EFFICIENT MAXIMUM INNER PRODUCT SEARCH FOR TOP-k ATTENTION VIA HIERARCHICAL ROUTERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Attention enables parallel token retrieval but at quadratic memory and computation relative to the sequence length, limiting long-context applicability. Sparse top-k attention reduces this computation while maintaining competitive accuracy. Although GPU-friendly, these often rely on clustering or hashing to avoid computing the full attention matrix, employing approximate or decoupled partition-based routing that can limit retrieval accuracy. To this end, we propose ***Hierarchical Router*** (HIROUTER), a sparse top-k attention mechanism that leverages a learned routing function within a hierarchical structure, coupling retrieval with attention by co-training queries, keys, and routing functions end-to-end. To make the learned router effective, we design two routing rules: relevant tokens should co-route with the query, and bucket occupancy should remain balanced to avoid collapse and support uniform GPU workloads. By optimizing two surrogate losses over a learned multi-level hierarchy, HIROUTER realizes these rules, partitioning tokens into discrete buckets in linear time and enabling accurate per-sequence retrieval with GPU-friendly layouts. On standard benchmarks, HIROUTER matches full-attention performance, while matching/surpassing full-attention accuracy on sequences beyond 4K tokens with up to 2× faster inference than `FlashAttention`.

1 INTRODUCTION

Transformers (Vaswani et al., 2017) have become indispensable for sequence modeling across a wide range of domains (OpenAI, 2023) such as natural language processing (Devlin et al., 2019; Brown et al., 2020; OpenAI, 2023; Jiang et al., 2024) and computer vision (Dosovitskiy et al., 2021; Ramesh et al., 2021; Brooks et al., 2024). This is in part thanks to self-attention (Bahdanau et al., 2015), which constructs rich token representations through sequence-parallel information mixing. Yet despite these successes, self-attention incurs quadratic memory and computational costs as the sequence length T grows, posing a major obstacle for long-sequence applications (Child et al., 2019; Beltagy et al., 2020; Tay et al., 2021; Huang, 2025) and other real-world, large-scale settings.

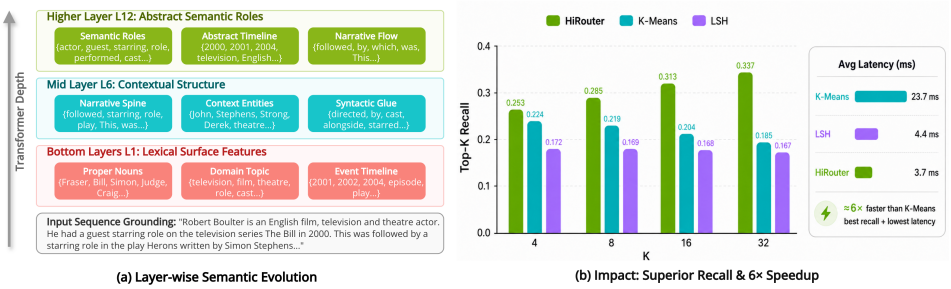

Figure 1: Structure and retrieval impact on WikiText-103. **(a)** Post-hoc summaries of different buckets shown for interpretability, where different layers show separation of different features/structures. **(b)** Top-k retrieval recall and average routing latency on sequences of length 4096. HIROUTER achieves higher recall with substantially lower latency than K-Means and LSH.

Several mitigation strategies have been proposed. Deng et al. (2024) notice an inherent sparsity to attention, motivating **top-k attention** that restricts computation to the k most relevant tokens,

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.

reducing memory usage and FLOPs while preserving performance (Roy et al., 2021; Kitaev et al., 2020; Gupta et al., 2021; Bertsch et al., 2023; Mao et al., 2024). Existing methods typically rely on partition-based routing mechanisms (Kitaev et al., 2020; Roy et al., 2021), such as hashing or clustering, to form a small candidate set for each query. These methods can fail in two ways. First, relevant top-k keys may be assigned to buckets different from the query bucket, causing recall loss before sparse attention is even computed. Second, token distributions can collapse into a few overloaded buckets under skewed data (Kitaev et al., 2020; Malkov and Yashunin, 2020), making the candidate set less selective and creating uneven GPU workloads. Thus, effective routed top-k attention should satisfy two design rules: top-k neighbors should co-route with the query, and bucket occupancy should remain balanced to avoid collapse while supporting efficient GPU execution.

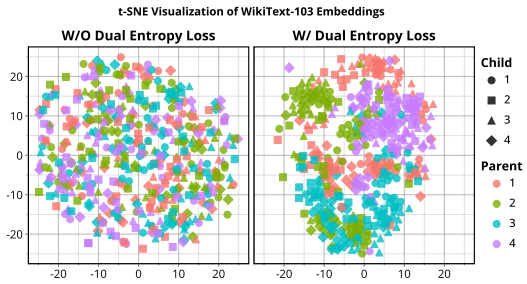

Figure 2: Two-level HIROUTER with four buckets per level, with colors/markers denoting first and second levels. Traditional attention (left) scatters embeddings while HIROUTER (right) clusters them, ensuring that similar tokens fall into the same bucket. This improves retrieval, as queries need only search within well-formed buckets.

We address these with HIROUTER, a hierarchical routing attention mechanism that couples routing alongside representation training. Rather than treating routing as a post-hoc operation on fixed embeddings, HIROUTER co-trains queries, keys, and routing centroids end-to-end, allowing representations to evolve toward a geometry the router can leverage. This makes both routing objectives achievable in practice without decoupling the router from the attention objective: it draws relevant tokens toward shared centroids while simultaneously balancing bucket usage. To make the two rules trainable, we introduce two entropy-based surrogate objectives: (1) a sample-wise entropy surrogate that sharpens token–centroid assignments, encouraging relevant tokens to co-route with the query and improving top-k recall (Figure 2), and (2) a balanced-assignment entropy surrogate to control average bucket occupancy, preventing collapse and enabling efficient block-wise attention (Yuan et al., 2025) with uniform GPU workloads. Together with a learned multi-level hierarchy, these yield accurate per-sequence retrieval while preserving linear-time routing in sequence length.

We evaluate HIROUTER on a broad range of benchmarks spanning language modeling and natural language understanding. Across tasks, hierarchical routing remains competitive with strong transformer baselines while substantially improving computational efficiency. In particular, our experiments demonstrate clear gains in both retrieval accuracy and runtime efficiency over existing partition-based top-k attention methods. We summarize our key contributions as follows:

1. **Parallel Hierarchical Top-k Attention.** We introduce HIROUTER, a hierarchical routing mechanism that clusters tokens in parallel into multi-level buckets, enabling efficient top-k attention while matching or surpassing full-attention accuracy.
2. **Routing Design Rules and Surrogate Objectives.** We formulate two design rules for routed top-k attention, co-routing relevant tokens and avoiding bucket collapse, and optimize them with tractable entropy surrogate losses under coupled routing and attention representation learning.
3. **Scalable and Efficient Long-Context Performance.** HIROUTER delivers consistent accuracy improvements, including on `WikiText-103` perplexity and numerous commonsense reasoning/retrieval tasks, as well as gains of $+3.35$ on `LongBench` and $+9.7$ on NIAH relative to full attention.

2 RELATED WORK

Efficient Attention. Location-based sparse patterns have long been used to curb the quadratic complexity of vanilla self-attention. Early works alternated coarse and local windows to reduce the receptive field (Liu et al., 2018). Strided/dilated patterns were later adopted for image generation (Child et al., 2019), while adaptive windows offered dynamic sparsity for sequence modeling (Sukhbaatar et al., 2019). Global-plus-local hybrids (Beltagy et al., 2020; Ainslie et al., 2020; Zaheer et al., 2020) designate a small set of global tokens that attend everywhere. Orthogonal to fixed patterns, low-rank or kernelized approaches approximate dense attention via linear projections (Katharopoulos et al.,

2020; Xiong et al., 2021; Wang et al., 2020) or random features (Choromanski et al., 2021; Peng et al., 2021). Yuan et al. (2025) meanwhile combine hierarchical token compression and selection. While these designs bring linear or near-linear complexity, they often under-use fine-grained, content-based interactions. KV-cache management methods have also introduced hierarchy-based clustering approaches (Fu et al., 2025; Wang et al., 2025; Liu et al., 2025; Tu et al., 2025; Zandieh et al., 2024) to optimize inference-time storage efficiency via budgeting, semantic clustering, or sublinear sampling. In contrast, HIROUTER optimizes the *retrieval* process itself, addressing how attention is computed.

Sparse Top-k Attention. Content-based sparsification keeps only the most relevant tokens per query (Pagliardini et al., 2023), with many approaches such as hashing (Roy et al., 2021; Kitaev et al., 2020; Gupta et al., 2021; Bertsch et al., 2023) direct neighbor search (Mao et al., 2024; Zeng et al., 2025; 2026a;b), or more complex patterns (Chalkidis et al., 2022; Chen et al., 2021). However, existing algorithms treat the top-k algorithm as separate from attention computation and do not explicitly regularize attention embeddings to improve retrieval quality.

Approximate Top-k Retrieval. Classical similarity search relies on graph or tree indices (Malkov and Yashunin, 2020; Douze et al., 2024; Guo et al., 2020), which are built sequentially and ill-suited for per-layer GPU parallelism. In the attention setting, these indices need to be frequently rebuilt or updated across layers and heads for each batch, making them prohibitively slow and incompatible with efficient GPU-parallel execution.

3 METHODOLOGY

In this section, we present HIROUTER, a hierarchical routing framework for efficient top-k attention. Inspired by learned-index approaches (Kraska et al., 2018; Li et al., 2023; Gupta et al., 2022), we propose a learnable hierarchical router that jointly trains centroids and routing logits, removing explicit indexing overhead and adapting dynamically to data. Unlike offline-trained partitioning (Dong et al., 2023), HIROUTER updates representations on-the-fly with entropy regularization, improves the precision of top-k retrieval, and balances buckets. We first review attention variants in Section 3.1 to motivate HIROUTER. Section 3.2 emphasizes how routing decisions are coupled with the query/key representations used by attention, which we use to formulate entropy-based surrogate objectives in Section 4 used to train the coupled router to satisfy the two routing rules: co-routing relevant tokens and maintaining balanced bucket occupancy. Finally, a hierarchical beam search (Section 4.2) retrieves the candidate buckets for sparse attention.

3.1 PRELIMINARIES

Self-Attention (Vaswani et al., 2017) enables each token to attend to all others, capturing long-range dependencies. Given queries, keys and values $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{T\times d}$, the scaled dot-product attention is

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\boldsymbol{Q}\boldsymbol{K}^\top/\sqrt{d}\right)\boldsymbol{V}, \tag{1}$$

where d denotes the model hidden dimension. **Top-k Attention** (Kitaev et al., 2020; Roy et al., 2021; Gupta et al., 2021) reduces the $\mathcal{O}\left(T^2\right)$ cost of full self-attention to $\mathcal{O}\left(T\cdot k\right)$ by restricting each query to its k most relevant keys. Specifically, for a query $\boldsymbol{q}\in\mathbb{R}^d$, the attention output is

$$\boldsymbol{A} = \textstyle\sum_{i\in I_q} \text{softmax}\left(\boldsymbol{q}\boldsymbol{K}_i^\top/\sqrt{d}\right)\boldsymbol{V}_i, \quad I_q = \text{Top-K}\left(\boldsymbol{q}\boldsymbol{K}^\top/\sqrt{d}, k\right) \tag{2}$$

where I_q is the set of indices corresponding to the k largest similarity scores. This preserves the expressivity of self-attention while substantially improving efficiency.

3.2 HIERARCHICAL ROUTING FOR EFFICIENT TOP-K RETRIEVAL

We introduce HIROUTER, a hardware-efficient router for top-k attention. The router operates on query/key representations used by attention: given hidden states $\boldsymbol{X} \in \mathbb{R}^{B\times T\times d}$, learned projections produce queries $\boldsymbol{Q}$ and keys $\boldsymbol{K}$, and the routing centroids are trained jointly with these projections. This coupling lets the attention representation geometry adapt to the retrieval structure, rather than forcing a fixed post-hoc partition on embeddings that were not trained for routing.

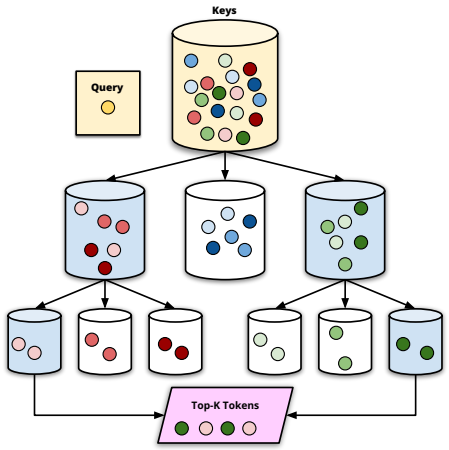

Figure 3: HIROUTER example using $M = 2$. Given a query (yellow dot), the keys and values of previous tokens are aggregated in a buffer. At every level, only the two highest-probability buckets (in blue) are kept. Leaf tokens are concatenated into a compact buffer for attention.

Algorithm 1 HIROUTER Pseudocode

Input: Key-Values $(\boldsymbol{K}, \boldsymbol{V})$, Query $\boldsymbol{q}$, Centroids $\mathcal{C}_{\text{all}}^{(l)}$ at $l \in \{1, \ldots, L\}$, beam width M. **Output:** Attention $\boldsymbol{A}$
Initialize: Accumulated probabilities $\boldsymbol{P}_{\boldsymbol{K}}^{0}, \boldsymbol{P}_{\boldsymbol{q}}^{0} = \mathbf{1}$, Root indices for keys and query $\boldsymbol{a}_{\boldsymbol{K}}^{0}$, $\boldsymbol{a}_{\boldsymbol{q}}^{0} = \mathbf{0} \in \mathbb{R}^{M}$
for level $l \in \{1, \ldots, L\}$ **do**
1. Gather active child centroids:
// Only consider child centroids of the current parents
// for keys: parents $\boldsymbol{a}_{\boldsymbol{K}}^{l-1}$; for query: beam parents $\boldsymbol{a}_{\boldsymbol{q}}^{l-1}$
$\mathcal{C}_{\boldsymbol{K}}^{(l)} = \text{Child}(\mathcal{C}_{\text{all}}^{(l)}, \boldsymbol{a}_{\boldsymbol{K}}^{l-1})$, $\mathcal{C}_{\boldsymbol{q}}^{(l)} = \text{Child}(\mathcal{C}_{\text{all}}^{(l)}, \boldsymbol{a}_{\boldsymbol{q}}^{l-1})$
2. Similarity computation:
// compute logits against children of the current parent
$\boldsymbol{\ell}_{\boldsymbol{K}}^{l} = \boldsymbol{K}\mathcal{C}_{\boldsymbol{K}}^{(l)\top}$, $\boldsymbol{\ell}_{\boldsymbol{q}}^{l} = \boldsymbol{q}\mathcal{C}_{\boldsymbol{q}}^{(l)\top}$
3. Local routing probabilities:
// softmax over the C children of each parent
$\boldsymbol{p}_{\boldsymbol{K}}^{l} = \text{softmax}(\boldsymbol{\ell}_{\boldsymbol{K}}^{l})$, $\boldsymbol{p}_{\boldsymbol{q}}^{l} = \text{softmax}(\boldsymbol{\ell}_{\boldsymbol{q}}^{l})$
4. Update path probability using chosen child
$\boldsymbol{P}_{\boldsymbol{K}}^{l} = \boldsymbol{P}_{\boldsymbol{K}}^{l-1} \odot \max_j(\boldsymbol{p}_{\boldsymbol{K},:,j}^{l})$
$\boldsymbol{P}_{\boldsymbol{q}}^{l} = \text{BeamUpdateProbs}(\boldsymbol{P}_{\boldsymbol{q}}^{l-1}, \boldsymbol{p}_{\boldsymbol{q}}^{l}, M)$
5. Hard assignment (keys: greedy, query: beam):
// keys: pick the best child index for each token
// query: maintain a beam of M best child indices
$\boldsymbol{a}_{\boldsymbol{K},i}^{l} = \boldsymbol{a}_{\boldsymbol{K},i}^{l-1} \cdot C + \arg\max_j \boldsymbol{p}_{\boldsymbol{K},i,j}^{l}$
$\boldsymbol{a}_{\boldsymbol{q}}^{l} = \text{BeamUpdateIndices}(\boldsymbol{a}_{\boldsymbol{q}}^{l-1}, \boldsymbol{p}_{\boldsymbol{q}}^{l}, M)$
end for
Retrieval: Select top-M buckets:
$\mathcal{I} = \text{Top-M-Buckets}(\boldsymbol{P}_{\boldsymbol{q}}^{L}, M)$
Return $\boldsymbol{A}$: Sparse attention over indices in $\mathcal{I}$.

Hierarchical Structure. We organize the routing as a C-ary tree with L levels. At any level $(l)_{l=1}^{L}$, the search space is partitioned into C^{l-1} parent nodes, where each parent p routes its assigned tokens into one of C child buckets. We define a bank of learnable centroids $\mathcal{C}^{(l)} \in \mathbb{R}^{C^{l-1} \times d \times C}$, where $\mathcal{C}_{p}^{(l)} \in \mathbb{R}^{d \times C}$ represents C centroids corresponding to the children of parent p.

