MoRE: Scaling Mixture of Experts with Hardware-Aware Low-Rank Routing

Honam Wong [1] **Surbhi Goel** [1] **Enric Boix-Adsera** [2]

Abstract

Mixture-of-Experts (MoE) layers are central to frontier language models, with recent architectures pushing toward larger total expert counts and finer-grained experts. In this regime, the standard linear router becomes a bottleneck: with M experts and hidden dimension h, its per-token cost $\Theta(Mh)$ grows to dominate the MoE layer once M is large. We introduce MoRE (Mixture of Rank-reduced-routed Experts), which factorizes the $h \times M$ routing matrix through a rank-r bottleneck, reducing routing cost to $O((h+M)r)$. We prove that rank logarithmic in M is sufficient to preserve the memorization capabilities. At matched active FLOPs, the factorization unlocks a $\Theta(h/r)$ factor more experts. However, realizing this gain in wall-clock requires more than the FLOP reduction, so we design a fused Triton kernel that avoids expensive memory operations on HBM. Empirically, MoRE's additional capacity translates into improved memorization on a synthetic phonebook task and stronger performance on knowledge-intensive Q&A after pretraining, while preserving reasoning ability. The code is available at https://github.com/Matheart/more_code.

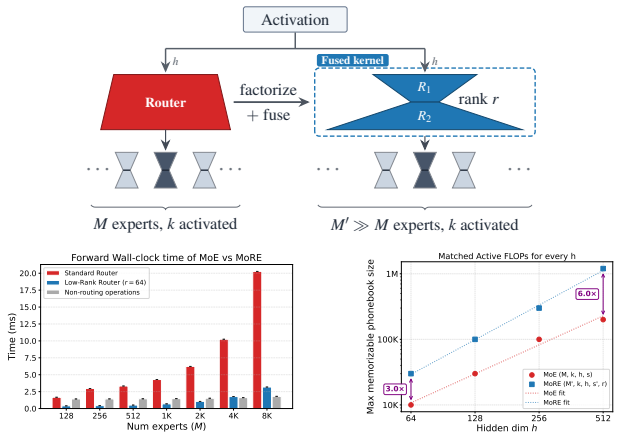

Figure 1. **MoRE unlocks more scaling with same active FLOPs. (Bottom Left)** The standard router's wall-clock cost grows linearly with the number of experts M and dominates the MoE layer at large M; MoRE's low-rank router stays nearly flat across the full sweep, $h = 512$. See Appendix E.3 for the end-to-end forward-pass measurement. **(Top)** Replacing the full-rank $h \times M$ router with two low-rank factors R_1, R_2 inside a fused kernel cuts routing cost from $O(hM)$ to $O(r(h+M'))$, enabling $M' \gg M$ experts at matched active FLOPs. **(Bottom Right)** This extra capacity translates into substantially better memorization on the synthetic phonebook task at matched FLOPs with different h; the gap widens with h, in agreement with the predicted $\Theta(h/r)$ scaling in Section 4.1.

1. Introduction

Mixture-of-experts (MoE) is now the dominant architectural paradigm for frontier language models (DeepSeek-AI, 2024a;b; Jiang et al., 2024; xAI, 2024; Yan et al., 2026). In these architectures, the standard dense MLP is replaced by a pool of independent experts and a router layer that activates only a small subset of experts per token. Recent advances in model scaling rely heavily on increasing the number of experts in MoE layers. This approach expands the total parameter count, allowing models to benefit from neural scaling laws without proportionally increasing active compute (Kaplan et al., 2020; Abnar et al., 2025). A major functional advantage of this sparsity is increased model capacity: at a fixed active compute budget, MoE layers achieve significantly better *memorization and factual recall* compared to dense MLPs (Jelassi et al., 2025). Recent frontier MoE models are increasingly operating in the *fine-grained regime* (Appendix A), in which the layer is divided into a larger number of smaller experts, each with intermediate width below the hidden dimension.

However, further scaling the number of experts hits a bottleneck: routing costs begin to dominate the expert cost as the expert pool grows. This is driven by two distinct factors: the increased active compute required to calculate the gating function, and the heavy memory bandwidth cost of reading

[1]Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA [2]Department of Statistics and Data Science, University of Pennsylvania, Philadelphia, PA, USA. Correspondence to: Honam Wong <hnwong@seas.upenn.edu>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

/ writing from logits. These combined costs can offset the efficiency gains of sparsity in the many-expert regime (see Figure 1) This motivates our main question:

Can we scale the number of experts in MoEs without paying the cost of the standard router?

