IGAL: IMPORTANCE-GUIDED ATTENTION LINEARIZATION OF LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

The quadratic cost of self-attention makes long-context processing expensive for pretrained Transformers. Converting self-attention to more efficient alternatives, such as sliding-window attention (SWA) and linear attention (LA), reduces this cost but can degrade accuracy, particularly on tasks requiring long-range retrieval. Attention heads exhibit substantial functional heterogeneity, with different heads specializing in local interactions and long-range retrieval. Existing conversion methods typically approximate each pretrained attention head with a combination of LA and SWA, relying on head-specific mixing weights and/or branch parameters to adapt this hybrid form to different functional roles. In contrast, we leverage pretrained head specialization to directly determine the attention operator assigned to each head. Specifically, we propose Importance-Guided Attention Linearization (IGAL), which repurposes existing head-importance scores—previously used to allocate larger KV-cache budgets—to guide heterogeneous attention conversion. IGAL converts a small subset of high-scoring heads to LA for long-range modeling, while converting the remaining heads to SWA for local interactions. We further find that the optimal proportion of LA heads differs between short- and long-context tasks. To accommodate both regimes within a single checkpoint, we introduce IGAL-R, which sets the LA budget using the prompt length and learns which heads receive it with a lightweight prompt-level router, fine-tuned on a mixture of general-language and retrieval data. Under task-specific fine-tuning on Llama 3.1 and Qwen3, IGAL improves general-language and long-context retrieval performance over the existing state-of-the-art linearization methods. On Llama-3.1-8B, assigning only 6.25% of GQA groups to LA achieves 89.3% needle-in-a-haystack retrieval accuracy at 32K after 1K-length data fine-tuning, with a $292\times$ smaller decoding cache than full attention. Under matched mixed-data fine-tuning, IGAL-R raises 16–32K retrieval accuracy from 21.7% to 78.3% over static IGAL, with comparable general-language performance.

1 INTRODUCTION

Pretrained Transformers provide strong language capabilities, but their reliance on full self-attention makes long-context deployment expensive (Vaswani et al., 2023; Llama Team, AI @ Meta, 2024; DeepSeek-AI, 2025). Attention computation grows quadratically with sequence length during prefill, while autoregressive decoding maintains a key–value (KV) cache that expands linearly with the growing context. Converting a pretrained Transformer into an architecture with linear computational complexity and bounded-memory offers a practical way to reduce these costs without repeating full pretraining (Zhang et al., 2025; Liu et al., 2025; Lan et al., 2025). However, preserving model quality after conversion remains challenging, particularly on tasks requiring long-range retrieval.

Sliding-window attention (SWA) and linear attention (LA) offer complementary approximations to full attention. SWA retains token-level softmax interactions within a local window but limits direct access to distant keys and values. LA summarizes the preceding context in a fixed-size recurrent state, carrying information beyond the local window without retaining every token explicitly (Yang et al., 2023; Kimi Team et al., 2025). Methods such as LoLCATs (Zhang et al., 2025) and Liger (Lan et al., 2025) approximate pretrained attention heads with combinations of LA and SWA, adapting mixing weights and/or branch parameters within a hybrid form. However, these hybrid designs failed to maintain performance on long-context retrieval tasks. Rather than combining both operators

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.

within each head, we therefore ask: *Can pretrained head specialization guide the choice of attention operator for each head?*

The functional heterogeneity of pretrained attention heads provides a useful basis for this decision. Retrieval-head analyses show that a small subset of heads contributes disproportionately to accessing distant information (Wu et al., 2024). Thus motivated, DuoAttention (Xiao et al., 2024) exploits head-level differences to preserve full attention for retrieval-critical heads, while HeadKV (Fu et al., 2025) allocates larger KV-cache budgets to heads important for retrieval and reasoning. These approaches use head importance to determine where token-level access to the past should be preserved.

We, in contrast, investigate a different use of this signal: guiding the allocation of recurrent memory when full attention is excluded. The intuition is to provide higher-scoring heads with a mechanism for carrying information beyond a fixed local window, while preserving sliding window attention in the remaining heads. We propose **Importance-Guided Attention Linearization (IGAL)**, which repurposes existing head-importance scores to guide the conversion from full self-attention to SWA or LA attention. In particular, for a given LA-head ratio, IGAL assigns higher-scoring heads to LA for long-range modeling and the remaining heads to SWA for local interactions. This makes pretrained head specialization the basis for operator assignment, rather than relying on parameter adaptation within each LA–SWA combination. Following Liger (Lan et al., 2025), we reuse the key projection for recurrent gating in LA and fine-tune the converted model without an additional distillation stage. The allocation strategy is compatible with a variety of recurrent update rules, such as GLA (Yang et al., 2023), allowing head selection to be studied independently of the specific LA mechanism.

We evaluate IGAL on Llama-3.1-8B and Qwen3-8B under task-specific fine-tuning for general-language and retrieval benchmarks, without full attention. On Llama-3.1-8B, assigning only 6.25% of GQA groups to LA, IGAL outperforms state-of-the-art (SOTA) linearization baselines on LM Evaluation Harness (Gao et al., 2024). With retrieval-specific fine-tuning on 1k-length sequences, it achieves $\mathbf{89.3\%}$ needle-in-a-haystack (NIAH) (Hsieh et al., 2024) accuracy at **32K**, where the other linearized baselines score nearly $\mathbf{0}$. As only the selected groups carry recurrent state, IGAL maintains a constant decoding cache over the evaluated 1K–32K contexts, approximately $\mathbf{292\times}$ smaller than full attention, and saving $\mathbf{25.5\%}$ total memory in a 32k context. In addition, under matched mixed-data fine-tuning, a single IGAL-R checkpoint raises 16–32K NIAH accuracy from 21.7% to $\mathbf{78.3\%}$ over static IGAL, with comparable general-language performance.

Overall, our contributions are threefold:

- We introduce IGAL, a full-attention-free conversion framework that repurposes pretrained head-importance scores to assign selected heads to LA and the remaining heads to SWA.
- We further introduce IGAL-R, a routing extension that sets the LA budget from the prompt length and learns which heads receive it with a prompt-level router.
- We demonstrate IGAL achieves the highest retention of general-language and long-context retrieval performance across Llama 3.1 and Qwen3 with $292\times$ less decoding-cache memory and IGAL-R achieves SOTA overall performance across both benchmarks with a single checkpoint, outperforming others by at least **56.6%** on 16–32K-length retrieval tasks.

