NESTING IS ALL YOU NEED: RECURSIVE MEMORY PLANE WITH LATENT SPECIALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Most language models treat depth as disposable computation: intermediate representations are transformed and discarded until only a terminal state remains. We introduce **Recursive Memory Plane**, a language-model architecture built around **nested specialization**: Primary (P), Intermediate (I), and Association (A) cells are structurally specialized for local, contextual, and global computation, forming a persistent P→I→A hierarchy whose entire representational trajectory remains jointly addressable. Rather than recur only over the terminal hidden state, each recursive step performs a **reason–readdress–integrate cycle**: a reasoning state R queries the full memory plane, then a discourse state D, conditioned on the updated reasoning state, independently re-queries the same plane to form the next prediction state. **LatentBank** further adds compact, persistent I- and A-specialist states, preserving nested specialization alongside recursion. This design yields an unexpected regime: **recursion is most valuable during training, while shallow inference performs best**. At 220M scale, LatentBank adds only **0.23%** parameters; training with three recursive steps and inferring with one achieves a **34.66% raw mean across nine benchmarks**, versus **33.87% for the strongest parameter matched baseline**. Eliminating inference recursion still reaches **34.52%**, whereas training without recursion falls to **32.99%**. These results suggest a different design principle: **nest specialized computation, preserve depth as memory, recursively re-address the full representational trajectory, and use deep training to produce powerful shallow inference**.

1 INTRODUCTION

Transformers process token representations through a fixed sequence of attention and feedforward blocks, using the final depth for prediction (Vaswani et al., 2017). Scaling this template has produced increasingly capable language models, while the organization of depth has remained comparatively stable. Intermediate representations contribute to the forward trajectory but are usually transient once later layers have consumed them. This raises a basic architectural question: if different depths develop different computational roles, should the full trajectory remain available when further computation is performed?

Several lines of work have explored computation beyond a fixed feedforward stack. Adaptive Computation Time allocates recurrent computation dynamically (Graves, 2016); Universal Transformers reuse a transition across depth (Dehghani et al., 2019); Deep Equilibrium Models define depth through convergence (Bai et al., 2019); and Looped Transformers study repeated Transformer computation (Yang et al., 2024). Recent reasoning models place latent recursion closer to the center of the architecture. Hierarchical Reasoning Models use recurrent modules with distinct computational rates (Wang et al., 2025), Tiny Recursive Models apply repeated latent refinement with a compact network (Jolicoeur-Martineau, 2025), and Ouro studies looped language models at pretraining scale (Zhu et al., 2025b). Other work enriches information flow through the residual pathway. Hyper Connections introduce multiple interacting residual streams (Zhu et al., 2025a), while manifold constrained Hyper Connections improve their stability (Xie et al., 2026). The Full Bandwidth Transformer carries a final hidden representation across autoregressive steps (Wang et al., 2026). Together, these results motivate richer use of internal computation and persistent latent state.

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

Our design is also motivated by evidence that biological language processing is distributed across differentiated systems (Hickok & Poeppel, 2007; Friederici, 2011; Hagoort, 2013). The dual stream account distinguishes complementary pathways for speech processing and sensorimotor integration (Hickok & Poeppel, 2007), while functional imaging has identified distributed language selective regions across frontal and temporal cortex (Fedorenko et al., 2010; 2024). Precision mapping further reveals closely juxtaposed specialized networks within individuals (Braga et al., 2020). Most directly, Du et al. (2026) identify nested, distributed, left lateralized language networks: an intermediate language network, intLANG, preferentially recruited by phonological demands, and a surrounding association language network, aLANG, preferentially recruited during meaning based sentence processing. This organization supports a broader view in which linguistic computation emerges from interacting systems with distinct functional roles (Friederici, 2011; Hagoort, 2013; Fedorenko et al., 2024).

We adopt this organizational principle as architectural inspiration. Figure 1 summarizes the resulting architecture. Our design contains *Primary* (P), *Intermediate* (I), and *Association* (A) specialists for local, contextual, and global computation. Their outputs form a heterogeneous depth trajectory. We preserve that trajectory as

$$\mathcal{M}_Y = [Y_1, Y_2, \ldots, Y_L], \tag{1}$$

and call the resulting structure the **Recursive Memory Plane**. Its axes are representational depth Y and recursive computation Z. The complete Y trajectory is computed once and remains jointly addressable during recursion.

Each recursive step performs two independent reads over this memory. A reasoning state R first queries the complete trajectory. The resulting R_z then conditions a second query for the discourse state D, which forms the next prediction state:

$$(D_{z-1}, R_{z-1}) \to \mathrm{Read}_R(\mathcal{M}_Y) \to R_z \to \mathrm{Read}_D(\mathcal{M}_Y) \to D_z. \tag{2}$$

Both updates reuse the terminal global Association operator while maintaining independent depth queries. We refer to this computation as a *reason, readdress, and integrate* cycle. Recursive computation can therefore recover information from early local states, intermediate contextual states, and late global states throughout the recursive trajectory.

We further introduce **LatentBank**, a compact persistent specialist memory aligned with Intermediate and Association computation. The Intermediate bank receives Primary and Intermediate signals, while the Association bank receives Intermediate bank and Association signals, preserving an asymmetric P→I→A organization. Both banks retain full token resolution and modulate the native contextual operators through lightweight feature wise transformations. LatentBank adds only 0.23% parameters at 220M scale.

Experiments reveal a separation between useful recursive depth during training and inference. Training with three recursive steps and evaluating with one yields a raw mean of 34.66% across nine benchmarks. Under the same evaluator, matched Transformer, TRM, and mHC baselines obtain 31.87%, 32.97%, and 33.87%, respectively. Removing inference recursion still gives 34.52%, while training the LatentBank architecture without recursion gives 32.99%. These results indicate that recursive computation can shape the underlying representations during training, allowing much of its benefit to persist at shallow inference depth.

Contributions. We introduce three connected ideas. First, nested P→I→A specialists assign different computational scopes across depth. Second, the **Recursive Memory Plane** preserves the complete heterogeneous depth trajectory as memory and readdresses it through independent reasoning and discourse queries. Third, **LatentBank** adds compact persistent specialist states with only 0.23% additional parameters. Together, these mechanisms result in an architecture in which deeper recursive training yields stronger shallow inference, reaching 34.66% against 33.87% for the strongest matched baseline.

2 METHOD

Our model combines three components: a nested specialist backbone along depth Y, a Recursive Memory Plane that retains and exposes the complete depth trajectory to recursive computation Z,

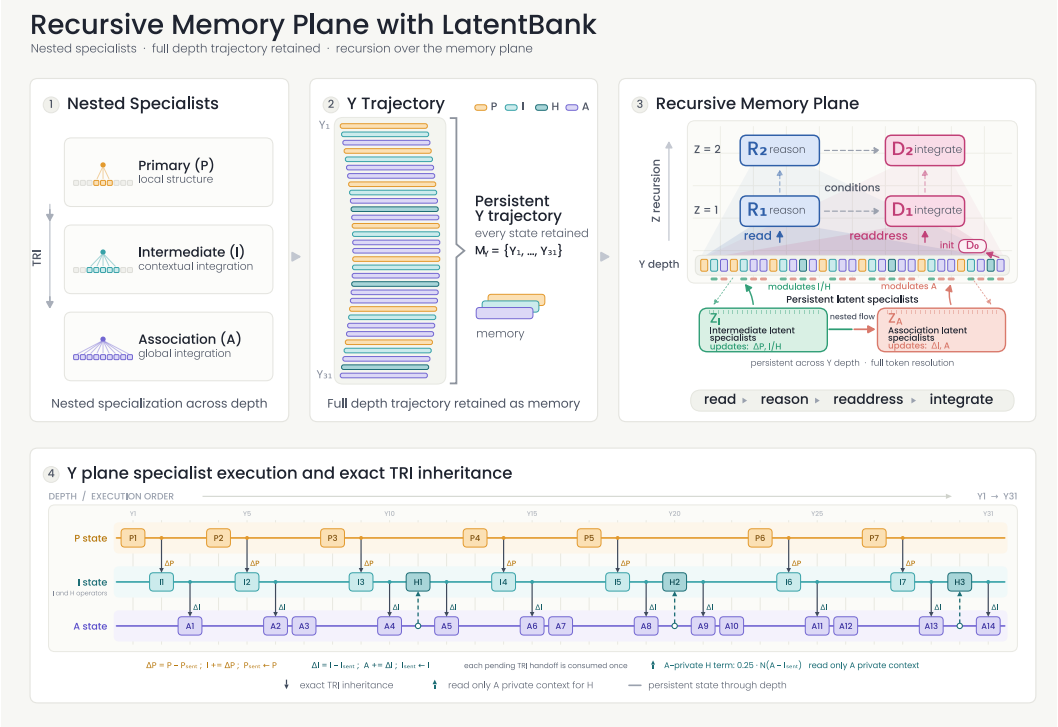


Figure 1: **Recursive Memory Plane with LatentBank.** Nested P/I/A specialists exchange information through TRI, while all 31 P/I/H/A states are retained as the depth memory $\mathcal{M}_Y$. The bottom panel expands the Y plane: persistent P, I, and A states evolve across the interleaved execution order. Each pending TRI handoff is consumed once by the next eligible downstream receiver; under the canonical schedule, ΔP is inherited by the seven I cells, while updates from I or H are inherited by the next Association cell. H additionally receives read only private Association context. Recursive states R and D independently address the complete Y trajectory across recursion Z. Persistent latent states Z_I and Z_A are updated by specialist signals, with $Z_I \to Z_A$ nested flow, and selectively modulate I/H and A computation.

and LatentBank states that preserve compact specialist information across the backbone. Figure 1 summarizes the resulting computation.

2.1 Nested Specialists

Given token embeddings $P_0 \in \mathbb{R}^{B\times T\times d}$, we initialize Intermediate and Association states as $I_0 = A_0 = 0$. The model contains 31 cells with 7 Primary cells, 7 Intermediate cells, 3 Hub cells, and 14 Association cells. Primary cells use causal depthwise convolution with kernel size 7 and dilation 1. Intermediate and Hub cells use causal grouped query attention over a contiguous window of 256 tokens. Association cells use global causal grouped query attention.

Information moves between specialist states through telescoping transport. When Primary computation advances from the last transmitted state P_{sent}, only its newly accumulated change is inherited by I:

$$\Delta P = P - P_{\text{sent}}, \qquad I \leftarrow I + \Delta P, \qquad P_{\text{sent}} \leftarrow P. \tag{3}$$

Intermediate to Association transport follows the same rule:

$$\Delta I = I - I_{\text{sent}}, \qquad A \leftarrow A + \Delta I, \qquad I_{\text{sent}} \leftarrow I. \tag{4}$$

Each newly accumulated source change is consumed exactly once by the first eligible downstream receiver. Thus, consecutive receiver cells do not repeat a handoff unless the source specialist has advanced since the previous inheritance event.