1.1. Our contributions

Low-rank router design In this work, we propose MoRE layers (Mixture of Rank-reduced-routed Experts) – an architectural design that allows for scaling to more experts with an asymptotically lower computational cost. **MoRE replaces the full-rank linear router with a low-rank router**. This factorization reduces the number of parameters in the gating function and the FLOPs required for a forward pass, allowing scaling to more experts. Our router design is inspired by the low-rank parameterization of key-query (KQ) weights in recent attention architectures (DeepSeek-AI, 2024a) as well as the low-rank design of LoRA adapters (Hu et al., 2021).

Theoretical guarantees One of the primary reasons to scale parameters with MoE layers is to increase memorization capacity. Any modification of these layers must maintain this important ability. As our theoretical contribution, in Section 3, we verify that the **low-rank routing in MoRE layers preserves memorization capabilities**. We prove that balanced expert assignment among M experts is possible even with a low logarithmic rank $r = \Theta(\log M)$. Such a balanced assignment ensures that the memorization capacity of each expert can be used fully.

Scaling experts efficiently In Section 4 we translate the FLOP saving of MoRE into a usable expert-count gain. At matched active FLOPs, a MoRE layer with rank-r routing and embedding dimension h supports a $\Theta(h/r)$ factor more experts than the standard router (Section 4.1). However, realizing this gain in wall-clock time requires more than a FLOP reduction: a naive two-kernel low-rank router implementation that materializes the logit tensor in High Bandwidth Memory (HBM) will be inefficient because it is memory-bound. Instead, **we design a fused Triton kernel** (Section 4.2) that keeps logits in on-chip SRAM and emits only the sparse top-k outputs, recovering the asymptotic $\Theta(h/r)$ gain at production-scale M.

Experimental validation Finally, we validate our architecture through pretraining experiments on both synthetic phonebook memorization tasks and real-world Internet data. We show that **MoRE models consistently match or improve performance in knowledge-intensive** tasks such as TriviaQA (Joshi et al., 2017) and Natural Questions (Kwiatkowski et al., 2019), compared to standard MoEs with an equal number of active FLOPs and comparable wall-clock time per step. Additionally we find low-rank routing does not hurt performance on reasoning tasks.

1.2. Related Work

Routing in MoE layers. The standard router used in modern MoE is a learned linear router that scores all M experts per token (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022). Several alternatives have been studied. *Hash routing* fixes the token-to-expert assignment to a deterministic hash, eliminating router parameters entirely (Roller et al., 2021). However, Clark et al. (2022); Dikkala et al. (2023) show that fully deterministic assignment underperforms learned routing at scale, motivating the search for *parameter-efficient* learned routers rather than discarding learning. Beyond linear routers, He (2024) introduces *PEER*, a top-1 product-key retrieval layer that scales to a pool of more than a million tiny experts and has asymptotic cost $O(h\sqrt{M})$. PEER targets a different portion of the design space, and in fact it could be combined MoRE's low-rank router idea to obtain a best-of-both worlds $O(r(\sqrt{M} + h))$-router cost. We leave the investigation of a PEER-MoRE hybrid and making it practical to future work. Boix-Adsera (2025) suggests low-rank routing but only evaluates it in a limited distillation context at small scale and leaves its study as a direction of future work. Chi et al. (2022) (X-MoE) reveals representation collapse in the small-expert regime $M \ll h$, and introduces low-rank routing to remedy this. MoRE addresses the opposite regime $M \gg h$ and low-rank routing is introduced for resolving the efficiency issue instead of representation collapse. More recently, L2R (Yang et al., 2026) proposes low-rank routing with Lipschitz-controlled scoring as a training-stability mechanism rather than as a means to efficiently scale to more experts. All three leave open both (i) the expressivity question we resolve in Section 3 for the large-M regime, and (ii) the efficient implementation we address in Section 4, without which the FLOP reduction does not translate to a wall-clock speedup at large M.

Fine-grained MoE and scaling. Recent open-weight architectures are increasingly designed in the *fine-grained* experts regime, where there is small expert intermediate dimension s paired with larger total expert count M. Krajewski et al. (2024) establish a granularity-aware scaling law showing that fine-grained MoE outperforms coarse-grained MoE at a fixed training budget, and Dai et al. (2024) argue that this design admits sharper expert specialization. Nevertheless, these works do not address the routing bottleneck that emerges as M grows. MoRE seeks these routing costs.

