrent Information | Injection Location | Fusion Module | BPTT |
|---|---|---|---|---|
| Transformer | Discrete token | Input embedding | – | N/A |
| Coconut | Last-layer hidden state | Post-embedding | Identity | ✓ |
| CODI | Last-layer hidden state | Post-embedding | 2-layer MLP | ✓ |
| Multiplex-Thinking | Soft token mixture | Post-embedding | Identity | × |
| HRPO | Soft token mixture<br>+ discrete token | Post-embedding (soft token)<br>+ Input embedding (discrete token) | Gated fusion | × |
| T$^2$MLR (ours) | Middle-layer hidden state<br>+ Discrete token | Earlier middle layer (hidden state)<br>+ Input embedding (discrete token) | Gated fusion | ✓ |

Table 2: Comparison with related architectures. Standard transformers communicate across time only through discrete tokens. Prior recurrent or latent-reasoning variants additionally propagate either last-layer hidden states or soft token mixtures, usually through input-level or post-embedding pathways. T$^2$MLR instead adds a recurrent middle-layer hidden-state pathway, fused into an earlier middle layer of the next token, while preserving the ordinary discrete token pathway and supporting end-to-end supervised training with a constant-depth BPTT.

mixtures in embedding space, which can shorten or eliminate verbose reasoning traces (Su et al., 2025; Tack et al., 2025; Zhang et al., 2025; Butt et al., 2025; Zhuang et al., 2025). Additionally, Zhu et al. (2025a) finds latent COT offers theoretical benefits on graph tasks.

Our work is most similar to Yue et al. (2025), which uses latent information by projecting the last-token latent state back to token space before the first layer. They use the resulting architecture for RL training, showing improvements on mathematical reasoning benchmarks. Crucially, in Yue et al. (2025) gradients do not flow to the past tokens, while in our work, we use supervised training and back-propagate through time. Furthermore, we emphasize *middle-layer* recurrence, avoiding constraints that force recurrent information to live in the embedding space.

**Looped Transformers** Looped (recurrent-depth) Transformers trade new parameters for repeated computation by applying the same Transformer block multiple times. Universal Transformers introduce this idea explicitly as recurrence over depth, optionally with adaptive halting Dehghani et al. (2019). ALBERT popularizes cross-layer parameter sharing in Transformers to reduce parameter count Lan et al. (2020). On the expressivity side, Giannou et al. show that looping expands the expressive power of Transformers Giannou et al. (2023). Saunshi et al. (2025) provide theoretical evidence that looping can approximate the benefits of much deeper Transformers, yielding large effective depth with fewer parameters.

Geiping et al. (2025) demonstrate empirically that increasing test-time compute via recurrent depth improves reasoning. Most recently, ByteDance's Ouro (Zhu et al., 2025b) looped LMs scale this design to billion-parameter models, reporting performance that rivals much larger-parameter Transformers. It is worth noting that both Geiping et al. (2025) and Zhu et al. (2025b) introduced leveraging the KV caches of the last loops when computing the forward pass for the shallower loops. This represents another form of creating a shortcut for utilizing more refined deeper representations at shallower layers.

**Recurrent memory for long-context modeling.** A line of work uses recurrence primarily as a *memory mechanism* to improve long-context performance, rather than to change per-token computation. Transformer-XL reuses hidden states across segments to extend effective context (Dai et al., 2019), while Compressive Transformer retains older history via a compressed memory (Rae et al., 2019). Recurrent Memory Transformer passes information across segments using dedicated memory tokens (Bulatov et al., 2022). More recently, Titans (Behrouz et al., 2024) augments standard attention over a fixed-length current context (motivated by attention's quadratic cost) with a neural long-term memory module that stores and retrieves information from much longer histories.

**State space and hybrid models.** State space models offer subquadratic alternatives to attention by maintaining and updating a compact state over the sequence, with representative examples including S4 (Gu et al., 2021), Hyena (Poli et al., 2023), and Mamba (Gu & Dao, 2023). Some hybrid architectures interleave attention with recurrent/SSM-style blocks to trade off content-based retrieval and efficiency, e.g., RetNet (Sun et al., 2023), Griffin (De et al., 2024), and Jamba (Lieber et al., 2024). In contrast, we do not introduce a new recurrent/SSM layer or replace attention; instead, we add a token-to-token recurrent connection that injects a previous token's later-layer residual stream into an earlier-layer residual stream of the current token.

## 6 Limitations and Future Works

### 6.1 Limitations and future work

**Training-time overhead.** Our batch approximate forward training loops through the recurrent middle layers multiple times per forward pass, increasing wall-clock training time. In Appendix A.3, we analyze gradient sensitivity to the batch-forward and batch-backward depths and find that a *moderate* approximation depth is sufficient in our pretraining setting. In practice, this corresponds to roughly a 2–4$\times$ training-time increase over parameter-matched Transformer baselines.

We believe this compute trade-off can be further improved. A natural direction is to design temporal mixing mechanisms that reduce the number of approximation steps while preserving long-range

state. For example, instead of mixing only the previous token, one can extend the recurrent input to a window of the past $k$ tokens, or maintain a small bank of recurrent states with lightweight updates.

