ATTENTION ROUTING STABILIZES EARLY: WORKING-SET INFERENCE FOR RECURRENT-DEPTH LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

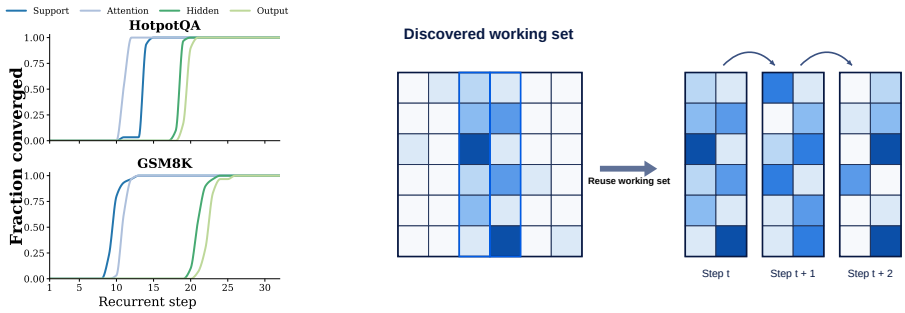
Recurrent-depth language models repeatedly apply shared network blocks to refine latent representations, enabling additional test-time computation without generating explicit intermediate reasoning tokens. However, each recurrent step recomputes full self-attention over the entire context, repeating global attention routing whose cost is quadratic in context length. We study how attention routing evolves across recurrent depth and uncover a consistent separation in convergence timescales: routing-related quantities, including the attention support and the attention distribution, stabilize substantially earlier than representation-related quantities such as hidden states and attention outputs. This suggests a two-stage structure in recurrent inference, where the model first discovers a sparse working set of relevant context and then continues refining representations over largely the same routing support. Motivated by this structure, we introduce `WISE` (**W**orking-set **I**nference with **S**upport **E**xploitation), a training-free method that uses unrestricted global attention during early recurrent steps to discover a block-structured working set, and then reuses it as the routing support in later steps while keeping recurrent refinement and within-support attention computation dynamic. Controlled interventions show that discovering the working set over multiple recurrent steps, rather than from the first step alone, yields more effective working sets, and that reusing only the routing support better preserves model behavior than more restrictive forms of late attention reuse. Across multi-hop QA benchmarks, `WISE` largely preserves the behavior of full-attention inference, while matched context-scaling experiments reveal an increasingly favorable quality–efficiency tradeoff as the routing support becomes increasingly sparse with longer contexts. A sparse-attention implementation translates this structured sparsity into practical GPU acceleration, achieving up to a $1.76\times$ late-step attention speedup over native FlashAttention at 4K context.

1 INTRODUCTION

Recurrent-depth language models provide a new axis for test-time computation by repeatedly applying shared network blocks to refine latent representations (Geiping et al., 2025; Dehghani et al., 2018; Yang et al., 2024; Giannou et al., 2023; Rodkin et al., 2026), rather than generating explicit intermediate reasoning tokens (Kojima et al., 2022). This decouples parameter count from inference-time compute, allowing additional computation to be allocated through recurrent depth rather than additional parameters (Dehghani et al., 2018; Graves, 2016). Prior work has shown that additional recurrent computation can improve multi-step reasoning (Geiping et al., 2025). However, existing recurrent inference recomputes full self-attention over the entire context at every recurrent step, repeating global attention routing (i.e., exchanging information across all token positions), whose cost is quadratic in context length. It remains unclear whether later steps still need to search the full context, or whether they mainly continue refining representations over context that earlier steps have already identified as relevant. Rather than asking how much recurrent computation to use, we take this budget as given and ask: *must every recurrent step continue to perform unrestricted global attention routing?*

To answer this question, we study how attention routing evolves across recurrent depth and uncover a consistent separation in convergence timescales: routing-related quantities, which describe *where*

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.



(a) Convergence timescales **(b) Working-set reuse**

Figure 1: **Routing stabilizes before representation refinement is complete.** **(a)** Fraction of examples converged at each recurrent step for Huginn, on 30 benchmark-native examples from each of HotpotQA and GSM8K. Routing-related quantities (diagnostic token-level attention support and the attention distribution) stabilize substantially earlier than representation-related quantities (hidden states and attention outputs). Whether the attention support or the attention distribution stabilizes first varies across tasks, but the separation between routing and representation is consistent across both tasks. **(b)** This separation motivates `WISE`: early recurrent steps use unrestricted global attention to discover a sparse block-structured working set, and later steps reuse only this routing support, while hidden representations, queries, keys, values, and within-support attention weights continue to evolve. Thus, `WISE` reuses *where* attention may route without freezing *how* information within the working set is processed: *discover early, reuse late*.

each token attends (the set of keys receiving most attention mass, i.e., the attention support, and the attention distribution over keys), stabilize substantially earlier than representation-related quantities, which describe *how* the retrieved information is processed (hidden states and attention outputs). Whether the attention support or the attention distribution stabilizes first varies across tasks and models, but routing consistently stabilizes before representation across benchmarks, a range of convergence thresholds, and two different recurrent-depth models. As illustrated in Figure 1a, recurrent models can settle on a stable region of information access while their representations are still evolving. In other words, the model largely determines *where to look* before it finishes refining *how to use* the retrieved information.

This separation suggests a two-stage structure in recurrent inference: early steps perform global search to discover a sparse *working set* of relevant context, while later steps continue refining representations over largely the same routing support. Standard recurrent inference does not exploit this structure, yet if late recurrent steps attend only to a working set of $m \ll n$ keys per query, where n is the context length, their routing cost can in principle drop from $O(n^2)$ to $O(nm)$ without reducing recurrent depth, with savings that grow as the context becomes longer.

Motivated by this structure, we introduce `WISE` (**W**orking-set **I**nference with **S**upport **E**xploitation), a training-free inference method illustrated in Figure 1b. `WISE` uses unrestricted global attention during an early discovery phase to identify the working set, and then keeps it fixed as the routing support for the remaining recurrent steps. To make this sparsity exploitable on GPUs, the working set is constructed directly over contiguous blocks of tokens, enabling efficient block-sparse execution. Crucially, `WISE` fixes only *where* attention may route: hidden representations, queries, keys, values, and attention weights within the working set continue to be recomputed at every step. Thus, `WISE` preserves late representation refinement while avoiding repeated global search over positions outside the discovered working set.

Our experiments support both the mechanism and its computational implications. Controlled interventions show that discovering the working set over multiple recurrent steps, rather than from the first step alone, yields more effective working sets, and that reusing only the routing support preserves model behavior better than more restrictive alternatives. Across multi-hop QA benchmarks, `WISE` largely preserves the behavior of full-attention inference, while matched context-scaling experiments reveal an increasingly favorable quality–efficiency tradeoff as the discovered working set becomes sparser with longer contexts. This structured sparsity translates into practical GPU acceleration,

yielding up to a $1.76\times$ late-step attention speedup over native FlashAttention at 4K context. Together, these results show that recurrent routing structure can become computationally reusable while late representation refinement remains valuable.

Contributions. We summarize our contributions as follows: (i) **Mechanism discovery.** We identify a consistent separation between routing and representation timescales in recurrent-depth language models: the attention support and the attention distribution stabilize substantially earlier than hidden representations and attention outputs, suggesting that information-access structure can become reusable before recurrent refinement is complete. (ii) **Method.** We introduce `WISE`, a training-free inference method that discovers a sparse block-structured working set using unrestricted attention in early recurrent steps and reuses it as the routing support in later steps. `WISE` preserves recurrent depth and keeps within-support computation dynamic, while reducing late attention routing cost from $O(n^2)$ to $O(nm)$. (iii) **Empirical validation and efficiency.** We causally validate support-only reuse through matched interventions, replicate the routing–representation separation on a second recurrent-depth model, and identify when multi-step discovery is most beneficial: when routing continues to reorganize during early recurrence. Across context lengths, `WISE` achieves a favorable quality–efficiency tradeoff, as increasingly sparse routing support translates into practical attention acceleration while largely preserving downstream quality through 2K context.

2 Related Work

Recurrent and Latent Test-Time Computation. Recurrent architectures increase effective computational depth by repeatedly applying shared network blocks, from **Universal Transformers** to more recent looped Transformers for iterative computation and reasoning (Dehghani et al., 2018; Giannou et al., 2023; Yang et al., 2024; Saunshi et al., 2025). Recent work extends this idea to language-model inference, using recurrent depth to allocate additional test-time computation and improve multi-step reasoning (Geiping et al., 2025; Rodkin et al., 2026; Zhang et al., 2026; Kohli et al., 2026). Related approaches reason directly in continuous latent states (Hao et al., 2024). These works establish recurrent refinement as a useful computational mechanism. `WISE` instead asks whether every recurrent step must continue to perform unrestricted global attention routing, and studies how routing and representation refinement evolve across recurrent depth.

Adaptive Transformer Computation. A complementary line of work reduces inference cost by adapting how much computation different inputs or tokens receive. **Adaptive Computation Time**, **Mixture-of-Depths**, and early-exit methods dynamically reduce recurrent or layer depth (Graves, 2016; Banino et al., 2021; Elbayad et al., 2019; Raposo et al., 2024; Xin et al., 2020; Zhou et al., 2020; Elhoushi et al., 2024). `WISE` is orthogonal: it preserves recurrent depth and reduces only the scope of late global routing, while keeping attention weights and hidden-state updates dynamic.

Efficient and Sparse Attention. Kernel-level methods such as **FlashAttention** and **FlashAttention-2** accelerate exact dense attention through IO-aware tiling and improved parallelism (Dao et al., 2022; Dao, 2024). Long-context methods further reduce attention cost through KV-cache compression or sparse retrieval, including $\mathbf{H_2O}$, **SnapKV**, **Quest**, and **MInference** (Zhang et al., 2023; Li et al., 2024; Tang et al., 2024; Jiang et al., 2024). Other work reuses attention patterns across layers or decoding steps (Xiao et al., 2019; Xu et al., 2025; Deshmukh et al., 2025). `WISE` instead targets recurrent-depth language models and reuses only the routing support after it stabilizes, while within-support attention and representations remain dynamic; this support-level sparsity complements efficient kernels such as FlashAttention.