Knowledge storage in language models. Two threads study how language models store factual knowledge. The first measures memorization in standard transformer and MoE architectures: Geva et al. (2021) interprets FFN layers as key-value memories, Allen-Zhu & Li (2024) establish knowledge capacity scaling laws for dense LMs, and Jelassi

et al. (2025) show that for MoE, memorization capacity scales with *total* parameters rather than active parameters, directly motivating designs with large M. A second thread replaces the FFN with a learnable key-value lookup: Memory Layers at Scale (Berges et al., 2024) scales product-key retrieval to billions of stored embeddings, Ultra-Sparse Memory (Huang et al., 2025) pushes to ultra-sparse activation patterns, and Cheng et al. (2026) extends retrieval-style storage to conditional access. MoRE is complementary to this thread: it preserves the MLP-expert structure of MoE, and inherits the memorization analysis of Jelassi et al. (2025) while reducing only the routing cost.

Efficient MoE systems. Existing fused MoE kernels target *expert dispatch*. ScatterMoE (Tan et al., 2024) and MegaBlocks (Gale et al., 2023) fuse scatter-gather with the per-expert grouped matmul, eliminating $O(Nkh)$ memory traffic from token routing, where N is the number of tokens, and k is the number of active experts. More recent kernels such as MoMoE (Costin et al., 2025) and SonicMoE (Guo et al., 2025) extend this with IO-aware optimizations on the dispatch path, and Megatron Core MoE (Yan et al., 2026) provides multi-GPU expert-parallel training infrastructure for frontier-scale runs. However, prior work leaves open the question of how to implement the router efficiently in the many-expert regime. MoRE introduces a fused *router* kernel (Section 4.2) that handles both low-rank matmuls and the top-k selection in a single pass, without materializing the logit tensor.

2. Preliminaries

Notation. We denote hidden dimension as h, expert intermediate dimension as s, experts per layer as M, active experts per token as k (with $k < M$), low-rank router rank as r, number of transformer layers as L. We denote batch size as B and sequence length as T, so number of tokens per iteration is $N = BT$. We write $[m] = \{1, \ldots, m\}$, use T for total parameter count, and τ for per-token active FLOPs.

Mixture-of-experts layer. An MoE layer has M experts $\{f_i\}_{i=1}^{M}$ and a router $g : \mathbb{R}^h \to \mathbb{R}^M$ that produces k-sparse routing weights. Each expert is usually a SwiGLU MLP (Shazeer, 2020)[1] with three weight matrices $W_i^{\text{gate}}, W_i^{\text{up}} \in \mathbb{R}^{s\times h}$ and $W_i^{\text{down}} \in \mathbb{R}^{h\times s}$:

$$f_i(x) = W_i^{\text{down}}\big(\sigma(W_i^{\text{gate}}x) \odot W_i^{\text{up}}x\big),$$
$$f_{\text{MoE}}(x) = \sum_{i=1}^{M} g_i(x) f_i(x),$$

with σ the SiLU activation and $\odot$ elementwise product. Each expert thus has $3hs$ parameters, hence the factor of 3 in the total below. The **standard router** (Shazeer et al., 2017; Fedus et al., 2022) is parametrized by a linear projection $R \in \mathbb{R}^{M\times h}$ with rows $r_i \in \mathbb{R}^h$, and applies softmax over the k largest scores: $g(x; R) = \mathrm{softmax}\big(\mathrm{TopK}(Rx)\big)$, where $\mathrm{TopK}(x)$ is the top-k values of $x \in \mathbb{R}^M$. We write its configuration as $\mathrm{MoE}(M, k, h, s)$.

MoRE layer. In MoRE layer, we replace R with R_1R_2 where $R_1 \in \mathbb{R}^{M\times r}$ and $R_2 \in \mathbb{R}^{r\times h}$, scoring through an r-dimensional bottleneck: $g(x; R_1, R_2) = \mathrm{softmax}\big(\mathrm{TopK}(R_1R_2x)\big)$. Routing parameters drop from Mh to $(M + h)r$. We write the low-rank configuration as $\mathrm{MoRE}(M, k, h, s, r)$.