Hub cells retain the Intermediate operator and receive read only feedback from Association state that has not already been inherited from I. Let

$$\mathcal{N}(x) = \frac{x}{\sqrt{d^{-1}\sum_{k=1}^{d} x_k^2 + \epsilon}}. \tag{5}$$

The Hub mixer input is

$$m_H = \frac{\mathcal{N}(I) + \gamma_H \mathcal{N}(A - I_{\text{sent}})}{\sqrt{1+\gamma_H^2}}, \qquad \gamma_H = 0.25. \tag{6}$$

We retain the output of every cell,

$$\mathcal{M}_Y = \{Y_1, \ldots, Y_L\}, \qquad L = 31, \tag{7}$$

so the complete heterogeneous depth trajectory remains available after the backbone has finished.

2.2 Recursive Memory Plane

The Y trajectory forms the persistent memory axis of the Recursive Memory Plane. Recursive depth Z forms its second computational axis. The 31 backbone cells execute once, after which every recursive step can independently address all Y states.

Each memory state supplies an RMS normalized key and an unmodified value:

$$K_j = \mathcal{N}(Y_j), \qquad V_j = Y_j. \tag{8}$$

For query q, the projection free depth read is

$$s_j(q) = \frac{\mathcal{N}(q)^\top K_j}{d\tau}, \tag{9}$$

$$w_j(q) = \frac{\exp s_j(q)}{\sum_{\ell=1}^{L} \exp s_\ell(q)}, \tag{10}$$

$$\text{Read}(q, \mathcal{M}_Y) = \sum_{j=1}^{L} w_j(q) V_j, \tag{11}$$

where τ is the depth temperature. A single softmax therefore competes across all 31 depth states.

For balanced combinations we use

$$\mathcal{C}(x_1, \ldots, x_n) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \mathcal{N}(x_i). \tag{12}$$

Recursion begins from the terminal Association state,

$$D_0 = A_L, \qquad R_0 = 0. \tag{13}$$

At recursive step z, reasoning first addresses the memory:

$$q_z^R = \mathcal{C}(D_{z-1}, R_{z-1}), \tag{14}$$

$$M_z^R = \text{Read}(q_z^R, \mathcal{M}_Y), \tag{15}$$

$$R_z = \Phi_A\big(R_{z-1}; \mathcal{C}(M_z^R, D_{z-1}, R_{z-1})\big), \tag{16}$$

where $\Phi_A(h; m)$ denotes the terminal global Association cell with residual state h and attention mixer input m.

The updated reasoning state then generates a fresh discourse query:

$$q_z^D = \mathcal{C}(D_{z-1}, R_z), \tag{17}$$

$$M_z^D = \text{Read}(q_z^D, \mathcal{M}_Y), \tag{18}$$

$$D_z = \Phi_A\big(D_{z-1}; \mathcal{C}(M_z^D, D_{z-1}, R_z)\big). \tag{19}$$

The same Φ_A parameters are reused for R and D, while their memory reads remain independent. The updated R_z therefore changes which depth states can contribute to D_z. For Z recursive steps, the model executes

$$C(Z) = 31 + 2Z \tag{20}$$

operator cells.

2.3 LATENTBANK

LatentBank augments the nested backbone with two compact persistent specialist states. The Intermediate bank contains four 16 dimensional banks, $Z_I \in \mathbb{R}^{B\times T\times 4\times 16}$, while the Association bank contains four 24 dimensional banks, $Z_A \in \mathbb{R}^{B\times T\times 4\times 24}$. Both retain the complete token axis.

For bank b, a source state is encoded as

$$u_b(x) = \tanh(W_b \mathcal{N}(x)) \,. \tag{21}$$

Each latent bank has an independent recurrent matrix R_b and learned retention $\lambda_b = \sigma(\ell_b)$, initialized at 0.5. Its update is

$$z_b' = \lambda_b z_b + \sqrt{1-\lambda_b^2}\, \tanh(u_b + R_b \mathcal{N}(z_b)) \,. \tag{22}$$

The connectivity follows the specialist hierarchy:

$$\Delta P,\ I/H \to Z_I \to Z_A, \qquad \Delta I,\ A \to Z_A. \tag{23}$$

Latent states modulate only the attention mixer inputs of Intermediate, Hub, and Association cells. Given latent state Z, a linear readout produces (α, β) and applies

$$x' = x \odot (1 + \varepsilon_F \tanh \alpha(Z)) + \varepsilon_F \beta(Z), \qquad \varepsilon_F = 10^{-2}. \tag{24}$$

The latent states are therefore kept outside the native residual transport. At 220M scale, LatentBank contributes 496,392 parameters, corresponding to 0.2255% of the backbone parameter count.

2.4 TRAINING AND INFERENCE

Training uses $Z_{\text{train}} = 3$ recursive steps. The language modeling head receives only the final discourse state:

$$p_\theta(x_{t+1} \mid x_{\le t}) = \text{softmax}\left(W_U\ \text{FinalNorm}(D_3[t])\right), \tag{25}$$

with the standard causal language modeling objective

$$\mathcal{L} = -\sum_t \log p_\theta(x_{t+1} \mid x_{\le t}). \tag{26}$$

There are no auxiliary losses at D_0, D_1, or D_2. At inference, Z can be selected independently of the training depth. We evaluate $Z \in \{0, 1, 2, 3, 4, 6\}$ and find that the strongest aggregate result occurs at $Z = 1$.

3 EXPERIMENTS AND RESULTS

Experimental setup. We compare against Transformer (Vaswani et al., 2017), TRM (Jolicoeur-Martineau, 2025), and mHC (Xie et al., 2026) at the 220M parameter scale. All models use the same pretraining configs and FineWeb Edu data (Penedo et al., 2024) within each parameter scale. Training uses sequences of 1,024 tokens, a global batch of 132 sequences, and 73,982 optimizer updates, corresponding to 10B processed tokens. We use AdamW with a peak learning rate of 3×10^{-4}, minimum learning rate 3×10^{-5}, 3700 warmup steps, weight decay 0.1, and gradient clipping at 1.0. Training runs use four V100 GPUs. Our model contains 220.59M parameters, including 496,392 LatentBank parameters (0.2255% over the native backbone), and uses $Z_{\text{train}} = 3$ unless stated otherwise.

Training depth controls use $Z_{\text{train}} \in \{0, 1, 3\}$, and a separate Z_I only model is trained with $Z_{\text{train}} = 3$ to measure the contribution of Association latent state during training.

We evaluate with one fixed protocol on BLiMP (Warstadt et al., 2020), BoolQ (Clark et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC Challenge (Clark et al., 2018), OpenBookQA (Mihaylov et al., 2018), MMLU (Hendrycks et al., 2021a), GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021b), and BBH (Suzgun et al., 2023). We report every task separately and use the unweighted mean of the nine accuracies as a compact aggregate across three seeds per model.

Table 1: Mean accuracy (%) at the 220M scale. Our model is trained with $Z = 3$ and evaluated with $Z = 1$. Mean is the unweighted average across the nine tasks.

Model	BLiMP	BoolQ	Wino	ARC	OBQA	MMLU	GSM8K	MATH	BBH	Mean
Transformer	74.52	47.31	51.07	19.40	14.40	25.52	0.99	0.00	53.62	31.87
TRM	72.19	50.12	51.54	**23.41**	14.40	25.14	**1.44**	0.00	58.49	32.97
mHC	72.81	57.77	51.54	21.74	**16.20**	25.67	0.60	0.00	58.53	33.87
RMP + LatentBank	**76.59**	**59.57**	**51.62**	21.74	15.80	**25.85**	0.99	0.00	**59.75**	**34.66**

Table 2: Measured inference efficiency at 220M scale. Time is median wall clock time per forward; memory is peak allocated GPU memory. Relative throughput is normalized to the Decoder.

Model	Params (M)	Mean (%)	Time (ms)	Throughput (k tok/s)	Rel. thr. ($\times$ Dec.)	Peak (GiB)
Decoder	219.66	31.87	57.01	**17.96**	**1.000**	0.96
TRM	221.43	32.97	61.06	16.77	0.934	0.97
mHC	220.09	33.87	64.30	15.93	0.887	**0.96**
RMP Z=0	220.59	34.52	63.14	16.22	0.903	1.05
RMP Z=1	220.59	**34.66**	69.21	14.80	0.824	1.16

Inference profiling. We measure full sequence forward inference on a single V100 16 GB GPU using the same profiler for all 220M models. We use batch size 1, sequence length 1024, FP32 inference, disabled gradients and compilation, five warmup forwards, and three repetitions of 20 timed forwards. We synchronize CUDA around every measurement and report the median over all 60 timed forwards. Throughput therefore measures controlled full sequence forward computation and does not measure cached autoregressive decoding..

Table 2 complements the operator count analysis with measured wall clock throughput. The Decoder reaches 17.96k tokens/s, with TRM, mHC, RMP at $Z = 0$, and RMP at $Z = 1$ retaining 93.4%, 88.7%, 90.3%, and 82.4% of Decoder throughput, respectively. RMP at $Z = 0$ reaches 16.22k tokens/s, 1.8% above mHC, while achieving a raw mean of 34.52% versus 33.87%. A single recursive step raises the mean to 34.66% at 14.80k tokens/s. Thus, $Z = 0$ preserves nearly all of the aggregate gain while recovering substantial inference throughput, whereas $Z = 1$ exchanges additional runtime for the highest aggregate score.

Comparison with matched baselines. RMP + LatentBank reaches 34.66%, improving over mHC by 0.79 percentage points, TRM by 1.69 points, and Transformer by 2.79 points. It gives the strongest result on BLiMP, BoolQ, WinoGrande, MMLU, and BBH. TRM leads on ARC Challenge and GSM8K, while mHC leads on OpenBookQA. The gains therefore span linguistic acceptability, question answering, knowledge, and reasoning rather than concentrating in a single benchmark.

Shallow inference after recursive training. Inference depth exhibits a clear optimum. The model trained with three recursive steps reaches its maximum mean accuracy at $Z = 1$, using 33 operator cell evaluations. Removing recursion at inference reduces the mean by only 0.14 points and requires 31 cells. Extending inference to the training depth gives 34.24%, and $Z = 6$ falls to 33.72% despite using 43 cells. Recursive inference therefore has a nonmonotonic accuracy profile. Under the present full sequence protocol, $Z = 0$ minimizes operator depth while $Z = 1$ gives the highest aggregate accuracy.

LatentBank contribution Panel 3(b) isolates runtime dependence using trained checkpoints. Disabling LatentBank reduces the $Z = 0$ mean by 0.52 points. Removing Z_I produces nearly the same reduction, whereas removing Z_A costs only 0.05 points. The Intermediate bank therefore carries most of the direct inference contribution in this checkpoint. Removing recurrent bank dynamics costs 0.34 points, and removing the $Z_I \to Z_A$ path costs 0.07 points.

The FiLM decomposition is also informative. Retaining only the additive term reaches 34.54%, while retaining only multiplicative scaling gives 34.02%. The learned additive modulation accounts for most of the immediate FiLM benefit.