Routing and Discrete Assignment. For a token $\boldsymbol{z}$ assigned to a parent p, we compute the routing logits $\boldsymbol{\ell}_{p}^{(l)} = \boldsymbol{z}\mathcal{C}_{p}^{(l)}$ and the resulting probability distribution $\boldsymbol{p}_{p}^{(l)} = \text{softmax}(\boldsymbol{\ell}_{p}^{(l)})$. To enforce a hard decision while maintaining differentiability, we employ the Gumbel-Softmax trick:

$$\tilde{\boldsymbol{p}}_{p}^{(l)} = \text{GumbelSoftmax}(\boldsymbol{\ell}_{p}^{(l)}), \quad a_{\text{local}} = \underset{j \in \{0,\ldots,C-1\}}{\arg\max}\ \tilde{p}_{p,j}^{(l)}. \tag{3}$$

Here, a_{local} is the local index of the chosen child bucket (relative to parent p). The unique global bucket index a for the token at this level is derived by offsetting the parent index: $a = p \cdot C + a_{\text{local}}$. Algorithm 1 shows the deterministic inference-time form of this routing step. During training, the hard key assignment in Step 5 uses the straight-through Gumbel-Softmax estimator in Eq. (3); Gumbel noise is removed at inference.

Physical Reorganization. For efficient parallel processing at the next level, tokens must be grouped physically by their assigned buckets. We sort the tokens based on their global indices $\{a_i\}$ such that $\tilde{a}_1 \leq \cdots \leq \tilde{a}_T$. The sorted feature tensor is then reshaped to: $\boldsymbol{z}^{(l+1)} \in \mathbb{R}^{B \times C^{l} \times \frac{T}{C^{l}} \times d}$. This reshaping operation treats the C^{l} buckets as independent batch dimensions for the next hierarchy level $(l+1)$, allowing the routing process to recurse efficiently. Our current optimized kernel uses an equal-capacity layout, which partitions the sorted buffer into fixed-size buckets for efficient batched execution. This slightly perturbs the original hard assignments when bucket occupancies are uneven.

Algorithm 1 highlights this parallelization: centroid–token similarities at each level are computed in parallel, depending only on the feature dimension d, an $\mathcal{O}(d)$ per-token cost. The routing probabilities are then evaluated only over the C children of each parent, a constant-time operation parallelized across all parent nodes in $\mathcal{O}(C)$. Finally, the hierarchical path probabilities are updated by multiplying each parent's probability with that of its selected child, operating in $\mathcal{O}(L)$. This offers an efficient, implementable routing procedure on modern hardware.

4 Routing Objectives and Retrieval Bounds

The two routing losses address different failure modes. **Rule 1** concerns relevant keys: if full attention ranks a key among the top-k, its routing decisions should remain compatible with the query so that the hierarchical search can keep it. **Rule 2** concerns selectivity: choosing a small number of buckets should not bring in most of the sequence. These roles are complementary. Sample entropy makes a route more decisive, but confidence alone does not make a relevant query–key pair agree. Balance does not identify relevant keys, but it limits how many unrelated keys enter with the retained ones.

4.1 Rule-to-Surrogate Reduction

For a query q and causal keys $\{z_i\}_{i=1}^{T}$, let $N_k(q) = I_q$ be the indices of the k largest attention scores $\langle q, z_i\rangle/\sqrt{d}$. At one routing parent, write p_q for the query distribution over its C children and p_i for key z_i's distribution over the same children. Let $\mathcal{D}_K$ denote a distribution over reference pairs (q, i) with $i \in N_k(q)$, and let $\bar{p} = T^{-1}\sum_i p_i$ be the average key distribution. As in the original formulation, the ideal analysis targets are $\mathcal{L}^{\star}_{\mathrm{smp}} = \mathbb{E}_{\mathcal{D}_K}[-\log(p_q^{\top} p_i)]$ and $\mathcal{L}^{\star}_{\mathrm{bal}} = \frac{1}{2}\|\bar{p} - u\|_2^2$, where $u_j = 1/C$. These quantities are used only to state the desired routing behavior; they are not optimized directly. Instead, training uses the tractable entropy objectives

$$\mathcal{L}_{\mathrm{smp}} = \frac{1}{T}\sum_i H(p_i), \qquad \mathcal{L}_{\mathrm{bal}} = -H(\bar{p}).$$

Proposition 4.1 (Rule 1: confidence and co-routing)**.** *For reference pairs distributed as $\mathcal{D}_K$,*

$$\mathcal{L}^{\star}_{\mathrm{smp}} \le \mathbb{E}_{\mathcal{D}_K}\left[H(p_i) + \mathrm{KL}(p_i\|p_q)\right], \qquad \mathbb{E}_{\mathcal{D}_K}[p_q^{\top} p_i] \ge \mathbb{E}_{\mathcal{D}_K} e^{-H(p_i)-\mathrm{KL}(p_i\|p_q)}. \tag{4}$$

The first term is exactly the quantity used by our sample-entropy loss: $\mathcal{L}_{\mathrm{smp}}$ averages $H(p_i)$ over keys, encouraging their routing decisions to become sharper. In the bound above, the same entropy is evaluated for relevant keys. Sharper routing helps these keys stay with the query when their routing preferences are already compatible. The KL term captures the remaining query–key mismatch, which sample entropy alone does not remove.

Proposition 4.2 (Rule 2: balance and candidate load)**.** *Let $\kappa(v) = \mathrm{KL}(v\|u) = \log C + \mathcal{L}_{\mathrm{bal}}(v)$. For $s \in (0,1)$, let $\mathsf{U}(s,\kappa)$ be the largest $x \in [s,1]$ such that the binary divergence $\mathrm{kl}(x\|s) \le \kappa$. If N_b is the observed number of keys in bucket b and $\widehat{\bar{p}}_b = N_b/T$, then every set B of buckets satisfies*

$$\mathcal{L}^{\star}_{\mathrm{bal}} \le \kappa(\bar{p}), \qquad N_B \le T\,\mathsf{U}\big(|B|/C, \kappa(\widehat{\bar{p}})\big). \tag{5}$$

The bound does not require perfectly equal bucket sizes. As long as the observed occupancy is reasonably balanced, selecting only a small fraction of the buckets also limits the fraction of keys that can enter the candidate set. Appendix A.2 gives the corresponding bound on bucket-size variation and extends the result across the hierarchy using the observed parent occupancies.

4.1.1 $\mathcal{L}_{\mathrm{smp}}$: from confidence to path retention

The local effect of sample entropy can be stated exactly.

Lemma 4.3 (Local entropy gradient)**.** *For $p_i = \mathrm{softmax}(\ell_i)$, $\partial H(p_i)/\partial \ell_{i,j} = -p_{i,j}(\log p_{i,j} + H(p_i))$. A direct descent step therefore increases $\ell_{i,j}$ exactly when $p_{i,j} > e^{-H(p_i)}$.*

Minimizing $\mathcal{L}_{\mathrm{smp}}$ sharpens the routing distribution, making the preferred routing choice more decisive. This is the effect we use in the path-retention analysis below. We next connect confidence to the search actually used by the router.

Assumption 4.1 (Probabilistic beam scores)**.** At each level, the query extends the current beam, adds the local log-probability of each child to the accumulated path score, and keeps the top M paths. Ties are resolved consistently. The path-retention analysis below uses these cumulative probabilities; any common implementation offset can be restored before evaluating the bound.

Let $\lambda_l(q)$ be the largest prefix log-probability discarded at level l. For key i, follow its greedy path. At level l, let $p_i^{(l)} = p^{l}_{K,i}$ be its local routing distribution. Let $\xi_{qi}^{(l)} = \left[\log p^{(l)}_{i,j_i^{(l)}} - \log p^{(l)}_{q,j_i^{(l)}}\right]_+$

denote the positive drop in log-probability that the query assigns to the child chosen by the key, where $[x]_+ = \max\{x, 0\}$. We summarize the accumulated uncertainty and mismatch before the path is pruned

$$\psi_{qi} = \max_{l \in \mathcal{P}_q} \frac{\sum_{s=1}^{l} [H(p_i^{(s)}) + \xi_{qi}^{(s)}]}{-\lambda_l(q)}, \tag{6}$$

where $\mathcal{P}_q$ is the set of pruning levels, and $\psi_{qi} = 0$ when no pruning occurs. The numerator records what can weaken the key's path; the denominator is the query-beam threshold that the path must fall below before it is removed.

Proposition 4.4 (Rule 1: individual path retention)**.** *For greedy key routing and the query's own beam,*

$$\mathbf{1}\{i \notin S_q\} \le \min\{1, \psi_{qi}\}, \qquad \psi_{qi} < 1 \Longrightarrow i \in S_q, \tag{7}$$

where S_q is the set of keys contained in the leaf buckets retained by the query beam.

The proof looks at the first level at which a missed key leaves the beam. The bound is applied to each key separately, so a poorly routed key does not affect the guarantees for the other relevant keys.

4.1.2 $\mathcal{L}_{\text{bal}}$ Improves Retrieval Selectivity and Efficiency

To improve routed top-k retrieval, preserving relevant neighbors is only one part of the problem. The selected buckets must also remain small enough to form a useful candidate set. We therefore separate these two effects: the path analysis above describes which relevant keys survive the query beam, while the balance loss controls how many total keys are contained in the selected buckets.

Let $\mathcal{B}_q$ denote the leaf buckets retained by the query beam and let $S_q = \bigcup_{b \in \mathcal{B}_q} S_b$ be the corresponding candidate keys. For each relevant key $i \in N_k(q)$, the quantity ψ_{qi} from the previous section measures how close its routing path is to being pruned. We aggregate these per-key contributions as

$$r_q = \frac{1}{k} \sum_{i \in N_k(q)} [1 - \psi_{qi}]_+$$

Each relevant key contributes separately to this bound. Thus, a poorly matched routing path for one key does not reduce the contribution of the others.

We next use $\mathcal{L}_{\text{bal}}$ to control how many keys enter the candidate set. Let N_b be the number of keys in leaf bucket b, so that $\widehat{\overline{p}}_b^{(L)} = N_b/T$ gives the observed leaf occupancy. We measure its deviation from uniform occupancy by

$$\kappa_L = \mathrm{KL}\left(\widehat{\overline{p}}^{(L)} \| u_{C^L}\right).$$

This leaf-level imbalance can be written exactly as a weighted sum of the local balance terms along the tree, with each parent weighted by the fraction of keys that reach it. Thus the same balance objective used during routing also controls occupancy at the leaves. If the query beam retains the leaf set $\mathcal{B}_q$, Proposition 4.2 gives

$$D_q = T \cup\left(\frac{|\mathcal{B}_q|}{C^L}, \kappa_L\right)$$

as an upper bound on the number of keys in those leaves. When the leaf occupancy is reasonably balanced, selecting only a small number of leaves keeps the candidate set under control: the retained leaves can not contain a disproportionately large fraction of all keys, so the number of keys passed to sparse attention remains small relative to the full sequence.

Proposition 4.5 (The two rules meet in recall and candidate precision)**.** *Under the path-retention conditions above,*

$$\mathrm{Recall@}\, k(q) \ge r_q, \qquad \mathrm{Precision}(q) \ge \frac{k r_q}{\max\{1, D_q\}}. \tag{8}$$

The two routing objectives therefore affect different parts of retrieval. The sample-entropy term, together with query–key routing compatibility, helps preserve relevant keys through the hierarchy and controls the numerator of the precision bound. The balance term controls the denominator by limiting how many keys accompany them in the selected buckets. This gives a direct connection between the two routing rules: relevant paths should survive the beam, while the buckets that survive should remain selective. The corresponding parent-weighted balance identity and full proof are given in Appendix A.2.

4.1.3 JOINT CO-TRAINING OF ROUTING AND ATTENTION REPRESENTATIONS.

A key feature of HIROUTER is that routing is not performed over fixed or externally defined embeddings. Instead, the key and query representations are *co-trained* with the routing centroids and assignment probabilities in an end-to-end manner.

Because HiRouter routes the same Q/K representations used by attention, the language-modeling and routing objectives act on a shared representation space. The language-modeling loss shapes Q/K according to task relevance, while the routing objective sharpens and balances assignments and adapts the centroids to this evolving geometry. This co-adaptation improves routing recall without assuming that entropy regularization alone creates semantic alignment. Empirically, improved retrieval recall helps language modeling, with diminishing returns once moderate recall is reached (Appendix C.9).

Overall Routing Objective. The final routing surrogate combines both terms: $\mathcal{L}_{\text{rtr}} = \mathcal{L}_{\text{bal}} + \mathcal{L}_{\text{smp}}$. In practice, $\mathcal{L}_{\text{rtr}}$ regularizes the query/key projections and routing centroids, promoting an emergent hierarchical representation space that supports the two routing rules. We integrate this with the downstream task loss to form the total objective $\mathcal{L} = \mathcal{L}_{\text{CE}} + \alpha \mathcal{L}_{\text{rtr}}$, where $\mathcal{L}_{\text{CE}}$ is the auto-regressive cross-entropy loss, $\mathcal{L}_{\text{rtr}}$ is our hierarchical routing surrogate, and $\alpha > 0$ controls the routing objective strength. In addition to the routed branch, we include a sliding-window branch for local context modeling and a learnable bias term in the attention final output; Appendix C ablates these components and shows that the routed branch is the main source of the gain.

4.2 CANDIDATE RETRIEVAL FOR ATTENTION

The router assigns each token to a bucket at each hierarchical level according to its routing probability. We first project them via learned matrices $\boldsymbol{W}_Q, \boldsymbol{W}_K$ and compute their similarity: $\boldsymbol{S} = \left(\boldsymbol{X}\boldsymbol{W}_Q^\top\right)\left(\boldsymbol{X}\boldsymbol{W}_K^\top\right)^\top$. At level l, each token has a conditional routing distribution $p^{(l)} \in \mathbb{R}^C$.

A brute-force strategy would enumerate all buckets with nonzero joint probability $p_{\text{joint}} = \prod_{l=0}^{L-1} p^{(l)}$, but this is prohibitive in both time and memory as a total $\prod_{l=0}^{L-1} C^l$ possibilities exit. Instead, we use a level-wise beam search of width M: at each level l, we retain the M buckets with largest partial joint probability $\prod_{i=0}^{l} p^{(i)}$. These M-element beams define a candidate set of key tokens for retrieval. We collect these M buckets and then compute sparse attention outputs following Equation 2.

5 EXPERIMENTS

5.1 SMALL-SCALE TASKS

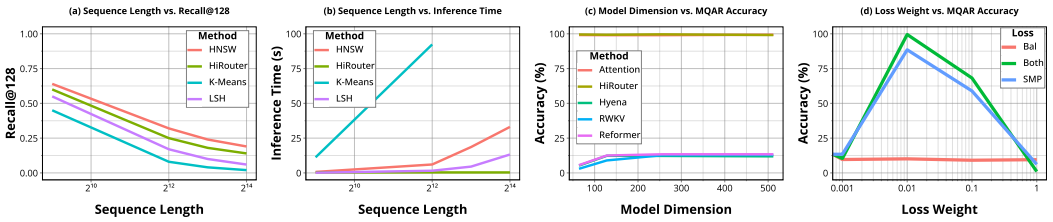

Figure 4: Synthetic experimental results, demonstrating the efficacy of HIROUTER. (a) and (b) show performance relative to contemporary retrieval methods in terms of recall and speed with relation to the sequence length T. (c) plot shows the effect of hidden size d on MQAR's accuracy, while (d) plot shows how the scale α of the auxiliary loss can influence performance on MQAR.

Synthetic Results. We first evaluate using a synthetic key–value (KV) retrieval task. N KV pairs are sampled from a multivariate Gaussian in $\mathbb{R}^d$, inserted them into our hierarchical router, and then issued M random key queries drawn from the same distribution (Kraska et al., 2018). We measure `recall@128` (i.e. the fraction of queries whose top-128 retrieved key matches the true maximum inner product keys) and end-to-end latency. As shown in Figure 4(a), HIROUTER maintains higher recall than LSH and vanilla k-means. HNSW achieves higher recall but with a superlinear query time, resulting in prohibitive latency. Baselines are implemented using FAISS (Douze et al., 2024).

Next, we benchmark on the Multi-Query Associative Recall (MQAR) task (Arora et al., 2024) where the model must store a sequence of N key–value pairs and retrieve the correct values for a set of queries. The total vocab size is 8192. Figure 4(c) shows that HIROUTER sustains high recall even for small d, unlike other methods. Finally, we sweep the weight α on our dual-entropy routing loss in Figure 4(d), where $\alpha \in [0.01, 0.1]$ yields a balanced structure with fine-grained semantic clustering.

WikiText-103. We next compare performance on `WikiText-103` using with 125M parameter models (Table 1). Against numerous sparse and efficient attention variants, HIROUTER is the only method to outperform the vanilla Transformer. We additionally visualize the learned routing structure and evaluate top-k recall and routing latency in Figure 1.

Table 1: WT-103 PPL.