Total parameters and active cost. The total MLP parameter count per MoE/MoRE layer is $3Mhs$, and the per-token active FLOPs per layer are

$$\tau = \begin{cases} C_e\, khs + C_r\, Mh & \text{(standard router)}, \\ C_e\, khs + C_r\, (M + h)\, r & \text{(low-rank router)}, \end{cases} \quad (1)$$

where C_e, C_r collect per-operation FLOP factors. Following Krajewski et al. (2024, App. E), full forward + backward training gives $(C_e, C_r) = (18, 14)$ and forward-only prefill gives $(C_e, C_r) = (6, 6)$, so $C_e \approx C_r$ in either regime (full per-component derivation in Appendix B.3). We set $C_e = C_r = 16$ in Section 4.1 for convenient computation .

3. Low-rank routing preserves memorization capabilities

MoRE replaces the standard router with a rank-r factorization, which raises two related theoretical questions. First, does the rank bottleneck prevent the router from expressing the needed expert assignments? Second, when there are many inputs to memorize, can the low-rank router still distribute them across experts in the way standard MoE could? We answer both questions positively. We first prove bounds for top-k routing, showing that logarithmic rank in the number of assignments suffices and is necessary up to precision factors. We then specialize to memorization and show that, in the Gaussian model of Jelassi et al. (2025), a rank-$\log_2 M$ router achieves the same load-balancing construction over M experts as standard MoE.

3.1. Routing expressivity

This subsection asks which expert subsets a low-rank router can realize at all. Formally, we fix a desired family of top-k subsets and ask whether a *single* router can select each subset for some input.

Setup. The rank-r router has a simple geometric form: the rows of R_1 define the M expert vectors $u_i \in \mathbb{R}^r$, and each input reaches the router only through its projected

[1] SwiGLU is the popular activation function choice in modern transformer-based LLMs, including LLaMA (Meta AI, 2025), Qwen (Yang et al., 2025), DeepSeek (DeepSeek-AI, 2024b), and Mixtral (Jiang et al., 2024).

representation $z(x) := R_2 x \in \mathbb{R}^r$. The router then selects the k experts whose vectors have the largest inner products with $z(x)$. Throughout this subsection, we normalize the scale by assuming $\|u_i\|_2 \leq 1$ for every expert vector and $\|z(x)\|_2 \leq 1$ for every input x under consideration.

Margin and precision. Because top-k is implemented at finite precision, selecting a subset robustly requires a positive gap between selected and unselected scores. We formalize this as follows.

Definition 1. *Fix a rank-r router $R = R_1 R_2$. A subset $S \in \binom{[M]}{k}$ is* realized at input x with margin $\eta > 0$ *if $S = \mathrm{TopK}(Rx)$ and*

$$\min_{i \in S} (Rx)_i - \max_{j \notin S} (Rx)_j \geq \eta.$$

Moreover, we say a family $\mathcal{F} \subseteq \binom{[M]}{k}$ is realizable at margin η *by R if every $S \in \mathcal{F}$ has some input representation x_S that realizes it at margin at least η.*

Note that for a p-bit implementation, a margin η is resolvable when $p = \Omega(\log(1/\eta))$, since that is equivalent to $\eta = \Omega(2^{-p})$.

Main Results. We first present an upper bound construction based on the Johnson–Lindenstrauss lemma (Johnson & Lindenstrauss, 1984):

Theorem 1 (Upper bound on required rank)**.** *There exists a universal constant $C > 0$ such that the following holds. For $1 \leq k \leq M - 1$, any top-k assignments $\mathcal{F} \subseteq \binom{[M]}{k}$, and any rank*

$$r \geq Ck \log(M + |\mathcal{F}|), \qquad h \geq r,$$

admits a rank-r router $R = R_1 R_2$ with $\|u_i\|_2 \leq 1$ that realizes every $S \in \mathcal{F}$ at margin $\Omega(1/\sqrt{k})$, using input representations $\{x_S\}_{S \in \mathcal{F}}$ with $\|R_2 x_S\|_2 \leq 1$.