3 Problem Setup

Recurrent inference. Consider a recurrent-depth language model that repeatedly applies a shared transformation to a sequence of hidden representations. Given a sequence of length n, let $\boldsymbol{H}^{(t)} \in \mathbb{R}^{n\times d}$ denote the hidden states after recurrent step t, where d is the hidden dimension. The recurrent computation is written as

$$\boldsymbol{H}^{(t+1)} = F_\theta\Big(\boldsymbol{H}^{(t)}\Big), \qquad t = 0, \ldots, T-1, \tag{1}$$

where the parameters θ are shared across recurrent steps and T denotes the recurrent depth. This formulation abstracts away model-specific non-recurrent components and focuses on the computation repeatedly applied within the recurrent core.

Recurrent attention. At each recurrent step, self-attention is recomputed from the current hidden representations. Suppressing recurrent-block and attention-head indices for clarity, we define $\boldsymbol{Q}^{(t)} = \boldsymbol{H}^{(t)}\boldsymbol{W}_Q$, $\boldsymbol{K}^{(t)} = \boldsymbol{H}^{(t)}\boldsymbol{W}_K$, and $\boldsymbol{V}^{(t)} = \boldsymbol{H}^{(t)}\boldsymbol{W}_V$, where $\boldsymbol{Q}^{(t)}, \boldsymbol{K}^{(t)}, \boldsymbol{V}^{(t)} \in \mathbb{R}^{n\times d_h}$ and d_h is the attention-head dimension. The corresponding attention matrix and attention output are

$$\boldsymbol{A}^{(t)} = \mathrm{softmax}\left(\frac{\boldsymbol{Q}^{(t)}\boldsymbol{K}^{(t)\top}}{\sqrt{d_h}} + \boldsymbol{M}\right), \qquad \boldsymbol{O}^{(t)} = \boldsymbol{A}^{(t)}\boldsymbol{V}^{(t)}, \tag{2}$$

where $\boldsymbol{M}$ denotes the attention mask. For query position i, we write $\boldsymbol{a}_i^{(t)} = \boldsymbol{A}_{i,:}^{(t)}$ for its attention distribution and let $\mathcal{K}_i \subseteq \{1, \dots, n\}$ denote the keys admissible under $\boldsymbol{M}$.

Attention support and working sets. For a mass threshold $\eta \in (0, 1)$, let $S_i^{(t,\eta)} \subseteq \mathcal{K}_i$ denote the smallest set of admissible keys whose cumulative attention mass under $\boldsymbol{a}_i^{(t)}$ is at least η; we refer to $S_i^{(t,\eta)}$ as the *attention support*. More generally, a *working set* is a sparse routing support discovered during early recurrent computation and reused at later recurrent steps. For structured execution, we partition the sequence into blocks of size B and denote by $\mathcal{B}_q^{(t,\eta)}$ the active key-block support associated with query block q at step t; its construction from block-level attention mass is specified in Section 5. Unless explicitly shown, recurrent-block and attention-head indices are suppressed for both $S_i^{(t,\eta)}$ and $\mathcal{B}_q^{(t,\eta)}$. Standard recurrent inference recomputes global attention over all admissible query–key pairs at every recurrent step, even when this routing structure has already stabilized.

4 Computational Structure of Recurrent Inference

The distinction between recurrent representation refinement and attention routing introduced in Section 3 has a direct computational consequence. For a sequence of length n, linear projections and feed-forward layers scale as $O(nd^2)$, whereas self-attention computes pairwise query–key interactions through $\boldsymbol{Q}^{(t)}\boldsymbol{K}^{(t)\top}$ and aggregates values through $\boldsymbol{A}^{(t)}\boldsymbol{V}^{(t)}$, resulting in $O(n^2 d)$ computation across attention heads. Over T recurrent steps, the leading-order cost of standard full-attention inference is therefore

$$C_{\text{full}} = O\big(Tnd^2\big) + O\big(Tn^2 d\big)\,, \tag{3}$$

where the second term is the only component that grows quadratically with context length and corresponds to repeatedly recomputing global attention routing. Figure 1a reveals a mismatch between this computation and the dynamics of recurrent inference: routing-related quantities stabilize substantially earlier than hidden representations and attention outputs, while standard recurrent inference continues to recompute attention over the full context at every step. If early recurrent computation has identified a working set containing on average $m \ll n$ relevant keys per query, late recurrent attention needs only to compare each query against these m keys, reducing the support-dependent attention cost from

$$O(n^2 d) \quad \longrightarrow \quad O(nmd). \tag{4}$$

Equivalently, defining the normalized working-set density $\rho = m/n$, the late-stage cost is $O(\rho n^2 d)$. The $O(nmd)$ form makes explicit that late routing depends on the size of the discovered working set rather than the full context: if m grows more slowly than n, the routing cost becomes subquadratic, and it becomes linear in n when m is bounded. This creates an opportunity to reduce repeated global routing while preserving the recurrent evolution of $\boldsymbol{Q}^{(t)}$, $\boldsymbol{K}^{(t)}$, $\boldsymbol{V}^{(t)}$, and the within-support attention weights, motivating `WISE`: *discover globally early, then reuse the working set late*.

5 WISE: Discover Early, Reuse Late

Section 4 shows that late recurrent attention can reduce its routing cost from $O(n^2 d)$ to $O(nmd)$ if computation is restricted to a sparse working set of $m \ll n$ keys. WISE realizes this structure by separating recurrent inference into two stages: unrestricted global attention discovers the relevant

routing support early, and the discovered support is reused later while recurrent refinement and all within-support computation remain dynamic. The support-reuse principle itself is agnostic to routing granularity, but unstructured token-level sparsity is difficult to translate directly into efficient GPU execution. We therefore instantiate WISE with block-structured support, trading some fine-grained sparsity for regular, reusable computation that block-sparse kernels can execute efficiently.

5.1 Block-Structured Working-Set Discovery

During the first t_{d} recurrent steps, WISE retains unrestricted attention over the full context. We partition the sequence into blocks of size B and let $\mathcal{I}_q$ denote the token positions in query block q and $\mathcal{I}_b$ those in key block b. Suppressing recurrent-block and attention-head indices as in Section 3, we define the attention mass assigned from query block q to key block b at step t as

$$r_{qb}^{(t)} = \frac{1}{|\mathcal{I}_q|} \sum_{i \in \mathcal{I}_q} \sum_{j \in \mathcal{I}_b} \left[\boldsymbol{A}^{(t)}\right]_{ij}. \tag{5}$$

For a mass threshold η, let $\mathcal{B}_q^{(t,\eta)}$ be the smallest set of admissible key blocks whose cumulative block-level mass reaches η. The working set reused after discovery is

$$\mathcal{B}_q = \bigcup_{t=t_{\mathrm{d}}-3}^{t_{\mathrm{d}}} \mathcal{B}_q^{(t,\eta)}, \qquad \mathcal{B}_q^{(t,\eta)} = \arg\min_{\mathcal{B}} |\mathcal{B}| \quad \text{s.t.} \quad \sum_{b \in \mathcal{B}} r_{qb}^{(t)} \geq \eta. \tag{6}$$

Our default configuration uses $T = 32$, $t_{\mathrm{d}} = 12$, $B = 32$, and $\eta = 0.95$, so the working set is the union of direct block-level 95%-mass supports from recurrent steps 9–12. We examine sensitivity to the cumulative-mass threshold in Appendix C.7. The support is constructed independently for each recurrent block, attention head, and query block. Importantly, WISE discovers support directly in block space rather than constructing a token-level support and subsequently rounding it to blocks. This aligns the discovery objective with the granularity the sparse kernel actually executes, avoiding a mismatch between fine-grained support selection and block-structured computation.

5.2 Working-Set Reuse

For recurrent steps $t > t_{\mathrm{d}}$, the discovered block support $\mathcal{B}_q$ remains fixed, but the hidden representations and attention computation continue to evolve. Let $q(i)$ and $b(j)$ denote the query and key blocks containing positions i and j, respectively, and define the working-set mask

$$\left[\boldsymbol{R}\right]_{ij} = \begin{cases} 0, & b(j) \in \mathcal{B}_{q(i)}, \\ -\infty, & \text{otherwise.} \end{cases} \tag{7}$$

The late-stage attention is recomputed as

$$\widetilde{\boldsymbol{A}}^{(t)} = \mathrm{softmax}\left(\frac{\boldsymbol{Q}^{(t)}\boldsymbol{K}^{(t)\top}}{\sqrt{d_h}} + \boldsymbol{M} + \boldsymbol{R}\right), \qquad \widetilde{\boldsymbol{O}}^{(t)} = \widetilde{\boldsymbol{A}}^{(t)}\boldsymbol{V}^{(t)}. \tag{8}$$

Thus, WISE reuses only the routing support: $\boldsymbol{H}^{(t)}$, $\boldsymbol{Q}^{(t)}$, $\boldsymbol{K}^{(t)}$, $\boldsymbol{V}^{(t)}$, and the attention weights within the working set remain dynamic throughout recurrence. It therefore fixes *where* attention may route without freezing *how* the retained information is weighted and transformed.

5.3 Computational Complexity

Let m denote the average number of key positions represented by the active key blocks for each query during the reuse phase, let $\rho = m/n$ denote the corresponding normalized working-set density, and let $\alpha = t_{\mathrm{d}}/T$ denote the fraction of recurrent steps devoted to discovery. Full attention incurs $O(n^2 d)$ routing cost at every recurrent step, whereas block-structured reuse reduces the late-stage cost to $O(nmd)$. Ignoring block-boundary and kernel overhead, the relative attention-routing cost is therefore

$$\frac{C_{\mathrm{WISE}}^{\mathrm{attn}}}{C_{\mathrm{full}}^{\mathrm{attn}}} \approx \alpha + (1-\alpha)\rho, \qquad \Delta C^{\mathrm{attn}} \approx (1-\alpha)(1-\rho). \tag{9}$$

The gain increases as the working set becomes sparse relative to the full context. Equivalently, the late recurrent routing term changes from $O(n^2 d)$ to $O(nmd)$, replacing dependence on the full key set with dependence on the discovered working-set size while preserving the recurrent refinement identified in Section 4.

6 THEORETICAL ANALYSIS

The design of WISE relies on the possibility that a discrete routing support becomes stable before the underlying recurrent representation converges. We formalize this behavior for the block-level cumulative-mass support used in Section 5.1. The analysis does not predict the empirical switching depth t_{d}; rather, it explains why a reusable routing structure can emerge at finite recurrent depth while representation refinement continues.