Model	ppl ↓
Transformer	19.2
Performer	26.8
Reformer	25.6
AFT-conv	28.2
RFA-Gaussian	27.5
CosFormer	23.1
IceFormer	31.4
RoutingTF	26.7
BigBird	23.3
ZETA	26.3
Mongoose	23.6
NSA	19.3
HIROUTER	**18.5**

5.2 LARGER SCALE LANGUAGE MODELING

Setup and Training. We next conduct a larger-scale evaluation against other efficient models (Biderman et al., 2023; Gu and Dao, 2024; Dao and Gu, 2024; Yang et al., 2024a;b; Zhang et al., 2024; Yuan et al., 2025). Models are trained under identical conditions with 410M parameters on 10B tokens from `FineWeb-Edu` (Penedo et al., 2024)[1] using a context length of 2048 tokens. We use AdamW (Loshchilov and Hutter, 2019) with a peak learning rate of 4e-4, weight decay of 0.1, and gradient clipping of 1.0. The learning rate follows a cosine annealing schedule with a warm-up period of 1% of the total steps and a batch size of 0.5M tokens. For HIROUTER, the router has 4 levels with 4 buckets/centroids each, the window size as 64 for the SWA branch, and a beam width of 4. Following our results on the synthetic task, we set α as 0.05.

Table 2: Performance comparison on language modeling and zero-shot common-sense reasoning.

Model	Wiki. ppl ↓	LMB. ppl ↓	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc_n ↑	SIQA acc ↑	BoolQ acc ↑	Avg.
Transformer	30.21	43.44	32.76	**67.68**	39.20	**53.51**	57.58	27.47	37.97	61.19	**47.17**
Mamba	32.63	61.68	27.79	65.61	38.47	51.22	57.41	26.62	38.64	**61.65**	45.93
Mamba2	30.15	49.83	28.70	66.81	38.94	51.38	**60.69**	28.75	37.67	59.69	46.83
HGRN2	30.07	**40.29**	31.32	66.54	**39.68**	50.12	59.30	27.05	38.84	58.72	46.45
GLA	31.50	51.56	29.01	66.49	38.60	50.12	57.83	26.11	39.25	57.77	45.40
DeltaNet	**28.82**	45.06	30.47	67.19	39.51	52.80	58.80	**29.10**	38.18	58.26	46.79
GSA	30.78	48.74	29.75	66.70	39.01	52.49	59.26	27.65	38.49	60.61	46.50
NSA	31.54	43.0	**36.48**	66.27	35.37	50.19	59.27	27.33	**39.41**	59.66	46.75
HIROUTER	31.09	42.94	33.09	66.81	38.03	50.75	59.47	28.50	38.08	61.31	47.01

Commonsense Reasoning. We present perplexity results as well as zero-shot commonsense reasoning performance on a number of different tasks (see Appendix B.4.1). These tasks are effective at evaluating the acquired knowledge of models through their general reasoning abilities. In Table 2, we observe that HIROUTER is effective in comparison to a number of modern methods commonly used as efficient language model backbones. In particular, we observe that while a baseline, full-attention Transformer remains the most effective model compared to other alternatives, HIROUTER remains highly effective on such tasks and performs comparably or outperforms recent models.

Recall-Intensive Tasks. We next compare the ability to recall information on recall-intensive tasks. As shown in Table 3, the Transformer fares best, while other efficient baselines generally struggle due to their fixed-size state. In contrast, HIROUTER remains capable of on-par performance relative to the Transformer while maintaining efficiency, demonstrating how HIROUTER can be beneficial for filtering irrelevant information.

Table 3: Real-world retrieval accuracy.

Model	FDA	SWDE	SQuAD	TQA	NQ	Drop	Avg.
Transformer	7.26	38.07	4.52	0.93	1.00	**2.48**	9.71
Mamba	1.36	6.84	3.10	**1.03**	1.00	2.12	2.91
Mamba2	4.26	10.53	4.49	0.55	1.25	2.43	3.92
HGRN2	2.00	10.17	4.02	0.97	1.02	3.12	3.88
GLA	3.27	9.72	2.72	0.40	1.36	1.96	3.24
DeltaNet	4.08	17.19	3.81	0.42	0.97	2.41	4.81
GSA	3.36	7.02	4.13	0.86	**1.47**	2.47	3.55
NSA	7.21	37.67	3.60	0.17	0.50	2.08	8.54
HIROUTER	**8.43**	**42.83**	3.38	0.67	0.93	2.32	**9.76**

[1] Mamba models use ≈430M parameters due to restrictions on the state size and the input dimension.

Long-Context Tasks. We next evaluate long-context performance. On LongBench (Bai et al., 2024, Table 4), Transformers struggle against linear models, but HIROUTER is capable of closing the gap between these two paradigms, highlighting the potential for improved long-context Transformers, outperforming baselines outside of Mamba without additional tuning.

Table 4: Accuracy on tasks from LongBench (Bai et al., 2024).

Model	Single-Doc QA			Multi-Doc QA			Summarization			Few-shot			Code		Avg.
	NQA	QQA	MFQ	HQA	2WM	Mus	GvR	QMS	MNs	TRC	TQA	SSM	LCC	RBP	
Transformer	0.67	3.23	3.86	0.33	1.37	0.11	8.29	11.87	**13.31**	1.50	3.02	5.61	9.82	9.61	5.19
Mamba	1.52	**3.55**	10.51	3.20	**6.82**	2.24	5.51	15.67	10.02	3.00	**14.04**	5.82	11.55	14.82	7.73
Mamba2	**1.80**	3.20	10.84	2.97	5.70	2.57	6.66	15.87	10.43	18.50	13.31	6.09	16.67	19.05	**9.55**
HGRN2	0.38	0.80	1.63	0.11	0.05	0.11	3.07	5.96	4.08	0.00	0.67	0.00	**20.88**	**20.71**	4.17
GLA	0.60	1.46	2.50	0.72	1.01	0.70	4.30	10.44	6.41	0.00	5.58	0.00	20.23	20.45	5.31
DeltaNet	0.38	0.76	1.63	0.11	0.05	0.11	3.21	7.40	4.50	0.00	5.58	**9.30**	20.03	19.89	5.21
GSA	0.37	0.73	1.60	0.11	0.05	**4.81**	3.28	8.61	4.81	0.00	4.71	8.27	19.33	20.16	5.49
NSA	0.87	3.51	4.02	0.64	1.36	0.33	**9.01**	13.21	13.16	1.75	4.21	6.21	9.42	13.81	5.82
HIROUTER	1.69	3.54	**11.25**	**4.54**	**6.82**	2.54	**8.80**	**16.21**	10.66	**21.17**	11.36	4.48	5.25	11.21	8.54

Table 5: Performance on Passkey-Retrieval tasks. For longer sequences, see Table 30.

Model	S-NIAH-1 (pass-key retrieval)				S-NIAH-2 (number in haystack)				S-NIAH-3 (uuid in haystack)				Avg.
	1K	2K	4K	8K	1K	2K	4K	8K	1K	2K	4K	8K	
Transformer	**94.8**	**96.0**	0.0	0.0	**95.6**	**70.8**	0.0	0.0	**91.6**	57.6	0.0	0.0	42.2
Mamba	8.8	4.0	1.2	0.8	27.2	3.6	2.4	2.4	0.0	0.0	0.0	0.0	4.2
Mamba2	35.2	9.6	0.8	0.0	25.2	6.4	11.6	1.6	0.8	1.6	0.4	0.0	7.7
HGRN2	76.0	4.8	0.0	0.0	36.4	7.6	0.0	0.0	0.0	0.0	0.0	0.0	10.4
GLA	0.0	0.0	0.0	0.0	3.2	1.6	1.2	0.8	0.0	0.0	0.0	0.0	0.6
DeltaNet	38.8	40.8	48.4	**34.8**	26.4	6.0	10.8	4.4	8.0	0.8	0.8	**2.4**	18.5
GSA	23.6	10.0	3.2	2.4	20.4	6.8	9.2	**4.8**	0.0	0.0	0.0	0.0	6.7
NSA	40.0	33.0	21.0	16.0	12.0	8.0	13.0	6.0	0.8	1.2	0.6	0.4	12.6
HIROUTER	93.6	86.8	**57.4**	22.4	84.2	67.6	**32.6**	4.4	88.4	**60.4**	**22.2**	**2.4**	**51.9**

On the Needle-in-a-Haystack (NIAH) task (Table 5), where models are tasked with retrieving a single element (the needle) from a large context (the haystack). While Transformers are more effective on lengths within the training context, highlighted by strong performance in different formats of the task, recurrent models demonstrate a better extrapolation, such as Mamba, DeltaNet, and GSA. HIROUTER again bridges this gap: on shorter contexts, the performance remains comparable to the Transformer, but as the length extends, HIROUTER retains an ability to extrapolate and still perform at par with models specifically trained for long contexts/extrapolation.

5.3 COMPUTATIONAL EFFICIENCY

To further quantify HIROUTER's runtime behavior, we benchmark against two sparse top-k attention baselines, ROUTINGTRANSFORMER (RT) and MONGOOSE (MG). Table 6 reports forward (FWD) and forward+backward (FWD+BWD) runtimes for lengths between 2^{12} and 2^{16}. HIROUTER consistently outpaces both while surpassing their scaling behavior: only HIROUTER remains efficient at the largest tested lengths.

Table 6: Time (ms) for forward (FWD) and complete (FWD+BWD) passes across varying lengths.

Method \ Length		4K	8K	16K	32K	65K
FA	FWD	**0.18**	**0.56**	1.93	7.14	30.76
	FWD+BWD	**0.61**	1.95	6.55	25.09	99.69
RT	FWD	2.88	3.76	7.98	15.76	33.79
	FWD+BWD	5.81	8.20	18.58	36.08	73.41
MG	FWD	2.21	3.76	6.76	13.29	29.66
	FWD+BWD	4.58	8.20	15.04	29.08	60.34
Ours	FWD	0.26	0.59	**1.68**	**3.12**	**8.11**
	FWD+BWD	0.75	**1.81**	**6.08**	**16.81**	**40.01**

6 CONCLUSION

We present HIROUTER, a novel hierarchical routing approach towards computing top-k attention via maximum inner product search. HIROUTER uses a bucket partitioning approach, partitioning tokens within the sequence into discrete buckets across multiple levels of a learned tree. The tree uses learned centroid-based routing logits and a Gumbel-Softmax trick with a dual-component routing loss for training. Our work provides empirical evidence to show that HIROUTER is both competitive with concurrent efficient LLM architectures as well as regular full-attention baselines. Furthermore, we provide an efficient Triton-based implementation to enable our method to outperform other efficient attention-based implementations in terms of efficiency.

Ethics Statement and Limitations

This work focuses on developing efficient attention mechanisms for large-scale language models. While our method improves the scalability and retrieval accuracy of Transformer models, the ethical considerations are largely consistent with those of general-purpose language modeling. Potential risks include misuse in generating harmful or misleading content, reinforcement of biases present in training corpora, and environmental concerns arising from large-scale training. We mitigate these risks by (i) benchmarking only on standard public datasets, (ii) avoiding the use of sensitive or private data, and (iii) providing transparent methodology to facilitate responsible replication. Moreover, the computational efficiency gains of HiRouter reduce energy consumption relative to dense or less efficient sparse baselines, contributing positively to the environmental impact of model deployment.

Our primary limitation is that we restrict the majority of our focus to auto-regressive large language models, but we contend that our method is also suitable for bi-directional models that use attention, such as vision-language models that use Transformer backbones (ex. ViT). We believe that our chosen datasets still provide valuable insights while remaining within our operational constraints and will further explore other directions as our computational capabilities expand.

AI Use Statement

An AI assistant was used to help restructure the manuscript, draft and edit prose, assist with mathematical arguments and proof checking, assemble tables from author-provided experiment reports, identify relevant references, generate LaTeX, and perform numerical sanity checks. The assistant did not run any reported pretraining or benchmark experiments. All AI-assisted content was reviewed and verified by the authors, who take responsibility for the final manuscript.

Reproducibility Statement

The model architecture and routing algorithm are described in Sections 3–4. Training settings and evaluation details are provided in Appendix B, additional ablations in Appendix C, Triton kernels in Appendix D, and complexity/system analysis in Appendix E. We provide code and configuration files in the anonymous supplementary material.

References

J. Ainslie, S. Ontañón, C. Alberti, et al. ETC: encoding long and structured inputs in transformers. In B. Webber, T. Cohn, Y. He, and Y. Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 268–284. Association for Computational Linguistics, 2020. URL https://doi.org/10.18653/v1/2020.emnlp-main.19.

J. Ainslie, J. Lee-Thorp, M. de Jong, et al. GQA: training generalized multi-query transformer models from multi-head checkpoints. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 4895–4901. Association for Computational Linguistics, 2023. URL https://doi.org/10.18653/v1/2023.emnlp-main.298.

S. Arora, B. Yang, S. Eyuboglu, et al. Language models enable simple systems for generating structured views of heterogeneous data lakes. *Proc. VLDB Endow.*, 17(2):92–105, 2023. URL https://www.vldb.org/pvldb/vol17/p92-arora.pdf.

S. Arora, S. Eyuboglu, A. Timalsina, et al. Zoology: Measuring and improving recall in efficient language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=LY3ukUANko.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning*

Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL http://arxiv.org/abs/1409.0473.

Y. Bai, X. Lv, J. Zhang, et al. Longbench: A bilingual, multitask benchmark for long context understanding. In L. Ku, A. Martins, and V. Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3119–3137. Association for Computational Linguistics, 2024. URL https://doi.org/10.18653/v1/2024.acl-long.172.

I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

A. Bertsch, U. Alon, G. Neubig, et al. Unlimiformer: Long-range transformers with unlimited length input. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

S. Bhakthavatsalam, D. Khashabi, T. Khot, et al. Think you have solved direct-answer question answering? try arc-da, the direct-answer AI2 reasoning challenge, 2021. URL https://arxiv.org/abs/2102.03315.

S. Biderman, H. Schoelkopf, Q. G. Anthony, et al. Pythia: A suite for analyzing large language models across training and scaling. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR, 2023. URL https://proceedings.mlr.press/v202/biderman23a.html.

Y. Bisk, R. Zellers, R. L. Bras, et al. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press, 2020. URL https://doi.org/10.1609/aaai.v34i05.6239.

P. Blanchard, D. J. Higham, and N. J. Higham. Accurately computing the log-sum-exp and softmax functions. *IMA Journal of Numerical Analysis*, 41(4):2311–2330, 2021. doi: 10.1093/imanum/draa038. URL https://academic.oup.com/imajna/article/41/4/2311/5893596.

T. Brooks, B. Peebles, C. Holmes, et al. Video generation models as world simulators, 2024. URL https://openai.com/research/video-generation-models-as-world-simulators.

T. B. Brown, B. Mann, N. Ryder, et al. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

I. Chalkidis, X. Dai, M. Fergadiotis, et al. An exploration of hierarchical attention transformers for efficient long document classification, 2022. URL https://arXiv.org/abs/2210.05529.

B. Chen, Z. Liu, B. Peng, et al. MONGOOSE: A learnable LSH framework for efficient neural network training. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=wWK7yXkULyh.

R. Child, S. Gray, A. Radford, et al. Generating long sequences with sparse transformers, 2019. URL http://arxiv.org/abs/1904.10509.

K. M. Choromanski, V. Likhosherstov, D. Dohan, et al. Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

T. Dao and A. Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=ztn8FCR1td.

P. Dasigi, K. Lo, I. Beltagy, et al. A dataset of information-seeking questions and answers anchored in research papers. In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tür, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 4599–4610. Association for Computational Linguistics, 2021. URL https://doi.org/10.18653/v1/2021.naacl-main.365.

Y. Deng, Z. Song, and C. Yang. Attention is naturally sparse with gaussian distributed input, 2024. URL https://arXiv.org/abs/2404.02690.

J. Devlin, M. Chang, K. Lee, et al. BERT: pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. URL https://doi.org/10.18653/v1/n19-1423.

Y. Dong, P. Indyk, I. P. Razenshteyn, et al. Learning space partitions for nearest neighbor search. *IEEE Data Eng. Bull.*, 47(3):55–68, 2023. URL http://sites.computer.org/debull/A23sept/p55.pdf.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

M. Douze, A. Guzhva, C. Deng, et al. The faiss library, 2024. URL https://arXiv.org/abs/2401.08281.

D. Dua, Y. Wang, P. Dasigi, et al. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2368–2378. Association for Computational Linguistics, 2019. URL https://doi.org/10.18653/v1/n19-1246.

A. R. Fabbri, I. Li, T. She, et al. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In A. Korhonen, D. R. Traum, and L. Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1074–1084. Association for Computational Linguistics, 2019. URL https://doi.org/10.18653/v1/p19-1102.

Y. Fu, Z. Cai, A. Asi, et al. Not all heads matter: A head-level KV cache compression method with integrated retrieval and reasoning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=FJFVmeXusW.

B. Gliwa, I. Mochol, M. Biesek, et al. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *CoRR*, abs/1911.12237, 2019. URL http://arxiv.org/abs/1911.12237.

R. Gomes, A. Krause, and P. Perona. Discriminative clustering by regularized information maximization. In *Advances in Neural Information Processing Systems*, volume 23, 2010. URL https://proceedings.neurips.cc/paper_files/paper/2010/file/42998cf32d552343bc8e460416382dca-Paper.pdf.