**Scale and architecture design space.** As an initial exploration, our experiments focus on relatively small model sizes. While $T^2$MLR shows consistent gains at this scale, it remains to be tested how these improvements translate to larger models and more complex benchmarks (e.g., strong math reasoning suites). A related direction is *retrofitting*: converting an existing pretrained Transformer into an $T^2$MLR model by inserting (and then finetuning) the recurrent mixing module, potentially reducing the cost of training from scratch.

Finally, the mixing module itself admits substantial design freedom. Our current instantiation is a simple mechanism for combining the recurrent and current-token streams; understanding its behavior more systematically may enable better architectural choices (e.g., alternative parameterizations, stability constraints, or more expressive fusion operators).

### 6.2 Conclusion

We presented $T^2$MLR, a generalized Transformer that integrates attention and recurrence via temporal recurrence at intermediate layers. By injecting previous-token intermediate representations into earlier layers of the current token, $T^2$MLR enables richer latent computation while retaining dense supervision and standard autoregressive inference cost. To make training scalable, we introduce a batch approximate forward procedure that computes recurrent states for all tokens in parallel.

Across a diverse set of tasks, $T^2$MLR consistently outperforms parameter-matched baselines, and it notably succeeds on S5-Retrieval where standard Transformers and LSTMs fail in isolation. Overall, $T^2$MLR provides a unified framework that interpolates between recurrent and attention-based computation, suggesting a path toward architectures that support stronger latent reasoning without relying on long explicit chains of thought.

## Contribution

ZC$^*$ designed the fusion gate, completed the majority of the the experiments including data generation and training, and contributed to writing. XZ$^*$ proposed the idea of $T^2$MLR, wrote the core training and inference script, and contributed to writing. YD$^\dagger$ contributed to general discussion on the architecture design, completed the future token prediction task and part of pre-training experiments, and contributed to writing. YH worked on the HotpotQA experiment.

## Acknowledgment

We acknowledge the support from NSF, Schmidt Foundation, DARPA AIQ Program, OpenAI and Google Inc. ZC and XZ are additionally supported by the Gordon Y.S. Wu Fellowship in Engineering.

We thank Abhishek Panigrahi for the initial discussion with XZ which motivated this idea. We would also want to thank Yun Cheng, Haoyu Zhao, Liam Fowl, Narutatsu Ri, and Zixuan Wang for helpful discussions during various stages of the project.

## REFERENCES


Adam Atanas and Kai Liu. A modular dataset to demonstrate llm abstraction capability, 2025. URL https://arxiv.org/abs/2503.17645.

Elie Bakouch, Carlos Miguel Patiño, Anton Lozhkov, Edward Beeching, Aymeric Roucher, Nouamane Tazi, Aksel Joonas Reedi, Guilherme Penedo, Hynek Kydlicek, Clémentine Fourrier, Nathan Habib, Kashif Rasul, Quentin Gallouédec, Hugo Larcher, Mathieu Morlon, Joshua Xenova, Vaibhav Srivastav, Xuan-Son Nguyen, Colin Raffel, Lewis Tunstall, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Smollm3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, July 2025. URL https://huggingface.co/blog/smollm3. Hugging Face blog post.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time, 2024. URL https://arxiv.org/abs/2501.00663.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Aydar Bulatov, Yuri Kuratov, and Mikhail S. Burtsev. Recurrent memory transformer, 2022. URL https://arxiv.org/abs/2207.06881.

Natasha Butt, Ariel Kwiatkowski, Ismail Labiad, Julia Kempe, and Yann Ollivier. Soft tokens, hard truths, 2025. URL https://arxiv.org/abs/2509.19170.

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.

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. *arXiv preprint arXiv:2110.14168*, 2021.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019. URL https://arxiv.org/abs/1901.02860.

Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, Arnaud Doucet, David Budden, Yee Whye Teh, Razvan Pascanu, Nando De Freitas, and Caglar Gulcehre. Griffin: Mixing gated linear recurrences with local attention for efficient language models, 2024. URL https://arxiv.org/abs/2402.19427.

Mostafa Dehghani, Stephan Gouws, et al. Universal transformers. In *International Conference on Learning Representations*, 2019.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation, 2023. URL https://arxiv.org/abs/2311.01460.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step, 2024. URL https://arxiv.org/abs/2405.14838.

Tianyu Fu, Yichen You, Zekai Chen, Guohao Dai, Huazhong Yang, and Yu Wang. Think-at-hard: Selective latent iterations to improve reasoning language models. *arXiv preprint arXiv:2511.08577*, 2025.

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. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

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*, 2025. URL https://openreview.net/forum?id=S3GhJooWIC.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *EMNLP*, 2021.

Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11398–11442. PMLR, 2023. URL https://arxiv.org/abs/2301.13196. arXiv:2301.13196.

Riccardo Grazzi, Julien Siems, Arber Zela, Jörg K. H. Franke, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear rnns through negative eigenvalues. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=UvTo3tVBk2.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2023. URL https://arxiv.org/abs/2312.00752.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces, 2021. URL https://arxiv.org/abs/2111.00396.