Table 3: Recursion and LatentBank analysis. Panel (a) varies inference recursion after training with $Z = 3$. Panel (b) applies runtime ablations to the same checkpoint at $Z = 0$; Δ is relative to the full $Z = 0$ model. Panel (c) compares independently trained controls, isolating training recursion and latent specialist capacity.

(a) Inference recursion

Z	Cells	Mean
0	31	34.52
1	33	**34.66**
2	35	34.33
3	37	34.24
4	39	34.36
6	43	33.72

(b) Same checkpoint ablations at $Z = 0$

Setting	Mean	Δ
Full LatentBank	34.520	0.000
LatentBank off	34.004	-0.517
Remove Z_I	34.022	-0.498
Remove Z_A	34.466	-0.055
Remove $Z_I \rightarrow Z_A$	34.453	-0.068
Additive FiLM only	**34.540**	$+0.019$
Multiplicative FiLM only	34.020	-0.501
Remove bank recurrence	34.180	-0.341

(c) Trained controls

Model	Train Z	Infer Z	Mean
Full	3	1	**34.66**
Full	3	0	34.52
Full	1	0	33.05
Full	0	0	32.99
Z_I only	3	0	33.14
NSN only	3	1	33.64

The Association bank shows a different pattern across training and inference. Disabling Z_A only at inference reduces the mean from 34.52% to 34.47%. In contrast, a model trained from scratch with only Z_I reaches 33.14% at the same $Z = 0$ inference depth, a reduction of 1.38 points relative to the full model trained with $Z = 3$. This indicates that Z_A contributes strongly to representation formation during training even though its direct contribution to the final shallow inference computation is small.

Recursive training shapes shallow representations. Controlling recursion during training reveals a strong depth effect. With inference fixed at $Z = 0$, training with $Z = 0$, $Z = 1$, and $Z = 3$ gives 32.99%, 33.05%, and 34.52%, respectively. One recursive training step improves the mean by only 0.06 points, while three steps improve it by 1.53 points over $Z = 0$ training. The gain from recursive training therefore emerges primarily with greater training recursion rather than from the presence of a single recursive update.

The recursive model receives language modeling supervision only at D_3, with no auxiliary losses at D_0, D_1, or D_2. Gradients through the recursive path consequently shape the shared specialist trajectory and LatentBank states. Their effect remains visible before recursive inference begins. We call this phenomenon *recursive nested representation shaping*.

Nested specialization alone reaches 33.64% at its strongest tested inference depth, compared with 34.66% for the complete architecture. The Z_I only retraining control reaches 33.14%, while the complete LatentBank model reaches 34.52% at the same $Z = 0$ inference depth. As a result, these controls indicate that the strongest result depends on the interaction among specialist computation, both levels of persistent latent state, and recursive training.

All training comparisons reported here use three different training seeds. The benchmark evaluations are deterministic conditional on their checkpoints.

Scaling to 400M and 800M. Appendix C extends the matched three seed comparison to approximately 400M and 800M parameters under the same 10B processed token budget. RMP with LatentBank reaches 37.81 ± 0.51 at 400M and 39.85 ± 0.46 at 800M, compared with 36.36 ± 0.70 and 38.86 ± 0.43 for mHC. The aggregate RMP advantage remains positive across all three tested scales, with differences of 0.79, 1.45, and 0.99 percentage points at 220M, 400M, and 800M, respectively, based on aggregate means. Figure 2 shows that this advantage persists as model size increases. Full per task results, seed level values, hardware configurations, and scale specific batch construction are reported in the appendix.

4 Discussion

Our results suggest that representational depth can serve as an addressable computational resource. Recursive Memory Plane retains the heterogeneous trajectory $\mathcal{M}_Y = \{Y_1, \ldots, Y_{31}\}$ after the specialist backbone is computed, so recursive states can revisit local, contextual, and global representations without recomputing the full depth sequence. The architecture therefore combines specialization along Y, persistent access to depth memory, and recursive computation along Z.

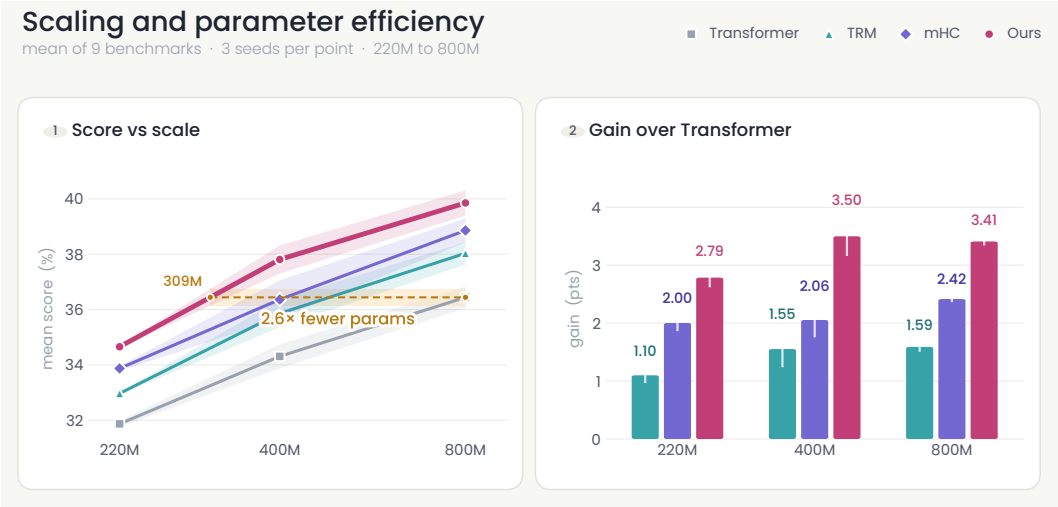


Figure 2: **Scaling and parameter efficiency.** Left: unweighted mean across the nine evaluation benchmarks for Transformer, TRM, mHC, and RMP with LatentBank at approximately 220M, 400M, and 800M parameters, averaged across three independent training seeds. RMP with LatentBank remains above all matched baselines at every tested scale. Right: improvement over the matched Transformer at each scale. The annotated 309M point is a log parameter interpolation between the measured 220M and 400M RMP results at the score reached by the measured 800M Transformer.

Recursive training shapes shallow representations. The central empirical result is the separation between useful training recursion and useful inference recursion. The 220M model is trained with $Z_{\text{train}} = 3$ using language modeling loss only at D_3, yet the strongest aggregate inference result occurs at $Z = 1$ with a raw mean of 34.66%, while $Z = 0$ still reaches 34.52%. Training the same architecture with $Z_{\text{train}} = 0$ gives only 32.99% at $Z = 0$. The intermediate control is also informative: training with $Z_{\text{train}} = 1$ yields 33.05%, only 0.06 percentage points above the zero recursion control, whereas three recursive training steps improve shallow inference by 1.53 percentage points. The benefit therefore emerges substantially with deeper recursive training.

We describe this effect as *recursive nested representation shaping*. Gradients passing through repeated R and D computation alter the specialist backbone and LatentBank states that construct the initial trajectory, allowing D_0 to retain benefits created by deeper training computation. No auxiliary losses are applied at D_0, D_1, or D_2. This result suggests that additional computation during learning can act as a scaffold for producing stronger shallow representations. At inference, $Z = 1$ uses 33 operator cell evaluations compared with 37 along the $Z = 3$ training path and measures 14.80k tokens/s under our controlled full sequence protocol. Removing inference recursion entirely raises throughput to 16.22k tokens/s while retaining a raw mean of 34.52%. This is slightly faster than mHC at 15.93k tokens/s under the same protocol, providing measured evidence that the shallow operating point can preserve most of the representation shaping benefit with lower inference cost.

Persistent specialization complements structural specialization. Structural specialization alone does not account for the full result. At 220M, the nested specialist model without the complete LatentBank mechanism reaches a best tested raw mean of 33.64%, below the full model at both $Z = 0$ and $Z = 1$. Runtime ablations further show that the Intermediate and Association banks contribute differently. Disabling LatentBank at inference reduces the $Z = 0$ mean from 34.52% to 34.00%, and removing Z_I gives 34.02%, indicating that Z_I carries most of the immediate shallow inference contribution. Removing Z_A from the trained checkpoint leaves 34.47%.

The retraining control reveals a larger role for Z_A during representation formation. A 220M model trained from scratch with only Z_I and $Z_{\text{train}} = 3$ reaches 33.14% at $Z = 0$, which is 1.38 percentage points below the full model. Thus, the Association latent state contributes strongly during training even though its direct removal from a trained checkpoint causes only a small immediate drop. This

difference highlights the importance of separating runtime ablation from retraining controls. The asymmetric $Z_I \rightarrow Z_A$ path is consistent with the intended nested organization, although removing this connection alone causes only a small shallow inference change, so its independent contribution remains unresolved.

Depth retrieval matters, but access is redundant. The canonical recursive process independently reads the complete retained trajectory for R and D. Same checkpoint interventions in Appendix E show that zeroing both retrieved vectors at $Z = 1$ gives 34.53%, essentially the $Z = 0$ value of 34.52%, whereas restricting both reads to Primary, Intermediate/Hub, Association, or terminal Y_{31} states gives 34.68%, 34.68%, 34.63%, and 34.74%, respectively. The unrestricted reader nevertheless allocates aggregate read mass across the trajectory. Thus recursive retrieval contributes at $Z = 1$, but this trained checkpoint does not require a particular specialist family or the complete 31 state read to preserve aggregate performance. Because the checkpoint was trained with full trajectory access, these interventions do not establish that complete depth access was unnecessary during optimization. The nonmonotonic Z sweep still makes $Z = 1$ the best aggregate operating point under the present evaluator.

The gain comes from computational organization. LatentBank increases the 220M parameter count by only 0.23%. Under the fixed nine benchmark evaluator, RMP LatentBank reaches a raw mean of 34.66%, compared with 33.87% for mHC, the strongest external baseline, a gain of 0.79 percentage points. The small parameter increase rules out a large increase in parameter count as an explanation for the result. The ablations support contributions from specialist computation, persistent state, and recursive training, although the present experiments do not provide a complete factorial decomposition of these mechanisms. The aggregate is an unweighted raw mean, so individual benchmark results remain the primary measurements. The same ordering persists in the larger scale experiments: RMP with LatentBank exceeds the strongest aggregate baseline at 400M and 800M as well, with full three seed scaling results reported in Appendix C.

Neuroscience motivation. The nested P, I, and A organization was inspired by evidence for partially segregated and nested language systems. We use this evidence as an architectural prior rather than as a claim of biological fidelity. P, I, and A are engineered operator families specialized for local, contextual, and global computation. The relevant design principle is that specialist systems can remain partially distinct while maintaining persistent interaction across levels.

Limitations. The model retains all 31 depth states, so memory cost grows with width, sequence length, and retained depth. Finally, the current design fixes the 31 cell Y trajectory and the recursive training depth. Future experiments should vary specialist depth, model width, LatentBank capacity, training recursion, and inference recursion independently.

5 Conclusion

We introduced **Recursive Memory Plane with LatentBank**, a language model architecture that combines nested Primary, Intermediate, and Association specialization with persistent access to the complete representational depth trajectory. Telescoping Residual Inheritance transfers changes between specialist states, LatentBank maintains compact Intermediate and Association state across depth, and recursive reasoning and discourse states repeatedly address the retained Y trajectory.