A. Gu and T. Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling, COLM 2024, Philadelphia, PA, USA, October 7-9, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

X. Gu, T. Pang, C. Du, et al. When attention sink emerges in language models: An empirical view. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=78Nn4QJTEN.

D. Guo, C. Xu, N. Duan, et al. Longcoder: A long-range pre-trained language model for code completion. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 12098–12107. PMLR, 2023. URL https://proceedings.mlr.press/v202/guo23j.html.

R. Guo, P. Sun, E. Lindgren, et al. Accelerating large-scale inference with anisotropic vector quantization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3887–3896. PMLR, 2020. URL http://proceedings.mlr.press/v119/guo20h.html.

A. Gupta, G. Dar, S. Goodman, et al. Memory-efficient transformers via top-k attention. In N. S. Moosavi, I. Gurevych, A. Fan, T. Wolf, Y. Hou, A. Marasovic, and S. Ravi, editors, *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing, SustaiNLP@EMNLP 2021, Virtual, November 10, 2021*, pages 39–52. Association for Computational Linguistics, 2021. URL https://doi.org/10.18653/v1/2021.sustainlp-1.5.

N. Gupta, P. H. Chen, H. Yu, et al. ELIAS: end-to-end learning to index and search in large output spaces. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

X. Ho, A. D. Nguyen, S. Sugawara, et al. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. In D. Scott, N. Bel, and C. Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6609–6625. International Committee on Computational Linguistics, 2020. URL https://doi.org/10.18653/v1/2020.coling-main.580.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. doi: 10.1080/01621459.1963.10500830. URL https://www.jstor.org/stable/2282952.

J. Huang. How well can a long sequence model model long sequences? comparing architectural inductive biases on long-context abilities. In O. Rambow, L. Wanner, M. Apidianaki, H. Al-Khalifa, B. D. Eugenio, and S. Schockaert, editors, *Proceedings of the 31st International Conference on Computational Linguistics*, pages 29–39, Abu Dhabi, UAE, Jan. 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.3/.

L. Huang, S. Cao, N. N. Parulian, et al. Efficient attentions for long document summarization. In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tür, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 1419–1436. Association for Computational Linguistics, 2021. URL https://doi.org/10.18653/v1/2021.naacl-main.112.

K. Jasińska-Kobus, M. Wydmuch, K. Dembczyński, M. Kuznetsov, and R. Busa-Fekete. Probabilistic label trees for extreme multi-label classification. *arXiv preprint arXiv:2009.11218*, 2020. URL https://arxiv.org/abs/2009.11218.

A. Q. Jiang, A. Sablayrolles, A. Roux, et al. Mixtral of experts, 2024. URL https://arXiv.org/abs/2401.04088.

M. Joshi, E. Choi, D. S. Weld, et al. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In R. Barzilay and M. Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1601–1611. Association for Computational Linguistics, 2017. URL https://doi.org/10.18653/v1/P17-1147.

A. Katharopoulos, A. Vyas, N. Pappas, et al. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR, 2020. URL http://proceedings.mlr.press/v119/katharopoulos20a.html.

N. Kitaev, L. Kaiser, and A. Levskaya. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rkgNKkHtvB.

T. Kociský, J. Schwarz, P. Blunsom, et al. The narrativeqa reading comprehension challenge. *Trans. Assoc. Comput. Linguistics*, 6:317–328, 2018. URL https://doi.org/10.1162/tacl_a_00023.

T. Kraska, A. Beutel, E. H. Chi, et al. The case for learned index structures. In G. Das, C. M. Jermaine, and P. A. Bernstein, editors, *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 489–504. ACM, 2018. URL https://doi.org/10.1145/3183713.3196909.

T. Kwiatkowski, J. Palomaki, O. Redfield, et al. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466, 2019. URL https://doi.org/10.1162/tacl_a_00276.

W. Li, C. Feng, D. Lian, et al. Learning balanced tree indexes for large-scale vector retrieval. In A. K. Singh, Y. Sun, L. Akoglu, D. Gunopulos, X. Yan, R. Kumar, F. Ozcan, and J. Ye, editors, *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pages 1353–1362. ACM, 2023. URL https://doi.org/10.1145/3580305.3599406.

X. Li and D. Roth. Learning question classifiers. In *19th International Conference on Computational Linguistics, COLING 2002, Howard International House and Academia Sinica, Taipei, Taiwan, August 24 - September 1, 2002*, 2002. URL https://aclanthology.org/C02-1150/.

G. Liu, C. Li, J. Zhao, et al. Clusterkv: Manipulating LLM KV cache in semantic space for recallable compression. In *62nd ACM/IEEE Design Automation Conference, DAC 2025, San Francisco, CA, USA, June 22-25, 2025*, pages 1–7. IEEE, 2025. URL https://doi.org/10.1109/DAC63849.2025.11132479.

P. J. Liu, M. Saleh, E. Pot, et al. Generating wikipedia by summarizing long sequences. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=Hyg0vbWC-.

T. Liu, C. Xu, and J. J. McAuley. Repobench: Benchmarking repository-level code auto-completion systems. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=pPjZIOuQuF.

C. Lockard, P. Shiralkar, and X. L. Dong. Openceres: When open information extraction meets the semi-structured web. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 3047–3056. Association for Computational Linguistics, 2019. URL https://doi.org/10.18653/v1/n19-1309.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Y. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(4):824–836, 2020. URL https://doi.org/10.1109/TPAMI.2018.2889473.

Y. Mao, M. Ester, and K. Li. Iceformer: Accelerated inference with long-sequence transformers on cpus. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=6RR3wU4mSZ.

C. Meister, T. Vieira, and R. Cotterell. Best-first beam search. *Transactions of the Association for Computational Linguistics*, 8:795–809, 2020. doi: 10.1162/tacl_a_00346. URL https://aclanthology.org/2020.tacl-1.51/.

S. Merity, C. Xiong, J. Bradbury, et al. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Byj72udxe.

OpenAI. GPT-4 technical report, 2023. URL https://arXiv.org/abs/2303.08774.

M. Pagliardini, D. Paliotta, M. Jaggi, et al. Fast attention over long sequences with dynamic sparse flash attention. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

D. Paperno, G. Kruszewski, A. Lazaridou, et al. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016. URL https://doi.org/10.18653/v1/p16-1144.

G. Penedo, H. Kydlícek, L. B. Allal, et al. The fineweb datasets: Decanting the web for the finest text data at scale. In A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

H. Peng, N. Pappas, D. Yogatama, et al. Random feature attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=QtTKTdVrFBB.

Y. Polyanskiy and Y. Wu. Lecture notes on information theory: f-divergences, n.d. URL https://people.lids.mit.edu/yp/homepage/data/LN_fdiv.pdf. Chapter 7, author-hosted lecture notes.

P. Rajpurkar, R. Jia, and P. Liang. Know what you don't know: Unanswerable questions for squad. In I. Gurevych and Y. Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 784–789. Association for Computational Linguistics, 2018. URL https://aclanthology.org/P18-2124/.

A. Ramesh, M. Pavlov, G. Goh, et al. Zero-shot text-to-image generation. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR, 2021. URL http://proceedings.mlr.press/v139/ramesh21a.html.

A. Roy, M. Saffar, A. Vaswani, et al. Efficient content-based sparse attention with routing transformers. *Trans. Assoc. Comput. Linguistics*, 9:53–68, 2021. URL https://doi.org/10.1162/tacl_a_00353.

K. Sakaguchi, R. L. Bras, C. Bhagavatula, et al. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8732–8740. AAAI Press, 2020. URL https://doi.org/10.1609/aaai.v34i05.6399.

M. Sap, H. Rashkin, D. Chen, et al. Social iqa: Commonsense reasoning about social interactions. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 4462–4472. Association for Computational Linguistics, 2019. URL https://doi.org/10.18653/v1/D19-1454.

S. Sukhbaatar, E. Grave, P. Bojanowski, et al. Adaptive attention span in transformers. In A. Korhonen, D. R. Traum, and L. Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 331–335. Association for Computational Linguistics, 2019. URL https://doi.org/10.18653/v1/p19-1032.

Y. Tay, M. Dehghani, S. Abnar, et al. Long range arena : A benchmark for efficient transformers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=qVyeW-grC2k.

H. Trivedi, N. Balasubramanian, T. Khot, et al. Musique: Multihop questions via single-hop question composition. *Trans. Assoc. Comput. Linguistics*, 10:539–554, 2022. URL https://doi.org/10.1162/tacl_a_00475.

D. Tu, D. Vashchilenko, Y. Lu, et al. Vl-cache: Sparsity and modality-aware KV cache compression for vision-language model inference acceleration. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=HMrcv7Q4Ub.

A. Vaswani, N. Shazeer, N. Parmar, et al. Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

S. Wang, B. Z. Li, M. Khabsa, et al. Linformer: Self-attention with linear complexity, 2020. URL https://arxiv.org/abs/2006.04768.

Z. Wang, B. Cui, and S. Gan. Squeezeattention: 2d management of kv-cache in LLM inference via layer-wise optimal budget. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=9HK2rHNAhd.

Y. Xiong, Z. Zeng, R. Chakraborty, et al. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Event, February 2-9, 2021*, pages 14138–14148. AAAI Press, 2021. URL https://doi.org/10.1609/aaai.v35i16.17664.

S. Yang, B. Wang, Y. Shen, et al. Gated linear attention transformers with hardware-efficient training. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024a. URL https://openreview.net/forum?id=ia5XvxFUJT.

S. Yang, B. Wang, Y. Zhang, et al. Parallelizing linear transformers with the delta rule over sequence length. In A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024b.

Z. Yang, P. Qi, S. Zhang, et al. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics, 2018. URL https://doi.org/10.18653/v1/d18-1259.

J. Yuan, H. Gao, D. Dai, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 23078–23097. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.1126/.

M. Zaheer, G. Guruganesh, K. A. Dubey, et al. Big bird: Transformers for longer sequences. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html.

A. Zandieh, I. Han, V. Mirrokni, et al. Subgen: Token generation in sublinear time and memory, 2024. URL https://arXiv.org/abs/2402.06082.

R. Zellers, A. Holtzman, Y. Bisk, et al. Hellaswag: Can a machine really finish your sentence? In A. Korhonen, D. R. Traum, and L. Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. Association for Computational Linguistics, 2019. URL https://doi.org/10.18653/v1/p19-1472.

Q. Zeng, J. Huang, P. Lu, et al. ZETA: leveraging z-order curves for efficient top-k attention. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=j9VVzueEbG.

Q. Zeng, J. Huang, P. Lu, R. Fang, G. Xu, Z. Jing, Y. Cui, C. Ling, G. Niu, and B. Wang. Arm: Attention with routed-memory for learnable sparse control, 2026a. URL https://arxiv.org/abs/2609.24417.

Q. Zeng, J. Huang, P. Lu, R. Fang, G. Xu, Z. Jing, Y. Cui, C. Ling, G. Niu, and B. Wang. Attention with routed-memory for learnable sparse control. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=6zbO21ySVJ.

Y. Zhang, S. Yang, R. Zhu, et al. Gated slot attention for efficient linear-time sequence modeling. In A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

M. Zhong, D. Yin, T. Yu, et al. Qmsum: A new benchmark for query-based multi-domain meeting summarization. In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tür, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5905–5921. Association for Computational Linguistics, 2021. URL https://doi.org/10.18653/v1/2021.naacl-main.472.

Appendix

A Theoretical Results and Proofs **20**

A.1 Rule 1: proof of Proposition 4.1 . . . 20
A.2 Rule 2: proof of Proposition 4.2 . . . 21
A.3 Sample-entropy dynamics: proof of Lemma 4.3 . . . 22
A.4 Individual path retention: proof of Proposition 4.4 . . . 22
A.5 Joint retrieval: proof of Proposition 4.5 . . . 24
A.6 Query/key roles, reference weighting, and implementation scope . . . 24
A.7 Attention-output consequence and optional re-ranking . . . 25

B Additional Experimental Details **26**

B.1 Implementation Details . . . 26
B.2 Optimized Implementations for Enhancing GPU Efficiency . . . 26
B.3 FAISS Baseline Configuration for Synthetic Gaussian Retrieval . . . 27
B.4 Language Task Details . . . 28
B.4.1 Language Model Evaluation Harness Tasks . . . 28
B.4.2 Recall Intensive Tasks . . . 28
B.4.3 LongBench . . . 28
B.4.4 Single Needle-in-a-Haystack . . . 28
B.5 Experimental Reproducibility . . . 29

C Additional Experiments **30**

C.1 Scaling Results at 1B Parameters . . . 30
C.2 Ablation: Beam Width . . . 30
C.3 Ablation: Top-K Recall on Synthetic Gaussian Retrieval . . . 31
C.4 Ablation on Window Size of the SWA branch . . . 31
C.5 Comparison Against NSA Components . . . 31
C.6 Ablation on Grouped-Query Attention (GQA) . . . 32
C.7 Ablation on Regularization Loss Weighting . . . 32
C.8 Success Rate comparisons with retrieval baselines . . . 32
C.9 Ablation on the Recall Level and Accuracy . . . 33
C.10 Empirical Evidence: Training vs. Non-Training . . . 33
C.11 Hard-Forward Balance and the KL-Uniform Comparison . . . 33
C.12 Comparison with ScaNN-PQ: Training Overhead and Recall–Speed Tradeoff . . . 34
C.13 Extended NIAH Results . . . 34
C.14 Comparison Against Inference-Time KV-Cache Methods . . . 34

C.15 Continual Tuning Ablation 35

D Triton Kernels **37**

E Time Complexity Analysis **39**

E.1 Cache-aware autoregressive decoding 39

E.2 Sorting Latency 40

E.3 Beam Search Latency 40

E.4 Sorting Overhead 40

E.5 Memory Bandwidth 40

A Theoretical Results and Proofs

All logarithms are natural. We use $0 \log 0 = 0$ for occupancies. Finite logits give positive softmax probabilities. Untagged p_i and $p_i^{(l)}$ denote key-side distributions, while query-side distributions use p_q, p_{q_i}, or an explicit argument q. At a common parent, $p_p(x) = \mathrm{softmax}(\ell_p(x))$; thus $p_{q_i} = p_p(q_i)$ and $p_i = p_p(z_i)$. On a greedy key path, $p_i^{(l)}$ is the original algorithm's $p_{K,i}^l$. The representations, centroid parameters, and causal key population are fixed in each statement. The main path certificate uses greedy keys and Assumption 4.1; categorical keys are treated separately.

A.1 Rule 1: proof of Proposition 4.1

For independent $B_i^{\mathrm{qry}} \sim \mathrm{Cat}(p_{q_i})$ and $B_r \sim \mathrm{Cat}(p_r)$ over the same children, $\Pr(B_i^{\mathrm{qry}} = B_r) = (p_{q_i})^\top p_r$. Applying Jensen's inequality for $-\log$ with the *key* distribution as the weights gives

$$-\log\left((p_{q_i})^\top p_r\right) \le -\sum_j p_{r,j} \log p_{q_i,j} = H(p_r) + \mathrm{KL}(p_r \| p_{q_i}). \tag{A.1}$$

Averaging under $\mathcal{D}_K$ gives the co-routing loss inequality in Proposition 4.1; exponentiating before averaging gives its probability inequality. Define the reference marginals $\pi_K^{\mathrm{qry}}(i) = \sum_r \mathcal{D}_K(i,r)$ and $\pi_K^{\mathrm{key}}(r) = \sum_i \mathcal{D}_K(i,r)$, and let $\mathcal{A}_K^{\mathrm{key}} = \mathbb{E}_{\mathcal{D}_K} \mathrm{KL}(p_r \| p_{q_i})$. The same inequalities and Jensen for e^{-x} give the equivalent marginal form:

$$\mathcal{L}_{\mathrm{smp}}^\star \le \sum_r \pi_K^{\mathrm{key}}(r) H(p_r) + \mathcal{A}_K^{\mathrm{key}}, \tag{A.2}$$

$$\mathbb{E}_{\mathcal{D}_K}[p_{q_i}^\top p_r] \ge e^{-\sum_r \pi_K^{\mathrm{key}}(r) H(p_r) - \mathcal{A}_K^{\mathrm{key}}}. \tag{A.3}$$

No tied query/key projections or equality between p_r and p_{q_i} is used.

The original query-oriented form is also valid. Using p_{q_i} as the Jensen weights instead gives

$$-\log\left((p_{q_i})^\top p_r\right) \le H(p_{q_i}) + \mathrm{KL}(p_{q_i} \| p_r).$$

Retaining the original notation $\mathcal{A}_K = \mathbb{E}_{\mathcal{D}_K} \mathrm{KL}(p_{q_i} \| p_r)$,

$$\mathcal{L}_{\mathrm{smp}}^\star \le \sum_i \pi_K^{\mathrm{qry}}(i) H(p_{q_i}) + \mathcal{A}_K. \tag{A.4}$$

Its exponential probability version follows identically. The query and key forms are alternatives, not equal quantities: $\mathcal{A}_K$ and $\mathcal{A}_K^{\mathrm{key}}$ use opposite KL directions. Uniform query weighting recovers a query-entropy average; uniform key weighting recovers the key-side $\mathcal{L}_{\mathrm{smp}}$ defined in the main text. Even uniform query sampling does not generally induce uniform key-neighbor frequencies.