Proof idea. First let us consider the M expert vectors as the standard basis directions $e_1, \ldots, e_M \in \mathbb{R}^M$ in the M-dimensional space. Then the representation $z_S \propto \frac{\mathbf{1}_S}{k} - \frac{\mathbf{1}_{[M] \setminus S}}{M-k}$ routes to exactly the experts in S: it has positive coordinates on S and negative coordinates outside S. The resulting margin is $\Theta(1/\sqrt{k})$. Now using Johnson–Lindenstrauss, we can project these $M + |\mathcal{F}|$ vectors $\{e_i\}_{i=1}^M \cup \{z_S : S \in \mathcal{F}\}$ to a r-dimensional space while preserving inner products up to additive error ε with $r = \Theta(\log(M + |\mathcal{F}|)/\varepsilon^2)$, so setting $\varepsilon = \Theta(1/\sqrt{k})$ preserves a constant fraction of the original margin and gives $r = O(k \log(M + |\mathcal{F}|))$. We defer the full proof to the Appendix G.

Next, we use a volume packing argument to show that a lower bound on the rank is necessary.

Theorem 2 (Lower bound on required rank)**.** *Let $\mathcal{F} \subseteq \binom{[M]}{k}$. If a rank-$r$ router $R = R_1 R_2$ with $\|u_i\|_2 \leq 1$ and $\|R_2 x_S\|_2 \leq 1$ realizes every top-k assignment $S \in \mathcal{F}$ at margin η, then*

$$|\mathcal{F}| \leq (1 + 2/\eta)^r, \qquad r \geq \frac{\log |\mathcal{F}|}{\log(1 + 2/\eta)}.$$

Proof idea. If two different assignments $S, T \in \mathcal{F}$ are both realized with margin η, their projected representations must be separated. Indeed, choose $i \in S \setminus T$ and $j \in T \setminus S$. At x_S, expert i must beat expert j by at least η; at x_T, expert j must beat expert i by at least η. Combining these two inequalities and using $\|u_i - u_j\|_2 \leq 2$ gives $\|R_2 x_S - R_2 x_T\|_2 \geq \eta$. Thus realizing all assignments in $\mathcal{F}$ requires placing $|\mathcal{F}|$ points in the unit ball of $\mathbb{R}^r$ with pairwise distance at least η. A standard volume-packing argument then gives $|\mathcal{F}| \leq (1 + 2/\eta)^r$, which rearranges to the stated lower bound on r.

Finally, taking $\mathcal{F}$ to be the collection of all top-k subsets gives the following consequence:

Corollary 1 (Full expressivity)**.** *At precision $p = \Omega(\log k)$, rank $r = O(k^2 \log(M/k))$ suffices to realize every top-k assignment over M experts. Conversely, any rank-r router that realizes every top-k assignment at precision scale p must have $r = \Omega(k \log(M/k)/p)$.*

3.2. Memorization capacity

A core function of MoE layers is to expand memorization capacity at bounded active compute (Du et al., 2022; Geva et al., 2021; Jelassi et al., 2025), so the router should preserve factual recall. The expressivity results above say that low-rank routing can address the required expert configurations. Memorization additionally requires the router to use those configurations on many examples without overloading a small number of experts. In the pure capacity abstraction, one expert is enough to store one example, so the natural case is top-1 routing, where each example is sent to one of the M experts.

Jelassi et al. (2025) study this mechanism in a Gaussian memorization model. They consider n independent Gaussian examples and construct a top-1 router that sends examples approximately evenly across M experts. Each expert then memorizes the examples routed to it. We show that this same load-balancing construction can be implemented with a rank-$\log_2 M$ router.

Proposition 1 (Gaussian load balancing with logarithmic rank)**.** *Let $x_1, \ldots, x_n \sim \mathcal{N}(0, I_h)$ be independent, and suppose $M = 2^r$ with $h \geq r$. There exists a rank-r top-1 router with M experts such that each example is routed uniformly over the M experts. Consequently, if N_a is the number of examples routed to expert a, then with probability at least $1 - \delta$,*

$$\max_{a \in [M]} \left| N_a - \frac{n}{M} \right| \leq \sqrt{\frac{n}{2} \log \frac{2M}{\delta}}.$$

The proposition shows that low-rank routing achieves the same Gaussian load-balancing used in their memorization construction. The router distinguishes the M expert labels with rank $r = \log_2 M$. The dependence on the number of examples n enters through how many examples each expert must memorize. Thus the low-rank factorization preserves the memorization mechanism of standard MoE, while reducing routing complexity from $O(Mh)$ to $O((M+h)\log M)$.