6.1 FINITE-TIME IDENTIFICATION OF THE WORKING SET

Fix a recurrent block, attention head, and query block q, and let $\boldsymbol{r}_q^{(t)} = [r_{qb}^{(t)}]_b$ denote its block-level attention-mass distribution as defined in Equation 5. Let $\mathcal{B}_q^{(t,\eta)}$ be the corresponding cumulative-η support and suppose $\boldsymbol{H}^{(t)} \to \boldsymbol{H}^\star$, inducing a limiting block-mass distribution $\boldsymbol{r}_q^\star$ and support $\mathcal{B}_q^\star$. Let $k_q^\star = |\mathcal{B}_q^\star|$. We assume that the limiting support is nondegenerate: the $k_q^\star$-th and $(k_q^\star+1)$-th largest block masses are strictly separated, and η lies strictly between the cumulative masses of the first $k_q^\star - 1$ and $k_q^\star$ blocks. Let $\Delta_q > 0$ denote the minimum margin to these ranking and cumulative-mass boundaries.

Theorem 6.1 (Finite-Time Identification of Recurrent Routing Support)**.** *Suppose the mapping from $\boldsymbol{H}^{(t)}$ to $\boldsymbol{r}_q^{(t)}$ is locally L_q-Lipschitz around $\boldsymbol{H}^\star$ and*

$$\|\boldsymbol{H}^{(t)} - \boldsymbol{H}^\star\| \le C\beta^t$$

for some $0 < \beta < 1$.

Let $T_{0,q}$ be a finite step after which the local Lipschitz bound applies along the trajectory. Then there exists a finite $T_{S,q}$ such that $\mathcal{B}_q^{(t,\eta)} = \mathcal{B}_q^\star$ for all $t \ge T_{S,q}$. One valid choice is

$$T_{S,q} = \max\left\{T_{0,q}, \left\lfloor \frac{\log(L_q C/\Delta_q)}{-\log\beta} \right\rfloor + 1\right\}. \tag{10}$$

Theorem 6.1 shows that exact routing-support identification can occur at finite recurrent depth even when $\boldsymbol{H}^{(t)}$ approaches $\boldsymbol{H}^\star$ only asymptotically. Once $\boldsymbol{r}_q^{(t)}$ lies sufficiently far from every ranking and cumulative-mass boundary, further continuous changes cannot alter the selected block support. Since the model contains finitely many recurrent blocks, attention heads, and query blocks, taking the maximum of their identification times yields a finite depth after which all nondegenerate supports are simultaneously stable. If the temporal-union window used by WISE lies beyond this depth, the union reduces exactly to the limiting support; before exact stabilization, the union provides a conservative buffer against local routing fluctuations. Formal definitions of Δ_q and the proof are provided in Appendix A.

6.2 SUPPORT STABILIZATION AND REPRESENTATION CONVERGENCE

For query block q, define the routing region

$$\mathcal{R}_q(\mathcal{B}_q^\star) = \left\{\boldsymbol{H} : \mathcal{B}_q^{(\eta)}(\boldsymbol{H}) = \mathcal{B}_q^\star\right\}.$$

Proposition 1 (Support Stabilization Does Not Imply Representation Convergence)**.** *Under the assumptions of Theorem 6.1, $\boldsymbol{H}^{(t)} \to \boldsymbol{H}^\star$ implies finite-time stabilization of the cumulative-mass routing support. Moreover,*

$$\mathbb{B}\left(\boldsymbol{H}^\star, \frac{\Delta_q}{L_q}\right) \subseteq \mathcal{R}_q(\mathcal{B}_q^\star), \tag{11}$$

so support stabilization alone is insufficient to imply convergence of $\boldsymbol{H}^{(t)}$.

Proposition 1 shows that a fixed routing support identifies a region of representation space rather than a unique representation. The recurrent state can therefore continue to evolve while remaining inside the same routing region. This is precisely the regime exploited by WISE: the block support can be reused while $\boldsymbol{H}^{(t)}$, $\boldsymbol{Q}^{(t)}$, $\boldsymbol{K}^{(t)}$, $\boldsymbol{V}^{(t)}$, and the within-support attention weights continue to change. The proof is provided in Appendix A. Importantly, this result does not imply that recurrence should terminate when the support stabilizes; rather, it explains why a discrete routing structure can become reusable while recurrent representation refinement continues.

7 EXPERIMENTS

7.1 EXPERIMENTAL SETUP

Models and benchmarks. Our primary experiments use the publicly available Huginn recurrent-depth language model with recurrent depth $T = 32$ (Geiping et al., 2025). To assess whether the observed recurrent dynamics generalize beyond the primary backbone, we additionally evaluate a recurrent Llama-3.2 model trained with $T = 32$ recurrent steps (Recurrent-Llama-T32) (McLeish et al., 2025). The replication uses the same HotpotQA (Yang et al., 2018) and GSM8K (Cobbe et al., 2021) mechanism diagnostics and the same HotpotQA and 2WikiMultiHopQA (Ho et al., 2020) behavioral evaluations. Unless otherwise specified, benchmark-level comparisons use 100 deterministically selected examples with matched inputs and recurrent initialization across methods, while mechanism diagnostics use fixed 30-example cohorts. The Recurrent-Llama replication preserves the same `WISE` configuration used for Huginn; model-specific instrumentation and prompting details are provided in Appendix B.

`WISE` configuration. We use a single frozen configuration throughout the final experiments: block size $B = 32$, discovery depth $t_{\mathrm{d}} = 12$, mass threshold $\eta = 0.95$, and a four-step discovery window spanning recurrent steps 9–12. The working set is constructed directly in block space and reused for $t > t_{\mathrm{d}}$, while hidden states, queries, keys, values, and within-support attention weights remain dynamic. We use the same configuration across benchmarks and context lengths without benchmark-specific tuning.

Evaluation. We report token-level F1 for downstream quality, together with paired behavioral comparisons, answer changes, teacher-forced gold-answer loss for causal interventions, block density, and retained future Full-attention mass. Mechanism analysis uses fixed convergence criteria for attention support, attention distributions, hidden states, and attention outputs; Appendix B provides complete metric definitions, statistical procedures, and systems protocols.

7.2 MECHANISM DISCOVERY

Routing Stabilizes Before Representation Refinement. We quantify how different components of recurrent computation stabilize across depth. For the diagnostic token support, convergence is defined as the earliest step at which consecutive supports maintain Jaccard similarity of at least 0.90 for three successive transitions; for the attention distribution, hidden state, and attention output, convergence is defined using a matched three-transition criterion based on changes relative to their early-trajectory scale. Full definitions are provided in Appendix B. As shown in Figure 1a, routing-related quantities consistently stabilize earlier than representation-related quantities across both HotpotQA and GSM8K, although the precise ordering within routing is task-dependent. This separation suggests that recurrent inference enters a regime in which the model has largely settled on where to retrieve information while representation refinement is still ongoing. Because the diagnostic token support differs from the block-structured support used by `WISE`, we separately analyze the deployed $B = 32$, $\eta = 0.95$ cumulative-mass support. The resulting steps-9–12 temporal union enters a high-stability regime near the discovery depth $t_{\mathrm{d}} = 12$, providing a method-aligned basis for late support reuse (Appendix C.2).

Robustness and cross-backbone replication. The routing–representation separation persists across consistently varied convergence criteria and replicates on Recurrent-Llama-T32. Under the same $T = 32$ diagnostic protocol, all $60/60$ Recurrent-Llama examples across HotpotQA and GSM8K exhibit a positive routing–representation gap, despite substantial differences in absolute convergence speed and within-routing ordering. Full threshold-sensitivity and cross-backbone results are reported in Appendix C.

7.3 CAUSAL VALIDATION OF SUPPORT-ONLY REUSE

We next test which parts of recurrent computation can be safely reused once routing has stabilized. Truncating recurrence at the discovery depth worsens teacher-forced gold-answer loss, indicating that representation refinement remains useful after global routing has largely stabilized. Among methods that preserve recurrent depth, `WISE` better preserves Full-model behavior than early-static controls, supporting the importance of recurrent working-set discovery. We further compare against

Table 1: **Controlled interventions validate support-only reuse.** ΔAns denotes answer changes relative to the matched Full run. TRUNCATE@12 stops recurrence at the discovery depth; SIZE-MATCHED exactly matches the support cardinality of WISE; and FREEZE-A@12 uses exactly the same support as WISE but freezes its step-12 within-support attention distribution.

	HotpotQA				2WikiMultiHopQA			
Method	**F1**	**ΔAns**	**Density**	**Future**	**F1**	**ΔAns**	**Density**	**Future**
Full	0.2263	–	100.0%	100.0%	0.2516	–	100.0%	100.0%
Truncate@12	0.2141	59/100	–	–	0.2091	43/100	–	–
Static-95	0.1467	87/100	30.0%	88.0%	0.1548	74/100	34.6%	88.4%
MassMatched	0.1509	78/100	36.8%	92.4%	0.2055	66/100	42.2%	92.7%
SizeMatched	0.1836	59/100	45.0%	94.7%	0.2446	54/100	49.6%	94.9%
Freeze-A@12	0.2164	55/100	45.0%	96.7%	0.2001	48/100	49.6%	96.9%
WISE	**0.2317**	**24/100**	**45.0%**	**96.7%**	**0.2497**	**16/100**	**49.6%**	**96.9%**

FREEZE-A@12, which uses exactly the same support as WISE but also freezes the within-support attention distribution at the discovery endpoint. This stronger intervention is more disruptive than support-only reuse, indicating that the reusable structure is the routing support rather than the full attention distribution. Together, these results support the central design of WISE: preserve late recurrent refinement while reusing only the routing structure that has already stabilized.

Cross-backbone replication. Recurrent-Llama-T32 yields a more nuanced behavioral replication: the WISE–SIZEMATCHED F1 differences are $+0.0083$ on HotpotQA and -0.0043 on 2WikiMultiHopQA, indicating broadly comparable downstream quality across the two methods. At the same time, WISE retains approximately 97% of future Full-attention mass and changes fewer answers on both benchmarks. As analyzed in Appendix C.4, early static routing is substantially more predictive on this backbone, suggesting that delayed discovery is most beneficial when routing continues to reorganize during early recurrence.