Information-maximization interpretation. Let I be a uniformly sampled key index and let $B \mid I = r \sim \mathrm{Cat}(p_r)$. Then

$$\mathcal{L}_{\mathrm{smp}} + \mathcal{L}_{\mathrm{bal}}(\bar{p}) = H(B \mid I) - H(B) = -\mathrm{I}(I;B). \tag{A.5}$$

This is the conditional-entropy/marginal-entropy decomposition used in information-maximization clustering (Gomes et al., 2010). It explains the complementary preference for decisive assignments and broad utilization. It does not identify clusters with attention relevance. The identity uses the *soft* marginal $\bar{p}$ of this categorical law. Substituting the realized hard marginal $\widehat{p}$ in only the second term does not preserve this exact identity; Proposition 4.2 separately analyzes those counts.

When the co-routing bound is informative. A key entropy plus reverse-KL discrepancy at most $-\log\gamma$ certifies co-routing probability at least γ. With $p_{q_i} = (0.8, 0.1, 0.05, 0.05)$ and $p_r = (0.75, 0.12, 0.08, 0.05)$, the exact probability is 0.6185 and the key-oriented lower bound is $0.434694\ldots$. Neither distribution is a point mass. These are formula evaluations, not checkpoint observations. Conversely, $(1-\epsilon, \epsilon)$ and $(\epsilon, 1-\epsilon)$ both have small entropy but co-route with probability $2\epsilon(1-\epsilon)$. This can occur with shared centroids and true inner-product relevance: take $q = (a, 1)$, $z_1 = (a, -1)$, $z_2 = (0, 1)$, and centroids $(0, t), (0, -t)$ with $a > \sqrt{2}$. The top-one key is z_1, yet it and the query have opposite modes as t grows. Thus compatibility cannot be removed by citing an entropy objective alone.

A.2 RULE 2: PROOF OF PROPOSITION 4.2

For completeness, the main-text binary envelope uses natural-log divergence

$$\mathrm{kl}(x\|s) = x\log\frac{x}{s} + (1-x)\log\frac{1-x}{1-s}, \qquad \mathsf{U}(s,\kappa) = \sup\{x \in [s,1] : \mathrm{kl}(x\|s) \le \kappa\}, \tag{A.6}$$

for $0 < s < 1$, finite $\kappa \ge 0$, and endpoint conventions $\mathsf{U}(0,\kappa) = 0$, $\mathsf{U}(1,\kappa) = 1$. For any probability vector v on C children,

$$\mathrm{KL}(v\|u) = \sum_j v_j \log(Cv_j) = \log C + \mathcal{L}_{\mathrm{bal}}(v). \tag{A.7}$$

Let B have uniform mass $s = m/C \in (0,1)$ and actual mass $x = v(B)$. Applying the log-sum inequality (equivalently, relative-entropy data processing under $j \mapsto \mathbf{1}\{j \in B\}$; Polyanskiy and Wu, n.d., Chapter 7) separately on B and its complement yields

$$\mathrm{KL}(v\|u) \ge x\log\frac{x}{s} + (1-x)\log\frac{1-x}{1-s} = \mathrm{kl}(x\|s). \tag{A.8}$$

For fixed s, the derivative in x is $\log\frac{x(1-s)}{s(1-x)}$. Hence the divergence increases on $[s,1]$, is zero at s, and equals $\log(1/s)$ at one. The definition of U gives $v(B) \le \mathsf{U}(s,\kappa(v))$, whether $x \ge s$ or $x < s$. Applying the same argument to the complement gives

$$v(B) \ge 1 - \mathsf{U}(1-s,\kappa(v)).$$

Empty and full subsets satisfy the stated endpoint conventions. Write $N_B = \sum_{b\in B} N_b$. These are deterministic inequalities for every B, so selecting B after observing v does not consume a failure probability.

Norm control for the ideal loss. For fixed s, the second derivative of $\mathrm{kl}(x\|s)$ is $1/[x(1-x)] \ge 4$, and the divergence and its first derivative vanish at $x = s$. Thus $\mathrm{kl}(x\|s) \ge 2(x-s)^2$. Apply equation A.8 to $B = \{j : v_j > u_j\}$, for which $v(B) - u(B) = \|v-u\|_1/2$, to obtain

$$\mathrm{KL}(v\|u) \ge \tfrac{1}{2}\|v-u\|_1^2 \ge \tfrac{1}{2}\|v-u\|_2^2. \tag{A.9}$$

The case $v = u$ is immediate. Taking $v = \bar{p}$ proves the ideal-loss statement. We do not use the weaker quadratic relaxation in the candidate-load certificate.

Count and variation bounds. For $v = \widehat{\bar{p}}$, $Tv(B) = N_B$, proving equation 5. For each singleton,

$$1 - \mathsf{U}(1-1/C,\kappa(v)) \le v_b \le \mathsf{U}(1/C,\kappa(v)).$$

Subtracting the lower bound from the upper and multiplying by T gives the workload-variation consequence:

$$\max_b N_b - \min_b N_b \le T\big[\mathsf{U}(1/C,\kappa(\widehat{\bar{p}})) + \mathsf{U}(1-1/C,\kappa(\widehat{\bar{p}})) - 1\big]. \tag{A.10}$$

Exact equality of counts is not assumed.

Sharpness and evaluation. For any $x \in [s,1]$, assign mass x/m to each child in B and $(1-x)/(C-m)$ elsewhere. Equality then holds in equation A.8. Therefore U is the smallest universal upper envelope given only κ and s; integer counts can impose additional restrictions. It obeys

$$\mathsf{U}(s,\kappa) = 1 \iff \kappa \ge \log(1/s) \qquad (0 < s < 1).$$

For observed counts this is equivalent to $H(\widehat{\bar{p}}) \le \log m$. Consequently a nontrivial load certificate does not require balance arbitrarily close to uniform. Evaluate the envelope by returning s when $\kappa = 0$, one when $\kappa \ge \log(1/s)$, and otherwise bisecting $[s,1]$. The upper root bracket is conservative in exact arithmetic; ordinary floating-point evaluation is not formal interval certification.

Observed counts and finite populations. κ_L is calculated from the observed population, so it includes any sampling variation. Under the additional model of T independent uniform assignments to $J = C^L$ leaves, its leading expected value is $(J-1)/(2T)$ when bins are well sampled. Indeed, writing $\widehat{\bar{p}}_b = 1/J + \delta_b$, a second-order expansion gives $\kappa_L \simeq (J/2)\sum_b \delta_b^2$, and $\mathbb{E}\delta_b^2 = (1/J)(1-1/J)/T$. This is a reference scale, not a lower bound for deterministic or token-dependent learned assignments. For $J = 1024, T = 32768$, substituting this scale gives $\kappa_L \simeq 0.015610$ and $\mathsf{U}(8/J,\kappa_L) \simeq 0.027883$, or an upper load near 914 keys. This is a formula evaluation, not a predicted HiRouter load. We do not subtract a sampling-bias correction from observed KL in a deterministic certificate.

Which distribution enters the loss. A hard-forward straight-through assignment has forward value e_{B_i}, so its forward marginal is $\widehat{\bar{p}} = T^{-1}\sum_i e_{B_i}$. A soft-marginal loss instead evaluates $\bar{p} = T^{-1}\sum_i p_i$. These values need not agree for greedy assignments. The main load bound uses the former directly. Standard Gumbel-max sampling follows $\mathrm{Cat}(p_i)$; a finite-temperature relaxed vector provides derivatives, not a different categorical law.

Local balance across the hierarchy. For observed counts, each nonempty leaf probability factors into empirical conditional child proportions along its path. Expanding $\log(\widehat{\bar{p}}_b^{(L)}/C^{-L})$, then summing over leaves, assigns weight N_p/T to every parent p. This gives

$$\mathrm{KL}(\widehat{\bar{p}}^{(L)}\|u_{C^L}) = \sum_{l=1}^{L}\sum_{p \text{ at level } l-1}\frac{N_p}{T}\,\mathrm{KL}(\widehat{\bar{p}}_p^{(l)}\|u_C).$$

Empty parents contribute zero. Applying equation A.7 at each parent gives the parent-mass-weighted leaf identity described in the main text. Applying the subset envelope to C^L leaves proves $|S_q| \le D_q$. All counts refer to the same prefix; a pooled batch loss does not automatically certify each prefix.

A.3 Sample-entropy dynamics: proof of Lemma 4.3

The softmax Jacobian is $\partial p_{i,h}/\partial \ell_{i,j} = p_{i,h}(\mathbf{1}\{h=j\} - p_{i,j})$. Therefore

$$\frac{\partial H(p_i)}{\partial \ell_{i,j}} = -p_{i,j}(\log p_{i,j} + 1) + p_{i,j}\sum_h p_{i,h}(\log p_{i,h} + 1) = -p_{i,j}(\log p_{i,j} + H(p_i)).$$

Negating the derivative proves the update statement. At a uniform distribution the gradient is zero. For inner-product logits, a parameter update combines all centroid directions; the scalar sign statement does not assert geometric contraction. As a separate useful identity, scaling fixed logits by $a > 0$ gives

$$\frac{d}{da}H(\mathrm{softmax}(a\ell)) = -a\,\mathrm{Var}_{\mathrm{softmax}(a\ell)}(\ell) \le 0.$$

Sharpening alone does not determine whether two vectors favor the same route.

A.4 Individual path retention: proof of Proposition 4.4

Probability scores and permitted storage normalization. Under Assumption 4.1, each local score is $\log p_{p,j}^{(l)}(q) = \ell_{p,j}^{(l)}(q) - \mathrm{LSE}(\ell_p^{(l)}(q))$. A query prefix has canonical score

$$F_q(a^l) = \sum_{s=1}^{l}\log p_{a^{s-1},j_s}^{(s)}(q).$$

Each conditional distribution sums to one. Thus the probabilities of all expanded prefixes at any depth sum to at most one. A discarded prefix has at least M expanded prefixes with scores at least as high, so the largest discarded probability is at most $1/(M+1)$ and $-\lambda_l(q) \ge \log(M+1) > 0$. This uses the probability-chain scoring convention, rather than raw-logit sums, familiar from probabilistic beam search (Meister et al., 2020).

A common level-wise renormalization does not change the beam order. If all stored candidate scores at depth l are $\widetilde{F}_q(a^l) = F_q(a^l) - c_l$, then the canonical rejected score is

$$\lambda_l(q) = \widetilde{\lambda}_l(q) + c_l. \tag{A.11}$$

Track or recompute this common offset before evaluating the certificate. The offset must be the one applied to the compared candidates, not a later normalization of retained entries. A parent-specific shift can instead change cross-parent rankings. Likewise, summing raw $\langle q, c\rangle$ scores omits parent-dependent log normalizers and is not the same search. Numerically stable shifted log-sum-exp computations remain valid when they produce the same normalized probabilities (Blanchard et al., 2021).

The first-exit implication. For the child chosen by a greedy key, $p^{(s)}_{i,j_i^{(s)}} = \max_j p^{(s)}_{i,j}$. For any categorical distribution, $-\log \max_j p_j \le H(p)$. The definition of $\xi^{(s)}_{qi}$ therefore gives

$$-\log p^{(s)}_{a^{s-1}_{K,i}, j_i^{(s)}}(q) \le -\log p^{(s)}_{i,j_i^{(s)}} + \xi^{(s)}_{qi} \le H(p^{(s)}_i) + \xi^{(s)}_{qi}.$$

Summing along a prefix yields

$$-F_q(a^l_{K,i}) \le \sum_{s=1}^{l} [H(p^{(s)}_i) + \xi^{(s)}_{qi}]. \tag{A.12}$$

If the key's leaf is absent, let l be the first depth where its prefix exits. Its parent was retained, so this prefix was considered and rejected: $F_q(a^l_{K,i}) \le \lambda_l(q)$. equation A.12 implies $\psi_{qi} \ge 1$. For a retained key the miss indicator is zero. Hence the pointwise result equation 7 holds, and $\psi_{qi} < 1$ certifies retention. No pruning means every leaf is selected. This is a first-exit certificate, not a claim that finite-width beam search always returns the globally best leaves.

Finite sharpening is not homogeneity. At one binary level with $M = 1$, query logits $(2a, 0)$ and key logits $(5a, 0)$ give

$$-\lambda_1(q) = \log(1 + e^{2a}), \qquad \xi_{qi} = \log \sigma(5a) - \log \sigma(2a), \quad \sigma(x) = (1 + e^{-x})^{-1}.$$

Neither expression is exactly homogeneous of degree one in a. The continuous certificate is $0.924590\ldots$ at $a = 1$ and $0.999587\ldots$ at $a = 3$; the greedy route is retained at both scales. The example evaluates explicit logits, not model data.

A general large-scale interpretation. Scale every fixed local routing logit vector by $\beta > 0$ and keep fixed tie-breaking. Define its query-side gap $g_{p,j}(q) = \max_h \ell_{p,h}(q) - \ell_{p,j}(q)$ and path cost $G_q(a^l) = \sum_{s=1}^{l} g_{a^{s-1}, j_s}(q)$. The elementary log-sum-exp bounds give

$$-\log p_{p,j}(q;\beta) = \beta g_{p,j}(q) + \rho_p(q;\beta), \qquad 0 \le \rho_p(q;\beta) \le \log C. \tag{A.13}$$

Indeed, factor out the largest exponential; the residual sum lies in $[1, C]$. For a greedy key child its own key-side gap is zero, so $\xi^{(s)}_{qi}(\beta)/\beta \to g_{a^{s-1}_{K,i}, j_i^{(s)}}(q)$. Also $H(p^{(s)}_i(\beta))/\beta \to 0$ because entropy is bounded by $\log C$.

For this asymptotic interpretation only, suppose the beam and rejected-prefix sets eventually stabilize, and at each pruning level the limiting rejected set $\mathcal{R}_l$ has $b_l = \min_{a^l \in \mathcal{R}_l} G_q(a^l) > 0$. Then

$$\frac{-\lambda_l(q;\beta)}{\beta} \longrightarrow b_l, \qquad \psi_{qi}(\beta) \longrightarrow \max_{l \in \mathcal{P}_q} \frac{G_q(a^l_{K,i})}{b_l}. \tag{A.14}$$

The finite maximum and the uniformly bounded residuals justify both limits. Thus the limiting certificate compares query costs on the key's branches to rejected-path costs. If all those key branches are query maximizers, $G_q = 0$ and the certificate tends to one. Different branch-gap magnitudes can give a different limit even with the same pattern of mode agreement. Tied or zero-leading-cost cases do not justify dividing by b_l; the exact finite-scale certificate remains valid there. None of these limiting assumptions is needed for Proposition 4.4.

A.5 JOINT RETRIEVAL: PROOF OF PROPOSITION 4.5

From Proposition 4.4,

$$\mathbf{1}\{i \in S_q\} \geq [1 - \psi_{qi}]_+.$$

Averaging over $N_k(q)$ proves $\text{Recall@}\, k(q) \geq r_q$. The observed-count envelope gives $|S_q| \leq D_q$ for the same assignment. Hence

$$\text{Precision}(q) = \frac{k\,\text{Recall@}\, k(q)}{\max\{1, |S_q|\}} \geq \frac{k r_q}{\max\{1, D_q\}}.$$

No stochastic independence is needed for this deterministic result. Empty candidate sets yield zero recall and precision. Every certified neighbor contributes separately, so a large mismatch on one key cannot cancel another key's contribution. For instance, clipped contributions corresponding to ratios $(0.1, 0.2, 0.3, 8)$ average to 0.6, whereas clipping their mean yields zero.

A stronger discrete certificate. The same proof gives

$$\text{Recall@}\, k(q) \geq \frac{1}{k} \sum_{i \in N_k(q)} \mathbf{1}\{\psi_{qi} < 1\} \geq r_q. \tag{A.15}$$

Replacing $H(p_i^{(s)})$ in ψ_{qi} by the chosen-child surprisal $-\log p^{(s)}_{i,j_i^{(s)}}$ can only strengthen both versions. The main text keeps the key entropy to expose its connection to the key-side sample functional. Appendix A.6 distinguishes that functional from an implementation's stream and node weights.

Categorical paths. Fix the query beam and representations independently of the current key-path draws. Let $P_i(b)$ be the categorical probability of path b. For a sampled path, replace greedy entropy by realized surprisal and define

$$\psi^{\text{cat}}_{qi}(b) = \max_{l \in \mathcal{P}_q} \frac{-\log P_i(b_{1:l}) + \sum_{s=1}^{l} \xi^{(s)}_{qi}(b)}{-\lambda_l(q)}, \quad r^{\text{cat}}_q = \frac{1}{k} \sum_{i \in N_k(q)} \mathbb{E}_{b \sim P_i}[1 - \psi^{\text{cat}}_{qi}(b)]_+. \tag{A.16}$$

Here ξ uses the actual child of b, whether or not it is modal. The first-exit proof gives $\mathbb{E}[\text{Recall@}\, k(q)] \geq r^{\text{cat}}_q$. Entropy enters only after expectation: $\mathbb{E}[-\log P_i(b)] = H(P_i) = \mathbb{E}\sum_{p \prec b} H(p_p(z_i))$. It would be incorrect to replace a sampled path's surprisal pointwise by entropy.