At 220M scale, the architecture adds approximately 0.23% parameters and reaches a raw mean of 34.66% across nine benchmarks at $Z = 1$, compared with 33.87% for the strongest external baseline, mHC. More importantly, training with three recursive steps produces a $Z = 0$ model score of 34.52%, while training without recursion reaches 32.99%. Training with one recursive step reaches 33.05%, indicating that the effect becomes substantially stronger with deeper recursive training.

These results support a simple architectural principle: preserve heterogeneous depth during training, maintain persistent specialist state, and use recursive computation to shape representations that remain effective at shallow inference. Same checkpoint interventions further show that recursive retrieval contributes at $Z = 1$ while its useful information is redundantly recoverable across depth.

AI USE STATEMENT

In this work, we used generative AI tools for refactoring source code, editing portions of the manuscript, organizing the presentation of results, and improving readability.

All AI assisted works as highlighted above, and manuscript content were reviewed by the authors. Refactored code was inspected and tested for correctness using architecture specific unit tests, compatibility tests, causality checks, checkpoint comparisons, and end to end training and evaluation runs. Numerical results reported in the paper were obtained from the executed experimental pipelines and were checked against the corresponding outputs.

We did not use generative AI to generate synthetic training or evaluation data, translate research material, conduct qualitative or thematic data analysis, or produce formal mathematical proofs. No generative AI system is listed as an author. **We hereby declare that as authors of this manuscript, we take serious responsibility for the final content of this work, including text, claims, code, analyses, and artifacts**.

ETHICS STATEMENT

This work studies language model architecture and does not involve experiments with recruited human participants or the collection of new personal data. Our models are trained and evaluated using existing language modeling data and established public benchmarks described in the experimental setup.

REPRODUCIBILITY STATEMENT

We provide the architectural specification, training procedure, inference procedure, evaluation protocol, and principal hyperparameters in the main paper and appendix. The Method section specifies the specialist execution sequence, Telescoping Residual Inheritance, Recursive Memory Plane computation, LatentBank updates, and recursive R and D states. The experimental sections report model sizes, training token counts, recursive depths, optimization settings, benchmark tasks, and the inference profiling protocol. Additional ablations and training controls are reported in the appendix. The source code, model configurations, training and evaluation scripts, and commands used for the reported experiments, including the fixed benchmark evaluator and throughput profiler will be made online upon acceptance of our paper.

REFERENCES

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/01386bd6d8e091c2ab4c7c7de644d37b-Paper.pdf.

Rodrigo M Braga, Lauren M DiNicola, Hannah C Becker, and Randy L Buckner. Situating the left-lateralized language network in the broader organization of multiple specialized large-scale distributed networks. *Journal of neurophysiology*, 124(5):1415–1448, 2020.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

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, 2021. URL https://arxiv.org/abs/2110.14168.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Jingnan Du, Anne Billot, Wendy Sun, Gregory Hickok, Mark C Eldaief, and Randy L Buckner. Multiple nested distributed language networks in the human brain. *bioRxiv*, pp. 2026–08, 2026.

Evelina Fedorenko, Po-Jang Hsieh, Alfonso Nieto-Castañón, Susan Whitfield-Gabrieli, and Nancy Kanwisher. New method for fmri investigations of language: defining rois functionally in individual subjects. *Journal of neurophysiology*, 104(2):1177–1194, 2010.

Evelina Fedorenko, Anna A Ivanova, and Tamar I Regev. The language network as a natural kind within the broader landscape of the human brain. *Nature Reviews Neuroscience*, 25(5):289–312, 2024.

Angela D Friederici. The brain basis of language processing: from structure to function. *Physiological reviews*, 91(4):1357–1392, 2011.

Alex Graves. Adaptive computation time for recurrent neural networks, 2016.

Peter Hagoort. Muc (memory, unification, control) and beyond. *Frontiers in psychology*, 4:416, 2013.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021b. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Gregory Hickok and David Poeppel. The cortical organization of speech processing. *Nature reviews neuroscience*, 8(5):393–402, 2007.

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

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 30811–30849. Curran Associates, Inc., 2024. doi: 10.52202/079017-0970. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/370df50ccfdf8bde18f8f9c2d9151bda-Paper-Datasets_and_Benchmarks_Track.pdf.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, August 2021. ISSN 0001-0782. doi: 10.1145/3474381. URL https://doi.org/10.1145/3474381.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model, 2025. URL https://arxiv.org/abs/2506.21734.

Xi Wang, Ziyang Cai, Zheng Zhan, Harry Dong, Ying Fan, Gustavo de Rosa, Tim Pearce, and John Langford. Full-bandwidth transformer, 2026.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. Blimp: The benchmark of linguistic minimal pairs for english. *Transactions of the Association for Computational Linguistics*, 8:377–392, 07 2020. ISSN 2307-387X. doi: 10.1162/tacl_a_00321. URL https://doi.org/10.1162/tacl_a_00321.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Mingyu Xu, Kuai Yu, Liang Zhao, Shangyan Zhou, Zhean Xu, Zhengyan Zhang, Wangding Zeng, Shengding Hu, Yuqing Wang, Jingyang Yuan, Lean Wang, and Wenfeng Liang. mHC: Manifold-constrained hyper-connections. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=mDhyxu8WRb.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 42195–42214, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/b8402301e7f06bdc97a31bfaa653dc32-Paper-Conference.pdf.

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 97183–97219, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/f1e8ff97057e0c3f6bcb7018c78d4df1-Paper-Conference.pdf.

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, 2025b.

Table A.1: Exact Y execution schedule. Horizontal position in the architecture figure corresponds to the global Y index, while the specialist subscript counts executions within that specialist family.

Y	Cell	Y	Cell	Y	Cell	Y	Cell
1	P_1	9	I_3	17	P_5	25	A_{11}
2	I_1	10	A_4	18	I_5	26	A_{12}
3	A_1	11	H_1	19	A_8	27	P_7
4	P_2	12	A_5	20	H_2	28	I_7
5	I_2	13	P_4	21	A_9	29	A_{13}
6	A_2	14	I_4	22	A_{10}	30	H_3
7	A_3	15	A_6	23	P_6	31	A_{14}
8	P_3	16	A_7	24	I_6		

A Exact Architecture and Reproducibility Details

This appendix provides the execution semantics, scale configurations, training protocol, evaluator behavior, three seed results, and implementation details used in the experiments. Our aim is to make the reported comparisons reproducible and to disambiguate several aspects of the architecture that are briefly outlined in the main text.

A.1 Persistent specialist states and exact execution schedule

The native backbone maintains three persistent full width token states, Primary P, Intermediate I, and Association A. Hub cells operate on the persistent Intermediate state and therefore do not introduce a fourth persistent state. The input embedding initializes P, while I and A are initialized to zero. The TRI sentinels P_{sent} and I_{sent} are also initialized to zero. LatentBank states Z_I and Z_A are initialized to zero for each sequence.

The exact 31 cell execution order is

$$\begin{aligned} &P, I, A,\ P, I, A, A,\ P, I, A, H, A,\ P, I, A, A, \\ &P, I, A, H, A, A,\ P, I, A, A,\ P, I, A, H, A. \end{aligned} \tag{A.1}$$

This gives seven Primary cells, seven Intermediate cells, three Hub cells, and fourteen Association cells. The indexed execution schedule is shown in Table A.1.

Each cell output is retained as one element of the depth memory $\mathcal{M}_Y$. A tap labeled H_j is the updated persistent I state after the corresponding Hub operator has executed.

A.2 Primary, Intermediate, Hub, and Association operators

All operator cells use a pre normalized residual mixer followed by a pre normalized SwiGLU residual update. Primary cells replace attention with a gated causal convolution. Let x denote the normalized Primary mixer input. A bias free projection produces two width c_P branches,

$$(u, v) = W_{\text{in}} x. \tag{A.2}$$

The u branch undergoes left padded depthwise causal convolution with kernel size seven and dilation one, after which the mixer output is

$$\text{ConvMix}(x) = W_{\text{out}} \left[\text{SiLU}(\text{DWConv}_7(u)) \odot v\right]. \tag{A.3}$$

The Primary residual update is

$$P \leftarrow P + \text{ConvMix}(\mathcal{N}(P)), \tag{A.4}$$

$$P \leftarrow P + \text{SwiGLU}(\mathcal{N}(P)). \tag{A.5}$$

Primary cells therefore perform strictly causal local sequence mixing without content addressed attention.

Intermediate and Hub cells use causal grouped query attention over a contiguous window of 256 tokens. Association cells use full causal grouped query attention. For an attention cell with persistent state h and mixer input m,

$$h \leftarrow h + \text{GQA}(m), \tag{A.6}$$

$$h \leftarrow h + \text{SwiGLU}(\mathcal{N}(h)). \tag{A.7}$$

For an ordinary Intermediate cell, $h = I$ and the native mixer input is $\mathcal{N}(I)$. For an Association cell, $h = A$ and the native mixer input is $\mathcal{N}(A)$ after any pending TRI inheritance has been applied. LatentBank modulation, when active, is applied to this already constructed native mixer input and never directly to the residual state.

The canonical 220M configuration uses model width 768, SwiGLU width 2056, 12 query heads, 4 key value heads, and head dimension 64. The Primary convolution width is 672. The model supports a maximum position length of 2048, uses RoPE with base 10000, RMS normalization with $\epsilon = 10^{-5}$, zero dropout, and tied token embedding and output weights. The training sequence length is 1024.

A.3 Telescoping Residual Inheritance

Information moves between persistent specialist states through **Telescoping Residual Inheritance (TRI)**. The sentinels P_{sent} and I_{sent} record the source state at the most recent completed inheritance event. When Primary has advanced since the previous Primary handoff, the next eligible Intermediate or Hub event consumes

$$\Delta P = P - P_{\text{sent}}, \qquad I \leftarrow I + \Delta P, \qquad P_{\text{sent}} \leftarrow P. \tag{A.8}$$

After an Intermediate or Hub operator updates I, the next Association event consumes

$$\Delta I = I - I_{\text{sent}}, \qquad A \leftarrow A + \Delta I, \qquad I_{\text{sent}} \leftarrow I. \tag{A.9}$$

The implementation uses pending source update flags, so each newly accumulated source change is consumed exactly once by the first eligible downstream receiver. No detach is applied to P_{sent} or I_{sent}, and the TRI path therefore remains part of the ordinary autograd graph.

This event semantics is important for interpreting the canonical schedule. Primary inheritance occurs at $I_1, \ldots, I_7$. The Hub cells H_1, H_2, H_3 occur after an ordinary Intermediate cell has already consumed the most recent Primary change, so the canonical schedule does not perform a new ΔP handoff at those Hub events. Each Hub nevertheless updates the persistent Intermediate state, which creates a new ΔI for the next Association cell. The ten Association inheritance events occur at

$$A_1, A_2, A_4, A_5, A_6, A_8, A_9, A_{11}, A_{13}, A_{14}. \tag{A.10}$$

The consecutive Association cells A_3, A_7, A_{10}, A_{12} execute their Association operator on the persistent A state without a second TRI handoff because no new Intermediate or Hub update has occurred since the previous Association inheritance event.