2 Related Work

Converting pretrained Transformers. Mamba-in-the-Llama (Wang et al., 2024) progressively replaces attention layers with Mamba blocks, while LoLCATs (Zhang et al., 2025) combines linear attention with a local softmax window and matches its outputs to pretrained attention outputs through layer-wise distillation, followed by LoRA fine-tuning. LAWCAT (Liu et al., 2025) improve generalization beyond the training context length through causal convolutions and normalized gated linear attention. Liger (Lan et al., 2025) reuses the K-projection as the recurrent gate, enabling direct LoRA fine-tuning without a separate attention-distillation stage.

Beyond them, Benfeghoul et al. (2025) show that disabling linear attention often leaves accuracy largely unchanged, whereas disabling SWA causes substantial degradation. A complementary study by Jolicoeur-Martineau et al. (2026) replaces full attention with a sliding window and a small set

of initial attention-sink tokens, and found it maintain competitive performance without updating pretrained weights. Both work highlights the special importance of the SWA.

Head specialization and memory allocation. Attention heads differ in how strongly they depend on information from the distant context. Wu et al. (2024) identify a small subset of retrieval heads that attend to answer-bearing tokens across long distances. DuoAttention (Xiao et al., 2024) turns this distinction into an inference strategy by learning which heads can be restricted to recent tokens and attention sinks while preserving the frozen model's outputs on synthetic retrieval examples. At inference, retrieval heads retain full-history KV caches, while streaming heads use constant-length caches containing only sink and recent tokens. HeadKV (Fu et al., 2025) instead uses graded importance scores to distribute cache capacity across heads. It estimates these scores from retrieval and retrieval–reasoning calibration examples, then redistributes a shared KV-cache budget so that more important heads retain more context.

Selective attention and hybrid memory. Hybrid architectures also differ in how they preserve access to distant context. NHA (Du et al., 2026) represents long-term history as recurrently updated key–value slots and combines these slots with recent tokens from a sliding window. STILL (Meng et al., 2026) estimates token saliency from local attention distributions, retaining salient tokens beyond the local window while summarizing the remainder through linear attention. NAtS-L (Deng et al., 2026) learns a related partition at chunk granularity, using pooled chunk representations to score alternative attention operations. A complementary line of work adapts when global history is queried rather than which historical tokens are retained. AHA (Luo et al., 2025) uses a binary router for each token–head pair to choose between full-history attention and a local sliding window. L2A (Choudhary et al., 2026) first processes every token with local attention, then uses the resulting representation to decide whether an additional global-attention operation is needed.

3 Proposed Method

3.1 Importance-Guided Head Allocation

As shown in Fig. 1 (a), we use offline head-importance scores from HeadKV (Fu et al., 2025), which score individual query heads based on their attention to answer-relevant context during generation. In grouped-query attention (GQA) (Ainslie et al., 2023), multiple query heads share a KV head. To preserve this grouping, we assign all query heads in a KV group to the same attention operator. We score each group by the maximum importance of its query heads, preventing a high-scoring head from being diluted by lower-scoring heads in the same group.

For a Transformer with L layers and G KV heads per layer, we rank all LG groups and assign the top $K = \mathrm{round}(\rho LG)$ groups to LA, with the remaining groups assigned to SWA. Here, ρ denotes the target ratio of groups assigned to LA, and K is the corresponding *LA budget*. We index KV groups across all layers by j and denote the set of selected group indices by $\mathcal{S}_K$. For a token t, each group implements only its statically assigned operator:

$$\mathbf{o}_t = \begin{cases} \mathbf{o}_t^{\mathrm{LA}}, & j \in \mathcal{S}_K, \\ \mathbf{o}_t^{\mathrm{SWA}}, & \text{otherwise.} \end{cases} \tag{1}$$

Within each layer, head outputs retain their pretrained order and dimensions and are combined using the pretrained attention output projection.

3.2 Attention Conversion and Adaptation

Sliding-window attention. SWA retains the pretrained softmax computation and rotary position embedding (RoPE), but restricts each query to the current token and at most $w - 1$ preceding tokens. With a fixed window, each token attends to a bounded number of positions, yielding linear computational complexity in the sequence length and a bounded-size KV cache during decoding.

Normalized linear attention. Our method is compatible with various linear attention mechanisms. For simplicity, we use gated linear attention (GLA) (Yang et al., 2023) to illustrate the formulation below and in Fig. 1 (b). Different form the original GLA, we integrate it with Liger-style key-derived gates (Lan et al., 2025) and LAWCAT-style Q/K convolutions and attention-weight normalization (Liu et al., 2025). For a given LA head, let $\mathbf{q}_t, \mathbf{k}_t, \mathbf{v}_t \in \mathbb{R}^d$ denote the projected features,

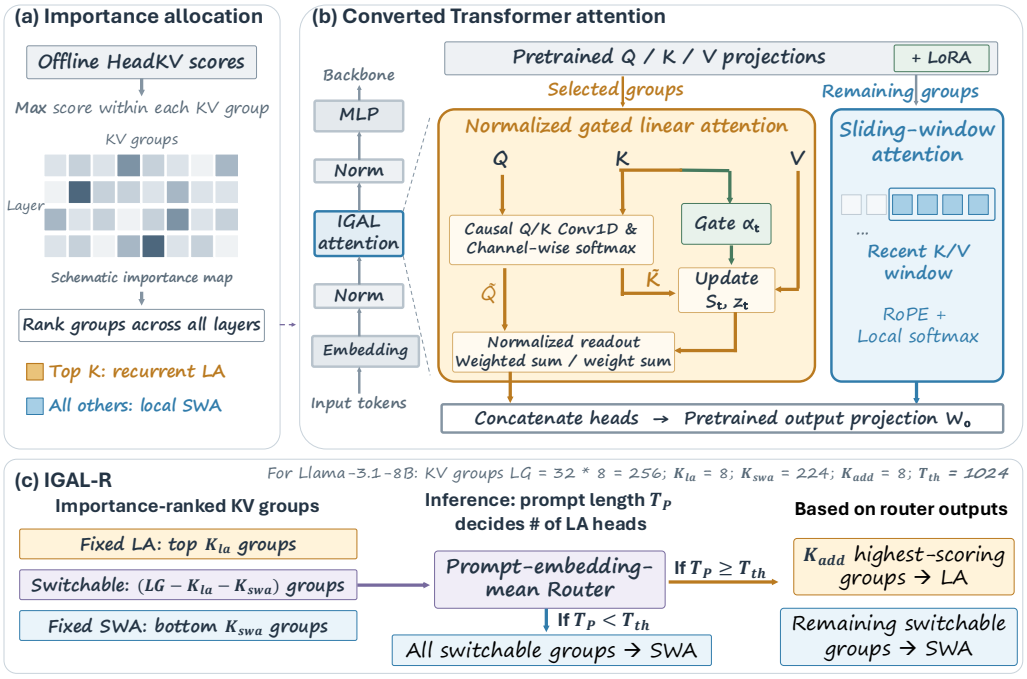

Figure 1: The overall framework of IGAL and IGAL-R.

with queries and keys taken before RoPE. We construct recurrent Q/K features as:

$$\widetilde{\mathbf{q}}_t = \mathrm{softmax}([\mathrm{Conv1D}_Q(\mathbf{q}_{1:t})]_t)\,, \qquad \widetilde{\mathbf{k}}_t = \mathrm{softmax}([\mathrm{Conv1D}_K(\mathbf{k}_{1:t})]_t)\,. \tag{2}$$

Here $\mathrm{Conv1D}_Q$ and $\mathrm{Conv1D}_K$ are separate causal depthwise convolutions over tokens (Gu & Dao, 2023), with head-specific kernels. Softmax is applied over the channel dimension of each head, transforming the convolutional features into non-negative, scale-normalized queries and keys suitable for the subsequent GLA.