Applying Hoeffding's bounded-sum inequality (Hoeffding, 1963), if key paths are independent conditional on the fixed layer, the k retention indicators are independent. For $0 < \rho < 1$, set $\delta_q = \sqrt{\log(1/\rho)/(2k)}$. Then, simultaneously with probability at least $1 - \rho$,

$$\text{Recall@}\, k(q) \geq [r^{\text{cat}}_q - \delta_q]_+, \qquad \text{Precision}(q) \geq \frac{k[r^{\text{cat}}_q - \delta_q]_+}{\max\{1, D_q\}}. \tag{A.17}$$

For completeness, an independent sum of k variables in $[0, 1]$ has centered log moment-generating function at most $kt^2/8$: the second derivative of each log moment-generating function is a tilted variance, at most $1/4$. Integrating twice proves this bound. Markov's inequality and optimization over t yield the lower-tail probability $e^{-2s^2/k}$. Taking $s = \sqrt{(k/2)\log(1/\rho)}$ proves equation A.17. The realized-count D_q holds for every outcome and needs no additional failure budget. This sampling result is not an independence claim about a full autoregressive rollout.

A.6 QUERY/KEY ROLES, REFERENCE WEIGHTING, AND IMPLEMENTATION SCOPE

Which entropy is regularized. The key-side functional used in the main text is $T^{-1}\sum_i H(p_p(z_i))$. A query-side functional would instead be $T^{-1}\sum_i H(p_p(q_i))$ on the same index population. They need not agree when $q_i = x_i W_Q^\top$ and $z_i = x_i W_K^\top$, even if the centroid bank is shared. More generally, an implementation may pool them with nonnegative weights,

$$\mathcal{L}^{\text{impl}}_{\text{smp}} = w_Q \mathcal{L}^{\text{qry}}_{\text{smp}} + w_K \mathcal{L}^{\text{key}}_{\text{smp}},$$

with its existing node/layer reductions also included. This is bookkeeping, not a proposed extra objective. If $w_K > 0$, the key entropy terms appear explicitly in that scalar; if $w_K = 0$, they are diagnostics, not directly regularized terms. The same distinction applies to query-side terms. The main Proposition 4.1 is key-oriented so that it and Proposition 4.4 refer to the same stream. equation A.4 supplies the query-oriented alternative. Equality of Q/K distributions is never required.

Weights and visited parents also matter. For greedy key paths on the same causal population,

$$\mathcal{L}_{\mathrm{smp},k}(q) = \frac{1}{k} \sum_{i \in N_k(q)} \sum_{l=1}^{L} H(p_i^{(l)})$$

is a reference-neighbor reweighting of *key-side* local entropies, and

$$\mathcal{L}_{\mathrm{smp}}^{\mathrm{path}} = \frac{1}{T} \sum_{i=1}^{T} \sum_{l=1}^{L} H(p_i^{(l)}) = \sum_{l=1}^{L} \sum_{p \text{ at level } l-1} \frac{N_p}{T} \left[\frac{1}{N_p} \sum_{i \text{ at } p} H(p_i^{(l)}) \right]. \tag{A.18}$$

Empty parents contribute zero. This is an identity for the declared key population and routing mode, not an equality to an arbitrary pooled batch loss. Query-beam visits, sampled key visits, and greedy key visits can involve different parents. An implementation-to-certificate comparison must use its actual stream weights, node weights, and assignment mode. Nonnegativity gives $\mathcal{L}_{\mathrm{smp},k}(q) \le (T/k)\mathcal{L}_{\mathrm{smp}}^{\mathrm{path}}$, but this possibly loose transfer is not used to instantiate r_q.

Relation to existing tree theory. Probabilistic label trees connect local probability estimators to global label retrieval (Jasińska-Kobus et al., 2020). Those supervised models have label-dependent node targets; their consistency results cannot be transferred merely by replacing the node loss with entropy. Here the valid connection is the explicitly proved first-exit inequality for the specified normalized beam, together with the classical data-processing load bound. Neither trained calibration nor guaranteed reduction of $\mathcal{A}_K^{\mathrm{key}}$ is assumed.

Diagnostic inputs and cache policy. Observed counts, local key entropies, and beam thresholds need no dense attention matrix. Computing ξ_{qi} or ψ_{qi} for a specified key can require extra query-to-centroid scores along that key's path. Identifying $N_k(q)$ for reference recall requires an oracle score comparison for each diagnostic query; no free dense oracle during sparse training is claimed. Evicting e_q reference keys lowers the full-cache recall certificate by at most e_q/k (clipped at zero); a new count bound uses the actual truncated cache. Occupancy-dependent sampled eviction needs separate dependence handling. Grouped queries use the actual group representative in routing scores and thresholds, but the individual query in attention scores and $N_k(q)$. Candidate selection must remain causal. Sorting changes layout, not logical bucket sizes.

The theoretical bounds apply to the logical routing assignments before the optional equal-capacity physical layout. If this layout changes e_q^{layout} reference top-k keys, the corresponding recall lower bound decreases by at most e_q^{layout}/k, analogously to cache eviction.

A.7 Attention-output consequence and optional re-ranking

Attention over the whole candidate set. Fix scores $s_i = \langle q, z_i \rangle / \sqrt{d}$, full softmax weights $a_i = e^{s_i} / \sum_j e^{s_j}$, and values with $\|v_i\| \le V$. In this paragraph the sparse branch applies softmax over *all* indices in S_q, without an additional token filter. For nonempty S_q, write $m_{S_q} = \sum_{i \in S_q} a_i$ and $r_{S_q} = m_{S_q}^{-1} \sum_{i \in S_q} a_i v_i$. Partitioning the full weighted sum gives

$$r_{\mathrm{full}} - r_{S_q} = (1 - m_{S_q})(r_{S_q^c} - r_{S_q}), \qquad \|r_{\mathrm{full}} - r_{S_q}\| \le 2V(1 - m_{S_q}).$$

For $m_{S_q} = 1$ the difference is zero. This is also the standard total-variation expectation bound: the full law and its conditional law on S_q have total variation $1 - m_{S_q}$ (Polyanskiy and Wu, n.d., Chapter 7). It does not require a geometric radius bound.

Define $E_q = \sum_i a_i \min\{1, \psi_{qi}\} \in [0, 1]$. Weighting the first-exit inequality gives

$$1 - m_{S_q} \le E_q, \qquad \|o_{\text{full-route}} - o_{S_q\text{-route}}\| \le 2 g_r V E_q, \tag{A.19}$$

where the routed gate $0 \le g_r \le 1$, representations, values, local branch, and other branches are fixed. Empty routed sets output zero, for which $2g_rV$ is a conservative bound.

An additional token-level filter has its own mass loss. Top-k attention explicitly restricts the score support (Gupta et al., 2021). Let $R_q \subseteq S_q$ be the indices actually used after any further selection, such as exact top-k re-ranking. For nonempty R_q, S_q, define the within-candidate discarded mass

$$\eta_q = 1 - \frac{\sum_{i \in R_q} e^{s_i}}{\sum_{i \in S_q} e^{s_i}} = 1 - \frac{m_{R_q}}{m_{S_q}}. \tag{A.20}$$

Then the *exact* mass identity and the resulting error bound are

$$1 - m_{R_q} = \eta_q + (1 - \eta_q)(1 - m_{S_q}), \tag{A.21}$$

$$\|o_{\text{full-route}} - o_{R_q\text{-route}}\| \le 2 g_r V \big[\eta_q + (1 - \eta_q) E_q\big]. \tag{A.22}$$

The identity follows by $m_{R_q} = (1 - \eta_q) m_{S_q}$; the error inequality follows from the same conditional-expectation argument on R_q. When $R_q = S_q$, $\eta_q = 0$ and equation A.19 is recovered. For an empty final support, take $\eta_q = 1$ and use the zero-output convention. Evaluating η_q needs only scores inside S_q, though evaluating full attention weights in E_q is a separate diagnostic cost. Under categorical routing, take expectation of the entire right side using the sampled-path certificate; do not factor η_q from dependent quantities.

Exact re-ranking preserves candidate top-k recall, not candidate mass. Let R_q be the first $\min\{k, |S_q|\}$ entries of S_q under the same score and total tie order used to define $N_k(q)$. Then

$$N_k(q) \cap R_q = N_k(q) \cap S_q. \tag{A.23}$$

Any globally top-k key has at most $k - 1$ predecessors in the full ordering, hence at most $k - 1$ in its candidate subset. It cannot be discarded by exact top-k re-ranking when present. If $|S_q| < k$, $R_q = S_q$ directly. Thus the reference recall bound remains valid, but candidate precision in Proposition 4.5 has denominator $|S_q|$, not the final support size. Extra score approximation, a different tie rule, or selection of fewer than k tokens requires separate accounting. Even when reference recall is unchanged, non-top-k candidate keys can carry substantial softmax mass, which is why the η_q term must remain. All output bounds concern a fixed-layer support substitution, not final perplexity or separately trained models.

B Additional Experimental Details

B.1 Implementation Details

Top-k Attention with Softmax Gating. We extend our architecture into a three-branch structure combining a Softmax-Weighted Average (SWA) branch, a learned bias branch, and a HiRouter sparse top-k attention branch. A gating head produces mixing weights through a $\mathrm{softmax}$, adaptively balancing contributions from the three branches. The SWA branch provides dense contextual aggregation; the learned bias branch adds a trainable bias vector weighted by the gate to absorb uncertain queries and stabilizes training similar to attention sinks (Gu et al., 2025); and the HiRouter branch delivers high-precision retrieval by selecting a small set of relevant tokens.

Grouped-Query Retrieval. In addition, we employ grouped-query attention (GQA) (Ainslie et al., 2023) to enhance computational efficiency. Instead of retrieving buckets for each query independently, we compute the average of queries within a group and use this group representative to identify the top candidate buckets. All queries in the group then share these buckets during retrieval.

All experiments were conducted on a single machine with 8 NVIDIA H100 80GB GPUs connected with HBM3. Experiments were run in an environment using CUDA version 12.6 and PyTorch 2.6.0.

B.2 Optimized Implementations for Enhancing GPU Efficiency

HiRouter supports both variable-size and equal-capacity physical layouts. An offset-based layout preserves the logical bucket assignments exactly by storing per-bucket counts and boundaries. Our current optimized kernel instead uses an equal-capacity layout for more regular batched execution. When logical occupancies are uneven, this layout can slightly alter the physical assignment. Balanced routing reduces this discrepancy and keeps the resulting workloads regular.

KV-cached decoding. During prefill, each key is routed once and inserted into the leaf bucket corresponding to its greedy path. Each layer maintains bucketed K/V storage, per-bucket write pointers and valid masks, together with the local sliding window. At each decoding step, HIROUTER computes the new Q/K/V, routes the query to its top-M leaf buckets, routes the new key to one leaf bucket, inserts that K/V pair into the corresponding cache, and attends only to the selected routed entries and the local window. With a fixed-capacity cache, the oldest entry in a full bucket is replaced. Past key assignments are not recomputed and routing probabilities need not be stored.

End-to-end benchmark configuration. The cache-aware measurements use an approximately 410M-parameter modified Llama3 model with 24 layers, 16 attention heads, hidden size 1024, batch size 1, BF16 precision, and a single NVIDIA A100 40GB GPU. Routing, sorting, and gathering overheads are included. Memory is reported as peak GPU usage minus the steady-state memory with only the model loaded. Generation latency is measured after prefill has constructed the KV cache.

Table 7: Autoregressive generation latency (ms/token) after different prefill lengths.

Prefill length	1K	2K	4K	8K
FA (full cache)	16.2	24.7	34.5	58.4
HIROUTER (full cache)	15.3	19.6	25.5	40.8
FA (SWA-1024)	15.9	16.1	16.0	15.5
HIROUTER (fixed cache-1024)	15.5	15.8	15.6	15.7

Table 8: Peak incremental GPU memory (GB) after different prefill lengths.

Prefill length	1K	2K	4K	8K
FA (full cache)	2.6	4.3	8.4	15.4
HIROUTER (full cache)	2.5	4.0	7.2	13.1
FA (SWA-1024)	2.7	2.7	2.7	2.7
HIROUTER (fixed cache-1024)	2.5	2.5	2.5	2.5

Table 9: End-to-end prefill latency (ms).

Prompt length	1K	4K	16K	64K
FA	11.5	24.3	48.3	92.2
HIROUTER	10.8	21.1	32.5	49.6

B.3 FAISS BASELINE CONFIGURATION FOR SYNTHETIC GAUSSIAN RETRIEVAL

To ensure reproducibility and clarify the interpretation of our comparisons, we provide the explicit configuration parameters used for the Faiss-GPU baselines.

K-MEANS (`faiss.Kmeans`).

- `num_clusters = max(1, sequence_length // 32)`: one cluster is allocated per 32 samples, with at least one cluster enforced.
- `niter = 20`: the number of iterations.

LSH (`faiss.IndexLSH`).

- `n_bits = 10`: number of bits used to represent each vector in the LSH index.

HNSW (`faiss.IndexHNSWFlat`).

- `M = 32`: maximum number of links (neighbors) maintained per node.

- `efConstruction = 40`: size of the candidate list during index construction, where larger values improve recall at the cost of higher construction time.
- `efSearch = 128`: size of the candidate list during query search, trading recall for search efficiency.

These parameter settings follow standard recommendations in the Faiss library, where `M`, `efConstruction`, and `efSearch` are the primary controls for the accuracy–efficiency tradeoff in LSH and HNSW.

Implementation-matched Flash-KMeans baseline. To reduce backend effects in the clustering comparison, we additionally use the official Triton implementation of Flash-KMeans (version 0.3.1). We use FP16 inputs on the same RTX 4090, with $B \times H = 4$, $d = 128$, and the same number of Flash-KMeans clusters and HIROUTER leaf buckets ($K = C^L = 256$). Flash-KMeans uses 10 Lloyd iterations. We compare its complete fitting time with HIROUTER's total prompt-routing time, including both query and key routing.

Table 10: Implementation-matched routing/clustering time (ms).

Prompt length	8K	16K	32K	65K
Flash-KMeans	5.636	5.267	6.781	7.069
HIROUTER total routing	1.188	1.304	1.657	2.462
Speedup	4.7×	4.0×	4.1×	2.9×

B.4 LANGUAGE TASK DETAILS

Here we list some additional details regarding the different tasks on which we conduct language model evaluation.

B.4.1 LANGUAGE MODEL EVALUATION HARNESS TASKS

The following are recall-intensive tasks on which we evaluate. All tasks are evaluated directly using accuracy for commonsense reasoning tasks and perplexity for language modeling.

Table 11: Harness tasks.

Task	Task Type
PIQA (Bisk et al., 2020)	Physical Commonsense Reasoning
ARC (Bhakthavatsalam et al., 2021)	Commonsense Reasoning
HELLASWAG (Zellers et al., 2019)	Commonsense Natural Language Inference
WINOGRANDE (Sakaguchi et al., 2020)	Pronoun Resolution
SIQA (Sap et al., 2019)	Social Commonsense Reasoning
BOOLQ	Yes/No Commonsense QA
WIKITEXT (Merity et al., 2017)	Language Modeling
LAMBADA (Paperno et al., 2016)	Text Understanding

B.4.2 RECALL INTENSIVE TASKS

The following are recall-intensive tasks on which we evaluate. All tasks are evaluated directly with accuracy reported as the metric of choice.

B.4.3 LONGBENCH

We evaluate the following tasks from LongBench (Bai et al., 2024) (Table 13). Due to our pre-training on an English dataset, we choose to use only the English language tasks included in the benchmark.

B.4.4 SINGLE NEEDLE-IN-A-HAYSTACK

We utilize the Single Needle-in-a-Haystack (S-NIAH) task on three settings.

Table 12: Recall-intensive tasks.

Task	Task Type
STRUCTURED WEB DATA EXTRACTION (SWDE) (Lockard et al., 2019)	Structure HTML Relation Extraction
FDA (Arora et al., 2023)	PDF Key-Value Retrieval
SQUAD (Rajpurkar et al., 2018)	Question Answering
TRIVIAQA (Joshi et al., 2017)	Question Answering
DROP (Dua et al., 2019)	Question Answering
NATURAL QUESTIONS (Kwiatkowski et al., 2019)	Question Answering

Table 13: Tasks from LONGBENCH.