Does the expressivity result still transfer in training? The results above show, by explicit construction, that rank logarithmic in M suffices to preserve memorization capabilities. However, it remains open whether a trained router realizes this in practice. To study this, we consider the synthetic memorization task — Phonebook (details in Section 5.1): we feed progressively larger subsets of the training data through a MoRE that is trained to memorize 1.2M entries, and record the number of activated experts per layer, we find that the low-rank router engages a majority of experts (Figure 2) after only 1000 examples. Interestingly, the number of activated experts decreases with layer depth, a pattern that holds in other configurations as well (see Appendix E.1). This suggests that adaptive rank and number of experts per layer may be useful, and we leave a full investigation to future work.

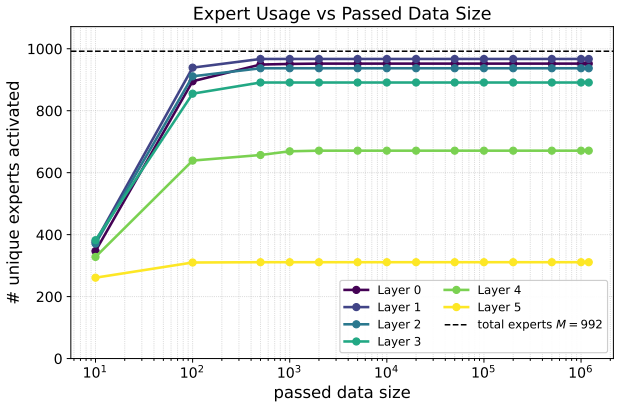

Figure 2. Expert usage vs. the number of passed examples for trained MoRE (h=64, s=64, M=992, k=4, L=6, r=32). Each curve gives the number of activated experts in different layers. After only 1000 examples, the early layers already activate the bulk of the M=992 experts, indicating that the low-rank router does not bottleneck access to the enlarged expert pool.

4. Scaling experts efficiently with fused implementation of MoRE

Section 3 establishes that logarithmic rank suffices to preserve memorization, so the low-rank factorization does not sacrifice routing expressivity. We can therefore ask: *at a fixed active-FLOP budget, how much additional capacity does the low-rank router unlock?* Below we sketch the calculation, showing a $\Theta(h/r)$ gain in the expert count. Then we show that realizing this gain in wall-clock time requires care. The large-M regime that the calculation pushes us into introduces a memory-bandwidth bottleneck on the routing logit tensor, which we resolve by implementing a new fused Triton (Tillet et al., 2019) kernel for the router.

4.1. Scaling under fixed active-FLOPs

Fix the active-FLOP budget τ, the hidden dimension h, and the active expert count k, so that all non-MoE components have identical parameter counts across the two settings. We compare $\mathrm{MoE}(M, k, h, s)$ and $\mathrm{MoRE}(M', k, h, s', r)$ on total MLP parameters $T = 3Mhs$. We sketch the argument below and refer the reader to Appendix F, Proposition 2 for the full calculation.

The active budget τ decomposes into an expert cost $C_e khs$ and a router cost. For MoE the router cost is $C_r Mh$, i.e. $C_r h$ FLOPs per expert. For MoRE the router cost is $C_r M'r + C_r hr$, where the per-expert term $C_r r$ is an h/r reduction over MoE, and $C_r hr$ is a fixed projection overhead independent of M'. Since $T = 3Mhs \propto Ms$ is a product, and the budget trades off linearly between scaling M (via the router) and scaling s (via the experts), it is maximized when the two halves of the budget are spent equally on router and experts. This gives

$$M = \frac{\tau}{2C_r h} \quad \text{(MoE)}, \qquad M' = \frac{\tau - C_r hr}{2C_r r} \quad \text{(MoRE)}.$$

In the regime of interest $\tau \gg C_r hr$ (equivalently $M, M' \gg r$) the projection overhead is negligible, and the ratio

$$\frac{M'}{M} \approx \frac{h}{r}$$

recovers exactly the per-expert routing-cost ratio: each MoE expert costs only a fraction r/h of the routing cost of MoE, so the same router budget supports h/r times as many experts. For example, at $h = 2048$ and $r = 16$, MoRE thus admits roughly $128\times$ more experts than MoE at the same active cost. In section 5, we choose configuration according to this derivation, with $C_e = C_r = 16$.