Following Liger (Lan et al., 2025), we reuse the key vector before convolution to construct a channel-wise forget gate:

$$\boldsymbol{\alpha}_t = \sigma(\mathbf{k}_t)^{1/\gamma}, \tag{3}$$

where σ is the sigmoid function, the power is element-wise, and $\gamma > 0$ controls the decay rate. Each LA head maintains a KV state $\mathbf{S}_t \in \mathbb{R}^{d\times d}$ and a normalization state $\mathbf{z}_t \in \mathbb{R}^d$, both initialized to 0:

$$S_t = \mathrm{diag}(\boldsymbol{\alpha}_t)S_{t-1} + \widetilde{\mathbf{k}}_t\mathbf{v}_t^\top, \qquad \mathbf{z}_t = \boldsymbol{\alpha}_t \odot \mathbf{z}_{t-1} + \widetilde{\mathbf{k}}_t. \tag{4}$$

The same gate discounts past contributions in both states. And the attention output is

$$\mathbf{o}_t^{\mathrm{LA}} = \frac{\mathbf{S}_t^\top \widetilde{\mathbf{q}}_t}{\max(\widetilde{\mathbf{q}}_t^\top \mathbf{z}_t, \epsilon)}, \tag{5}$$

where the denominator normalizes the accumulated attention weights and $\epsilon > 0$ ensures numerical stability.

One-stage adaptation. To reduce conversion cost, we follow Liger's direct adaptation recipe (Lan et al., 2025) and fine-tune the converted model without the separate attention-distillation stage used by LoLCATs (Zhang et al., 2025) and LAWCAT (Liu et al., 2025). Under our experiment setup, this uses half as many training tokens and approximately half the wall-clock training time as these two-stage baselines. To ensure a stable and smooth transition during one-stage adaptation, we not only reuse the pretrained projection weights as Liger, but initialize the newly introduced Q/K convolutions near identity, thereby minimizing their perturbation to the pretrained representations at initialization.

We jointly train Q/K/V LoRA adapters (Hu et al., 2022) for both LA and SWA heads, together with the LA branch's Q/K convolutions, while all pretrained base model's weights remain frozen.

Because gates reuse the effective key projection, the K-projection adapters update keys and gates jointly. Training minimizes the standard next-token cross-entropy loss $\mathcal{L}_{\text{task}}$ over supervised target tokens. Initialization and implementation details are provided in Appendix A.2.

3.3 IGAL-R: Prompt-Conditioned Head Routing

IGAL uses the fixed allocation across inputs, but general-language and retrieval tasks can favor different LA fractions (Fig. 3). We further extend the IGAL to IGAL-R to accommodate this difference within a single checkpoint by keeping some KV groups fixed to LA or SWA and routing the remaining groups based on the input prompt.

Fixed and switchable groups. As shown in Fig. 1 (c), we first partition all the LG KV groups using the static head importance score ranking. Let $\mathcal{S}_{\text{la}}$ contain the K_{la} highest-scoring groups, which always use LA, and let $\mathcal{S}_{\text{swa}}$ contain the K_{swa} lowest-scoring groups, which always use SWA. The remaining $LG - K_{\text{la}} - K_{\text{swa}}$ groups are switchable between the two operators. A prompt with $T_p < T_{\text{th}}$ will be assigned only $\mathcal{S}_{\text{la}}$ for LA, where T_p is the prompt length in tokens and T_{th} is a pre-defined length threshold. For a prompt with $T_p \geq T_{\text{th}}$, the router scores all switchable groups and assigns LA to the K_{add} highest-scoring groups and SWA to the rest. These assignments remain fixed during both pre-fill and response generation for a given prompt and are thus recomputed once per prompt. The total LA count is therefore K_{la} for short prompts and $K_{\text{la}} + K_{\text{add}}$ for long prompts.

Prompt-conditioned selection. For a prompt x, we average its input embeddings and then apply parameter-free RMS normalization to obtain $\bar{\mathbf{e}}(x)$. Then a linear router, with weight matrix W^{R} and bias vector $\mathbf{b}^{\text{R}}$, produces one score per switchable group:

$$\mathbf{u}(x) = W^{\text{R}}\bar{\mathbf{e}}(x) + \mathbf{b}^{\text{R}}. \tag{6}$$

For long prompts, the K_{add} highest-scoring switchable groups use LA, with selection performed globally across all layers. The fixed sets $\mathcal{S}_{\text{la}}$ and $\mathcal{S}_{\text{swa}}$ are unchanged by routing.

Router training. We jointly train the router, Q/K/V adapters for both operators, and LA convolutions using the task loss $\mathcal{L}_{\text{task}}$ on a mixture of Alpaca and S-NIAH examples. We use a straight-through estimator: forward outputs use discrete selection, while the backward pass uses a smooth approximation with the same selection count. The router is initialized to recover the static importance-guided allocation at the larger budget. Appendix A.3 provide the training regimes, gradient estimator, and initialization.

Optional inference-time gate scaling. To reduce gate-induced decay for long prompts, we optionally rescale the forget gates at inference:

$$s(T_p) = 1 + \beta \log_2 \frac{\max(T_p, T_{\text{th}})}{T_{\text{th}}}, \qquad \boldsymbol{\alpha}_t^{\text{inf}} = \boldsymbol{\alpha}_t^{1/s(T_p)}. \tag{7}$$

where $\beta \in \{0, 1\}$ disables or enables scaling, and the power is applied element-wise. The scale is computed once from the prompt length and applied to forget gates throughout prefill, recurrent-state preparation, and subsequent decoding. Scaling is disabled (beta is set to zero) for short contexts $T_p \leq T_{\text{th}}$. Appendix A.4 details the motivation and interpretation of this scaling approach.