7.4 PRACTICAL GPU EFFICIENCY

Native FlashAttention is highly optimized for regular dense tiled execution, but it does not directly exploit the input-dependent block-sparse schedules discovered by WISE: applying the support only as a mask would preserve semantics while still traversing masked tiles. We therefore implement an exact-$B = 32$ Triton kernel (Tillet et al., 2019) for the Huginn attention geometry on NVIDIA Ampere GPUs, using a reusable GPU-resident sparse schedule to skip inactive key blocks while fusing query–key scoring, causal masking, online softmax, and value aggregation. The final implementation further specializes arithmetic for the native head dimension and tunes the execution configuration, without changing WISE support or attention semantics; Appendix B provides implementation, correctness, and timing details. As shown in Table 2, this converts the increasingly sparse routing structure into substantial wall-clock gains, reaching a $1.76\times$ setup-inclusive late-reuse speedup over native FlashAttention at 4K; the complete $T = 32$ attention trajectory remains $1.36\times$ faster after including unrestricted discovery, and a matched exact-$B = 32$ backend comparison yields a $2.50\times$ speedup from reducing active routing support alone. These results show that WISE's structured sparsity is practically exploitable, not merely a reduction in theoretical attention work. Our implementation is nevertheless a workload-specialized sparse kernel rather than a fully co-designed sparse counterpart to FlashAttention, leaving tighter integration with architecture-specific scheduling and memory-movement pipelines as a complementary direction for further gains.

7.5 QUALITY–EFFICIENCY SCALING WITH CONTEXT LENGTH

Figure 2 summarizes how the quality–efficiency tradeoff evolves with context length. As context grows from 512 to 4K, working-set density decreases from roughly 57% to 37% while retaining about 96–98% of future Full-attention mass. This increasing sparsity translates into progressively larger practical acceleration: setup-inclusive late-reuse speedup over native FlashAttention increases from $1.15\times$ at 512 to $1.32\times$ at 1K, $1.61\times$ at 2K, and $1.76\times$ at 4K. At 4K, the complete $T = 32$ attention trajectory remains $1.36\times$ faster even after including the unrestricted discovery phase. Matched quality

Table 2: **WISE translates structured block sparsity into practical GPU acceleration at 4K context.** Latency is measured on the canonical $N = 30$ Huginn systems cohort using an NVIDIA RTX A6000, with values reported as medians over 10 balanced timing passes. Native FlashAttention provides the optimized dense baseline. The late-reuse comparison includes one-time sparse-schedule construction, while the matched exact-$B = 32$ comparison uses the same optimized backend for Full-support and WISE-support execution to isolate the benefit of reducing active routing support.

Workload	Comparison	Full	WISE	Speedup
Late reuse (20 steps)	Native FA	161.382 ms	91.820 ms	**1.758**×
Full $T = 32$ attention trajectory	Native FA	258.787 ms	190.973 ms	**1.355**×
Late reuse (20 steps)	Matched exact $B = 32$	231.130 ms	92.317 ms	**2.504**×

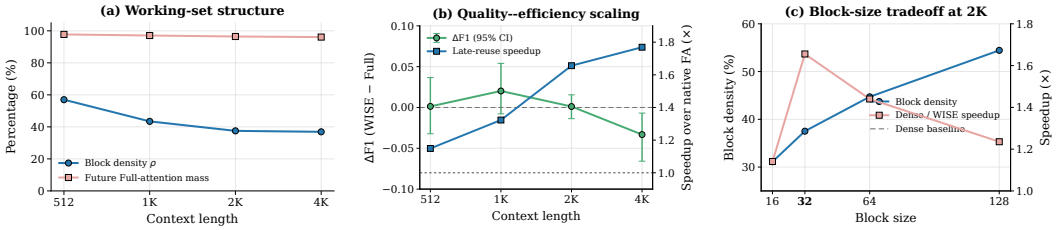


Figure 2: **WISE exposes a context-dependent quality–efficiency frontier and a sparsity–execution tradeoff. (a)** Working-set density decreases with context length while retaining most future Full-attention mass. **(b)** Late-reuse speedup increases with context length, while matched HotpotQA quality shows no detected average F1 loss through 2K and degrades at 4K; quality and systems use fixed $N = 100$ and $N = 30$ cohorts, respectively. **(c)** Although $B = 16$ is sparsest, $B = 32$ achieves the highest setup-inclusive speedup on the matched 2K workload, forming the systems-facing knee. Full protocols and block-size results are in Appendices B and C.6.

comparisons show no detected average F1 loss through 2K, while a 3.3-point reduction emerges at 4K. Thus, longer contexts expose an increasingly favorable efficiency opportunity while preserving downstream quality through moderate context lengths, with a measurable quality cost appearing only at the longest evaluated context.

Block-size tradeoff. Under equal implementation-tuning budgets, $B = 16$ yields the sparsest support, while $B = 32$ achieves the highest setup-inclusive speedup at both 2K and 4K; coarser blocks are progressively denser and slower. No tested granularity consistently dominates downstream quality across both benchmarks, so we use $B = 32$ as the systems-facing knee between sparsity and executable regularity; complete results are in Appendix C.6.

8 Conclusion

We identify a separation in recurrent-depth language-model inference: attention routing stabilizes substantially earlier than representation refinement. Controlled interventions show that routing can become reusable before recurrent refinement is complete, motivating WISE, which preserves recurrent depth while reusing only the sparse routing support discovered during early global attention. Across two recurrent backbones, our results support the routing–representation separation while revealing that delayed discovery is most useful when early routing remains predictive of later attention only after several recurrent steps. On the primary backbone, WISE converts this reusable routing structure into practical attention acceleration, exposing a context-dependent quality–efficiency tradeoff. Because WISE's block-sparse routing is complementary to FlashAttention-style kernel optimization, tighter kernel co-design may unlock further efficiency beyond the specialized sparse implementation studied here.

AI USE STATEMENT

We used large language models (LLMs) as research assistants throughout the development and preparation of this work. Specifically, LLMs assisted with drafting and refining the manuscript, discussing and reviewing theoretical formulations, mathematical arguments, and proofs, implementing and debugging research code, and providing feedback on experimental design, analysis, and result interpretation. The central research questions and core methodological ideas originated from the authors. All AI-assisted theoretical arguments, implementations, and experimental analyses were subject to human review and validation. LLM-generated suggestions were treated as advisory, and all final methodological decisions and scientific conclusions were made by the authors. The authors take full responsibility for the correctness of the theoretical results, experimental findings, and all other content of this paper.

Reproducibility Statement. We provide detailed information to support reproducibility of our results. Section 7.1 and Appendix B describe the models, datasets, evaluation cohorts, interventions, metrics, statistical procedures, and systems protocols, while Appendix A provides the proofs for our theoretical analysis. Appendix C reports robustness analyses, cross-backbone replication, and additional ablations. All experiments use fixed evaluation cohorts and publicly available datasets and model checkpoints. The implementation code is provided in the Supplementary Material.

REFERENCES

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

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.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 35549–35562, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/98ed250b203d1ac6b24bbcf263e3d4a7-Paper-Conference.pdf.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35: 16344–16359, 2022.

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

Dhruv Deshmukh, Saurabh Goyal, Nipun Kwatra, and Ramachandran Ramjee. Kascade: A practical sparse attention method for long-context llm inference. *arXiv preprint arXiv:2512.16391*, 2025.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. *arXiv preprint arXiv:1910.10073*, 2019.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. LayerSkip: Enabling early exit inference and self-speculative decoding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12622–12642, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.681. URL https://aclanthology.org/2024.acl-long.681/.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz,

M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 41340–41391. Curran Associates, Inc., 2025. doi: 10.52202/085713-1380. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/3b01972cf31e6fa0fe29e4b8b5c2a0a1-Paper-Conference.pdf.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning*, pp. 11398–11442. PMLR, 2023.

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

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. *arXiv preprint arXiv:2412.06769*, 2024.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. 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. 52481–52515. Curran Associates, Inc., 2024. doi: 10.52202/079017-1663. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5dfbe6f5671e82c76841ba687a8a9ecb-Paper-Conference.pdf.

Harsh Kohli, Srinivasan Parthasarathy, Huan Sun, and Yuekun Yao. Loop, think, & generalize: Implicit reasoning in recurrent-depth transformers. *arXiv preprint arXiv:2604.07822*, 2026.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 22199–22213. Curran Associates, Inc., 2022. doi: 10.52202/068431-1613. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/8bb0d291acd4acf06ef112099c16f326-Paper-Conference.pdf.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Sean McLeish, Ang Li, John Kirchenbauer, Dayal Singh Kalra, Brian R Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Jonas Geiping, Tom Goldstein, and Micah Goldblum. Teaching pretrained language models to think deeper with retrofitted recurrence. *arXiv preprint arXiv:2511.07384*, 2025.

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

Ivan Rodkin, Daniil Orel, Konstantin Smirnov, Arman Bolatov, Bilal Elbouardi, Besher Hassan, Yuri Kuratov, Aydar Bulatov, Preslav Nakov, Timothy Baldwin, Artem Shelmanov, and Mikhail Burtsev. Beyond memorization: Extending reasoning depth with recurrence, memory and test-time compute scaling. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 42385–42404, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.2103. URL https://aclanthology.org/2026.findings-acl.2103/.

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*, volume 2025, pp. 14855–14881, 2025.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*, 2024.

Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19, 2019.

Tong Xiao, Yinqiao Li, Jingbo Zhu, Zhengtao Yu, and Tongran Liu. Sharing attention weights for fast transformer. *arXiv preprint arXiv:1906.11024*, 2019.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. Deebert: Dynamic early exiting for accelerating bert inference. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 2246–2251, 2020.

Fangyuan Xu, Tanya Goyal, and Eunsol Choi. RefreshKV: Updating small KV cache during long-form generation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24878–24893, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1211. URL https://aclanthology.org/2025.acl-long.1211/.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *International conference on learning representations*, volume 2024, pp. 42195–42214, 2024.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Xiaojing Zhang, Haifeng Wu, Gang He, Jiyang Shen, Bochen Lyu, and Zhanxing Zhu. Modr: Mixture-of-depth-recurrent transformers for test-time reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 85952–85975, 2026.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710. Curran Associates, Inc., 2023. doi: 10.52202/075280-1506. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6ceefa7b15572587b78ecfcebb2827f8-Paper-Conference.pdf.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems*, 33: 18330–18341, 2020.