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, 2024. URL https://arxiv.org/abs/2412.06769.

Samy Jelassi, David Brandfonbrener, Sham Girotti, Yuanzhi Li, and Alessandro Lazaric. Repeat after me: Transformers are better than state space models at copying. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2402.01032.

Deqian Kong, Minglu Zhao, Dehong Xu, Bo Pang, Shu Wang, Edouardo Honig, Zhangzhang Si, Chuan Li, Jianwen Xie, Sirui Xie, and Ying Nian Wu. Latent thought models with variational bayes inference-time computation, 2025. URL https://arxiv.org/abs/2502.01567.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://arxiv.org/abs/1909.11942. arXiv:1909.11942.

Belinda Z. Li, Zifan Carl Guo, and Jacob Andreas. (how) do language models track state? In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=8SXosAVIFH.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. Jamba: A hybrid transformer-mamba language model, 2024. URL https://arxiv.org/abs/2403.19887.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=De4FYqjFueZ.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *NeurIPS*, 2022.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=QZgo9JZpLq.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Preetum Nichani, Alex Vitushinsky, Song Mei, Hila Gonen, and Yuanzhi Li. Understanding factual recall in transformers via associative memories. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2406.06484. Spotlight paper.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles

Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. Gpt-4o system card, 2024. URL https://arxiv.org/abs/2410.21276.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In Waleed Ammar, Annie Louis, and Nasrin Mostafazadeh (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pp. 48–53, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-4009. URL https://aclanthology.org/N19-4009/.

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, 2024. URL https://arxiv.org/abs/2406.17557.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y. Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models, 2023. URL https://arxiv.org/abs/2302.10866.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling, 2019. URL https://arxiv.org/abs/1911.05507.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Nikunj Saunshi, Stefani Karp, Shankar Krishnan, Sobhan Miryoosefi, Sashank Jakkam 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 *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2502.17416. arXiv:2502.17416.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation, 2025. URL https://arxiv.org/abs/2502.21074.

DiJia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qinqing Zheng. Token assorted: Mixing latent and text tokens for improved language model reasoning, 2025. URL https://arxiv.org/abs/2502.03275.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models, 2023. URL https://arxiv.org/abs/2307.08621.

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, 2025. URL https://arxiv.org/abs/2502.08524.

Yao Tang, Li Dong, Yaru Hao, Qingxiu Dong, Furu Wei, and Jiatao Gu. Multiplex thinking: Reasoning via token-wise branch-and-merge, 2026. URL https://arxiv.org/abs/2601.08808.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. In *ACL*, 2019.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. Attention is all you need. In *Advances in Neural Information Processing Systems*, 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. *arXiv preprint arXiv:2201.11903*, 2022.

Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*, 2017.

Ronald J Williams and David Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. In *Backpropagation*, pp. 433–486. Psychology Press, 2013.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Zhenrui Yue, Bowen Jin, Huimin Zeng, Honglei Zhuang, Zhen Qin, Jinsung Yoon, Lanyu Shang, Jiawei Han, and Dong Wang. Hybrid latent reasoning via reinforcement learning, 2025. URL https://arxiv.org/abs/2505.18454.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, and Xin Eric Wang. Soft thinking: Unlocking the reasoning potential of LLMs in continuous concept space. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=ByQdHPGKgU.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. *arXiv preprint arXiv:2505.12514*, 2025a. URL https://arxiv.org/abs/2505.12514.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Shanda Li, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, Qiyang Min, Hongzhi Huang, Xun Zhou, Wei Ye, Jiaheng Liu, Jian Yang, Yunfeng Shi, Chenghua Lin, Enduo Zhao, Tianle Cai, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jason Eshraghian. Scaling latent reasoning via looped language models. *arXiv preprint*, 2025b. URL https://arxiv.org/abs/2510.25741. arXiv:2510.25741.

Yufan Zhuang, Liyuan Liu, Chandan Singh, Jingbo Shang, and Jianfeng Gao. Mixture of inputs: Text generation beyond discrete token sampling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=l6C6Pw30Gl.

Jiaru Zou, Xiyuan Yang, Ruizhong Qiu, Gaotang Li, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. Latent collaboration in multi-agent systems, 2025. URL https://arxiv.org/abs/2511.20639.

# A Additional Information on Training T$^2$MLR

## A.1 Temporal-Parallel Approximated Training

Here we formally describe the fixed-point iteration algorithm used to train T$^2$MLR, which has been informally introduced in Section 2.4.

Let $\boldsymbol{H}^{(0)} \in \mathbb{R}^{T\times d}$ denote the input embedding for a sequence of $T$ tokens, and with slight abuse of notation let $\mathcal{F}_{\ell_1:\ell_2} : \mathbb{R}^{T\times d} \to \mathbb{R}^{T\times d}$ denote the mapping on embedding sequences implemented by layers between $\ell_1$ and $\ell_2$. After doing a standard forward pass (assuming no recurrent cache exists), we store $\boldsymbol{H}^{(\ell_{\text{start}}-1)} := \mathcal{F}_{1:\ell_{\text{start}}-1}(\boldsymbol{H}^{(0)})$ as the exact input into layer $\ell_{\text{start}}$. This also corresponds to the exact input for the sequence if recurrence has been done sequentially.