Task	Context Type	Average Length	Metric	Data Samples
NARRATIVEQA (Kociský et al., 2018)	Literature/Film	18409	F1	200
QASPERQA (Dasigi et al., 2021)	Science	3619	F1	200
MULTIFIELDQA (Bai et al., 2024)	Multi-Field	4559	F1	150
HOTPOTQA (Yang et al., 2018)	Wikipedia	9151	F1	200
2WIKIMULTIQA (Ho et al., 2020)	Wikipedia	4887	F1	200
MUSIQUE (Trivedi et al., 2022)	Wikipedia	11214	F1	200
GOVREPORT (Huang et al., 2021)	Government Reports	8734	Rouge-L	200
QMSUM (Zhong et al., 2021)	Meetings	10614	Rouge-L	200
MULTINEWS Fabbri et al. (2019)	News	2113	Rouge-L	200
TREC (Li and Roth, 2002)	Web Questions	5117	Accuracy	200
TRIVIAQA (Joshi et al., 2017)	Wikipedia/Web	8209	F1	200
SAMSUM (Gliwa et al., 2019)	Dialogue	6258	Rouge-L	200
LCC (Guo et al., 2023)	Github	1235	Edit Similarity	500
REPOBENCH-P (Liu et al., 2024)	Github Repositories	4206	Edit Similarity	500

- S-NIAH-1: The key type is a word and the value type is a number. The haystack consists of repeated sentences. This is referred sometimes as passkey retrieval.
- S-NIAH-2: The key type is a word and the value type is a number. The haystack consists of Paul Graham Essays. This is referred to as vanilla NIAH.
- S-NIAH-3: The key type is a word and the value type is a UUID. The haystack consists of Paul Graham Essays.

For evaluating correctness on NIAH, the model is made to generate a sequence. If the generation contains the correct value, the model is considered correct. Performance is reported in terms of accuracy.

B.5 EXPERIMENTAL REPRODUCIBILITY

For full transparency, we provide our code within the supplemental material. This includes the code used directly to evaluate our models. Our code is based directly on the packages used for evaluating the models:

- `lm-evaluation-harness`: We use this package to evaluate on commonsense reasoning (Table 2) and real-world recall tasks (Table 3).
  - https://github.com/EleutherAI/lm-evaluation-harness
- LongBench: We use this to evaluate on LongBench tasks (Table 4).
  - https://github.com/THUDM/LongBench
- RULER: We use this package to evaluate on NIAH tasks (Table 5).
  - https://github.com/NVIDIA/RULER

For training baselines, we utilized the `flame` (https://github.com/fla-org/flame) package along with their provided model configurations. We change the tokenizer to use the `EleutherAI/gpt-neox-20b` tokenizer and make according changes to the special token ids to support the tokenizer.

C Additional Experiments

C.1 Scaling Results at 1B Parameters

We further evaluate HiRouter at the 1B parameter scale. As shown in Table 14, the method continues to demonstrate strong performance, extending the robustness observed at the 410M scale (see Table 2). These results reinforce that HiRouter scales effectively with model size across diverse language modeling and reasoning tasks. We also note that additional scaling studies, particularly on parameter and hyperparameter tuning, would further support broader adoption, which we leave to future work.

Results. Table 14 compares two variants: one without the learned bias branch (`w/o bias`) and one with a learned bias branch (`w/ bias`). The bias branch yields consistent improvements, highlighting its role in stabilizing training and enhancing generalization as model size grows. Beyond task-wise comparisons, the tighter gap between HiRouter-1B and the Transformer baseline in Table 15 suggests that HiRouter continues to scale predictably under substantially longer training horizons (65B tokens). The fact that the performance differences remain small, yet consistently favorable to HiRouter-1B on several tasks, indicates that the routing mechanism does not introduce optimization instability as models grow larger and training runs become more compute-intensive. This behavior is encouraging, as many sparse or structured attention variants face degradation or divergence when extended to higher-capacity regimes. Overall, the results in Table 15 reinforce that HiRouter is not only effective at moderate scales but also robust under realistic, long-sequence, billion-parameter training conditions.

Table 14: Results at the 1B scale on language modeling and zero-shot common-sense reasoning with 10 billion tokens of training data.

Model	Wiki. ppl ↓	LMB. ppl ↓	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc_n ↑	SIQA acc ↑	BoolQ acc ↑	Avg.
Transformer	102.64	24.37	38.07	69.48	45.73	54.38	57.45	31.40	39.61	58.59	**49.33**
Mamba	26.36	28.89	34.47	69.53	44.77	52.01	56.40	30.55	39.30	59.72	48.34
HiRouter `w/o bias`	26.29	30.56	35.80	68.23	41.90	53.12	63.17	29.69	39.30	58.99	48.78
HiRouter `w/ bias`	26.01	27.21	36.06	68.82	42.99	53.35	62.96	29.27	39.30	59.79	<u>49.07</u>

Table 15: Results at the 1B scale on language modeling and zero-shot common-sense reasoning with 65 billion tokens of training data.

Model	Wiki. ppl ↓	LMB. ppl ↓	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc_n ↑	SIQA acc ↑	BoolQ acc ↑	Avg.
Transformer	18.53	14.60	46.15	72.74	56.29	57.38	66.12	37.03	**42.78**	**63.43**	**55.24**
Mamba	20.37	16.54	43.43	72.36	54.89	56.12	64.35	37.12	40.47	56.64	53.17
Mamba2	19.43	15.09	44.22	**72.96**	56.37	57.62	65.82	38.05	41.91	50.00	53.57
DeltaNet	**17.80**	14.23	45.02	72.78	56.31	56.91	63.93	36.95	41.20	58.75	53.98
GDN	18.02	14.49	45.37	72.72	56.20	58.19	65.58	37.04	42.12	61.49	54.83
GSA	18.34	14.31	46.68	72.61	55.98	**62.62**	**67.07**	**37.99**	39.28	57.49	54.97
HiRouter	17.87	**13.80**	**47.12**	72.54	**56.67**	57.27	66.44	36.29	42.52	62.59	55.18

C.2 Ablation: Beam Width

We investigate how increasing beam width (without retraining) affects performance in two settings: SNIAH-1 and WikiText-103. The results are shown in Tables 16 and 17.

Table 16: SNIAH-1: Retrieval accuracy at different beam widths

	1K	2K	4K	8K
HiRouter (width = 4)	93.6%	86.8%	57.4%	22.4%
HiRouter (width = 8)	96.4%	88.0%	60.2%	33.2%

Even without retraining, increasing beam width from 4 to 8 in SNIAH-1 leads to higher recall. Yet on WikiText-103, further increases beyond width 3 or 4 show diminishing gains in perplexity.

Table 17: WikiText-103: Perplexity (↓) vs beam width

Width	1	2	4	8
Perplexity	20.7	19.4	18.5	18.6

This suggests a moderate beam width yields the best practical trade-off between accuracy and computational cost.

C.3 Ablation: Top-K Recall on Synthetic Gaussian Retrieval

We perform ablations for the recall task on synthetic Gaussian retrieval, on datasets of total length 2^{12} tokens. We examine how beam width, number of buckets, and routing levels each affect Recall@128 under fixed budget settings.

Beam Width Ablation (with $\mathsf{num_levels} = 4,\ \mathsf{num_bucket} = 4$**)**

Beam Width M	4	8	12	16	32
Recall @128 (%)	14.0	25.3	34.0	39.4	62.1

Recall increases monotonically with beam width, confirming that enlarging the search beam systematically improves top-k retrieval accuracy (at the cost of higher runtime).

Bucket Count Ablation (adjusted beam width for fairness, $\mathsf{num_levels} = 4$**)**

num_bucket	2	4	6	8
Recall @128 (%)	33.1	**39.4**	38.0	36.4

We see the best recall at $\mathsf{num_bucket} = 4$. Fewer buckets make the tree too broad and reduce specialization; too many buckets fragment retrieval too finely, decreasing recall.

Level Depth Ablation (adjusted beam width, $\mathsf{num_bucket} = 4$**)**

num_level	2	3	4	5
Recall @128 (%)	33.8	**39.4**	39.1	37.6

A routing depth of 3 levels achieves the best recall. Both shallower and deeper trees reduce performance, due respectively to coarse bucket granularity or excessive fragmentation of tokens.

Summary of Findings These ablations indicate that for sequence length 2^{12} and top-128 recall: (i) moderate beam widths (e.g. 12 or 16) yield strong gains without excessive overhead; (ii) a balance of bucket width (4) gives the right granularity; and (iii) a mid-level tree depth (3 levels) maximizes recall efficiency. Overly coarse or overly fine configurations degrade performance.

C.4 Ablation on Window Size of the SWA branch

We further ablate the impact of the attention window size on language modeling and zero-shot reasoning performance. Table 18 reports results for window sizes 32, 64, and 128 across the same evaluation benchmarks as in the main paper.

We observe that moderate window sizes (e.g., 64) provide the best overall performance, balancing perplexity and accuracy across tasks. Too small a window (32) reduces model expressiveness, while larger windows (128) slightly degrade recall and downstream accuracy.

C.5 Comparison Against NSA Components

To isolate HiRouter's effect, we can provide controlled ablations in Table 19.

Table 18: Ablation on window size for HIROUTER. We report accuracy (%) on common-sense reasoning benchmarks.

Window Size	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc_n ↑	SIQA acc ↑	BoolQ acc ↑	Avg.
32	27.38	66.16	38.18	52.64	58.88	28.41	38.33	61.13	46.38
64	33.09	66.81	38.03	50.75	59.47	28.50	38.08	61.31	**47.01**
128	30.33	65.72	37.62	52.57	58.75	27.82	38.54	52.48	45.48

Table 19: Comparision of individual components of HIROUTER.

Setup	Wiki. ppl ↓	LMB. ppl ↓	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc_n ↑	SIQA acc ↑	BoolQ acc ↑	Avg.
SWA	32.56	44.21	30.56	64.48	36.56	49.88	58.09	24.72	37.65	56.30	44.78
Top-K	33.52	44.65	31.63	63.59	35.27	50.12	57.10	25.91	37.92	59.15	45.08
NSA	31.54	43.0	36.48	66.27	35.37	50.19	59.27	27.33	39.41	59.66	46.75
NSA*	30.74	43.08	36.56	66.35	38.08	50.48	59.17	28.23	39.06	59.20	47.14
Ours	31.09	42.94	33.09	66.81	38.03	50.75	59.47	28.50	38.08	61.31	47.01

From this table, we observe that

- HIROUTER's top-k branch alone outperforms SWA, validating the routing mechanism.
- NSA* (Replacing NSA's sparse branch with HIROUTER) yields +0.39 avg accuracy, demonstrating HIROUTER to be a superior drop-in replacement for block-based top-k attention in NSA.

C.6 Ablation on Grouped-Query Attention (GQA)

To study the effect of grouped-query attention (GQA) (Ainslie et al., 2023), we conduct an ablation on WIKITEXT-103. We vary the group size G while keeping other hyper-parameters fixed, and report perplexity in Table 20.

We observe that larger group sizes ($G = 16$) slightly degrade performance due to excessive parameter sharing, while reducing the group size consistently improves perplexity. At $G = 1$, which corresponds to standard multi-head attention without grouping, the model achieves the best perplexity (18.5). These results highlight the trade-off between efficiency and modeling capacity: GQA provides computational savings at the cost of a small increase in perplexity, while smaller groups preserve model expressivity.

C.7 Ablation on Regularization Loss Weighting

Separate contribution of the two routing losses. We first ablate the two routing terms independently on WikiText-103.

Neither term alone reproduces the full improvement. This is consistent with their complementary roles: $\mathcal{L}_{\text{smp}}$ encourages confident routing decisions, while $\mathcal{L}_{\text{bal}}$ discourages collapse and controls candidate load.

We ablate the effect of the dual-entropy regularization weight on performance across language modeling and zero-shot commonsense reasoning tasks. Table 23 reports results when varying the regularization α coefficient from 0.00 (no regularization) to 0.10.

We find that moderate weighting (e.g., 0.05) achieves the best overall trade-off across tasks, improving average performance compared to both no regularization (0.00) and heavier weighting (0.10). This supports the view that the dual-entropy loss is most effective when applied as a lightweight regularizer, sharpening token-to-bucket assignments without overwhelming the training objective.

C.8 Success Rate comparisons with retrieval baselines

We compare HIROUTER with standard Approximate Nearest Neighbor (ANN) methods, including HNSW, LSH, and K-MEANS, as summarized in Figure 4. As shown in Table 25, evaluated on

Table 20: Ablation of GQA group size on WIKITEXT-103. Smaller group sizes correspond to fewer queries sharing key-value projections.

Group Size	16	4	2	1
Perplexity	19.1	18.8	18.6	**18.5**

Table 21: WikiText-103 perplexity under separate routing-loss ablations.

Routing objective	PPL ↓
No routing loss	24.6
$\mathcal{L}_{\text{smp}}$ only	23.8
$\mathcal{L}_{\text{bal}}$ only	24.3
$\mathcal{L}_{\text{smp}} + \mathcal{L}_{\text{bal}}$	**18.5**

$X \in \mathbb{R}^{32\times4096\times64}$, HIROUTER achieves a success rate competitive with or superior to classical ANN baselines while operating orders of magnitude faster.

C.9 ABLATION ON THE RECALL LEVEL AND ACCURACY

To obtain a denser measure of retrieval quality than task-average accuracy, we also control the fraction of the full-attention top-k keys made available to the routed branch on WikiText-103 at Table 26.

Perplexity improves monotonically as more reference neighbors are recovered, but the gain shows diminishing returns: most of the improvement is obtained by moderate recall, while the difference between 75% and 100% is small. We therefore view high-fidelity retrieval as useful up to a saturation regime, rather than expecting downstream quality to improve proportionally with recall.

In Table 24, we provide results on the performance when we dynamically use k such that the top-p percent of the true tokens from full-attention are retrieved. We observe here that a moderate level of p is sufficient for good performance, and that the results show a clear degradation once recall becomes low, while performance changes only slightly between moderate and high recall. We therefore interpret this as a saturation effect: recovering a moderate fraction of the most relevant keys provides most of the benefit, with diminishing returns as recall approaches 100%.

C.10 EMPIRICAL EVIDENCE: TRAINING VS. NON-TRAINING

To quantify the impact of joint optimization, we compare the recall of our trained HIROUTER against a baseline where the router is applied to untrained representations (`num_levels=4`, `num_buckets=4`, `seq_length=4096`). Table 28 reports the Recall@128 under different beam widths.

Conclusion. The results demonstrate a **2×–3× improvement** in recall when the router and representations are co-trained. This confirms that the performance gains arise from the *synergy* between the hierarchical routing mechanism and our dual-entropy loss design, rather than from the router architecture alone.

C.11 HARD-FORWARD BALANCE AND THE KL-UNIFORM COMPARISON

We use the Gumbel–Softmax straight-through estimator to combine discrete forward assignments with relaxed backward derivatives. For a hard-forward assignment, the balance functional is evaluated on the observed bucket proportions $\widehat{\bar{p}}$, whereas a soft-marginal implementation evaluates $\bar{p}$. These quantities need not coincide, as discussed in Appendix A.2.

For either specified probability vector v, the identity $\mathrm{KL}(v\|u) = \log C + \mathcal{L}_{\text{bal}}(v)$ holds. Therefore, Table 22 should be interpreted as a comparison between the reported implementation configurations rather than as evidence that KL-based objectives cannot balance buckets in general. The observed difference can arise from the assignment estimator, marginal on which the loss is evaluated, temperature, and reduction used in each configuration.

Table 22: Top-128 recall of $\mathcal{L}_{\text{bal}}$ and KL-uniform regularization across hierarchy levels.

Level	2	3	4	5
$\mathcal{L}_{\text{bal}}$	33.8	39.4	39.1	37.6
KL-Unif	12.4	14.7	15.2	14.2

Table 23: Ablation on the weighting of the dual-entropy regularization loss. We report perplexity (Wiki., LMB.) and accuracy (%) across commonsense reasoning benchmarks.

Reg. Weight	Wiki. ppl ↓	LMB. ppl ↓	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc_n ↑	SIQA acc ↑	BoolQ acc ↑	Avg.
0.10	31.53	44.74	32.58	65.78	38.02	50.36	58.00	27.22	38.13	60.64	46.34
0.05	31.09	42.94	33.09	66.81	38.03	50.75	59.47	28.50	38.08	61.31	47.01
0.01	30.30	39.99	34.47	66.76	38.36	50.59	58.80	28.24	39.41	58.99	46.95
0.00	30.04	43.22	33.08	67.00	38.25	51.18	59.73	28.21	38.29	57.02	46.59

Table 24: Downstream performance under controlled reference top-k recall. We report perplexity (Wiki., LMB.) and accuracy (%) across commonsense reasoning benchmarks.

Recall Level	Wiki. ppl ↓	LMB. ppl ↓	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc_n ↑	SIQA acc ↑	BoolQ acc ↑	Avg.
100%	30.85	44.02	39.32	66.37	34.42	51.50	59.56	26.45	39.29	59.10	47.00
75%	30.90	44.04	39.28	66.33	34.38	51.46	59.52	26.41	39.25	59.06	46.96
50%	30.95	44.08	39.24	66.29	34.34	51.42	59.48	26.37	39.21	59.02	46.92
25%	31.80	48.00	33.12	65.20	33.50	49.88	58.50	25.50	38.50	58.20	45.30
0%	32.75	53.45	30.08	64.21	33.00	48.50	57.50	24.50	37.50	57.00	44.04

Table 25: Comparison between HiRouter and classical ANN methods.