TRI has two useful properties. First, a receiver obtains only the source change accumulated since its previous handoff, avoiding repeated copying of stale source state. Second, the sum of successive transmitted changes telescopes to the net source change over the corresponding interval. The mechanism adds no learned transport matrix, router, or gating parameter.

A.4 Hub feedback

Hub cells are Intermediate operators with an additional read only Association private context. Immediately before a Hub attention operation, define

$$A_{\text{private}} = A - I_{\text{sent}}. \tag{A.11}$$

The Hub attention mixer input is

$$m_H = \frac{\mathcal{N}(I) + \gamma_H \mathcal{N}(A - I_{\text{sent}})}{\sqrt{1 + \gamma_H^2}}, \qquad \gamma_H = 0.25. \tag{A.12}$$

The Association signal is read only. It modifies the attention computation and is never directly added to the Intermediate residual stream. The Hub output remains an update of the persistent I state. After the Hub executes, the next Association cell receives the resulting ΔI through TRI.

A.5 Recursive Memory Plane

After the 31 cell backbone has executed once, all cell outputs are retained as

$$\mathcal{M}_Y = \{Y_1, \ldots, Y_{31}\}. \tag{A.13}$$

The implementation stores raw Y states as retrieval values and uses RMS normalized query and key geometry for depth scoring. For a query q,

$$K_j = \mathcal{N}(Y_j), \qquad V_j = Y_j, \tag{A.14}$$

$$s_j(q) = \frac{1}{\tau d}\mathcal{N}(q)^\top K_j, \tag{A.15}$$

$$w_j(q) = \frac{\exp s_j(q)}{\sum_{\ell=1}^{31} \exp s_\ell(q)}, \tag{A.16}$$

$$\mathrm{Read}(q, \mathcal{M}_Y) = \sum_{j=1}^{31} w_j(q) V_j. \tag{A.17}$$

The canonical depth temperature is $\tau = 1$. There is one softmax across all 31 states. The depth read uses no learned query, key, or value projection, no family equalization, no family prior, no score standardization, and no normalization of the raw retrieval values.

Balanced combinations use

$$\mathcal{C}(x_1, \ldots, x_n) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \mathcal{N}(x_i). \tag{A.18}$$

Recursion begins with $D_0 = A_{14}$ and $R_0 = 0$. At recursive step z, the reasoning state first performs an independent memory read,

$$q_z^R = \mathcal{C}(D_{z-1}, R_{z-1}), \tag{A.19}$$

$$M_z^R = \mathrm{Read}(q_z^R, \mathcal{M}_Y), \tag{A.20}$$

$$R_z = \Phi_A\big(R_{z-1}; \mathcal{C}(M_z^R, D_{z-1}, R_{z-1})\big)\,, \tag{A.21}$$

then the updated reasoning state changes the discourse query,

$$q_z^D = \mathcal{C}(D_{z-1}, R_z), \tag{A.22}$$

$$M_z^D = \mathrm{Read}(q_z^D, \mathcal{M}_Y), \tag{A.23}$$

$$D_z = \Phi_A\big(D_{z-1}; \mathcal{C}(M_z^D, D_{z-1}, R_z)\big)\,. \tag{A.24}$$

Here Φ_A is the terminal global Association operator. The same terminal operator parameters are reused for the R and D updates, while the two depth memory reads are independent. LatentBank does not add a direct edge into R or D.

The full Y sheet is not recomputed at each recursive step. The base sheet always costs 31 operator evaluations and each recursive step adds one terminal Association evaluation for R and one for D,

$$C(Z) = 31 + 2Z. \tag{A.25}$$

Thus $Z = 0, 1, 3, 6$ use 31, 33, 37, and 43 operator evaluations, respectively.

A.6 LatentBank state and connectivity

LatentBank adds two compact persistent states while preserving full token resolution:

$$Z_I \in \mathbb{R}^{B\times T\times 4\times 16}, \qquad Z_A \in \mathbb{R}^{B\times T\times 4\times 24}. \tag{A.26}$$

This corresponds to 64 Intermediate latent features and 96 Association latent features per token. There is no pooling, striding, temporal dilation, token averaging, or temporal compression in either bank.

A source state is RMS normalized, projected without bias into the bank feature space, reshaped into four banks, and bounded with $\tanh$. Each bank has a separate recurrent matrix and a learned retention coefficient. For bank b,

$$u_b(x) = \tanh(W_b \mathcal{N}(x)), \tag{A.27}$$

$$\lambda_b = \sigma(\ell_b), \tag{A.28}$$

$$z_b' = \lambda_b z_b + \sqrt{1-\lambda_b^2}\,\tanh(u_b + R_b \mathcal{N}(z_b))\,. \tag{A.29}$$

Table B.1: RMP with LatentBank scale configurations. Head dimension is 64 at all scales. All scales use the same 31 cell Y schedule, Primary kernel size 7, Intermediate and Hub attention window 256, four 16 dimensional Z_I banks, four 24 dimensional Z_A banks, and $Z_{\text{train}} = 3$.

Scale	d	SwiGLU	Q heads	KV heads	Head dim.
220M	768	2056	12	4	64
400M	1024	3072	16	4	64
800M	1536	3840	24	8	64

Retention logits are initialized to zero, giving $\lambda_b = 0.5$ at initialization. Recurrent matrices are bank local and do not mix the four banks.

The update graph is asymmetric:

$$\Delta P \to Z_I, \qquad I/H \to Z_I, \qquad Z_I \to Z_A, \qquad \Delta I \to Z_A, \qquad A \to Z_A. \tag{A.30}$$

There is no $Z_A \to Z_I$ edge and no direct $P \to Z_A$ edge. In execution order, Z_I is updated when a Primary TRI handoff is consumed and again after every Intermediate or Hub operator. Z_A is updated when an Intermediate TRI handoff is consumed, jointly with the current one way $Z_I \to Z_A$ drive, and again after every Association operator.

Latent readout is applied only after the native Intermediate, Hub, or Association attention mixer input has been constructed. A bias free readout produces (α, β) and applies

$$x' = x \odot (1 + \varepsilon_F \tanh \alpha(Z)) + \varepsilon_F \beta(Z), \qquad \varepsilon_F = 10^{-2}. \tag{A.31}$$

The readout weights are initialized to normal, so the LatentBank paths are active from the first forward pass. The small fixed ε_F keeps the initial perturbation close to the native backbone while allowing every latent path to receive gradient immediately. Primary cells are not directly modulated by LatentBank, and latent features are never directly added to the native residual states.

At 220M scale, the native Recursive Memory Plane contains 220,091,808 parameters. LatentBank contributes 496,392 additional parameters, giving 220,588,200 parameters in total, an increase of 0.2255%. We also expose an exact same checkpoint control in which LatentBank is disabled and the forward pass delegates to the native backbone using the same native weights.

B Training, Hardware, and Evaluation Protocol

B.1 Model scale configurations

The specialist schedule, TRI semantics, local attention window, Hub feedback rule, LatentBank dimensions, and recursive training depth are unchanged across scales. Width, feedforward size, and attention head counts are scaled as shown in Table B.1.

The matched comparison models are scaled using the same width and attention geometry at each capacity. At the 220M scale, the Decoder, TRM, and mHC contain 219,659,520, 221,430,528, and 220,089,879 parameters, respectively. Their configurations use model width 768, feedforward width 2048, 12 query heads, 4 key value heads, head dimension 64, and 31 base layers.

At the 400M scale, the baseline configurations use model width 1024, feedforward width 3072, 16 query heads, 4 key value heads, head dimension 64, and 31 base layers. The resulting parameter counts are 406,649,856 for the Decoder, 409,797,632 for TRM, and 407,223,575 for mHC.

At the 800M scale, the baseline configurations use model width 1536, feedforward width 3840, 24 query heads, 8 key value heads, head dimension 64, and 31 base layers. The resulting parameter counts are 792,820,224 for the Decoder, 799,901,184 for TRM, and 793,680,663 for mHC.

Across all three scales, the baselines use maximum position length 2048, RoPE base 10000, RMS normalization with $\epsilon = 10^{-5}$, zero dropout, and tied token embedding and output weights. The TRM configuration retains two reasoning layers, one training reasoning step, and zero detached prefix steps at every scale. The mHC configuration retains two substreams, six Sinkhorn iterations, dynamic mHC, and initial $\alpha = 0.01$ at every scale.

Table B.2: Scale specific training schedule. The reduction from six to three sequences per GPU at 800M is deliberately aimed for memory management. Doubling accumulation from 11 to 22 preserves the 400M effective global batch and tokens per optimizer update.

Scale	Hardware	Batch/GPU	Accum.	Global batch	Seq. len.	Tokens/update	Updates	Processed tokens
220M	4×V100 16 GB	3	11	132	1024	135,168	73,982	9,999,998,976
400M	4×V100 32 GB	6	11	264	1024	270,336	36,991	9,999,998,976
800M	4×V100 32 GB	3	22	264	1024	270,336	36,991	9,999,998,976

B.2 Token budget, batch construction, and hardware

All reported scale experiments process approximately 10B training tokens per run. Sequence length is 1024 throughout. Table B.2 gives the exact batch construction and optimizer update counts.

The 800M microbatch change was made to fit the larger model within 32 GB device memory. The effective batch remains identical to the 400M run. The 220M experiments use 16 GB V100 cards and a smaller effective global batch of 132 sequences. Consequently, comparisons among architectures within a given scale are fully matched in batch construction, hardware class, sequence length, token budget, and optimizer schedule. The 400M to 800M scaling comparison also preserves the effective global batch exactly. The 220M to 400M scaling comparison changes effective global batch from 132 to 264, so scale to scale gains across that boundary should be interpreted as the result of the complete scale specific training configuration rather than a pure model width intervention.

All models use AdamW with peak learning rate 3×10^{-4}, minimum learning rate 3×10^{-5}, weight decay 0.1, $\beta_1 = 0.9$, $\beta_2 = 0.95$, and gradient clipping at 1.0. Warmup is approximately 5% of training: 3,700 updates at 220M and 1,850 updates at 400M and 800M. RMP with LatentBank is trained with $Z_{\text{train}} = 3$. Cross entropy is applied only to the final prediction state D_3. There are no auxiliary losses at D_0, D_1, or D_2.

Given our available computational resources as a small academic lab, extending training beyond this budget while retaining three seed replication and the full ablation suite was not feasible. All compared architectures use the same FineWeb Edu sample and tokenizer protocol.

B.3 Random seeds and aggregation

All principal comparisons use three independent training seeds, 42, 123, and 456. Evaluation is deterministic conditional on a checkpoint because dropout is zero and generation uses greedy decoding. For each task we report the arithmetic mean across the three independently trained checkpoints. The final "Mean" column is the unweighted arithmetic mean of the nine task accuracies.