During the same forward pass, we set the right-shifted version of $\boldsymbol{H}^{(\ell_{\text{end}})}_{\langle 0\rangle} := \mathcal{F}_{1:\ell_{\text{end}}}(\boldsymbol{H}^{(0)})$ as the first iterate of the approximated recurrent cache. Formally, denote $\mathbf{R}_{\langle 1\rangle} := \text{ShiftRight}(\boldsymbol{H}^{(\ell_{\text{end}})}_{\langle 0\rangle})$. Here, we use subscripts in angle brackets to denote the index of iteration for the recurrent cache and the subsequent representation induced by that cache.

After getting this approximated cache, we merge it back with the already cached $\boldsymbol{H}^{(\ell_{\text{start}}-1)}$ with the fusion module $\Phi(\cdot,\cdot)$. Then we pass the fused representation through the middle layers, getting

$$\boldsymbol{H}^{(\ell_{\text{end}})}_{\langle 1\rangle} := \mathcal{F}_{\ell_{\text{start}}:\ell_{\text{end}}}\Big(\Phi\left(\boldsymbol{H}^{(\ell_{\text{start}}-1)}_{\langle 0\rangle}, \mathbf{R}_{\langle 1\rangle}\right)\Big). \tag{A.1}$$

This gives us the post-$\ell_{\text{end}}$ representation conditioned on the first iterate of the cache $\mathbf{R}_{\langle 1\rangle}$. We then combine the right-shifted $\mathbf{R}_{\langle 1\rangle}$ with right shifted $\boldsymbol{H}^{(\ell_{\text{end}})}_{\langle 1\rangle}$ following the temporal cache residual rule to yield the next iterate of cache $\mathbf{R}_{\langle 2\rangle}$. We continue this iterative update for $d_{\text{forward}}$ time, settle on $\mathbf{R}_{\langle D_{\text{forward}}\rangle}$, and use it as the recurrent cache for the final forward pass.

To avoid too deep of a gradient backward pass, we also allow selective detachment analogous to TBPTT in standard RNN training (Williams & Zipser, 2013). We present the full temporal-parallel approximated forward algorithm in Algorithm 1, where we adopt a temporal residual connection for the cache as described in Section 2.3.

When no temporal cache residual is used, line 8 in Algorithm 1 simply becomes

$$\mathbf{R}_{\langle k\rangle} \leftarrow \text{ShiftRight}\Big(\boldsymbol{H}^{(\ell_{\text{end}})}_{\langle k-1\rangle}\Big).$$

**Algorithm 1:** Temporal-Parallel Approximated Forward for T$^2$MLR

**Input:** $D_{forward}$, $D_{backward}$, input embeddings $\boldsymbol{H}^{(0)} \in \mathbb{R}^{T\times d}$
**Output:** $\boldsymbol{H}^{(L)}_{\langle D_{forward}\rangle} \in \mathbb{R}^{T\times d}$

1 $\boldsymbol{H}^{(\ell_{\text{start}}-1)} \leftarrow \mathcal{F}_{1:\ell_{\text{start}}-1}\big(\boldsymbol{H}^{(0)}\big)$ `// prefill to` $\ell_{\text{start}}-1$
2 $\boldsymbol{H}^{(\ell_{\text{end}})}_{\langle 0\rangle} \leftarrow \mathcal{F}_{\ell_{\text{start}}:\ell_{\text{end}}}\big(\boldsymbol{H}^{(\ell_{\text{start}}-1)}\big)$ `// single pass to seed cache`
3 $\mathbf{R}_{\langle 1\rangle} \leftarrow \text{ShiftRight}\Big(\boldsymbol{H}^{(\ell_{\text{end}})}_{\langle 0\rangle}\Big)$ `// initialize deep cache`
4 **for** $k = 2$ **to** $D_{forward}$ **do**
5 $\quad \boldsymbol{H}^{(\ell_{\text{end}})}_{\langle k-1\rangle} \leftarrow \mathcal{F}_{\ell_{\text{start}}:\ell_{\text{end}}}\Big(\Phi\left(\boldsymbol{H}^{(\ell_{\text{start}}-1)}_{\langle 0\rangle}, \mathbf{R}_{\langle k-1\rangle}\right)\Big)$ `// recurrent refinement`
6 $\quad$ **if** $k > D_{forward} - D_{backward}$ **then**
7 $\quad\quad \boldsymbol{H}^{(\ell_{\text{end}})}_{\langle k-1\rangle} \leftarrow \text{StopGrad}\Big(\boldsymbol{H}^{(\ell_{\text{end}})}_{\langle k-1\rangle}\Big)$ `// truncate gradients`
8 $\quad \mathbf{R}_{\langle k\rangle} \leftarrow \text{Normalize}\left(\text{ShiftRight}\Big(\boldsymbol{H}^{(\ell_{\text{end}})}_{\langle k-1\rangle} + \mathbf{R}_{\langle k-1\rangle}\Big)\right)$ `// update cache`
9 $\boldsymbol{H}^{(L)}_{\langle D_{\text{forward}}\rangle} \leftarrow \mathcal{F}_{\ell_{\text{start}}:L}\Big(\Phi\left(\boldsymbol{H}^{(\ell_{\text{start}}-1)}_{\langle 0\rangle}, \mathbf{R}_{\langle D_{\text{forward}}\rangle}\right)\Big)$ `// forward w/ refined cache`
10 **return** $\boldsymbol{H}^{(L)}_{\langle D_{forward}\rangle}$