4.2. Realizing the speedup at large M

Even with a low-rank router, a naive two-kernel implementation must materialize the full (N, M) routing-score tensor in HBM before performing the top-k selection. As M is large in our regime, this $O(NM)$ HBM round-trip[2] dominates the layer (Figure 3): the TopK contribution overwhelms the rank-r matmuls, and the router becomes memory-bound on its own output rather than compute-bound on its FLOPs.

[2]See Appendix D.1 for a detailed GPU memory-hierarchy description.

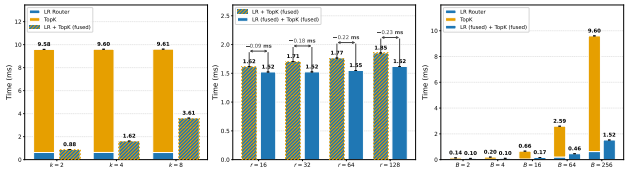

Figure 3. **(Left)** Fusing TopK with the LR matmul eliminates the (N, M) HBM round-trip on the logits tensor, giving up to $\sim 11\times$ wall-clock speedup at k=2; the fused-kernel cost still scales linearly with k as the in-SRAM merge-sort runs k iterations. **(Middle)** The benefit of additionally folding R_1 into the fused kernel grows with r, since it avoids writing the (N, r) intermediate tensor to HBM. The single-kernel variant wins by 0.09 ms at r=16 and by 0.23 ms at r=128. **(Right)** The full-fusion speedup grows with batch size, since the avoided HBM traffic on the (N, M) logits scales linearly with $N{=}B \cdot T$. At B=2 the speedup is 1.4×; by B=256 it reaches 6.3×. h=2048, M=4,096, T=1024, $N{=}B \cdot T$; mean of 200 trials with 50 warm-ups, 95% C.I. on a single NVIDIA B200 GPU.

Fix: a single fused Triton kernel. Recall from Section 2 that $R_1 \in \mathbb{R}^{M\times r}$ and $R_2 \in \mathbb{R}^{r\times h}$, so the routing logits factorize as $\ell = R_1 R_2 x$, equivalently $z = XR_2^\top$ followed by $\ell = zR_1^\top$ in batched form. Since the intermediate dimension r is small, we fuse *both* low-rank matrix multiplications $(N, h) \times (h, r)$ and $(N, r) \times (r, M)$ together with the top-k selection into a single Triton kernel. (*Phase 1*) For each block of tokens, the kernel tiles over h, accumulating $z = X_{\text{block}} R_2^\top$ in on-chip SRAM registers; the slice has shape (N_{block}, r) and occupies only a few kilobytes. (*Phase 2*) The kernel then tiles over M, computing logits $y = zR_1^\top$ per tile and updating a sequentially running top-k buffer in SRAM. The full (N, M) logits are *never materialized to HBM*: only the sparse (N, k) top values and indices are written out, cutting router output traffic from $O(NM)$ to $O(Nk)$. The top-k softmax weights and the load-balancing auxiliary loss are computed outside the kernel from the (N, k) outputs (see Appendix D.2); no (N, M) tensor is ever materialized end to end. This simultaneously eliminates the duplicated kernel-launch overhead and the intermediate (N, r) HBM round-trip that a two-kernel approach would incur. Since the top-k operation is sequential in our algorithm, the time scales linearly with k (Figure 3 Left) [3]. Full forward-, backward-pass pseudocode and analysis is in Appendix D.2–D.3.

Wall-clock gains in the large-M regime. As shown in Figure 4, the fused MoRE kernel consistently outperforms the standard router for $M > 128$, with the speedup growing monotonically in M to roughly $3 \sim 4\times$ at $M = 4{,}096$ and $M = 8{,}192$. Equivalently, at matched wall-clock time MoRE supports an order of magnitude more experts: for example, at $h = 512$, MoRE scales to $M = 2{,}048$ while the standard router is capped near $M = 128$, bringing the $\Theta(h/r)$ parameter advantage of Section 4.1 within reach in practice.

[3]For detailed discussion of our Top-K algorithm, see Appendix D.4.