4 Experiments

4.1 Setup

We utilize **Llama-3.1-8B-Instruct** (Llama Team, AI @ Meta, 2024) and **Qwen3-8B** (Yang et al., 2025)as our baseline models. For the LA mechanism, we use KDA (Kimi Team et al., 2025) for IGAL and GLA (Yang et al., 2023) for IGAL-GLA and IGAL-R. Unless otherwise stated, we use the head importance scores from HeadKV (R) (Fu et al., 2025) for all experiments. For IGAL, we set the LA group ratio ρ to 6.25% for Llama-3.1-8B-Instruct and 12.5% for Qwen3-8B. We adopt the one-stage fine-tuning only framework from Liger (Lan et al., 2025) and the training data generation pipeline and model architecture from LAWCAT (Liu et al., 2025). For the mix dataset, we interleave Alpaca-clean (Taori et al., 2023) and NIAH datasets (Hsieh et al., 2024) during training, alternating between samples from each dataset. We evaluate performance on the LM Evaluation Harness (Gao et al., 2024) for zero-shot general capabilities and the RULER benchmark (Hsieh et al., 2024) for

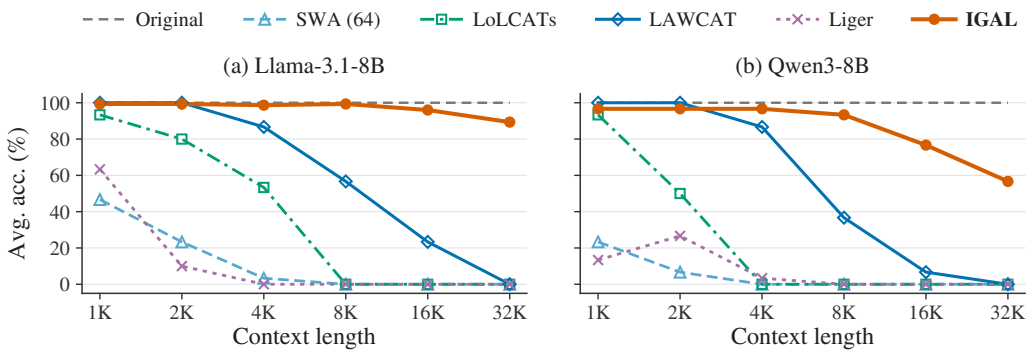

Figure 2: Comparison of different linearization methods on S-NIAH across Llama-3.1-8B and Qwen-3-8B models (fine-tuned on NIAH dataset).

Table 1: Comparison of different linearization methods on LM-EVAL with Llama-3.1-8B and Qwen3-8B (fine-tuned on the Alpaca-clean dataset).

Model	**PiQA** acc ↑	**ARC-e** acc ↑	**ARC-c** acc_norm ↑	**Hella.** acc_norm ↑	**Wino.** acc ↑	**Avg.** ↑
Llama-3.1-8B	80.25	81.86	55.20	79.31	73.72	74.07
Llama-3.1-8B (SWA64)	80.20	82.79	55.20	76.00	73.01	73.44
LoLCATs	80.03	80.98	54.35	**77.3**	**74.43**	73.42
LAWCAT	79.27	80.01	49.57	71.78	58.88	67.90
Liger	79.22	81.31	52.3	75.35	72.3	72.10
IGAL (Ours)	**80.68**	**82.99**	**55.8**	76.45	72.53	**73.69**
Qwen-3-8B	76.60	83.50	56.70	74.86	68.40	72.01
Qwen-3-8B (SWA64)	77.64	**84.30**	**60.41**	74.62	**73.00**	**73.99**
LoLCATs	78.56	83.8	58.45	**75.96**	72.06	73.77
LAWCAT	77.58	82.58	56.66	70.39	61.8	69.80
Liger	78.13	83.63	59.56	73.11	70.24	72.93
IGAL (Ours)	**78.78**	84.09	**60.41**	73.06	71.9	73.65

long-context retrieval (S-NIAH tasks). Full hyperparameters and router training details are provided in Appendix A.1.

4.2 Task-Specific Fine-Tuning

Short-context language understanding. Table 1 shows that IGAL preserves competitive general-language performance across both backbones. On Llama-3.1-8B, IGAL achieves the highest average score among the evaluated linearization methods (73.69), trailing the original model by only 0.38 points. On Qwen3-8B, IGAL reaches 73.65, exceeding the original model by 1.64 points and remaining close to SWA (73.99) and LoLCATs (73.77). At the task level, IGAL achieves the best or joint-best scores on both PiQA and ARC-c across the two backbones. These results indicate that IGAL retains strong short-context capabilities, providing a foundation for improving long-context retrieval without the substantial average-score reductions observed for LAWCAT.

Long-context retrieval. Fig. 2 shows that IGAL's advantage becomes more pronounced on NIAH. Averaged over NIAH-1/2/3 and six context lengths from 1K to 32K, IGAL achieves 97.00% accuracy on Llama-3.1-8B and 86.11% on Qwen3-8B, surpassing the strongest retrieval baseline, LAWCAT, by 35.89 and 31.11 percentage points, respectively. SWA, LoLCATs, and Liger reach zero accuracy by 8K on both backbones, despite some of these methods remaining competitive on LM-EVAL. In contrast, IGAL retains 89.33% and 56.67% accuracy at 32K on Llama-3.1-8B and Qwen3-8B, respectively, where all evaluated linearized baselines score zero. Although retrieval

Table 2: Comparison of different linearization methods on Llama-3.1-8B under the mixed-data protocol. IGAL-R uses 3.125% LA for prompts shorter than T_{th} and 6.25% otherwise.

Method	Fine-tuning data	Allocation (%)	LM Eval Avg. (%)	NIAH (2-8K) Avg. (%)	NIAH (16-32K) Avg. (%)
IGAL-GLA	Alpaca	6.25	72.93	1.11	0.00
IGAL-GLA	NIAH	6.25	56.87	98.67	83.33
IGAL-GLA	Mix	6.25	72.06	70.00	21.67
SWA	Mix	–	72.77	5.00	0.00
LoLCATs	Mix	–	**73.76**	38.89	0.00
LAWCAT	Mix	–	68.47	84.44	5.00
Liger	Mix	–	71.90	30.83	0.00
IGAL-R (Ours)	Mix	3.125/6.25	72.47	**94.44**	**78.33**

still degrades at longer contexts, particularly on Qwen3-8B, IGAL substantially extends effective retrieval while maintaining competitive short-context performance.