We report seed level values and sample standard deviations. The paired ordering against the strongest matched baseline is nevertheless consistent across the three seeds at each reported scale, as shown in Section C.

B.4 Fixed benchmark evaluator

Every architecture, scale, seed, recursion setting, and ablation is evaluated with the same evaluator implementation and task definitions. Full evaluations use all available examples within each task. The task splits are BLiMP held configurations, BoolQ validation, WinoGrande XL validation, ARC Challenge validation, OpenBookQA validation, MMLU all test, GSM8K test, MATH 500 test, and all 27 BIG Bench Hard configurations. The corresponding example counts are 14,000 for BLiMP, 3,270 for BoolQ, 1,267 for WinoGrande, 299 for ARC Challenge, 500 for OpenBookQA, 14,042 for MMLU, 1,319 for GSM8K, 500 for MATH, and 6,511 for BBH.

The evaluator is zero shot. BLiMP compares conditional log probability of the grammatical and ungrammatical sentences. BoolQ, ARC Challenge, OpenBookQA, and MMLU score each answer option by the sum of conditional token log probabilities and select the largest score. WinoGrande inserts each candidate into the sentence and selects the completion with the larger conditional log probability. The multiple choice score is the sum of completion token log probabilities and is not length normalized. This exact rule is shared by all models.

Table C.1: Seed level aggregate accuracy. Values are the unweighted mean of the nine task accuracies for each independently trained checkpoint. The final column reports mean ± sample standard deviation across seeds.

Scale	Model	Seed 42	Seed 123	Seed 456	Mean ± std
220M	Transformer	31.78	32.00	31.83	31.87 ± 0.12
	TRM	33.01	32.97	32.93	32.97 ± 0.04
	mHC	33.79	33.86	33.97	33.87 ± 0.09
	RMP + LatentBank	34.69	34.60	34.68	34.66 ± 0.05
400M	Transformer	34.18	34.79	33.95	34.31 ± 0.43
	TRM	35.40	36.39	35.79	35.86 ± 0.50
	mHC	35.97	37.17	35.96	36.36 ± 0.70
	RMP + LatentBank	37.33	38.33	37.77	37.81 ± 0.51
800M	Transformer	36.34	36.87	36.11	36.44 ± 0.39
	TRM	38.03	38.44	37.63	38.03 ± 0.40
	mHC	38.80	39.32	38.46	38.86 ± 0.43
	RMP + LatentBank	39.77	40.35	39.44	39.85 ± 0.46

Table C.2: 220M accuracy in percent, mean ± sample standard deviation across three training seeds.

Model	BLiMP	BoolQ	Wino	ARC	OBQA	MMLU	GSM8K	MATH	BBH	Mean
Transformer	74.52 ± 0.36	47.31 ± 1.03	51.07 ± 0.55	19.40 ± 0.83	14.40 ± 0.35	25.52 ± 0.27	0.99 ± 0.33	0.00	53.62 ± 1.00	31.87 ± 0.12
TRM	72.19 ± 0.64	50.12 ± 0.92	51.54 ± 0.32	23.41 ± 0.69	14.40 ± 0.69	25.14 ± 0.19	1.44 ± 0.46	0.00	58.49 ± 0.84	32.97 ± 0.04
mHC	72.81 ± 0.53	57.77 ± 1.24	51.54 ± 0.44	21.74 ± 0.52	16.20 ± 0.53	25.67 ± 0.23	0.60 ± 0.26	0.00	58.53 ± 0.63	33.87 ± 0.09
RMP + LatentBank	76.59 ± 0.42	59.57 ± 0.86	51.62 ± 0.38	21.74 ± 0.47	15.80 ± 0.53	25.85 ± 0.19	0.99 ± 0.33	0.00	59.75 ± 0.73	34.66 ± 0.05

GSM8K, MATH, and BBH use greedy generation with at most 128 new tokens. GSM8K extracts the final numeric answer, first preferring the conventional "####" form and otherwise using the final number in the completion. MATH uses normalized exact matching of a generated $\backslash\mathrm{boxed}\{\cdot\}$ answer against the normalized reference box. BBH counts a prediction as correct when the normalized target string occurs in the normalized generated completion. These evaluation choices define the raw accuracies in this paper and should be kept in mind when comparing them with scores from external evaluation harnesses that use different prompts, normalization, few shot examples, or answer extraction.

B.5 Inference throughput protocol

The throughput experiment in the main paper measures full sequence forward inference contrary to autoregressive decoding. All 220M models are profiled on one V100 16 GB GPU with batch size one, sequence length 1024, FP32 computation, gradients disabled, and compilation disabled. Each model receives five warmup forwards followed by three repetitions of 20 timed forwards. CUDA is synchronized immediately before and after each timed forward. Checkpoint loading is excluded from timing. We report the median over the resulting 60 forward measurements. The reported token rate is therefore 1024 divided by the median wall clock time per forward. Peak memory is peak allocated GPU memory during the profile. No KV cache behavior is measured by this protocol.

C Scaling and Reproducibility Results

C.1 Seed level aggregate results

Table C.1 reports the unweighted nine task mean for every training seed. The RMP with LatentBank row uses $Z_{\mathrm{train}} = 3$ and $Z_{\mathrm{infer}} = 1$ at all three scales.

Seed consistency. Figure C.1 shows the individual training runs. Across three scales and three matched seeds, RMP with LatentBank obtains the highest aggregate mean in all nine scale and seed comparisons. The average paired margin over the strongest baseline is 0.78, 1.45, and 0.99 points at 220M, 400M, and 800M, respectively.

Task specific scaling. Figure C.2 shows that the aggregate scaling gain does not arise from a uniform improvement across benchmarks. RMP with LatentBank remains particularly strong on

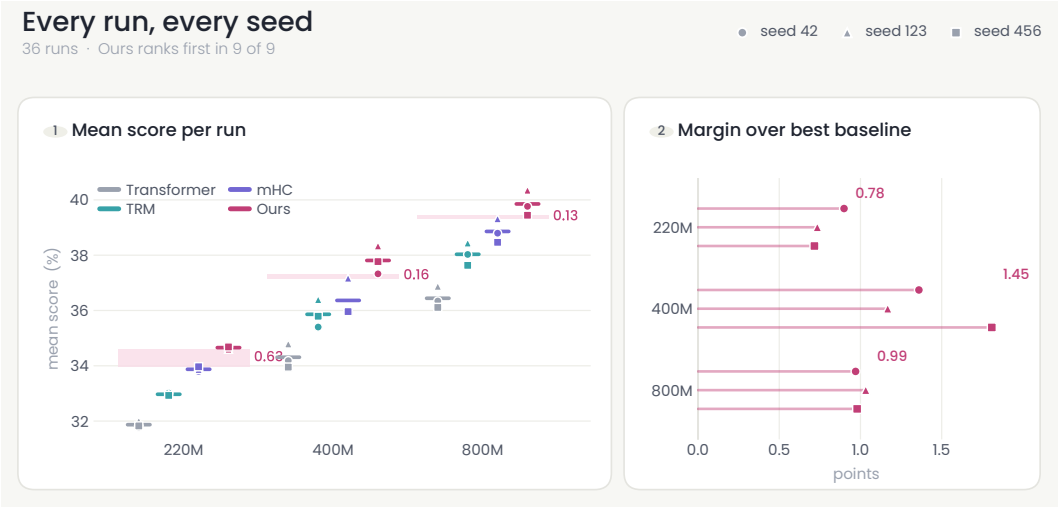


Figure C.1: **Consistency across independent training seeds.** Left: aggregate nine benchmark mean for every independently trained checkpoint across the three model scales and three seeds. Each scale contains twelve runs, comprising four model families and three seeds. RMP with LatentBank ranks first in all nine matched scale and seed comparisons. The annotated separation at each scale measures the gap between the lowest scoring RMP run and the highest scoring competing run at that scale. Right: paired RMP margin over the strongest matched baseline for each seed; the displayed values are the mean paired margins at each scale.

Table C.3: 400M accuracy in percent, mean ± sample standard deviation across three training seeds.

Model	BLiMP	BoolQ	Wino	ARC	OBQA	MMLU	GSM8K	MATH	BBH	Mean
Transformer	74.67 ± 1.54	47.10 ± 0.65	52.67 ± 0.40	22.56 ± 0.60	17.51 ± 0.32	28.44 ± 0.52	1.64 ± 0.02	0.00	64.20 ± 1.06	34.31 ± 0.43
TRM	71.29 ± 1.19	52.56 ± 0.58	51.90 ± 0.37	28.65 ± 0.58	16.72 ± 0.45	27.35 ± 0.73	1.59 ± 0.04	0.00	72.70 ± 1.66	35.86 ± 0.50
mHC	72.99 ± 1.59	57.93 ± 1.03	51.40 ± 1.25	27.03 ± 0.21	18.89 ± 0.33	27.90 ± 0.85	1.00 ± 0.02	0.00	70.15 ± 1.30	36.36 ± 0.70
RMP + LatentBank	77.07 ± 1.19	63.01 ± 0.77	51.77 ± 1.33	25.94 ± 0.11	19.24 ± 0.38	27.91 ± 0.73	1.07 ± 0.02	0.00	74.26 ± 0.53	37.81 ± 0.51

BLiMP, BoolQ, and BBH, while ARC and GSM8K favor competing models at larger scales. At 800M, the RMP margin over the strongest baseline is $+2.63$ points on BLiMP, $+3.08$ on BoolQ, and $+1.18$ on BBH, while the corresponding margins are -2.99 on ARC and -0.77 on GSM8K. These heterogeneous trajectories motivate reporting every benchmark separately alongside the unweighted aggregate.

Figure C.3 separates improvement over the standard Transformer from improvement over the strongest task specific baseline. Across the 24 displayed benchmark and scale comparisons against the strongest baseline, RMP records 13 positive differences, one tie, and 10 negative differences. Its most persistent advantages occur on BLiMP, BoolQ, and BBH. The matrix also shows that the aggregate improvement does not imply dominance on every task: TRM remains stronger on ARC and GSM8K at several scales, and the leading baseline can vary with both task and model size.

C.2 Scaling behavior

RMP with LatentBank improves from 34.66 at 220M to 37.81 at 400M and 39.85 at 800M. Its gain from 220M to 800M is 5.20 points. The corresponding total gains are 4.57 points for the Transformer, 5.06 points for TRM, and 4.99 points for mHC. The aggregate gap between RMP with LatentBank and mHC is 0.79 points at 220M, 1.45 points at 400M, and 0.99 points at 800M when computed from the displayed aggregate means. The advantage observed at 220M therefore remains positive at both larger scales.

Because the seed identifiers are matched across model families, we can also inspect paired seed differences in the aggregate mean. RMP with LatentBank exceeds mHC by 0.78 ± 0.10, 1.45 ± 0.33, and 0.99 ± 0.03 points at 220M, 400M, and 800M, respectively, where the uncertainty is the

Table C.4: 800M accuracy in percent, mean ± sample standard deviation across three training seeds.