| Model | Train (per seq) | Infer (per token) |
|---|---|---|
| Vanilla | $O\big(L\,(N^2d + Nd^2)\big)$ | $O\big(L\,(Nd + d^2)\big)$ |
| Full-loop | $O\big(KL\,(N^2d + Nd^2)\big)$ | $O\big(KL\,(Nd + d^2)\big)$ |
| Mid-loop | $O\big((L + (K-1)D)\,(N^2d + Nd^2)\big)$ | $O\big((L + (K-1)D)\,(Nd + d^2)\big)$ |
| T$^2$MLR | $O\big((L + d_{\text{forward}}D)\,(N^2d + Nd^2)\big)$ | $\approx O\big(L\,(Nd + d^2)\big)$ |

Table 3: **Asymptotic compute comparison.** Here $N$ is the sequence length, $L$ the total number of layers, $d$ the hidden width, and $D = \ell_{\text{end}} - \ell_{\text{start}} + 1$ the size of the recurrent middle block.

### A.2 Computation Overhead of Temporal-Parallel Jacobi Training

We summarize the asymptotic computation cost of T$^2$MLR in Table 3. Let $N$ denote the sequence length, $L$ the total number of Transformer layers, $d$ the hidden width, and $D = \ell_{\text{end}} - \ell_{\text{start}} + 1$ the size of the recurrent middle block. For full-sequence training without KV caching, each layer incurs cost $O(N^2d + Nd^2)$, where the $O(N^2d)$ term comes from dense self-attention and the $O(Nd^2)$ term comes from projections and MLPs. For autoregressive inference with KV caching at context length $N$, each layer costs $O(Nd + d^2)$ per generated token.

Relative to a vanilla Transformer, the training overhead of T$^2$MLR comes from the $d_{\text{forward}}$ Jacobi-style refinement steps used to approximate the recurrent cache in parallel across token positions. Importantly, these extra passes only re-run the recurrent middle block of size $D$, rather than the full $L$-layer network. As a result, the training cost of T$^2$MLR is $O\big((L + d_{\text{forward}}D)\,(N^2d + Nd^2)\big)$, compared with $O\big(L\,(N^2d + Nd^2)\big)$ for a vanilla Transformer.

This should be contrasted with full-layer looping approaches, whose cost scales as $O(KL\,(N^2d + Nd^2))$, and middle-block looping approaches, whose cost scales as $O((L+(K-1)D)\,(N^2d+Nd^2))$. Thus, the additional cost of T$^2$MLR scales only with the recurrent block size $D$ and the Jacobi depth $d_{\text{forward}}$, which is substantially cheaper than repeatedly looping over the entire stack when $D \ll L$.

At inference time, T$^2$MLR does not require extra serial Jacobi refinement. Once the recurrent cache from the previous token is available, the current token is processed with a single forward pass together with one application of the fusion module. Therefore, under KV caching, the per-token inference complexity remains approximately the same as that of a standard Transformer, $O\big(L\,(Nd + d^2)\big)$ up to a small constant-factor overhead from the fusion computation. This distinguishes T$^2$MLR from looped-Transformer approaches, whose inference cost grows linearly with the number of serial loops.

The same argument is particularly favorable in RL settings. During online rollout generation, the model already computes the exact recurrent latent states sequentially token by token, so for rollout tokens there is no need to perform additional forward-side Jacobi refinement. Instead, one can cache these exact latent representations during decoding and reuse them during policy optimization. Temporal-parallel approximation is only needed in settings such as prefill, teacher-forced training, or replayed trajectories where token-parallel processing is desired.

It is worth noting that in RL-style training, $d_{\text{forward}}$ need not be chosen to improve the approximation quality of recurrent cache themselves. It only needs to be large enough to construct the temporal computation graph needed for BPTT. Therefore one can set $d_{\text{forward}} = d_{\text{backward}}$, which will significantly reduce the practical overhead relative to supervised pretraining.

### A.3 On the Validity of Approximation Depth

Due to the existence of the $\mathrm{ShiftRight}$ operation, the $t$-th column of $\mathbf{R}$ will converge after $t$-iterations. In this subsection, we show that empirically, we might need way fewer depth for training.

We take a middle checkpoint from the FineWeb-Edu pretraining runs (with $\ell_{\text{start}} = 5, \ell_{\text{end}} = 26, d_{\text{forward}} = 16, d_{\text{backward}} = 4$), sample a random training batch with 2048 tokens per sequence, and compute the gradients under different approximation-depth settings $(d_{\text{forward}}, d_{\text{backward}})$ in Algorithm 1. We treat the gradients under the largest setting $(d_{\text{forward}}, d_{\text{backward}}) = (32, 32)$ as an *anchor*.