4.3 Mixed-Data Fine-Tuning

Table 2 evaluates whether a single checkpoint can support both general-language tasks and long-context retrieval. Except for the two task-specific IGAL-GLA reference checkpoints, all methods in Table 2 are fine-tuned on the same Alpaca/NIAH mixture. Task-specific IGAL-GLA checkpoints exhibit a pronounced trade-off: Alpaca-only fine-tuning achieves 72.93% on LM Eval but 0.00% on 16–32K NIAH, whereas NIAH-only fine-tuning reaches 83.33% on the latter but reduces LM Eval to 56.87%. Mixed-data fine-tuning substantially improves retrieval over Alpaca-only training while maintaining similar general-language performance, but its 16–32K accuracy remains limited to 21.67%. Thus, combining the two data sources alone does not recover the strengths of both task-specific checkpoints.

Under mixed-data fine-tuning, IGAL-R improves over static IGAL-GLA by 24.44 and 56.66 percentage points on 2–8K and 16–32K NIAH, respectively, while slightly improving LM Eval from 72.06% to 72.47%. It achieves the highest retrieval accuracy among all mixed-data methods in both length ranges, although LoLCATs obtains the highest LM Eval score. Relative to the task-specific references, IGAL-R comes within 0.46 percentage points of the Alpaca-only checkpoint on LM Eval and 5.00 points of the NIAH-only checkpoint on 16–32K retrieval. These results show that the complete IGAL-R configuration substantially narrows the gap to separate task-specific fine-tuning while serving both evaluations with a single checkpoint. The contributions of each component in the routing design are examined in Table 5.

Table 3: Decode-time memory at batch size 1. Persistent cache excludes model weights, while serving memory denotes the driver-reported GPU memory held by the process during decoding.

	Persistent cache (MiB)			Serving memory (GiB)		
Method	**1K**	**8K**	**32K**	**1K**	**8K**	**32K**
Llama-3.1-8B	128.0	1,024.0	4,096.0	15.45	16.69	20.54
SWA64	8.0	8.0	8.0	15.31	15.31	15.31
LAWCAT	66.5	66.5	66.5	15.47	15.51	15.60
Liger	74.5	74.5	74.5	15.48	15.52	15.61
IGAL	14.0	14.0	14.0	15.31	15.31	15.31

4.4 Memory Efficiency

Table 3 show the comparison of persistent decoding cache and whole serving GPU memory among different methods. IGAL maintains a 14.0 MiB cache across the evaluated 1K–32K contexts, re-

Table 4: Comparison of different head importance metrics with IGAL-GLA on LM-EVAL & NIAH with Llama-3.1-8B. (fine-tuned on Alpaca-clean dataset and NIAH dataset respectively).

Metric	LM Eval Avg. (%)	NIAH (1-8K) Avg. (%)	NIAH (16-32K) Avg. (%)
Random	71.90	85.83	28.33
DuoAttn	72.25	94.17	76.67
HeadKV (R2)	72.26	96.67	80.00
HeadKV (R)	**72.93**	**98.67**	**83.33**

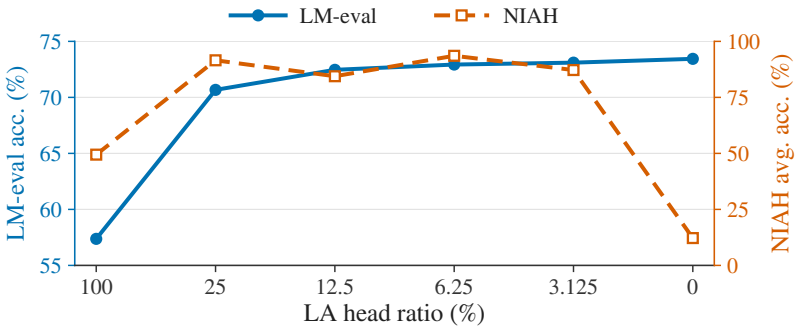

Figure 3: Comparison of LA head ratios on LM-Eval and NIAH with Llama-3.1-8B. We use IGAL-GLA with the importance score from HeadKV (R).

ducing cache size by approximately 292× relative to full attention at 32K. Moreover, sparse recurrence yields roughly a fivefold cache reduction over the dense-recurrence baselines (66.5–74.5 MiB), while requiring only about 6 MiB more than the all-SWA variant. As for the total serving memory, the shared 14.96 GiB backbone dominates overall GPU memory, so the relative reduction is smaller than that in persistent cache. At 32K, IGAL uses 15.31 GiB compared with 20.54 GiB for full attention, a reduction of approximately 25.5%. Thus, sparse allocation reduces the persistent-state cost of bounded-memory decoding while keeping the overall footprint close to the all-SWA reference. Measurement details are provided in Appendix A.5.

4.5 Ablation Study

Effect of the head-importance metric. Table 4 compares random allocation with allocation guided by DuoAttention scores (Xiao et al., 2024) and HeadKV retrieval (R) and retrieval–reasoning (R2) scores (Fu et al., 2025). All variants use global allocation on Llama-3.1-8B under a fixed 6.25% LA budget. All importance-guided variants outperform random selection, with HeadKV (R) achieving the best results across the three evaluation settings. On 16–32K NIAH, it reaches 83.33%, compared with 80.00% for HeadKV (R2) and 28.33% for random allocation. Unlike HeadKV's original setting, which distributes KV entries across heads to preserve retrieval and reasoning capabilities, IGAL selects a small subset of groups to retain information beyond the local attention window. We hypothesize that this distinction favors R on NIAH: its direct-retrieval criterion prioritizes groups involved in recovering distant target information, aligning the limited recurrent budget more closely with needle retrieval than the broader retrieval–reasoning criterion.

Effect of the LA head ratio Fig. 3 shows that reducing the LA-head fraction consistently improves LM-eval performance, whereas NIAH accuracy varies non-monotonically. Among the evaluated configurations, a fraction of 6.25% achieves the highest NIAH accuracy (93.56%) while maintaining an LM-eval score of 72.93, only 0.51 points below the best observed score. Further reducing the fraction to 3.125% improves LM-eval by just 0.17 points but decreases NIAH accuracy by 6.36 percentage points; removing LA entirely reduces NIAH accuracy to 12.2%. These results suggest that a small LA component helps preserve long-range retrieval, while allocating more heads to LA does not necessarily improve performance. We therefore adopt 6.25% as the default static LA-

Table 5: Router ablations on Llama-3.1-8B after mixed-data fine-tuning. "prompt" denotes the mean-pooled prompt embedding; "prompt+Q+len" additionally concatenates this embedding with the mean of the last 64 prompt-token embeddings and scaled log prompt length. "LA freeze" suppresses LA-branch gradients from Alpaca batches (T: enabled; F: disabled). β is the optional inference-time gate scaling, defined in Eq. 7; "0" indicates that scaling is not used.