Model	BLiMP	BoolQ	Wino	ARC	OBQA	MMLU	GSM8K	MATH	BBH	Mean
Transformer	75.39 ± 1.10	48.88 ± 0.82	52.40 ± 0.93	27.16 ± 0.65	19.98 ± 0.26	30.02 ± 0.35	1.87 ± 0.17	0.00	72.27 ± 0.91	36.44 ± 0.39
TRM	72.89 ± 0.36	52.90 ± 1.64	52.32 ± 1.44	33.13 ± 0.11	19.68 ± 0.49	29.54 ± 0.15	2.43 ± 0.21	0.00	79.40 ± 0.20	38.03 ± 0.40
mHC	73.63 ± 0.53	60.28 ± 1.66	52.75 ± 1.10	30.90 ± 0.12	22.00 ± 0.66	30.21 ± 0.32	1.14 ± 0.13	0.00	78.82 ± 0.58	38.86 ± 0.43
RMP + LatentBank	78.02 ± 1.11	63.35 ± 0.76	52.76 ± 0.71	30.14 ± 0.78	21.87 ± 0.20	30.30 ± 0.60	1.66 ± 0.17	0.00	80.58 ± 0.55	39.85 ± 0.46

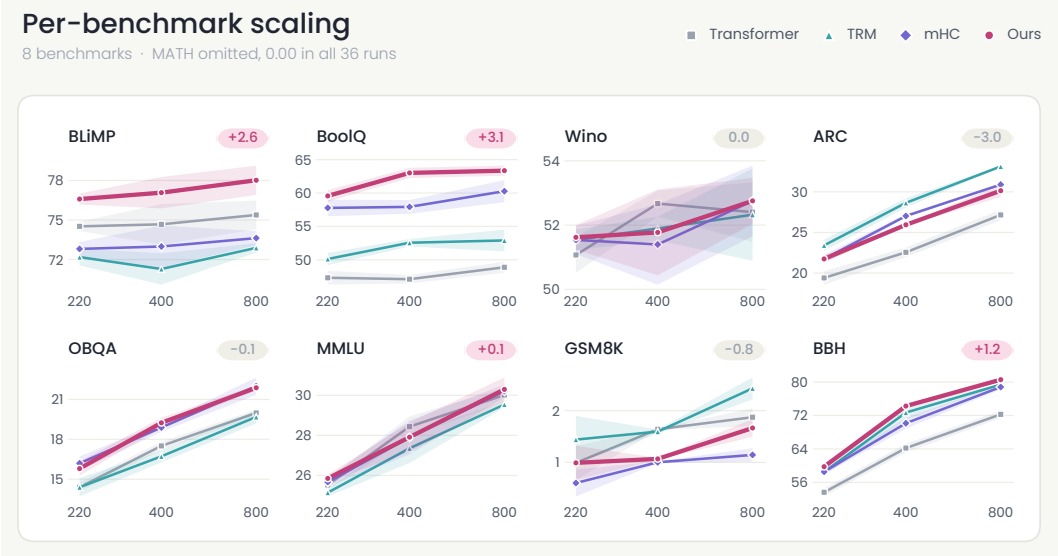


Figure C.2: **Per benchmark scaling across model size.** Mean performance across three training seeds for Transformer, TRM, mHC, and RMP with LatentBank at approximately 220M, 400M, and 800M parameters. Shaded regions visualize variation across the three independently trained checkpoints. The annotation in each panel reports the RMP margin over the strongest baseline at 800M. MATH is omitted because all four model families obtain 0.00 under the fixed evaluator at every tested scale.

sample standard deviation of the three paired seed differences. Every one of the nine scale by seed comparisons has a positive RMP minus mHC difference.

The per task scaling trajectory is heterogeneous. RMP with LatentBank rises from 59.75 to 74.26 to 80.58 on BBH, from 21.74 to 25.94 to 30.14 on ARC Challenge, and from 25.85 to 27.91 to 30.30 on MMLU. MATH remains at zero under the fixed evaluator for all compared models and scales. The unweighted mean therefore summarizes a set of task trajectories with different scales and difficulty levels; individual task values remain the primary measurements.

D Ablation Semantics and Reproducibility Controls

The main paper reports runtime and retraining controls. For completeness, the exact runtime switches are:

- **LatentBank off**: disables the complete latent mechanism and delegates to the exact native RMP backbone with the same native checkpoint weights.
- **Remove** Z_I: disables Z_I updates and reads and also removes the $Z_I \to Z_A$ cross path.
- **Remove** Z_A: disables Z_A updates and reads while leaving Z_I active.
- **Remove** $Z_I \to Z_A$: removes only the one way cross bank projection.
- **Additive FiLM only**: removes the multiplicative scale term and retains the additive β term.
- **Multiplicative FiLM only**: removes the additive β term and retains the multiplicative scale term.

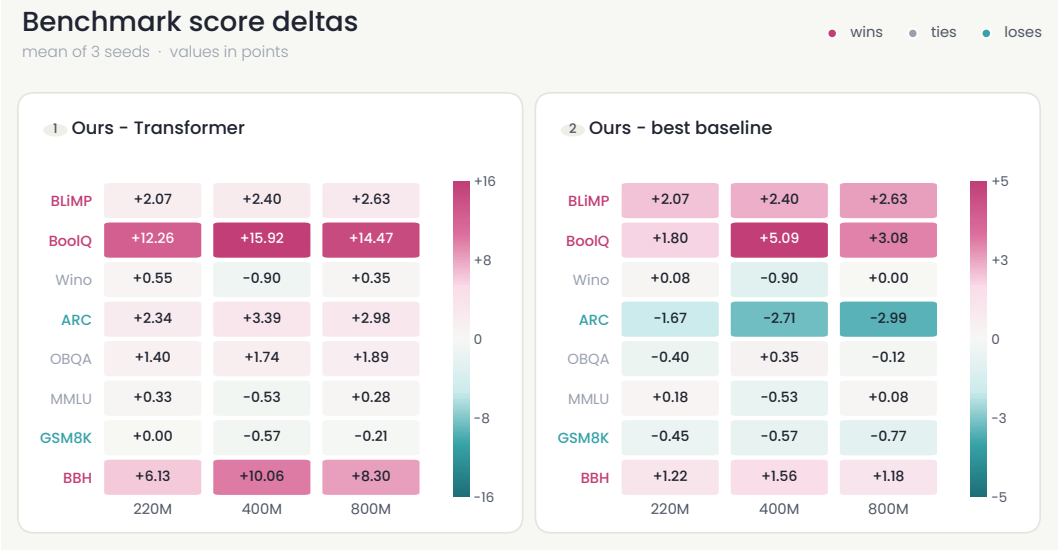


Figure C.3: **Benchmark level performance differences.** Differences are computed from three seed mean accuracies and reported in percentage points. Left: RMP with LatentBank minus the matched Transformer for each benchmark and model scale. Right: RMP with LatentBank minus the strongest external baseline for that benchmark and scale, where the reference baseline may therefore differ across cells. Positive values indicate an RMP advantage, negative values indicate a baseline advantage, and zero indicates a tie. MATH is omitted because all compared models obtain zero throughout.

Table C.5: Unweighted nine task mean across model scale. All entries are averages across three training seeds.

Model	220M	400M	800M	$\Delta_{400-220}$	$\Delta_{800-400}$
Transformer	31.87	34.31	36.44	+2.44	+2.13
TRM	32.97	35.86	38.03	+2.89	+2.17
mHC	33.87	36.36	38.86	+2.49	+2.50
RMP + LatentBank	**34.66**	**37.81**	**39.85**	+3.15	+2.04

- **Remove bank recurrence**: sets the recurrent $R_b\mathcal{N}(z_b)$ contribution to zero while retaining source drives and learned retention.

These are runtime interventions on a trained checkpoint and therefore answer a different question from retraining controls. The Z_I only training experiment removes Association latent capacity throughout optimization, while the runtime Z_A removal tests immediate dependence of an already trained representation on that state. The large difference between these two interventions is evidence that a component can be important during representation formation even when its immediate contribution at shallow inference is small.

The training depth controls change Z_{train} while keeping the $Z = 0$ inference point fixed. Since the language modeling loss is applied only at the final training discourse state, the improvement of D_0 after deeper recursive training cannot be attributed to auxiliary supervision of the shallow state. This is the basis for the term *recursive nested representation shaping* used in the main paper.

E Recursive Depth Read Interventions and Diagnostics

All experiments in this section are same checkpoint inference interventions on the trained 220M RMP with LatentBank model. They do not retrain the architecture. All 31 backbone cells, TRI updates, LatentBank states, the initialization $D_0 = Y_{31}$ and $R_0 = 0$, the recursive R and D updates, the reused terminal Association operator, and the requested recursion depth remain unchanged. Only the representation supplied by the recursive depth read is modified.

Table E.1: Same checkpoint recursive depth read interventions. All parameters and recursive update operators are fixed. "Zero" removes the retrieved depth vector while retaining the recursive computation. Full trajectory values are the matched unrestricted checkpoint references at the displayed precision.

Read source	Z	Mean (%)
No recursion	0	34.52
Zero depth retrieval	1	34.53
Association only	1	34.63
Intermediate/Hub only	1	34.68
Primary only	1	34.68
Full 31 state trajectory	1	34.66
Terminal Y_{31} only	1	34.74
Full 31 state trajectory	3	34.24
Terminal Y_{31} only	3	34.30

Table E.2: Per task accuracy (%) for same checkpoint recursive read interventions.

Condition	BLiMP	BoolQ	Wino	ARC	OBQA	MMLU	GSM8K	MATH	BBH	Mean
Y_{31} only, $Z = 1$	76.65	59.97	51.38	22.41	16.00	25.82	1.06	0.00	59.39	34.7419
Primary only, $Z = 1$	76.75	59.57	51.30	22.07	15.80	25.87	1.06	0.00	59.68	34.6787
Intermediate/Hub only, $Z = 1$	76.54	59.69	51.46	22.07	15.80	25.79	1.06	0.00	59.65	34.6755
Association only, $Z = 1$	76.58	59.57	51.54	21.74	15.60	25.90	0.99	0.00	59.71	34.6255
Zero retrieval, $Z = 1$	76.39	59.51	51.54	21.40	15.40	25.81	1.06	0.00	59.68	34.5334
Y_{31} only, $Z = 3$	75.81	59.51	51.38	21.07	15.20	25.74	1.06	0.00	58.92	34.2997

For the family restricted controls, the canonical 31 depth logits are computed normally. Weights outside the selected family are then set to zero and the remaining weights are renormalized. For fixed logits this is equivalent to applying the softmax only over the selected family. The Primary set contains seven states,

$$\{Y_1, Y_4, Y_8, Y_{13}, Y_{17}, Y_{23}, Y_{27}\},$$

the Intermediate/Hub set contains ten states,

$$\{Y_2, Y_5, Y_9, Y_{11}, Y_{14}, Y_{18}, Y_{20}, Y_{24}, Y_{28}, Y_{30}\},$$

and the Association set contains fourteen states,

$$\{Y_3, Y_6, Y_7, Y_{10}, Y_{12}, Y_{15}, Y_{16}, Y_{19}, Y_{21}, Y_{22}, Y_{25}, Y_{26}, Y_{29}, Y_{31}\}.$$

The terminal state intervention supplies Y_{31} directly to both recursive reads. The zero retrieval control sets both retrieved depth vectors to zero.