A Proofs for the Theoretical Analysis

A.1 Proof of Theorem 6.1

Proof. Fix a recurrent block, attention head, and query block q, and let $\boldsymbol{r}_q^{(t)} = [r_{qb}^{(t)}]_b$ denote the block-level attention-mass distribution defined in Equation 5. Let $\boldsymbol{r}_q^\star$ denote the limiting block-mass distribution induced by $\boldsymbol{H}^\star$, and write its entries in descending order as $r_{q,(1)}^\star \geq \cdots \geq r_{q,(N_B)}^\star$, where N_B is the number of admissible key blocks. Let $k_q^\star = |\mathcal{B}_q^\star|$ be the smallest integer such that

$$\sum_{s=1}^{k_q^\star} r_{q,(s)}^\star \geq \eta. \tag{12}$$

For the cumulative-mass support to be locally identifiable, we assume a strict ranking boundary, $r_{q,(k_q^\star)}^\star > r_{q,(k_q^\star+1)}^\star$, and a strict cumulative-mass boundary,

$$\sum_{s=1}^{k_q^\star-1} r_{q,(s)}^\star < \eta < \sum_{s=1}^{k_q^\star} r_{q,(s)}^\star.$$

Define

$$\delta_q^{\mathrm{rank}} = \frac{r_{q,(k_q^\star)}^\star - r_{q,(k_q^\star+1)}^\star}{2}, \qquad \delta_q^- = \eta - \sum_{s=1}^{k_q^\star-1} r_{q,(s)}^\star, \qquad \delta_q^+ = \sum_{s=1}^{k_q^\star} r_{q,(s)}^\star - \eta,$$

and let

$$\Delta_q = \min\left\{\delta_q^{\mathrm{rank}}, \frac{\delta_q^-}{k_q^\star - 1}, \frac{\delta_q^+}{k_q^\star}\right\}, \tag{13}$$

where the second term is interpreted as $+\infty$ when $k_q^\star = 1$. By the nondegeneracy assumptions, $\Delta_q > 0$.

We first show that $\|\boldsymbol{r}_q^{(t)} - \boldsymbol{r}_q^\star\|_\infty < \Delta_q$ is sufficient for exact support identification. For any $b \in \mathcal{B}_q^\star$ and $c \notin \mathcal{B}_q^\star$,

$$r_{qb}^{(t)} - r_{qc}^{(t)} \geq r_{qb}^\star - r_{qc}^\star - 2\left\|\boldsymbol{r}_q^{(t)} - \boldsymbol{r}_q^\star\right\|_\infty > 0, \tag{14}$$

so every block in $\mathcal{B}_q^\star$ remains ranked above every block outside it. The cumulative mass of any $k_q^\star - 1$ blocks is bounded above by

$$\sum_{s=1}^{k_q^\star-1} r_{q,(s)}^\star + (k_q^\star - 1)\left\|\boldsymbol{r}_q^{(t)} - \boldsymbol{r}_q^\star\right\|_\infty < \eta,$$

while the mass of the $k_q^\star$ blocks in $\mathcal{B}_q^\star$ is bounded below by

$$\sum_{s=1}^{k_q^\star} r_{q,(s)}^\star - k_q^\star\left\|\boldsymbol{r}_q^{(t)} - \boldsymbol{r}_q^\star\right\|_\infty > \eta.$$

Hence $k_q^\star$ remains the smallest number of blocks whose cumulative mass reaches η, and their membership is unchanged. Therefore, $\mathcal{B}_q^{(t,\eta)} = \mathcal{B}_q^\star$ whenever $\|\boldsymbol{r}_q^{(t)} - \boldsymbol{r}_q^\star\|_\infty < \Delta_q$.

By the local L_q-Lipschitz assumption and geometric convergence of the recurrent representations, for all sufficiently large t,

$$\left\|\boldsymbol{r}_q^{(t)} - \boldsymbol{r}_q^\star\right\|_\infty \leq L_q\left\|\boldsymbol{H}^{(t)} - \boldsymbol{H}^\star\right\| \leq L_q C\beta^t. \tag{15}$$

Since $0 < \beta < 1$, there exists a finite $T_{S,q}$ such that $L_q C\beta^t < \Delta_q$ for every $t \geq T_{S,q}$. Thus, $\mathcal{B}_q^{(t,\eta)} = \mathcal{B}_q^\star$ for every $t \geq T_{S,q}$. If the Lipschitz bound is valid from recurrent step $T_{0,q}$ onward, one valid choice is

$$T_{S,q} = \max\left\{T_{0,q}, \left\lfloor \frac{\log(L_q C/\Delta_q)}{-\log\beta} \right\rfloor + 1\right\}, \tag{16}$$

with the second term replaced by 0 whenever $L_q C < \Delta_q$. Consequently,

$$T_{S,q} = O\left(\frac{\log(L_q C/\Delta_q)}{-\log \beta}\right),$$

up to the finite entry time into the local Lipschitz neighborhood. For a finite collection of recurrent blocks, attention heads, and query blocks, taking $T_S = \max_q T_{S,q}$ over all such routing units yields a finite recurrent depth after which all nondegenerate block supports are simultaneously identified. □

A.2 Proof of Proposition 1

Proof. For query block q, define the routing region

$$\mathcal{R}_q(\mathcal{B}_q^\star) = \left\{ \boldsymbol{H} : \mathcal{B}_q^{(\eta)}(\boldsymbol{H}) = \mathcal{B}_q^\star \right\}.$$

The proof of Theorem 6.1 establishes that any representation satisfying

$$\left\| \boldsymbol{r}_q(\boldsymbol{H}) - \boldsymbol{r}_q^\star \right\|_\infty < \Delta_q$$

induces cumulative-mass support $\mathcal{B}_q^\star$. By local L_q-Lipschitz continuity,

$$\left\| \boldsymbol{r}_q(\boldsymbol{H}) - \boldsymbol{r}_q^\star \right\|_\infty \le L_q \left\| \boldsymbol{H} - \boldsymbol{H}^\star \right\|,$$

and therefore

$$\mathbb{B}\left(\boldsymbol{H}^\star, \frac{\Delta_q}{L_q}\right) \subseteq \mathcal{R}_q(\mathcal{B}_q^\star). \tag{17}$$

If $\boldsymbol{H}^{(t)} \to \boldsymbol{H}^\star$, then there exists a finite T such that

$$\boldsymbol{H}^{(t)} \in \mathbb{B}\left(\boldsymbol{H}^\star, \frac{\Delta_q}{L_q}\right)$$

for every $t \ge T$, and hence $\mathcal{B}_q^{(t,\eta)} = \mathcal{B}_q^\star$ thereafter. Eventual support stabilization therefore follows from convergence to a nondegenerate limiting representation.

The converse does not hold. Since $\Delta_q / L_q > 0$, the routing region contains a nontrivial neighborhood of $\boldsymbol{H}^\star$ and hence distinct representations $\boldsymbol{H}_1 \neq \boldsymbol{H}_2$ that induce the same block support $\mathcal{B}_q^\star$. A recurrent trajectory may therefore continue to move within this region while preserving an identical routing support. Thus, without additional constraints on the recurrent dynamics, support stabilization is insufficient to imply representation convergence. □

B Experimental Details

Models, benchmarks, and evaluation cohorts. Our primary experiments use the publicly available Huginn recurrent-depth language model with recurrent depth $T = 32$. We additionally evaluate Recurrent-Llama-T32, a recurrentized Llama-3.2 backbone trained with 32 recurrent steps, to test whether the observed routing dynamics and behavioral effects generalize beyond the primary checkpoint. The Recurrent-Llama model contains six recurrent blocks and uses grouped-query attention with 32 query heads and 8 key–value heads. Because it is not instruction-tuned, we use a single fixed plain-completion prompt for all Recurrent-Llama conditions; prompting is held fixed across Full, `WISE`, and all static controls within each backbone. Model parameters, numerical precision, decoding settings, and recurrent initialization are held fixed across matched inference conditions, and no model training or parameter update is performed. Our primary benchmark is the HotpotQA distractor validation split, evaluated with its original questions, answers, and benchmark-native contexts. Benchmark-level behavioral comparisons use fixed 100-example cohorts from HotpotQA and 2WikiMultiHopQA, while mechanism diagnostics use fixed 30-example cohorts from HotpotQA and GSM8K. No example is selected or removed according to Full or `WISE` performance, and all paired methods within a backbone receive identical inputs and recurrent initialization.

WISE configuration. All WISE experiments use the block-structured configuration defined in Section 5. We set the recurrent depth to $T = 32$, block size to $B = 32$, discovery depth to $t_{\mathrm{d}} = 12$, and block-mass threshold to $\eta = 0.95$. For each recurrent block, attention head, and query block q, the reused working set is

$$\mathcal{B}_q = \bigcup_{t=t_{\mathrm{d}}-3}^{t_{\mathrm{d}}} \mathcal{B}_q^{(t,\eta)}, \tag{18}$$

corresponding to the union of direct block-level 95%-mass supports from recurrent steps 9–12. Support is constructed directly in block space rather than by selecting token-level supports and subsequently rounding them to blocks. For $t > t_{\mathrm{d}}$, $\mathcal{B}_q$ remains fixed while $\boldsymbol{H}^{(t)}$, $\boldsymbol{Q}^{(t)}$, $\boldsymbol{K}^{(t)}$, $\boldsymbol{V}^{(t)}$, and the within-support attention weights continue to be recomputed. The same $B = 32$, $t_{\mathrm{d}} = 12$, $\eta = 0.95$, and steps-9–12 discovery window are used for Huginn and Recurrent-Llama-T32 without backbone-specific tuning.

Intervention controls. We compare WISE against three early-static controls that use the same block representation and the same intervention timing. For each recurrent block, attention head, and query block q, let $\boldsymbol{a}_q^{(t)}$ denote the causal block-attention masses at recurrent step t, and let $\mathcal{B}_q^{(t,\rho)}$ denote the smallest set of highest-mass admissible key blocks whose cumulative mass under $\boldsymbol{a}_q^{(t)}$ is at least ρ. The STATIC-95 control freezes the direct 95%-mass support identified at the first recurrent step,

$$\mathcal{S}_q^{\mathrm{Static95}} = \mathcal{B}_q^{(1,0.95)}. \tag{19}$$

To distinguish the effect of recurrent discovery from simply retaining more early attention mass, MASSMATCHED uses the recurrence-1 ranking but matches the amount of recurrence-1 attention mass covered by the final WISE working set. Writing

$$m_q^{\mathrm{WISE}} = \sum_{j \in \mathcal{W}_q} a_{q,j}^{(1)}, \qquad \mathcal{W}_q = \bigcup_{t=t_{\mathrm{d}}-3}^{t_{\mathrm{d}}} \mathcal{B}_q^{(t,\eta)}, \tag{20}$$

the mass-matched control is

$$\mathcal{S}_q^{\mathrm{MassMatched}} = \mathcal{B}_q^{(1,m_q^{\mathrm{WISE}})}. \tag{21}$$

Finally, SIZEMATCHED controls exactly for support cardinality. Let $K_q = |\mathcal{W}_q|$. Using the recurrence-1 block scores, it retains the K_q highest-scoring admissible blocks,

$$\mathcal{S}_q^{\mathrm{SizeMatched}} = \mathrm{TopK}\left(\boldsymbol{a}_q^{(1)}, K_q\right). \tag{22}$$

Thus, MASSMATCHED matches the recurrence-1 attention mass retained by WISE, whereas SIZEMATCHED matches its block cardinality exactly for every local routing unit.