Method	Router Input	LA Freeze	Retention β	LM Eval Avg.(%)	NIAH (2-8K) Avg.(%)	NIAH (16-32K) Avg.(%)
IGAL-GLA	–	F	0	72.06	70.00	21.67
Token-wise router	hidden states	F	0	72.16	72.22	20.00
IGAL-R	prompt+Q+len	T	1	73.05	95.56	56.67
IGAL-R	prompt+Q+len	F	0	72.90	75.56	16.67
IGAL-R	prompt+Q+len	F	1	72.90	87.78	40.00
IGAL-R	prompt	T	1	**73.12**	**96.67**	50.00
IGAL-R	prompt	F	0	72.47	90.00	51.67
IGAL-R	prompt	F	1	72.47	94.44	**78.33**

head fraction, retaining strong retrieval performance with a limited reduction in general-language accuracy.

IGAL-R ablations. Table 5 compares static IGAL, our token-wise router baseline, and IGAL-R after mixed-data fine-tuning. The token-wise router uses a linear binary routing function to dynamically select the attention operator of each head at each token. It offers limited gains over static IGAL, suggesting that changing the head type for each token is difficult to train and does not improve the model performance.

We also examine router inputs, comparing the mean-pooled prompt embedding ("prompt") with an augmented input ("prompt+Q+len") that concatenates this embedding with the mean of the last 64 prompt-token embeddings and scaled log prompt length. Across matched configurations, the simpler prompt-mean input generally achieves better retrieval performance, suggesting that the additional features may complicate router training without consistently improving generalization. Besides, we also check the impact of "LA freeze", which suppresses LA-branch gradients from Alpaca batches. It modestly improves LM Eval but reduces 16–32K NIAH for prompt-mean routing, although it benefits the augmented-input configuration. Restricting LA updates may regularize general-language adaptation, while preventing the recurrent branch from co-adapting with the rest of the model on Alpaca examples, potentially limiting length generalization.

Finally, we ablate the effect of β, which controls the optional inference-time gate scaling (0: disabled; 1: enabled). Enabling scaling consistently improves NIAH while leaving paired LM Eval scores unchanged. The gains are consistent with reduced gate-induced attenuation of stored prompt information: moving forget gates closer to one helps preserve this information during response generation. We therefore use prompt-mean input, no LA freeze, and $\beta = 1$ as the default IGAL-R configuration, which achieves the best overall performance.

5 Conclusions

We introduced **IGAL**, a full-attention-free conversion framework that uses pretrained head importance to assign recurrent LA and SWA to different heads. Our results show that a small, carefully selected subset of linear attention can preserve general-language performance and retain substantial long-range retrieval with a fixed-size decoding cache. Matched-budget ablations highlight the value of head selection beyond the amount of recurrent memory. We further propose **IGAL-R**, which uses the prompt length to decide LA budget and router a certain group heads between LA and SWA. It significantly improve long-context retrieval over static IGAL at comparable general-language performance in a single checkpoint trained on mixed general-language and retrieval data. Together, these results motivate treating specialization-aware memory allocation as a powerful design choice alongside recurrent operators and training recipes: future bounded-memory models should consider not only *how memory is updated*, but also *where it is allocated and how its budget adapts to the input*.

6 AI USE STATEMENT

We used generative AI tools to assist with drafting and editing selected parts of the manuscript, and generate the editable powerpoint version of Fig. 1. We comprehensively reviewed and revised all AI-assisted content, including Fig. 1, and manually prepared and verified all citations and references. We take full responsibility for the final manuscript, including all claims, results, and references.

7 REPRODUCIBILITY STATEMENT

Section 3 and Appendix A.2 describes IGAL and IGAL-R, including head-importance-based allocation, attention conversion, and prompt-level routing. Section 4 specifies the pretrained models, training data, baselines, and evaluation benchmarks. Appendices A.1 and A.3 provides additional implementation details, training hyper-parameters, and inference settings. We will release the code implementation and training and evaluation scripts upon publication.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints, 2023. URL https://arxiv.org/abs/2305.13245.

Martin Benfeghoul, Teresa Delgado, Adnan Oomerjee, Haitham Bou Ammar, Jun Wang, and Zafeirios Fountas. Untangling component imbalance in hybrid linear attention conversion methods, 2025. URL https://arxiv.org/abs/2510.05901.

Sakshi Choudhary, Aditya Chattopadhyay, Luca Zancato, Elvis Nunez, Matthew Trager, Wei Xia, and Stefano Soatto. Learning when to attend: Conditional memory access for long-context llms, 2026. URL https://arxiv.org/abs/2603.17484.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Difan Deng, Andreas Bentzen Winje, Lukas Fehring, and Marius Lindauer. Neural attention search linear: Towards adaptive token-level hybrid attention models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.03681.

Jusen Du, Jiaxi Hu, Tao Zhang, Weigao Sun, and Yu Cheng. Native hybrid attention for efficient sequence modeling, 2026. URL https://arxiv.org/abs/2510.07019.

Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. Not all heads matter: A head-level KV cache compression method with integrated retrieval and reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=FJFVmeXusW.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Saurav Goel. Paul graham essay collection dataset. https://huggingface.co/datasets/sgoel9/paul_graham_essays, 2023. Accessed: 2025-05-05.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Alexia Jolicoeur-Martineau, Rhea Sanjay Sukthanker, Pashmina Cameron, and Emy Gervais. Sliding-window beats linear attention, 2026. URL https://arxiv.org/abs/2608.28444.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, T. Y. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi linear: An expressive, efficient attention architecture, 2025. URL https://arxiv.org/abs/2510.26692.

Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. Booksum: A collection of datasets for long-form narrative summarization. 2021.

Disen Lan, Weigao Sun, Jiaxi Hu, Jusen Du, and Yu Cheng. Liger: Linearizing large language models to gated recurrent structures. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=1PfZs0xC2v.

Zeyu Liu, Souvik Kundu, Lianghao Jiang, Anni Li, Srikanth Ronanki, Sravan Bodapati, Gourav Datta, and Peter A. Beerel. Lawcat: Efficient distillation from quadratic to linear attention with convolution across tokens for long context modeling, 2025. URL https://arxiv.org/abs/2509.18467.

Llama Team, AI @ Meta. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

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

Xuan Luo, Kailai Zhang, and Xifeng Yan. Learning when not to attend globally, 2025. URL https://arxiv.org/abs/2512.22562.

Weikang Meng, Liangyu Huo, Yadan Luo, Jiawen Guan, Jingyi Zhang, Yingjian Li, and Zheng Zhang. Still: Selecting tokens for intra-layer hybrid attention to linearize llms, 2026. URL https://arxiv.org/abs/2602.02180.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Junxiong Wang, Daniele Paliotta, Avner May, Alexander M Rush, and Tri Dao. The mamba in the llama: Distilling and accelerating hybrid models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=uAzhODjALU.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. Retrieval head mechanistically explains long-context factuality, 2024. URL https://arxiv.org/abs/2404.15574.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. *arXiv*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

Michael Zhang, Simran Arora, Rahul Chalamala, Benjamin Frederick Spector, Alan Wu, Krithik Ramesh, Aaryan Singhal, and Christopher Re. LoLCATs: On low-rank linearizing of large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8VtGeyJyx9.