For each $(d_{\text{forward}}, d_{\text{backward}})$ setting, we measure the difference between the gradient and the anchor by (i) the cosine similarity and (ii) the $\ell_2$ distance normalized by the norm of the anchor gradients (which we denote by relative $\ell_2$ distance). As shown in Figure 8, setting $d_{\text{forward}} < 8$ and $d_{\text{backward}} \leq 4$ could lead to drastic gradient differences compared to the anchor, while further increasing depth yields diminishing benefits.

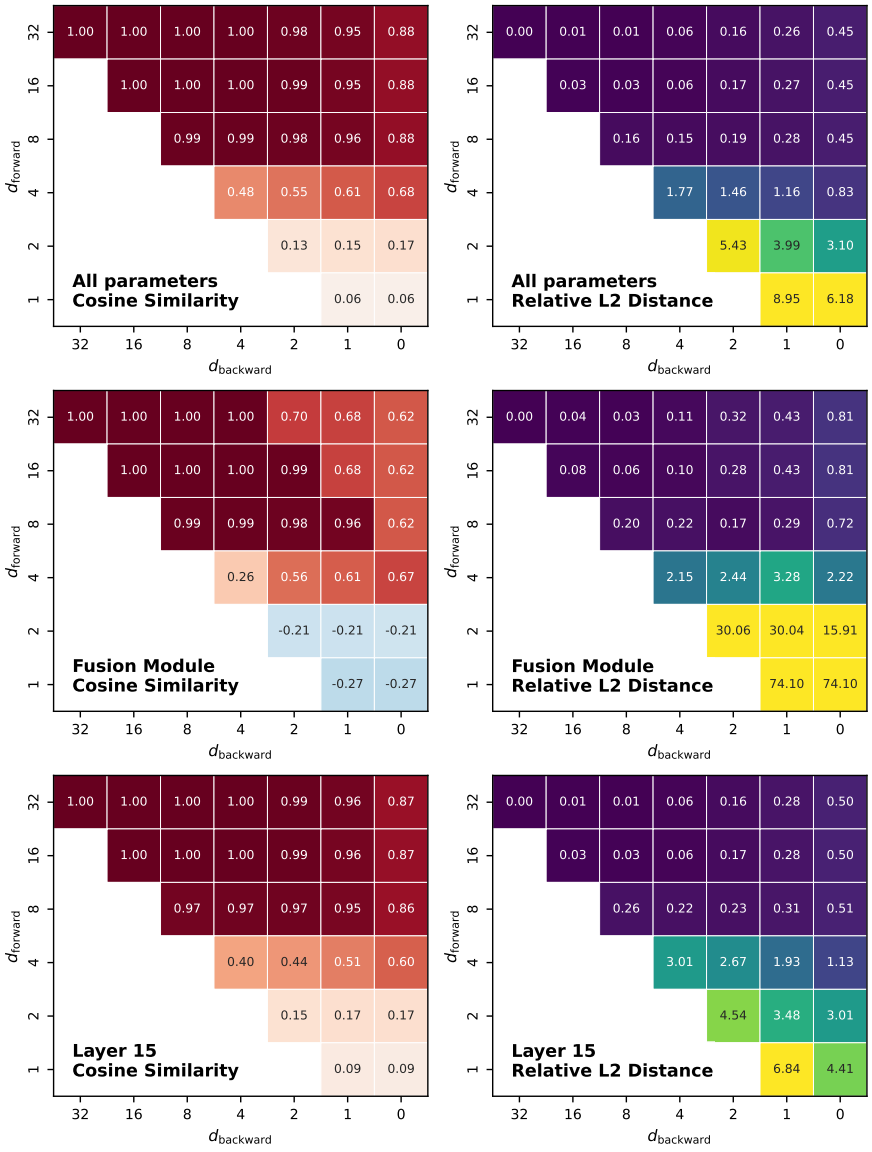


Figure 8: Cosine similarity and relative $\ell_2$ distance between gradients attained under different setup of $(d_{\text{forward}}, d_{\text{backward}})$ against setting $d_{\text{forward}} = 32, d_{\text{backward}} = 32$.

## B FUTURE TOKEN PREDICTION

Temporal middle-layer recurrence creates an explicit temporal shortcut: the recurrent cache computed from token $t$ is fused into the residual stream when processing token $t{+}1$, so the loss at position $t{+}1$ can backpropagate through this pathway into token $t$'s intermediate representation.