We additionally evaluate two interventions that probe what remains necessary after the discovery phase. FREEZE-A@12 uses exactly the same final working set W_q as WISE, but also freezes the within-support attention distribution at the discovery endpoint. Let $A^{(12)}$ denote the unrestricted post-softmax attention matrix at recurrent step 12. Using the query- and key-block mappings $q(i)$ and $b(j)$ from Section 5.2, we define the restricted and renormalized distribution

$$\bar{A}_{ij}^{(12)} = \frac{A_{ij}^{(12)}\,\mathbf{1}\big[b(j) \in W_{q(i)}\big]}{\sum_{k \in \mathcal{K}_i} A_{ik}^{(12)}\,\mathbf{1}\big[b(k) \in W_{q(i)}\big]},$$

where $\mathcal{K}_i$ denotes the keys admissible under the native attention mask. For all $t > t_d$, FREEZE-A@12 reuses this distribution,

$$\widetilde{A}^{(t)} = \bar{A}^{(12)}, \qquad \widetilde{O}^{(t)} = \bar{A}^{(12)} V^{(t)},$$

while values, hidden representations, and the remaining recurrent computation continue to evolve. Thus, FREEZE-A@12 and WISE use exactly the same routing support and differ only in whether within-support attention weights are recomputed after discovery.

TRUNCATE@12 instead follows the unrestricted Full trajectory through $t_d = 12$ and then terminates the recurrent core without executing steps 13–32. The model's native post-recurrent computation, including its exit path, final normalization, and output head, is otherwise left unchanged.

All intervention conditions are matched to the unrestricted Full trajectory through $t_d = 12$. For STATIC-95, MASSMATCHED, SIZEMATCHED, and WISE, recurrent computation continues through $T = 32$ using the corresponding fixed support while queries, keys, values, hidden states, and within-support attention weights remain dynamic. FREEZE-A@12 likewise continues through $T = 32$, but additionally reuses the fixed distribution $\bar{A}^{(12)}$ defined above. TRUNCATE@12 instead terminates recurrent computation immediately after step 12. These interventions therefore separately test support composition, within-support reweighting, and the value of continued recurrent refinement after the discovery depth.

Evaluation metrics. We report standard normalized exact match (EM) and token-level F1 for downstream answer quality. Because the main comparisons are paired at the example level, we additionally report paired F1 differences with 95% bootstrap confidence intervals and answer-change counts relative to the matched Full run. Controlled interventions additionally report paired changes in teacher-forced gold-answer adjusted loss as a continuous measure of behavioral perturbation, where a positive change denotes worse behavior relative to the reference condition. To characterize routing structure, we report block density, defined as the fraction of admissible query–key blocks retained during reuse, and retained future attention mass, defined as the fraction of unrestricted Full-attention mass at $t > t_{\mathrm{d}}$ that falls inside the frozen support. We compute these metrics using the same definitions for both backbones.

Mechanism diagnostics. The convergence analysis uses fixed 30-example cohorts from HotpotQA and GSM8K and tracks diagnostic attention support, attention distributions, hidden representations, and pre-projection attention outputs across recurrent depth. For query position i, the diagnostic support is the causal Top-$\min(16, i+1)$ attended-key set, which is distinct from the direct $B = 32$ cumulative-mass support deployed by WISE. Support stability is measured by Jaccard similarity between consecutive supports, with convergence defined by similarity ≥ 0.90. For $\boldsymbol{A}^{(t)}$, $\boldsymbol{H}^{(t)}$, and $\boldsymbol{O}^{(t)}$, consecutive recurrent states are compared using Jensen–Shannon divergence, relative ℓ_2 change, and relative ℓ_2 change, respectively; each convergence threshold is set to 0.05 times the example-specific early-trajectory scale computed from the first four finite transition distances of that quantity. All four criteria must hold for three consecutive transitions, and τ_X is the first transition in the qualifying run; trajectories that do not converge by $T = 32$ are recorded as $\tau_X = 33$. We apply the same diagnostic definitions and thresholds to Huginn and Recurrent-Llama-T32 without retuning. These diagnostics characterize recurrent dynamics only and do not determine the WISE switching depth or working-set configuration. We separately evaluate the stability of the exact block support deployed by WISE in Appendix C.2.

Cross-backbone replication. The Recurrent-Llama replication uses a model-specific instrumentation adapter to expose recurrent attention, hidden representations, and pre-projection attention outputs while preserving the model's native computation. The adapter accounts for Llama-style rotary position embeddings and grouped-query attention but does not modify model parameters, recurrent depth, attention scores, or support construction. The behavioral replication evaluates the original five support-selection conditions—Full, STATIC-95, MASSMATCHED, SIZEMATCHED, and WISE—using the same definitions and intervention timing as on the primary Huginn backbone. In particular, SIZEMATCHED exactly matches the number of retained blocks used by WISE for every local routing unit. FREEZE-A@12 and TRUNCATE@12 are additional causal interventions evaluated only on the primary Huginn backbone.

Context-length scaling. Matched HotpotQA quality scaling is performed only on the primary Huginn backbone and uses the same 100-example cohort at approximately 512, 1K, 2K, and 4K context lengths. Required answer evidence is preserved at every length, while additional context is formed from natural benchmark passages rather than synthetic padding or duplicated text. Full and WISE receive exactly the same input at each context length. Because absolute Full-model difficulty varies with context composition, the primary scaling statistic is the paired difference $\Delta\mathrm{F1} = \mathrm{F1}_{\mathrm{WISE}} - \mathrm{F1}_{\mathrm{Full}}$ rather than the absolute F1 trajectory across lengths. The Huginn checkpoint supports at most 4096 positions through its rotary-position representation; consequently, 4K is the maximum valid model-level context length considered in the scaling study, and we report no model-level 8K quality result.

Systems evaluation. Systems measurements are performed only for the primary Huginn configuration on NVIDIA RTX A6000 GPUs. The practical dense baseline uses PyTorch scaled dot-product attention dispatching to native FlashAttention, while the final WISE implementation uses a custom Triton kernel specialized for exact $B = 32$ block-structured reuse. Systems measurements use a fixed 30-example cohort of saved HotpotQA working-set workloads, distinct from the 100-example quality-scaling cohort. Headline measurements use 10 balanced timing passes on an uncontended RTX A6000 under a fixed software stack, with latency reported as the median across passes and the corresponding interquartile range retained to characterize timing variability. We report three complementary measurements: setup-inclusive late-reuse latency, which compares one-time sparse-schedule preparation plus 20 reuse calls against 20 native FlashAttention calls; complete $T = 32$ attention-trajectory latency, which includes the 12 unrestricted discovery steps together with schedule preparation and 20 reuse steps; and a controlled same-backend comparison in which the optimized exact-$B = 32$ kernel executes either all causally valid blocks or only the WISE support. Latency and answer-quality results are therefore treated as complementary measurements rather than as measurements on the same evaluation cohort.

Sparse-kernel implementation. The custom kernel consumes the exact block support discovered by WISE without introducing an additional approximation. For each recurrent block, attention head, and query block, the active key blocks are encoded as a reusable GPU-resident CSR schedule consisting of row offsets and active block indices. This schedule is constructed once after the discovery phase and reused unchanged across all subsequent recurrent steps. At execution time, the kernel skips blocks absent from the sparse schedule while fusing query–key scoring, causal masking, online softmax normalization, and value aggregation without materializing the dense attention matrix. Queries, keys, and values are recomputed at every recurrent step, so only the routing support is reused, and the measured sparse path contains no dense-attention fallback. For the primary Huginn workload, attention contains four recurrent attention layers with 55 query and key–value heads and native head dimension $d_h = 96$. The final implementation specializes arithmetic for this native head dimension and uses two warps and three pipeline stages. An equal-budget work-order sweep selects the natural within-head row order for the final configuration; these implementation choices affect only kernel realization and do not alter the discovered support or attention semantics. Before timing, the optimized kernel is validated against explicit exact masked attention under the same BF16 correctness criterion used throughout the systems study. Setup-inclusive measurements include GPU sparse-schedule construction and associated execution-metadata preparation, whereas prepared-schedule measurements exclude only this one-time reusable setup cost. All reported Full and WISE comparisons use the same hardware and software configuration, identical inputs, and identical warm-up, synchronization, and timing procedures.

Reproducibility. All example identities, context variants, intervention conditions, predictions, working-set masks, and timing records are stored in deterministic manifests. We verify that paired methods use identical inputs and recurrent initialization within each backbone, that WISE begins support reuse only after t_{d}, that the discovered block support remains fixed throughout reuse, and that hidden states, queries, keys, values, and within-support attention weights continue to evolve. For the cross-backbone replication, we additionally record the Recurrent-Llama checkpoint revision, tokenizer, fixed plain-completion prompt, grouped-query-attention instrumentation, and all exact support-cardinality audits. Exact checkpoint revisions, preprocessing details, prompts, generation parameters, hardware configuration, and implementation code will be released with the final artifact.