Metric	HNSW	HiRouter	K-Means	LSH
Time (s)	0.635	**0.072**	9.334	0.438
Success Rate	3.16%	**8.67%**	0.02%	0.65%

C.12 Comparison with ScaNN-PQ: Training Overhead and Recall–Speed Tradeoff

We train HiRouter jointly with the base model, so it introduces no separate training cost, and its parameter overhead is negligible. Routing is realized as dot-product operations with centroids (i.e. linear transforms), which allows rapid convergence alongside the main model. To validate this in practice, we benchmark both training and inference time of HiRouter against ScaNN-PQ (Guo et al., 2020) under a batch size of 128 (equivalent to top-k attention over 8 heads × 16 sequences).

Even for inference alone, HiRouter is substantially faster than ScaNN-PQ, and its training overhead remains modest.

While ScaNN-PQ attains higher recall at longer lengths, its large runtime cost makes it less practical for efficient sparse top-k attention. HiRouter offers a better balance of recall and efficiency, making it more suitable in real systems.

C.13 Extended NIAH Results

Table 30 provides some additional results for the NIAH task at even longer lengths.

C.14 Comparison Against Inference-Time KV-Cache Methods

We further compare our method against methods that use a base Transformer and use inference-time KV-cache management meant for improved transformer performance. In particular, we test against HeadKV (Fu et al., 2025), SqueezeAttention (Wang et al., 2025) and ClusterKV (Liu et al., 2025).

Table 26: WikiText-103 perplexity under controlled reference top-k recall.

Recall	0%	25%	50%	75%	100%
PPL ↓	23.6	21.2	18.8	18.5	18.4

Table 27: Accuracy on tasks from LongBench (Bai et al., 2024) given a minimum recall limit.

Model	Single-Doc QA			Multi-Doc QA			Summarization			Few-shot			Code		Avg.
	NQA	QQA	MFQ	HQA	2WM	Mus	GvR	QMS	MNs	TRC	TQA	SSM	LCC	RBP	
100%	1.69	3.54	11.25	4.54	6.82	2.54	8.80	16.21	10.66	21.17	11.36	4.48	5.25	11.21	8.54
75%	1.71	3.49	11.12	4.58	6.76	2.49	8.86	16.08	10.74	20.98	11.41	4.42	5.31	11.09	8.53
50%	1.66	3.57	11.03	4.47	6.88	2.51	8.74	16.19	10.58	20.81	11.22	4.55	5.18	11.27	8.48
25%	1.54	3.28	10.41	4.11	6.23	2.31	8.35	15.12	9.96	19.02	10.53	4.16	4.93	10.44	7.89
0%	1.27	2.94	9.36	3.62	5.51	1.97	7.71	13.84	9.04	16.88	9.27	3.81	4.51	9.33	7.15

Table 28: Recall@128 comparison between trained and non-trained HiRouter.

Beam Width (M)	**8**	**12**	**16**	**32**	**64**
Trained / Recall@128	25.3%	34.0%	39.4%	62.1%	83.0%
Without Training / Recall@128	8.8%	13.1%	15.6%	27.0%	46.1%

	0.5K	4K	8K	16K
HiRouter (train)	1.14 s	4.04 s	6.15 s	10.80 s
HiRouter (infer)	0.32 s	0.36 s	0.34 s	0.33 s
ScaNN-PQ (infer)	9.34 s	17.06 s	25.3 s	40.3 s

Table 29: Top-128 recall across varying sequence lengths.

	0.5K	4K	8K	16K
HiRouter (Recall@128)	60.1%	25.3%	17.9%	13.8%
ScaNN-PQ (Recall@128)	50.0%	42.9%	34.9%	26.5%

Table 30: Performance on Passkey-Retrieval tasks at longer lengths.

Model	S-NIAH-1 (pass-key retrieval)			S-NIAH-2 (number in haystack)			S-NIAH-3 (uuid in haystack)			Avg.
	16K	32K	64K	16K	32K	64K	16K	32K	64K	
Transformer	0.4	0.0	0.0	0.2	0.0	0.0	0.8	0.0	0.0	0.2
NSA	8.4	4.0	1.5	4.1	1.5	0.5	0.8	0.2	0.0	2.3
HiRouter	32.0	15.0	9.0	21.51	0.2	6.0	1.8	1.0	0.3	10.7

Our results in Table 31 demonstrate that our method outperforms such cache management techniques, confirming the benefits of this specific new architectural paradigm.

C.15 Continual Tuning Ablation

We further attempt to observe whether the proposed method able to fine-tune a full-attention Transformer into a sparse Transformer.

We use a `Llama3-8B` checkpoint on the same dataset (10B split of `Fineweb-Edu`); we explore the option of tuning just the routing structure as well as tuning the entire model in tandem. We also compare with doing an additional fine-tuning with the entire `Llama3-8B` model (**`Full`**) on the 10B split for fair comparison. We present the performance on `LongBench` below.

Table 31: Performance comparison on language modeling and zero-shot common-sense reasoning.

Model	Wiki. ppl ↓	LMB. ppl ↓	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc_n ↑	SIQA acc ↑	BoolQ acc ↑	Avg.
Transformer	30.21	43.44	32.76	**67.68**	39.20	**53.51**	57.58	27.47	37.97	61.19	**47.17**
+ HeadKV	34.01	43.67	27.29	67.43	37.86	51.62	58.34	24.23	37.82	61.77	45.80
+ SqueezeAttn	34.00	43.27	30.78	66.21	38.99	51.30	59.30	27.56	37.46	61.53	46.64
+ ClusterKV	33.01	45.67	30.51	65.41	36.75	49.82	58.92	26.51	37.17	60.12	45.65
RetNet	36.47	63.64	26.33	65.18	35.61	50.59	56.82	27.13	37.87	60.95	45.06
Mamba	32.63	61.68	27.79	65.61	38.47	51.22	57.41	26.62	38.64	**61.65**	45.93
Mamba2	30.15	49.83	28.70	66.81	38.94	51.38	**60.69**	28.75	37.67	59.69	46.83
HGRN2	30.07	**40.29**	31.32	66.54	**39.68**	50.12	59.30	27.05	38.84	58.72	46.45
GLA	31.50	51.56	29.01	66.49	38.60	50.12	57.83	26.11	39.25	57.77	45.40
DeltaNet	**28.82**	45.06	30.47	67.19	39.51	52.80	58.80	**29.10**	38.18	58.26	46.79
GSA	30.78	48.74	29.75	66.70	39.01	52.49	59.26	27.65	38.49	60.61	46.50
NSA	31.54	43.0	**36.48**	66.27	35.37	50.19	59.27	27.33	**39.41**	59.66	46.75
HIROUTER	31.09	42.94	33.09	66.81	38.03	50.75	59.47	28.50	38.08	61.31	47.01

Table 32: Accuracy on tasks from LongBench after continual tuning.

Model	Single-Doc QA			Multi-Doc QA			Summarization			Few-shot			Code		Avg.
	NQA	QQA	MFQ	HQA	2WM	Mus	GvR	QMS	MNs	TRC	TQA	SSM	LCC	RBP	
Llama3-8B	6.83	13.87	28.38	10.79	11.78	7.66	29.82	25.11	2.54	73.00	91.13	46.92	23.60	15.07	27.61
Router Only	8.06	13.37	27.82	10.72	11.69	7.53	30.48	25.36	2.51	72.50	91.47	47.16	23.86	16.10	27.76
Full	7.90	1.51	10.85	10.25	6.12	7.33	15.44	20.88	0.24	27.00	69.44	29.21	7.94	12.55	16.19

We can observe some improvements on the average performance by tuning the router during this phase; on the other hand, tuning the entire model leads to degradation in performance due to the large quality discrepancy between the original checkpoint and the tuning dataset.

D TRITON KERNELS

Algorithm 2 demonstrates how we perform our sparse top-k attention computations within our custom Triton kernels. Algorithm 4 demonstrates how we implement the hierarchical beam search within our custom Triton kernels.

Algorithm 2 Forward Pass for the HIROUTER Sparse Attention Kernel

Input: Query, key, and value tensors $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{BH \times T \times d}$;
Query/key index tensors $\texttt{q_idx}, \texttt{k_idx} \in \mathbb{Z}^{BH \times T \times S}$;
Block size *BS*, candidate padding *CAND_PAD*, and number of samples per query S.
Notation: B: batch size; H: number of attention heads; $BH = B \times H$: total head instances; G: number of queries grouped per head; B_K: block size in the key dimension; B_V: block size in the value dimension.
Output: Attention result $\mathbf{O} \in \mathbb{R}^{BH \times T \times d}$, and log-sum-exp buffer $\mathbf{LSE} \in \mathbb{R}^{BH \times T}$.
Initialize a 3-D launch grid over $(t, v, bh) \leftarrow (0..T{-}1,\ 0..d/B_V{-}1,\ 0..BH{-}1)$.
for each block (t, v, bh) in the grid **do**
  $b, h \leftarrow \lfloor bh/H \rfloor,\ bh \bmod H$
  Initialize accumulators: $\ell \leftarrow -\infty_{\,G},\ s \leftarrow 0_{\,G},\ w \leftarrow 0_{\,G \times B_V}$
  Determine padded group size: $G_{\text{pad}} \leftarrow \max(G,\ CAND_PAD)$
  Load query block: $q \leftarrow \mathbf{Q}[bh,\ t,\ 0{:}B_K] \in \mathbb{R}^{G_{\text{pad}} \times B_K}$
  Scale queries: $q \leftarrow q/\sqrt{d}$
  for each sampled index $i = 0{:}S{-}1$ **do**
    $\texttt{s_idx} \leftarrow \texttt{q_idx}[bh, t, i] \times BS$
    Load corresponding key/value blocks $\mathbf{K}_i, \mathbf{V}_i$ via $\texttt{k_idx}$
    Compute attention scores: $\text{score} \leftarrow q\,\mathbf{K}_i^T$
    Apply causal mask if $\texttt{s_idx} > t$: $\text{score} \leftarrow -\infty$
    Update statistics: $(\ell, s, w) \leftarrow \texttt{UPDATE_STATS}(\ell, s, w, \text{score}, \mathbf{V}_i)$
  end for
  Normalize outputs: $\mathbf{O}[bh, t, v] \leftarrow w/s, \quad \mathbf{LSE}[bh, t] \leftarrow \ell + \log s$
end for

Algorithm 3 UPDATE_STATS: Numerically Stable Softmax Statistics Update

Input: Given running softmax statistics $\ell \in \mathbb{R}^B$ (max logits), $s \in \mathbb{R}^B$ (sum of exps), $w \in \mathbb{R}^{B \times d}$ (weighted sum), a new block of logits $\text{scores} \in \mathbb{R}^{B \times N}$, and corresponding values $\mathbf{V} \in \mathbb{R}^{B \times N \times d}$
$m \leftarrow \max_j \text{scores}_{\cdot,j} \quad \in \mathbb{R}^B$
$\ell_{\text{new}} \leftarrow \max(\ell,\ m)$
$\text{scale} \leftarrow \exp(\ell - \ell_{\text{new}})$
$s \leftarrow s \times \text{scale}$
$w \leftarrow w \times \text{scale}$
$\Delta \leftarrow \exp(\text{scores} - \ell_{\text{new}}[:, \texttt{None}]) \quad \in \mathbb{R}^{B \times N}$
$s \leftarrow s + \sum_{j=1}^{N} \Delta_{\cdot,j}$
$w \leftarrow w + \sum_{j=1}^{N} \Delta_{\cdot,j}\, \mathbf{V}_{\cdot,j,:}$
Return: $\ell_{\text{new}},\ s,\ w$

Algorithm 4 Hierarchical Beam Search Kernel

Input: $Q \in \mathbb{R}^{Bs \times D}$, Offsets $\in \mathbb{Z}^{L+1}$, Counts $\in \mathbb{Z}^{L}$, beam, k, BLOCK_TOKENS, L, C, D
$b \leftarrow \text{program_id}(0)$
ids $\leftarrow b \cdot$ BLOCK_TOKENS $+ [0 :$ BLOCK_TOKENS$-1]$
valid $\leftarrow$ ids $< B_S$
$Q_{\text{tile}} \leftarrow \text{load}(q_{\text{ptr}},$ ids$)$
initialize beam_probs $\leftarrow [1, 0, \ldots, 0]$, beam_parents $\leftarrow [0, \ldots, 0]$
for $\ell = 0 \ldots L-1$ **do**
  $P_\ell \leftarrow \text{counts}[\ell]$, off $\leftarrow \text{offsets}[\ell]$
  W $\leftarrow \text{gather}(\text{route}_{\text{ptr}},$ beam_parents, off$)$
  scores $\leftarrow \exp\left(Q_{\text{tile}} \cdot W^{\top}\right)$
  normalize and weight by beam_probs
  reshape to [BLOCK_TOKENS, beam $\cdot C$]
  (sorted_s, sorted_idxs) $\leftarrow$ ARGSORT(scores, arange)
  take top-beam (top-M) entries from sorted_s, sorted_idxs
  update beam_probs, beam_parents
end for
store final beam_parents into output buffer

E TIME COMPLEXITY ANALYSIS

Let T be the sequence length, D the per-head feature dimension, L the number of routing levels, C the branching factor, and M the query beam width. For query q, let S_q be the set of candidate keys contained in its selected leaf buckets, and define the average candidate size $\bar{S} = T^{-1} \sum_q |S_q|$.

Greedy routing of all keys requires $O(TLCD)$ work. The query beam expands at most MC children per level, giving $O(TLMCD)$ routing work for all queries. The sparse-attention computation itself is proportional to the number of query–key pairs that are actually scored:

$$O\left(D \sum_{q=1}^{T} |S_q| \right) = O(T\bar{S}D).$$

The total prefill cost is therefore

$$O\big(TLCD + TLMCD + T\bar{S}D\big)$$

plus the cost of sorting/gathering the routed keys.

This expression makes the role of bucket selectivity explicit. When the candidate size $\bar{S}$ is bounded independently of T, the attention work is linear in sequence length. If the number of leaves is fixed while their occupancy grows with T, $\bar{S}$ can also grow with T; in that regime the method reduces the number of scored query–key pairs but does not change the worst-case quadratic order. The measured scaling of the implemented kernels is reported below.

E.1 CACHE-AWARE AUTOREGRESSIVE DECODING

During autoregressive decoding, the centroid banks are fixed model parameters. A newly generated key is routed when it is created and stored in its leaf bucket; previously cached keys do not need to be routed again.

Lemma E.1 (Cached keys do not require re-routing)**.** *Let z_t be the key representation produced at decoding step t, and let $a^l_{K,t}$ denote its greedy bucket index at level l,*

$$a^l_{K,t} = C a^{l-1}_{K,t} + \arg\max_j p^l_{K,t,j}.$$

Once z_t and the centroid banks are fixed, the leaf assignment $a^L_{K,t}$ is independent of all future queries and generated tokens. Therefore the assignment can be computed once when the key is created and reused at every later decoding step.

Proof. At each level, the next bucket is determined only by the key representation, the centroid bank at that level, and the bucket chosen at the preceding level. These quantities do not change when later queries are generated. Applying this argument recursively from the root to level L fixes the final leaf assignment, so previously cached keys do not need to be re-routed. □

Proposition E.2 (Per-token decoding cost)**.** *Let M be the query beam width, C the branching factor, L the hierarchy depth, and D the head dimension. Let S_q be the routed candidate keys for the current query and let W be the local-window size. One decoding step requires*

$$O(LMCD + (|S_q| + W)D)$$

work per head. If the total routed cache has fixed capacity $K_{\max}$, then $|S_q| \le K_{\max}$ and the bound becomes

$$O((LMC + K_{\max} + W)D)\,,$$

which is independent of the number of earlier generated tokens.

Proof. The newly generated key follows one greedy route, requiring C child scores at each of L levels. The query beam expands at most MC children per level. The attention kernel then scores only the selected routed keys and the local window. By Lemma E.1, previously cached keys contribute no additional routing work. A FIFO insertion or replacement changes only the bucket receiving the new key. □

The proposition describes algorithmic work, not wall-clock latency. Kernel utilization, memory traffic, padding, and cache layout are measured separately in the system experiments.

E.2 Sorting Latency

Table 33: Routing Overhead

Sequence Length	Router Time	Total Time	Router %
4K	0.10ms	0.26ms	39.0%
16K	0.12ms	1.68ms	6.9%
64K	0.46ms	8.11ms	5.6%

E.3 Beam Search Latency

Table 34: Beam Searching Overhead

Sequence Length	Router Time	Search Time	Router %
4K	0.10ms	0.022ms	21.8%
16K	0.12ms	0.048ms	40.0%
64K	0.46ms	0.240ms	52.2%

E.4 Sorting Overhead

Table 35: Sorting Overhead

Sequence Length	Sort Time	Sort %	Gather Time	Sorting + Gathering %
4K	0.018ms	6.9%	0.04ms	22.2%
16K	0.027ms	1.6%	0.08ms	6.4%
64K	0.057ms	0.7%	0.19ms	3.0%

E.5 Memory Bandwidth

Table 36: Memory Bandwidth using different model configurations (following `Llama-3`).

Model	Phase	Bytes (R+W)	Eff. BW	% of Peak
1B	FA	64.0 MiB	200 GB/s	6.8%
	Routing	46.1 MiB	194 GB/s	6.6%
7B	FA	256.0 MiB	144 GB/s	4.9%
	Routing	164.2 MiB	177 GB/s	6.1%