Restricted depth access. Table E.1 summarizes the aggregate same checkpoint results. At $Z = 1$, the unrestricted reader gives 34.66%. Restricting both recursive reads to Primary, Intermediate/Hub, or Association states gives 34.68%, 34.68%, and 34.63%, respectively. Restricting both reads to the single terminal Association state Y_{31} gives 34.74%. The same terminal restriction at $Z = 3$ gives 34.30%, compared with 34.24% for the unrestricted $Z = 3$ path.

In contrast, removing the retrieved memory signal entirely gives 34.53% at $Z = 1$, essentially matching the $Z = 0$ value of 34.52%. Thus, simply executing the recursive terminal Association updates does not reproduce the full $Z = 1$ gain. A retrieved representation contributes to recursive inference, while the information sufficient for the aggregate gain is recoverable from multiple portions of the trained trajectory.

Table E.2 reports the individual task values for the new interventions. The small aggregate differences are not caused by identical predictions: the restricted readers produce measurably different logits, as examined below.

Learned depth allocation. We also inspect the unrestricted learned depth reads at $Z = 3$ over 1,048,576 token positions. Table E.3 reports aggregate mass by specialist family for each of the three reasoning reads and three discourse reads. Association states receive approximately 50.6–53.4% of aggregate read mass, Primary states receive 17.2–19.1%, and Intermediate/Hub states receive 29.4–30.4%. The individual terminal state Y_{31} receives only 4.6–5.4%.

Table E.3: Aggregate learned read mass (%) by specialist family for the unrestricted 220M checkpoint at $Z = 3$. Y_{31} is included within the Association column and is also shown separately.

Read	Primary	I/H	Association	Y_{31}
R_1	17.18	29.42	53.39	5.38
D_1	18.94	30.29	50.76	4.74
R_2	18.95	30.25	50.80	4.78
D_2	19.05	30.38	50.57	4.56
R_3	19.07	30.35	50.57	4.60
D_3	19.01	30.36	50.63	4.62

Table E.4: Numerical validation of family restricted reads on a fixed 128 token sequence. Logit differences are measured against the unrestricted reader.

Read source	Mean $\lvert\Delta\rvert$	Max $\lvert\Delta\rvert$	Argmax agreement
Primary only	0.05958	0.67593	96.88%
Intermediate/Hub only	0.01068	0.15127	100.00%
Association only	0.00641	0.05198	99.22%

For comparison, a uniform 31 state distribution would assign 22.58% to the seven Primary states, 32.26% to the ten Intermediate/Hub states, 45.16% to the fourteen Association states, and 3.23% to Y_{31} individually. The learned unrestricted read is therefore distributed across the trajectory rather than concentrated on the terminal state. These quantities describe the distribution after aggregation across token positions; they should not be interpreted as per token entropy measurements.

Intervention validation. Because the similar benchmark means could otherwise suggest an inactive mask, we directly verify the family restricted implementation. For both the reasoning and discourse reads, every prohibited state receives exactly zero weight, while the allowed family weights sum to one within 1.192×10^{-7}. The resulting logits are also different from the unrestricted computation.

Table E.4 reports logit differences on a fixed 128 token diagnostic sequence. Primary only produces the largest change from the unrestricted reader, with mean absolute logit difference 0.0596 and maximum difference 0.676, while preserving the same argmax token at 96.88% of positions. Intermediate/Hub and Association restrictions alter the logits less strongly. Thus the similar downstream accuracies do not arise because the three interventions execute the same computation.

Pairwise family comparisons are also nonzero. Primary versus Intermediate/Hub gives mean absolute logit difference 0.06218 and maximum difference 0.63213 with 96.88% argmax agreement; Primary versus Association gives 0.05521, 0.68859, and 96.09%; Intermediate/Hub versus Association gives 0.01610, 0.20325, and 99.22%, respectively.

Terminal state inference efficiency. We additionally profile the terminal state intervention under the same V100 full sequence protocol used in the main paper. At $Z = 1$, replacing the full depth read with Y_{31} reduces median forward latency from 69.212 ms to 67.015 ms and increases throughput from 14.795k to 15.280k tokens/s. Peak allocated memory changes from 1.16 to 1.15 GiB. This corresponds to a 3.17% latency reduction and a 3.28% throughput increase in the measured implementation.

The current terminal intervention still prepares the depth memory before bypassing its recursive read, so this result measures the evaluated terminal state path rather than an optimized terminal only implementation.

For completeness, the family restricted diagnostic paths measure 70.490 ms and 14.527k tokens/s for Primary only, 70.800 ms and 14.463k tokens/s for Intermediate/Hub only, and 70.932 ms and 14.436k tokens/s for Association only, with 1.16 GiB peak allocated memory in each case. These measurements should not be interpreted as the cost of optimized family specific execution: the diagnostic implementation first computes the complete 31 state depth scores and softmax and only then masks and renormalizes the selected family. It therefore removes information access without pruning the underlying depth read computation.

Table E.5: External-corpus language modeling performance at 220M scale. NLL is token-weighted next-token cross-entropy in nats per token and PPL is $\exp(\mathrm{NLL})$. Lower is better.

Condition	Z	NLL	PPL
Decoder	0	3.8328	46.19
TRM	1	3.8886	48.84
mHC	0	3.8009	44.74
RMP full	0	4.0619	58.08
RMP full	1	3.9285	50.83
RMP full	3	**3.7951**	**44.48**
RMP Y_{31} only	1	3.9461	51.73
RMP Y_{31} only	3	3.8266	45.91
RMP Primary only	1	3.9220	50.50
RMP Intermediate/Hub only	1	3.9261	50.71
RMP Association only	1	3.9353	51.18
RMP zero retrieval	1	3.9769	53.35

Interpretation. The intervention results distinguish retrieval necessity from source specific necessity. Removing the retrieved vector returns $Z = 1$ performance almost exactly to the $Z = 0$ level, indicating that recursive reprocessing of D_0 alone does not explain the $Z = 1$ gain. At the same time, Primary, Intermediate/Hub, Association, and terminal state restrictions each preserve nearly the unrestricted aggregate score. Together with the distributed learned read allocation, this pattern is consistent with substantial redundancy or consolidation of task relevant information across the trained depth trajectory.

All restricted conditions use a checkpoint optimized with access to the complete trajectory. These experiments therefore measure inference dependence of an already trained model. They do not establish that the full depth memory was unnecessary during training, nor do they replace a separately trained terminal only or family restricted recursive control.

Held out language modeling. To complement the downstream benchmark suite, we evaluate next-token likelihood on a fixed external WikiText-103 (Merity et al., 2017) token stream using the same tokenizer and 1024-token context length for every model and intervention. We report token-weighted cross-entropy in nats per token and perplexity.

Language modeling performance improves monotonically with recursive depth for the trained RMP checkpoint. The $Z = 0$, $Z = 1$, and $Z = 3$ configurations obtain NLLs of 4.0619, 3.9285, and 3.7951, corresponding to perplexities of 58.08, 50.83, and 44.48. Thus, although the nine-task downstream aggregate is highest at $Z = 1$, held out next-token likelihood continues to improve to the training depth $Z = 3$.

At $Z = 3$, RMP obtains the lowest measured NLL among the compared 220M models: 3.7951 versus 3.8009 for mHC, 3.8328 for the Decoder, and 3.8886 for TRM. Restricting the $Z = 3$ RMP reads to the terminal state Y_{31} increases NLL to 3.8266. At $Z = 1$, removing recursive retrieval entirely raises NLL from 3.9285 to 3.9769, while family-restricted reads remain close to the unrestricted result. These results indicate that recursive memory retrieval contributes to held out language modeling likelihood, while information available to a single recursive step remains substantially redundant across specialist families.

F Comparability, Scope, and Clarifications

Parameter matching. At 220M scale, all compared models lie within a narrow parameter range around the target capacity. RMP with LatentBank has 220.59M parameters, mHC has 220.09M, TRM has 221.43M, and the Decoder has 219.66M. The LatentBank addition itself is 496,392 parameters, or 0.2255% relative to the native RMP backbone. Results at 400M and 800M use matched scale configurations and identical within scale training budgets.

Compute matching. The parameter matched comparison does not imply equal operator structure or equal wall clock cost. The main paper therefore reports both operator counts for RMP and measured 220M full sequence throughput. RMP at $Z = 0$ uses 31 operator evaluations and RMP at

$Z = 1$ uses 33. Throughput measurements are hardware specific and are reported separately from benchmark accuracy.

Training recursion and inference recursion. Training depth and inference depth are distinct controls. The principal RMP checkpoints are optimized with $Z_{\text{train}} = 3$ and one loss at D_3. At evaluation, the same checkpoint can be executed at $Z = 0, 1, 2, 3, 4, 6$. The best aggregate 220M operating point is $Z = 1$, while $Z = 0$ retains most of the aggregate score and provides greater measured throughput. No task specific routing is used.

Depth memory cost. RMP retains all 31 Y states until recursive computation has finished. Activation memory for the retained trajectory therefore grows linearly with batch size, sequence length, hidden width, and retained depth. The 800M microbatch reduction is a direct systems consequence of the larger model and activation footprint. Gradient accumulation is used to preserve the desired effective batch without changing the model computation.

Evaluation scope. The nine task mean is an unweighted descriptive aggregate and is not a standardized benchmark score. The tasks differ substantially in dataset size, chance level, answer format, and sensitivity to prompting. We therefore report every task separately and provide the exact evaluator behavior in Appendix B.4. Claims in the paper concern comparisons under this fixed evaluator rather than direct equivalence to numbers reported by other evaluation harnesses.

Variance and statistical scope. We had limited computing budget as a small academic group. Three seeds provide a direct check that the principal ordering is reproducible across independent training runs, and we admit three observations per model are insufficient for precise tail probability estimates. We report sample standard deviations and paired seed differences and avoid strong claims based solely on null hypothesis significance testing.

Scaling interpretation. The tested scaling range is 220M to 800M parameters with a fixed 10B processed token budget. The 400M and 800M runs share the same effective global batch of 264 sequences, while the 220M runs use 132. The scaling curves therefore demonstrate persistence of the architecture comparison across three practical training settings, while the 220M to 400M difference also includes the scale specific batch change. Larger parameter counts, longer training budgets, and other data configurations remain open.

Biological interpretation. The P, I, and A hierarchy is an engineering abstraction motivated by evidence for differentiated and nested language systems. The architecture does not claim anatomical correspondence, biological learning rules, or biological fidelity. The neuroscience literature motivates the organizational prior of interacting specialist systems, while all computational mechanisms in RMP with LatentBank are evaluated as machine learning design choices.

Causality. Primary convolution uses left only padding, Intermediate and Hub attention is causal within its local window, Association attention is globally causal, and the depth memory is constructed only from same token causal residual states. RMP depth reads mix depth for a fixed token position and therefore do not expose future token positions.