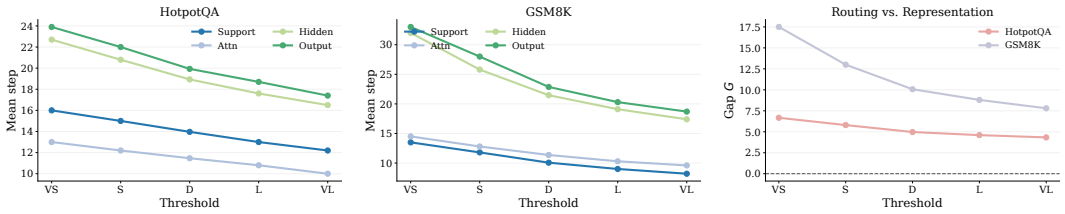
C Robustness Analysis

C.1 Sensitivity to Convergence Criteria

The routing–representation separation is robust to matched changes in the convergence criteria. We evaluate five settings ranging from very strict to very loose, varying the strictness of the routing and representation diagnostics consistently. To summarize the separation, we define the routing–representation gap as

$$G = \min(\tau_H, \tau_O) - \max(\tau_S, \tau_A),$$

such that $G > 0$ requires both routing quantities to stabilize before either representation quantity. Across all five matched settings, $G > 0$ for every diagnostic example: $30/30$ on HotpotQA and



(a) HotpotQA **(b) GSM8K** **(c) Routing–representation gap**

Figure 3: **Sensitivity of the convergence analysis to threshold choice. (a)–(b)** Fraction of examples classified as converged by recurrent step under five *matched* threshold settings: very loose (VL), loose (L), default (D), strict (S), and very strict (VS). Across these matched settings, routing-related quantities consistently converge earlier than representation-related quantities, although the exact convergence times and step gaps vary with threshold strictness. **(c)** Mean routing–representation gap under the same matched settings on HotpotQA and GSM8K. Positive values indicate earlier routing convergence. The gap remains positive throughout the matched sweep, supporting a robust routing–representation separation under reasonable threshold perturbations, while not implying threshold-invariance under arbitrary asymmetric criteria.

Table 3: **Boundary of the convergence claim on HotpotQA.** Matched calibration preserves the routing–representation separation, while deliberately asymmetric calibration can reverse the measured ordering.

Criteria	τ_S	τ_A	τ_H	τ_O	G
Default	13.97	11.47	18.93	19.93	4.97
Routing very strict / repr. loose	18.90	13.20	17.33	18.20	-1.57

$30/30$ on GSM8K. The mean gap ranges from 4.33 to 6.67 recurrent steps on HotpotQA and from 7.80 to 17.50 on GSM8K. The largest GSM8K gaps under the strictest criteria should be interpreted cautiously because representation convergence is strongly right-censored at the $T = 32$ endpoint. These results show that the qualitative routing–representation separation does not depend on the particular default threshold choice.

The ordering is not invariant to arbitrary cross-quantity calibration. In an intentionally unfavorable but still reasonable asymmetric setting, routing is evaluated using very strict criteria while representation is evaluated using loose criteria. On HotpotQA, the resulting mean convergence steps are $\tau_S = 18.90$, $\tau_A = 13.20$, $\tau_H = 17.33$, and $\tau_O = 18.20$, giving a mean gap of $G = -1.57$; only $1/30$ examples retains $G > 0$. We therefore do not claim a threshold-independent ordering across heterogeneous quantities. Rather, the empirical claim is that routing stabilizes earlier under the reported criteria and throughout the tested family of consistently calibrated threshold perturbations.

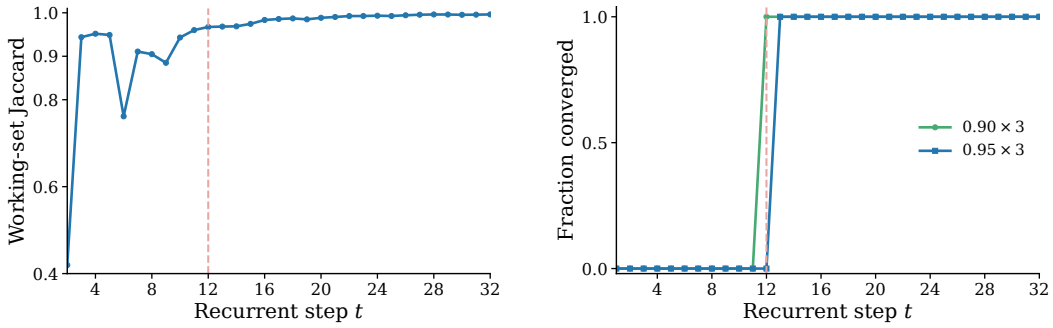
C.2 Stability of the Deployed Working Set

The diagnostic support used in Figure 1a is token-level, whereas `WISE` reuses a four-step union of direct $B = 32$, $\eta = 0.95$ cumulative-mass block supports. We therefore evaluate the stability of the exact working-set construction used by the method. For each recurrent step t, define the candidate rolling working set

$$\mathcal{W}_q^{(t)} = \bigcup_{s=\max(1,t-3)}^{t} \mathcal{B}_q^{(s,\eta)}. \tag{23}$$

For an example e, let $\mathcal{W}_e^{(t)}$ denote the collection of active key blocks across all recurrent blocks, attention heads, and query blocks. We measure adjacent rolling-set stability as

$$J_{\text{roll}}^{(t)} = \frac{1}{N} \sum_{e=1}^{N} \frac{\left| \mathcal{W}_e^{(t-1)} \cap \mathcal{W}_e^{(t)} \right|}{\left| \mathcal{W}_e^{(t-1)} \cup \mathcal{W}_e^{(t)} \right|}. \tag{24}$$



(a) Rolling working-set stability **(b) Fraction of converged examples**

Figure 4: **The deployed block support becomes highly stable near the `WISE` discovery depth. (a)** Adjacent-step Jaccard similarity between four-step rolling working sets constructed from the exact $B = 32$, $\eta = 0.95$ block supports used by `WISE`. The discovery depth $t_{\mathrm{d}} = 12$ lies near the onset of a high-stability regime. **(b)** Fraction of examples whose rolling-set Jaccard satisfies the stability criterion for three consecutive transitions. All 30 examples satisfy the 0.90 criterion by $t = 12$, while all satisfy the stricter 0.95 criterion by $t = 13$.

This statistic exactly matches the block-structured working-set construction used by `WISE`; for $t > 12$, the rolling sets are reconstructed from the saved unrestricted Full-attention trajectory for diagnostic purposes, whereas deployed `WISE` freezes $\mathcal{W}^{(12)}$ after discovery.

The rolling working set becomes highly stable near the chosen discovery depth. At $t = 12$, the mean adjacent rolling-set Jaccard is 0.9674 on the 30-example 4K HotpotQA audit cohort. Under the 0.90-for-three-transitions criterion, all $30/30$ examples first satisfy the stability condition at $t = 12$. Under the stricter 0.95 criterion, none satisfy it by $t = 12$, while all $30/30$ satisfy it at $t = 13$. Thus, $t_{\mathrm{d}} = 12$ lies at the onset of a high-stability regime rather than marking an exact convergence boundary.

Local stability does not imply that the support becomes globally fixed. Comparing the working set selected at the switching point with the hypothetical rolling set constructed from the final Full-attention steps gives

$$J\left(\mathcal{W}^{(12)}, \mathcal{W}^{(32)}\right) = 0.8987. \tag{25}$$

This slow cumulative drift is consistent with the role of the temporal union: `WISE` does not assume that routing becomes exactly constant at a single recurrent step, but instead freezes a working set after routing has entered a low-drift regime. Together with the high future Full-attention mass retained by `WISE`, this analysis provides a direct method-aligned bridge between the token-level mechanism diagnostic and the block-structured support actually reused during inference.

C.3 Cross-Backbone Replication

We next test whether the observed separation between routing stabilization and representation refinement is specific to the primary Huginn checkpoint. We repeat the mechanism analysis on Recurrent-Llama-T32 using the same recurrent depth $T = 32$, the same diagnostic definitions, and the same convergence thresholds described in Appendix B. The replication uses fixed 30-example HotpotQA and GSM8K cohorts and does not retune any convergence criterion for the new backbone. As shown in Table 4, the routing–representation separation replicates strongly. On HotpotQA, attention support and attention distributions converge at mean steps 5.07 and 7.00, respectively, while the hidden state and attention output converge at 8.93 and 16.13. On GSM8K, the corresponding values are 3.87, 7.10, 9.63, and 31.20. Most importantly, every diagnostic example exhibits a positive routing–representation gap: $30/30$ on HotpotQA and $30/30$ on GSM8K. The precise ordering within routing differs from the primary Huginn backbone—support stabilizes before the attention distribution on Recurrent-Llama-T32—but the broader separation between routing stabilization and continued representation refinement is preserved.

Table 4: **The routing–representation separation replicates on Recurrent-Llama-T32.** Mean convergence steps use the same diagnostic definitions and thresholds as the primary Huginn analysis. A positive gap requires both routing quantities to stabilize before both representation quantities.

Benchmark	τ_S	τ_A	τ_H	τ_O	Positive gap
HotpotQA	**5.07**	**7.00**	8.93	16.13	**30/30**
GSM8K	**3.87**	**7.10**	9.63	$31.20^{\dagger}$	**30/30**

†For GSM8K, $27/30$ attention-output trajectories do not converge by $T = 32$ and are recorded as $\tau_O = 33$ when computing the mean; 31.20 therefore reflects right censoring rather than an observed convergence time.

Table 5: **Behavioral replication on Recurrent-Llama-T32.** SIZEMATCHED and `WISE` use identical support cardinality for every routing unit. Future denotes retained late Full-attention mass. Paired confidence intervals are reported in the text.

Benchmark	Full	SizeMatched	`WISE`	Density	Future: Size / `WISE`	ΔF1
HotpotQA	0.2075	0.2076	**0.2159**	0.338	96.13% / **96.56%**	+0.0083
2Wiki	**0.2172**	0.2128	0.2085	0.395	96.41% / **96.81%**	−0.0043

We also repeat the main behavioral intervention study using the same frozen `WISE` configuration and the same Full, STATIC-95, MASSMATCHED, and SIZEMATCHED controls. Exact SIZEMATCHED cardinality matching succeeds for all $1{,}478{,}784$ audited local routing units. The result is more nuanced than on the primary backbone. On HotpotQA, `WISE` obtains F1 0.2159, compared with 0.2075 for Full and 0.2076 for SIZEMATCHED, corresponding to a paired `WISE`–SIZEMATCHED difference of $+0.0083$. On 2WikiMultiHopQA, the corresponding values are 0.2085, 0.2172, and 0.2128, giving a paired difference of -0.0043. Thus, unlike on Huginn, the cross-backbone replication does not establish a downstream F1 advantage of recurrently discovered support over an exactly size-matched early-static support.

C.4 WHEN DOES DELAYED DISCOVERY HELP?