Table 6: Reference IGAL / IGAL-R configuration for Llama-3.1-8B.

Group	Setting	Value
Backbone	layers L / query heads H / KV heads G	32 / 32 / 8
	head dim d, feature dim m	128, 128
Allocation	importance scores	HeadKV (Fu et al., 2025), group max
	granularity	global Top-K over all $LG = 256$ groups
	LA fraction ρ (static IGAL)	6.25% ($K = 16$ groups)
Operators	SWA window w	64
	Q/K conv width c	4
	gate exponent γ (Eq. 3)	16
	normalizer floor ϵ	10^{-6}
IGAL-R	$K_{\text{la}}/K_{\text{swa}}/K_{\text{add}}$	3.125% / 87.5% / 3.125% (8 / 224 / 8 groups)
	regime threshold T_{th}	1024 tokens
	question window W	64
	surrogate temperature T_{r}	0.5
	retention scaling β (Eq. 7)	0 in training, 1 at long-context evaluation
Optimization	optimizer / scheduler	AdamW, ReduceLROnPlateau
	learning rate (router learning rate)	5×10^{-4} (3×10^{-5})
	batch size / grad. accumulation / epochs	1 / 8 / 2
	gradient clipping	1.0
	LoRA rank / alpha / dropout / targets	8 / 8 / 0.0 / Q,K,V

A Appendix

A.1 Training Details

For the S-NIAH task, different from LAWCAT (Liu et al., 2025), we only generate 8456 samples with S-NIAH-3 format to build the training dataset, without mixing any S-NIAH-1 format data. We also use the BookSum dataset (Kryściński et al., 2021) as the contextual 'haystack' in training, and use the Paul Graham Essay Collection Dataset Goel (2023) in evaluation.

For the distillation-based baselines (LAWCAT, LoLCATs) we follow hyperparameters similar to LoLCATs (Zhang et al., 2025). For distillation, we use AdamW (Loshchilov & Hutter, 2019) with an initial learning rate of 0.01 and ReduceLROnPlateau for scheduling. The batch size is 1 with 8 gradient accumulation steps, and training runs for 2 epochs. For fine-tuning, we use the same optimizer and scheduler, but reduce the initial learning rate to 1e-4. All other settings remain unchanged, and training also runs for 2 epochs. LoRA (Hu et al., 2022) is applied to the query, key, value, and output projections with rank 8 and alpha 16.

IGAL and IGAL-R instead use the one-stage recipe of Liger (Lan et al., 2025): no attention-distillation stage, and the converted model is fine-tuned directly with the task objective. We use AdamW with an initial learning rate of 5×10^{-4} and the same ReduceLROnPlateau schedule, batch size 1 with 8 gradient accumulation steps, gradient clipping at norm 1.0, and 2 epochs; the IGAL-R prefill router forms a separate parameter group with learning rate 3×10^{-5}. LoRA is applied to the query, key and value projections only, with rank 8, alpha 8 and dropout 0, so the output projection W_O stays frozen as described in Section 3.2; the added Q/K convolutions (and the router, for IGAL-R) are trained in full and saved with the adapter. Table 6 lists the full configuration.

A.2 IGAL Implementation Details

Reference configuration. Unless stated otherwise, all Llama-3.1-8B results use the configuration in Table 6. The model has $L = 32$ layers, $H = 32$ query heads and $G = 8$ KV heads per layer, so head allocation ranges over $LG = 256$ GQA groups; at $\rho = 6.25\%$ exactly $K = 16$ of them are linear. The SWA window $w = 64$ is fixed in all experiments in this paper.

Initialization. The converted model starts as close to the pretrained backbone as the architecture allows, so that fine-tuning only has to correct the operator substitution. Pretrained Q/K/V/O projec-

tions are reused unchanged and W_O is frozen. The added depthwise Q/K convolutions are initialized near identity (a one-hot kernel at the current token), so at step zero the convolved features of Eq. 2 reduce to the pretrained projections. SWA heads retain the backbone's softmax attention and rotary embedding and therefore require no initialization beyond the LoRA adapters, which start at zero.

A.3 IGAL-R: Training Details

Prompt-feature normalization. For $\mathbf{v} \in \mathbb{R}^D$, parameter-free RMS normalization is

$$\mathrm{RN}(\mathbf{v}) = \frac{\mathbf{v}}{\sqrt{\|\mathbf{v}\|_2^2/D + \epsilon_{\mathrm{r}}}}, \tag{8}$$

with $\epsilon_{\mathrm{r}} > 0$. let $M = LG - K_{\mathrm{la}} - K_{\mathrm{swa}}$, then the global scorer has parameters $W^{\mathrm{R}} \in \mathbb{R}^{M \times D}$ and $\mathbf{b} \in \mathbb{R}^M$.

Fixed-budget sigmoid surrogate. For a long-regime example, given scores $\mathbf{u}$, we solve

$$p_j = \sigma\left(\frac{u_j - \eta}{T_{\mathrm{r}}}\right), \qquad \sum_{j=1}^{M} p_j = K_{\mathrm{add}}, \tag{9}$$

for a shared threshold η by bisection. For finite logits and $T_{\mathrm{r}} > 0$, the sum is continuous and strictly decreasing in η, with limits M and 0, so the threshold is unique. The constraint is differentiated implicitly, rather than treating the bisection result as a constant. Writing $\omega_j = p_j(1 - p_j)/T_{\mathrm{r}}$ gives

$$\begin{aligned} \frac{\partial \eta}{\partial u_k} &= \frac{\omega_k}{\sum_i \omega_i}, \\ \frac{\partial p_j}{\partial u_k} &= \omega_j \left(\delta_{jk} - \frac{\omega_k}{\sum_i \omega_i}\right). \end{aligned} \tag{10}$$

For upstream derivatives $v_j = \partial \mathcal{L}/\partial p_j$, an equivalent vector–Jacobian product is

$$\frac{\partial \mathcal{L}}{\partial u_k} = \omega_k \left(v_k - \frac{\sum_j \omega_j v_j}{\sum_j \omega_j}\right). \tag{11}$$

These identities follow directly from the stated constraint.