This encourages the model to maintain latent states that better anticipate upcoming tokens. To test this hypothesis, we perform a probing analysis: from each intermediate layer, we extract the representation of token $t$ and train a two-layer MLP to predict the next token $x_{t+1}$. Table 4 reports next-token prediction loss for T$^2$MLR and a parameter-matched Transformer baseline under two choices of $\ell_{\text{end}}$. We probe representations from layers $l_{-k}$, where $l_{-k}$ denotes the layer $k$ steps below the LM head.

| | $\ell_{\text{end}} = -3$ | | | $\ell_{\text{end}} = -5$ | | | | |
|---|---|---|---|---|---|---|---|---|
| | $\ell_{-1}$ | $\ell_{-2}$ | $\ell_{-3}$ | $\ell_{-1}$ | $\ell_{-2}$ | $\ell_{-3}$ | $\ell_{-4}$ | $\ell_{-5}$ |
| T$^2$MLR | **5.091** | **5.088** | **5.110** | **5.097** | **5.096** | **5.109** | **5.126** | **5.158** |
| Baseline | 5.135 | 5.131 | 5.145 | 5.135 | 5.131 | 5.145 | 5.173 | 5.182 |

Table 4: Each predictor is trained for 20,000 steps on Fineweb-edu 10B.

Table 4 shows that intermediate-layer representations learned by T$^2$MLR consistently yield lower next-token prediction loss than the parameter-matched baseline, providing direct evidence that T$^2$MLR promotes representations that better anticipate future tokens.

# C Extended Discussions

## C.1 T$^2$MLR as a generalized Transformer and recurrent model

T$^2$MLR subsumes both recurrent neural networks and the standard Transformer as special cases. In particular, when the learned self-attention pattern reduces to the identity—so that token interactions occur exclusively through recurrence—T$^2$MLR becomes a fully recurrent model. Conversely, when the learned gating parameters $\gamma_{\text{cur}}$ and $\gamma_{\text{rec}}$ are set to zero, the recurrent pathway is disabled and T$^2$MLR recovers the standard Transformer architecture.

Consequently, T$^2$MLR has the expressive capacity to smoothly interpolate between recurrent and attention-based computation. Since Transformers excel at in-context retrieval tasks (Jelassi et al., 2024; Nichani et al., 2025) while recurrent models are well suited for explicit state tracking (Merrill et al., 2024; Grazzi et al., 2025), T$^2$MLR has the expressivity to naturally support both modes of computation within a unified framework.

## C.2 Learned gating behavior

As formulated in Section 2.3, T$^2$MLR learns scalar, data-independent gates $\gamma_{\text{cur}}$ and $\gamma_{\text{rec}}$ that control the contributions of the input and recurrent streams, respectively. To better understand how the model allocates these streams relative to the residual branch $x_i^{(t)}$ in Equation 2.3, we analyze the learned gating parameters over the course of training.

Empirically, across all tasks we study, the recurrent gate $\gamma_{\text{rec}}$ consistently converges to positive values, while the input gate $\gamma_{\text{cur}}$ consistently converges to negative values. Figure 9 illustrates this behavior for pretraining on FineWeb. This pattern aligns with the structure of Equation 2.3: since the input representation $x_i^{(t)}$ is already added unconditionally via the residual connection, the role of the $\gamma$-gated terms is not to reintroduce the input signal, but rather to *modulate deviations from the identity mapping*. In particular, a negative $\gamma_{\text{cur}}$ suppresses redundant amplification of the input stream, while a positive $\gamma_{\text{rec}}$ selectively promotes the recurrent contribution, allowing information from previous time steps to be injected only when it provides additional predictive value beyond the current-token representation.

This asymmetry suggests that T$^2$MLR learns to treat recurrence as an additive refinement to the residual pathway, rather than as a competing source of information. In effect, the model preserves the standard Transformer computation as a default and leverages the recurrent stream to inject latent, temporally accumulated information when beneficial.

Finally, note that both $\gamma_{\text{cur}}$ and $\gamma_{\text{rec}}$ are initialized to zero, so the gating module in Equation 2.3 initially reduces to the identity function. As a result, T$^2$MLR starts training in a regime that is exactly equivalent to a standard Transformer, and only gradually departs from it as the gating parameters adapt. This design stabilizes optimization and ensures that recurrent computation is introduced smoothly, rather than being imposed a priori.

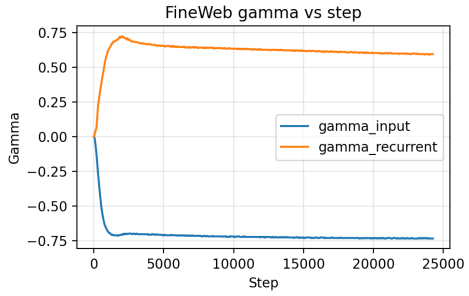


Figure 9: Plot of gating parameters $\gamma_{\text{cur}}$ and $\gamma_{\text{rec}}$ with respect to training steps on the FineWeb dataset. This illustrates the learned behavior of input and recurrent gating over the course of training.

# D Additional Experiment Setups and Results

## D.1 NLP benchmarks for evaluating pre-trained models

We use the following abbreviations for the NLP benchmarks: ARC-C/E: ARC-Challenge/Easy (Clark et al., 2018), HS: HellaSwag (Zellers et al., 2019), OBQA: OpenBookQA (Mihaylov et al., 2018), PIQA: PhysicalInteractionQA (Bisk et al., 2020), SciQ: ScienceExamQA (Welbl et al., 2017), WG: Winogrande (Sakaguchi et al., 2021).

This set of benchmarks follows standard evaluation setup in the pretraining literature.