The cross-backbone comparison suggests a natural boundary condition for the benefit of `WISE`. Although Recurrent-Llama-T32 exhibits the same routing-before-representation separation, its routing support stabilizes substantially earlier than on Huginn. On HotpotQA, the mean token-support convergence step decreases from 13.97 on Huginn to 5.07 on Recurrent-Llama-T32; on GSM8K, it decreases from 10.07 to 3.87. This earlier stabilization is reflected in how predictive an early static support is of later Full attention. On Huginn, the exactly size-matched early-static support retains 94.65% and 94.87% of late Full-attention mass on HotpotQA and 2WikiMultiHopQA, respectively. On Recurrent-Llama-T32, the corresponding values increase to 96.13% and 96.41%, despite lower retained block density. Consequently, recurrent discovery increases late-mass coverage by approximately 2 percentage points on Huginn but only about 0.4 percentage points on Recurrent-Llama-T32. These results suggest that delayed recurrent discovery is most useful when routing continues to reorganize during early recurrence. When an early support already predicts later attention well, an early static support can approximate the eventually discovered working set more closely, leaving less room for `WISE` to improve support composition. We treat this interpretation as suggestive rather than definitive: the saved cross-backbone trajectories do not contain the exact per-unit $B = 32$ support identities and attention masses required to measure direct early-to-late block-level Jaccard similarity or head-level routing specialization. Moreover, attention distributions and hidden representations also converge more quickly on Recurrent-Llama-T32, so the current evidence does not isolate stable routing roles from broader differences in recurrent dynamics.

C.5 `WISE` HYPERPARAMETER SENSITIVITY

Finally, we test whether the frozen `WISE` configuration is an isolated operating point on the primary Huginn backbone. Starting from $\eta = 0.95$, $t_{\mathrm{d}} = 12$, temporal-union width $w = 4$, and $B = 32$, we vary one parameter at a time while keeping all others fixed. This analysis is intended as a local robustness check rather than hyperparameter optimization; all initial comparisons use the same

Table 6: **Early routing is more predictive on Recurrent-Llama-T32.** SIZEMATCHED Future is the fraction of late unrestricted Full-attention mass retained by the exactly size-matched early-static support. Gain is the additional late-mass coverage obtained by `WISE`.

Backbone	Benchmark	SizeMatched Future	`WISE` gain
Huginn	HotpotQA	94.65%	+2.01 pp
Huginn	2Wiki	94.87%	+2.04 pp
Recurrent-Llama-T32	HotpotQA	**96.13%**	**+0.43 pp**
Recurrent-Llama-T32	2Wiki	**96.41%**	**+0.40 pp**

Table 7: **Local sensitivity of `WISE` on HotpotQA ($N = 50$).** Each row varies one parameter around the frozen default. Density is the retained $B = 32$ block fraction and Future Mass is the fraction of later unrestricted Full-attention mass covered by the frozen support.

Setting	F1	ΔF1	95% CI	Density	Future Mass
Default ($\eta = 0.95,\ t_d = 12,\ w = 4$)	0.1899	$+0.0101$	$[-0.0041, +0.0303]$	0.447	0.966
$\eta = 0.90$	0.1813	$+0.0015$	$[-0.0295, +0.0331]$	0.298	0.933
$\eta = 0.98$	0.1816	$+0.0019$	$[-0.0011, +0.0067]$	0.651	0.987
$t_d = 8$	0.1795	-0.0003	$[-0.0153, +0.0122]$	0.498	0.972
$t_d = 16$	0.1817	$+0.0019$	$[-0.0049, +0.0096]$	0.431	0.964
$w = 1$	0.1683	-0.0115	$[-0.0342, +0.0061]$	0.413	0.961
$w = 2$	0.1816	$+0.0018$	$[-0.0040, +0.0079]$	0.426	0.963

deterministic $N = 50$ HotpotQA native-context cohort, for which matched Full inference obtains F1 0.1798. As shown in Table 7, no tested neighboring configuration clearly dominates the frozen default. Varying η exposes the expected sparsity–coverage tradeoff: decreasing η from 0.95 to 0.90 reduces block density from 0.447 to 0.298 while reducing retained future attention mass from 0.966 to 0.933, whereas increasing η to 0.98 raises future-mass retention to 0.987 at the cost of density increasing to 0.651. Nearby discovery depths and temporal-union widths similarly yield comparable downstream behavior while changing the retained routing structure.

The initially larger degradation observed for $w = 1$ was evaluated on the untouched second half of the canonical $N = 100$ cohort. On the first 50 examples, $w = 1$ minus the default yielded $\Delta\mathrm{F1} = -0.0216$ with 95% CI $[-0.0458, -0.0029]$, whereas the untouched second 50 yielded $+0.0089$ with 95% CI $[-0.0111, +0.0425]$. Pooling all 100 examples gives $\Delta\mathrm{F1} = -0.0063$ with 95% CI $[-0.0234, +0.0140]$, so the initial quality deficit does not replicate. We therefore interpret the sensitivity analysis as evidence that the default is not an obviously fragile isolated point, rather than evidence of hyperparameter insensitivity or optimality. These experiments are restricted to benchmark-native contexts and one-at-a-time perturbations; they do not establish long-context hyperparameter robustness or robustness to joint interactions among η, t_{d}, and w.

C.6 BLOCK-SIZE TRADEOFF

At the finest extreme, token-level support would maximize routing granularity but produce highly irregular sparsity; block structure instead exposes a controllable systems tradeoff between sparsity and executable regularity. The block size B controls the granularity at which the discovered working set can be exploited by sparse execution. We therefore evaluate $B \in \{16, 32, 64, 128\}$ while keeping the remaining `WISE` configuration fixed: $\eta = 0.95$, $t_d = 12$, and the four-step discovery window spanning recurrent steps 9–12, and use an equal implementation-tuning budget for each block size. Finer blocks expose more routing sparsity, whereas coarser blocks conservatively retain more neighboring tokens. However, block density alone does not determine execution efficiency, since the granularity of the resulting sparse computation also affects kernel execution. All four implementations are validated against explicit exact masked attention under the same BF16 correctness criterion, with maximum absolute error below 0.007643 and no support or causal-mask mismatches. For $B = 64$ and $B = 128$, the independently selected block supports are executed using exact 32×32 physical subtiles without changing support membership or softmax semantics.

Table 8 summarizes the full tradeoff. On the matched $N = 30$ 2K systems panel, $B = 16$ produces the sparsest support at 31.12% density but reaches only a $1.141\times$ setup-inclusive speedup. Increasing

Table 8: **Block-size ablation for WISE.** Quality and routing statistics use the matched $N = 100$ benchmark cohorts, while systems statistics use the matched $N = 30$ 2K panel. Future denotes the fraction of future Full-attention mass retained by the discovered working set. Systems speedup is the setup-inclusive Dense/WISE attention-latency ratio under the matched block-size execution harness.

	HotpotQA			2WikiMultiHopQA			2K systems		
B	F1	Density	Future	F1	Density	Future	Density	Future	Speedup
16	0.2254	37.60%	96.34%	0.2831	41.19%	96.53%	31.12%	96.22%	1.141×
32	0.2317	44.99%	96.66%	0.2497	49.58%	96.92%	37.51%	96.46%	**1.656**×
64	0.2463	53.51%	97.11%	0.2599	58.97%	97.48%	44.71%	96.83%	1.440×
128	0.2309	64.26%	97.75%	0.2562	70.40%	98.24%	54.44%	97.36%	1.236×

Table 9: Sensitivity to the working-set mass threshold η. All settings other than η are fixed. ΔF1 is measured relative to the matched Full run, and Future denotes retained future Full-attention mass.

Benchmark	η	F1	ΔF1 (95% CI)	Density	Future
HotpotQA	0.90	0.2102	-0.0161 [-0.0489, $+0.0133$]	30.2%	93.3%
	0.95	0.2317	$+0.0054$ [-0.0171, $+0.0261$]	45.0%	96.7%
	0.99	0.2212	-0.0051 [-0.0123, 0.0000]	77.8%	99.4%
2WikiMultihopQA	0.90	0.2309	-0.0207 [-0.0542, $+0.0077$]	35.2%	93.7%
	0.95	0.2497	-0.0019 [-0.0133, $+0.0077$]	49.6%	96.9%
	0.99	0.2541	$+0.0025$ [0.0000, $+0.0075$]	80.0%	99.5%

to $B = 32$ raises density modestly to 37.51% while achieving the highest speedup of $1.656\times$. Coarser $B = 64$ and $B = 128$ retain 44.71% and 54.44% of blocks and reach $1.440\times$ and $1.236\times$, respectively. The same ordering persists at 4K, where $B = 32$ reaches $1.769\times$, compared with $1.146\times$, $1.554\times$, and $1.394\times$ for $B = 16$, 64, and 128. Thus, finer sparsity does not necessarily translate into faster execution: $B = 32$ forms a robust systems-facing knee between support sparsity and executable regularity.

Downstream quality exhibits a substantially weaker dependence on block size than execution efficiency. On HotpotQA, $B = 16$ differs from $B = 32$ by only -0.0064 F1, with a paired 95% bootstrap confidence interval of $[-0.0288, 0.0166]$. On 2WikiMultiHopQA, the corresponding difference is $+0.0334$, but its interval $[-0.0043, 0.0738]$ also includes zero. Likewise, although $B = 64$ attains higher observed F1 than $B = 32$ on both benchmarks, the paired difference is not established on HotpotQA ($+0.0145$, 95% CI $[-0.0097, 0.0450]$), while the smaller 2Wiki gain is positive ($+0.0102$, 95% CI $[0.0005, 0.0228]$). The sweep therefore does not identify a block size that consistently dominates downstream quality across both benchmarks. Instead, $B = 32$ is selected as the default systems-facing operating point because it provides the strongest overall balance between routing sparsity and realized execution efficiency, rather than because it is universally optimal for F1.

C.7 Sensitivity to the Working-Set Mass Threshold

We evaluate sensitivity to the cumulative-mass threshold η while holding all other aspects of WISE fixed: $T = 32$, $t_d = 12$, $B = 32$, and the four-step discovery window spanning recurrent steps 9–12. The threshold controls the conservativeness of working-set construction: lower values retain fewer blocks, while higher values preserve more of the Full-attention mass. Lowering η to 0.90 produces substantially sparser working sets, but also reduces retained future attention mass and increases behavioral changes relative to Full. Increasing η to 0.99 preserves nearly all future attention mass, but raises block density to roughly 80% without yielding a consistent quality improvement across benchmarks. The default $\eta = 0.95$ therefore provides a reasonable quality–sparsity operating point rather than relying on a narrowly tuned F1 optimum.