Hard forward, surrogate backward. For long-regime examples, let $\mathbf{g}^{\mathrm{hard}}$ indicate $\mathcal{J}(x)$ and define $p_j = \sigma((u_j - \eta)/T_{\mathrm{r}})$, with $T_{\mathrm{r}} > 0$ and η chosen so that $\sum_j p_j = Q$. We use the straight-through mask

$$\widetilde{\mathbf{g}} = \mathrm{sg}[\mathbf{g}^{\mathrm{hard}} - \mathbf{p}] + \mathbf{p}, \tag{12}$$

where sg denotes stop-gradient. For switchable groups at $t \geq t_0$, we re-define forward output to be:

$$\mathbf{o}_{\ell h t} = \widetilde{g}_{\ell g(h)} \mathbf{o}^{\mathrm{LA}}_{\ell h t} + (1 - \widetilde{g}_{\ell g(h)}) \mathbf{o}^{\mathrm{SWA}}_{\ell h t}. \tag{13}$$

This produces hard forward outputs and surrogate gradients from the same task loss; only long-regime response-token losses train the selector. The initial router scores recover the static allocation $\mathcal{I}_{K_1}$. The straight-through mask has the same value as $\mathbf{g}^{\mathrm{hard}}$ in the forward pass. In backward propagation, the task loss reaches $\mathbf{u}$ through Eq. 13 and the sigmoid surrogate. This uses both candidate outputs, or an equivalent backward computation; hard indexing alone would not implement the stated gradient path. The resulting training cost is distinct from selected-state decoding cost. Only response-token losses on long-regime examples train the selector.

Initialization. We initialize $W^R = 0$. Among the switchable groups, the K_{add} groups with the highest static importance scores receive bias $+1$, and the remaining groups receive bias -1. Together with the always-LA groups, this recovers the static importance-guided allocation with $K_{\mathrm{la}} + K_{\mathrm{add}}$ LA groups.

A.4 Inference-Time Gate Scaling: Motivation and Interpretation

For a fixed sequence of gate values and a fixed prompt scale $s \geq 1$, elementwise rescaling gives

$$\prod_{v=i+1}^{t} \alpha_v^{1/s} = \left(\prod_{v=i+1}^{t} \alpha_v\right)^{1/s}. \tag{14}$$

Equivalently, cumulative log-decay is divided by s. This reduces decay of past contributions for $s > 1$. For a time-invariant scalar gate $\alpha \in (0, 1)$, the continuous half-life is $\log(1/2)/\log\alpha$, and rescaling multiplies that quantity by s. For token-dependent gates, this does not imply that the number of token steps needed to reach a fixed retention level is always multiplied by exactly s. Moreover, the product identity conditions on the gate sequence; it does not assert that hidden states or gate inputs stay unchanged when the full model is evaluated with a modified recurrence. The value of s is a chosen logarithmic scaling rule based on context length T_p. With $T_{th} = 1024$ and scaling enabled, $s = 2$ at $T_p = 2048$ and $s = 6$ at $T_p = 32768$. For $T_p \le T_{th}$, it is exactly 1.

A.5 How the Decode State Is Measured

Cache size. The returned cache is walked exhaustively: every layer, and every tensor in every slot it holds (`recurrent_state`, `attn_state`, `conv_state`, `router_state`), with shapes and dtypes recorded as found rather than assumed. Tuple and dict slots are recursed into. Because `numel×element_size` reports what a tensor advertises while the allocator can only free a whole storage, the underlying storages are walked separately and deduplicated by address; the two agree for every configuration reported here, so the advertised sizes are also the resident ones. Routing metadata (the prompt length, the selected route, and any router state) is included in every total and is under 1 MiB in all configurations.

Cross-check. An analytic model derived from the live configuration — head counts taken from the assignment, state widths read from the instantiated tensors — is required to reproduce the measured recurrent, window and convolution state exactly, per layer, for every configuration at every length. It does. The two are independent only in the sense that a disagreement would indicate the walk had missed a slot; agreement is what licenses quoting a single architectural number per method.

Serving footprint. The driver-reported total for one decode step, taken after blocks left cached by the prefill are released, so the figure describes decoding rather than the prefill high-water mark. It includes the CUDA context (320 MiB, a fixed per-process cost) and allocator slack: memory PyTorch has reserved but is not using. Slack depends on where the allocator happened to place the cache during prefill and is therefore not a property of the architecture; the largest case we observed among converted models was 782 MiB of slack behind a 26.1 MiB cache.

Conditions. Batch size 1, unpadded, greedy, in bf16, with the 64-token SWA window fixed in the attention implementation. Recurrent states are allocated in fp32 regardless of the model dtype. Each configuration runs in its own process, since the peak-memory counters are process-wide and one shared process would report the largest configuration's peak for every row measured after it. Weights are captured once after loading and excluded throughout; every model here carries the same 14.96 GiB of bf16 parameters. Only one logit position is materialized per forward, so the $32{,}768 \times 128{,}256$ logit tensor never appears. No adapter is loaded: cache size is a function of shapes alone, so these figures hold for any trained checkpoint of the same architecture. State scales with batch, so a multi-sequence serving estimate requires its own measurement.

A.6 Computational Complexity

For a single layer with H_{LA} recurrent heads, our implementation evaluates the recurrence only on the recurrent heads and the windowed kernel on all H heads, discarding the windowed outputs of recurrent heads; the Q/K convolutions are likewise applied to all heads. Evaluating the windowed kernel on all heads avoids gathering and scattering head subsets, and with $w = 64$ its cost is small relative to the projections. The attention computation therefore has complexity

$$\mathcal{O}(T[Hwd + H_{\mathrm{LA}} md]), \tag{15}$$

excluding shared projections and fixed-width convolutions, whose computational costs are also linear in T for fixed model dimensions. During autoregressive decoding, the persistent attention state of a layer consists of the windowed key–value cache, the recurrent matrix and normalizer states, and the Q/K convolution buffers, totaling

$$\underbrace{2Gwd}_{\text{window KV}} + \underbrace{H_{\mathrm{LA}}\, m(d+1)}_{\text{recurrent}} + \underbrace{2Hcd}_{\text{conv}} \tag{16}$$

elements, where c is the convolution width. Only the recurrent term depends on the allocation: the window cache is kept for all G KV groups and the convolution buffers for all H query heads.

For IGAL-R, the prefill evaluates the recurrence over the switchable pool $\mathcal{P}$ (i.e., H_{LA} in Eq. 15 counts all switchable heads during prefill), but only the recurrent states of the routed groups $\mathcal{A}(x)$ are retained, so the decoding footprint equals that of static IGAL at the same budget.

With fixed w, m, d, and c, the attention computation scales linearly with sequence length, while the persistent attention-state memory remains constant with respect to context length. Since the remaining model components are also linear in sequence length for fixed model dimensions, the overall model has $\mathcal{O}(T)$ computational complexity.