## D.2 Data Curation for HotpotQA

To construct SFT data with step-by-step multi-hop reasoning, we curate a subset of HotpotQA and generate 19k chain-of-thought reasoning traces using GPT-4o-mini. As noted earlier, we restrict our experiments to the *easy* subset, since preliminary experiments show that both baseline models and T$^2$MLR struggle significantly on the medium and hard subsets, making it difficult to draw meaningful comparisons.

**Prompt for generating HotpotQA chain of thoughts**

According to the context, answer the question in a multi-hop reasoning process. Be sure to include a detailed and clear logic chain in your reasoning. Detail your thought process at each step. Put your final answer within \boxed{}. Output example: \boxed{Knox County Regional Airport}.
Question: {*question*}
Context: {*context*}
Note that the correct final answer is \boxed{*answer*}.

## D.3 Training details

We note the training details used for each task.

| Task | Model | LR | Schedule | Steps |
|---|---|---|---|---|
| S5-Retrieval | **Transformer:** 4-layer, 6-head 384 hidden dim, RoPE **LSTM:** 4-layer, 456 hidden size | 5e-4 | Cosine | 200k |
| FineWeb-Edu Pretraining | SmolLM2-135M-T$^2$MLR | 5e-4 | Cosine w/ min LR | 20k |
| GSM8K-Aug | SmolLM2-135M-T$^2$MLR | 5e-5 | Cosine | 8000 |
| HotpotQA | SmolLM2-135M-T$^2$MLR | 1e-4 | Cosine w/ min LR | 5000 |

Table 5: Training details for various experiments.

For pretraining on FineWeb-Edu, we used AdamW (Loshchilov & Hutter, 2019) optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and weight-decay of $\lambda = 0.01$. We set the minimum learning rate decay to be $0.001$ times the peak learning rate. We pack the pretraining data into sequences of length 2048 and train with batchsize 256. For finetuning runs we used AdamW (Loshchilov & Hutter, 2019) optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight-decay of $\lambda = 0.01$. Training is conducted on 4 $\times$ H100 GPUs.

**Data examples** We provide simple examples of the datasets used in our tasks.

| Dataset | Details |
|---|---|
| ProsQA | **Input:** Every zolufibus is a conarus. Every conarus is a tanirus. Every tanirus is a gedatus. Every cemapus is a lovaperus. Every lovaperus is a zucirarus. Every zucirarus is a pilevus. Every sasarus is a madusurus. Every madusurus is a cidodivus. Every cidodivus is a bumobus. Jordan is a sasarus. Is Jordan a gedatus or a pilevus?<br>**Steps:** Jordan is a sasarus.<br>Every sasarus is a madusurus.<br>Every madusurus is a cidodivus.<br>Every cidodivus is a bumobus.<br>Every bumobus is a nuronus.<br>Every nuronus is a metuvelus.<br>Every metuvelus is a resamus.<br>Every resamus is a gedatus.<br>**Output:** Jordan is a gedatus. |
| S5 Retrieval | **Input:**<br>`<A_32514>F4Nd <A_12543>9O8W <A_54213>Jccq | <A_54213> <A_43125> <A_52314>`<br>**Target:**<br>`<A_32514>F4Nd <A_12543>9O8W <A_54213>Jccq | <A_54213>Jccq <A_12543>9O8W <A_32514>F4Nd` |
| GSM-Aug | **Question:** Out of 600 employees in a company, 30% got promoted while 10% received bonus. How many employees did not get either a promotion or a bonus?<br>**Steps:** 600 x 30/100 = 180 employees were promoted.<br>600 x 10/100 = 60 employees received a bonus.<br>So a total of 180+60=240 employees received a promotion or a bonus.<br>Therefore, 600 - 240 = 360 employees did not get either a promotion or a bonus.<br>**Answer:** 360 |
| Variable assignment | **Input:** Variable assignment. Follow the assignments and fill in the blank.<br>a=5<br>b=2<br>c=8<br>x=c<br>y=x<br>z=y<br>w=b<br>v=w<br>z=___<br>Answer: \|<br>**Outputs:** 8 |
| HotpotQA | **Input:** Which magazine was founded first, *The Economist* or *The Atlantic*?<br>**Context:**<br>Document 1: *The Economist* is a weekly newspaper founded in 1843.<br>Document 2: *The Atlantic* is a magazine founded in 1857.<br>**Steps:**<br>The Economist was founded in 1843.<br>The Atlantic was founded in 1857.<br>1843 is earlier than 1857.<br>**Answer:** The Economist |

Table 6: Examples for each dataset used.

**Additional Pretraining Results** Table 7 shows the pretraining perplexities of $T^2$MLR and baseline Transformer at different model scales.

| | 135M | | 368M | |
|---|---|---|---|---|
| | Baseline | $T^2$MLR | Baseline | $T^2$MLR |
| Perplexity↓ | 22.64 | **21.54** | 16.12 | **15.79** |

Table 7: We pretrain the standard Transformer and $T^2$MLR, both based on SmolLM, at different scales on Fineweb-edu-10B, with $T^2$MLR consistently having improved test time performance.