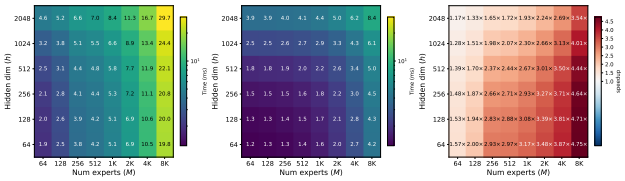

Figure 4. Forward pass wall-clock time for a single MoE layer on the (h, M) grid. (a) MoE Layer; (b) MoRE Layer ($r = 64$); (c) speedup $T_{\text{MoE}}/T_{\text{MoRE}}$. The speedup grows monotonically with M and is around $3 \sim 4\times$ at $M = 4{,}096$ and $M = 8{,}192$ across the entire h range. $s = 64$, $k = 4$, $r = 64$, $B = 256$, $T = 1024$; mean of 200 trials with 50 warm-ups on a single NVIDIA B200. See Appendix E for other configurations of s including $128, 256, 512$.

5. Experiments: MoRE improves performance at matched active FLOPs

In this section, we conduct experiments on synthetic memorization tasks and real-world tasks. Our results show that replacing MoE layers by MoRE layers in transformers can improve performance in knowledge-intensive tasks, while keeping active FLOPs fixed.

5.1. Synthetic task: increased memorization capabilities at matched active FLOPs

We first conduct a synthetic memorization study to test whether MoRE's parameter advantage at matched active FLOPs translates into capacity. We adopt the phonebook task from Jelassi et al. (2025) as a synthetic memorization dataset and find that MoRE memorizes substantially more pairs than MoE at the same active FLOPs across all scales we sweep.

In this dataset, each example is a five-letter name followed by an eight-digit number, such as "alice|12345678". We pretrain on these pairs, and report the largest number of training examples for which the model attains $\geq 90\%$ exact-matching accuracy on the digit numbers with given names, evaluated on a 1,000-example subset sampled uniformly at random from the training set. To ensure that we approach the model's full memorization capacity, we train up to 1,500 epochs, in line with the exposure thresholds per example necessary for memorization reported by Allen-Zhu & Li (2024).

In Figure 5, we sweep over hidden dimension $h \in \{64, 128, 256, 512\}$ at expert width $s = 8$ and expert width $s \in \{8, 16, 32, 64\}$ at $h = 64$. Then we find the optimal configurations by Proposition 2, such that MoE and

s, h	Mo(R)E-M	Active MoE FLOPs / layer	Total Param	Val PPL↓	TriviaQA F1↑	NQ-Open F1↑	Hella. acc_norm↑	ARC-e acc↑	SciQ acc↑	Wino. acc↑
32, 512	MoE-128	2.10M	135.6M	16.69	**13.62** ±0.21	4.61 ±0.26	32.76 ±0.47	51.43 ±1.03	73.20 ±1.40	**52.25** ±1.40
	MoRE-768	2.03M	400.1M	**15.26**	13.51 ±0.19	**5.01** ±0.25	**34.66** ±0.47	**53.45** ±1.02	**76.30** ±1.35	50.12 ±1.41
32, 1024	MoE-128	4.19M	287.0M	14.22	15.15 ±0.21	5.66 ±0.28	36.87 ±0.48	56.27 ±1.02	79.80 ±1.27	52.25 ±1.40
	MoRE-1536	4.06M	1.50B	**12.28**	**20.63** ±0.26	**8.03** ±0.35	**41.10** ±0.49	**61.99** ±1.00	**80.50** ±1.25	**52.57** ±1.40
64, 512	MoE-256	4.19M	362.9M	13.93	16.98 ±0.24	5.97 ±0.29	37.32 ±0.48	57.11 ±1.02	76.60 ±1.34	50.59 ±1.41
	MoRE-1792	4.13M	1.91B	**12.82**	**19.13** ±0.25	**6.57** ±0.30	**40.19** ±0.49	**60.44** ±1.00	**77.90** ±1.31	**52.49** ±1.40
128, 512	MoE-256	6.55M	664.9M	12.68	18.15 ±0.25	7.47 ±0.34	40.10 ±0.49	60.40 ±1.00	**78.10** ±1.31	49.25 ±1.41
	MoRE-1792	6.19M	3.76B	**11.92**	**21.48** ±0.27	**8.48** ±0.37	**43.03** ±0.49	**62.50** ±0.99	77.30 ±1.33	**53.12** ±1.40

Table 1. Pretraining results for MoE Transformers on 40B tokens. MoRE configuration ($r = 64$) matches active expert and routing FLOPs of the MoE configuration, while total parameters are scaled up. All tasks are evaluated 0-shot, with standard deviation in the parentheses. Active MoE FLOPs per token per layer are computed as $18khs + 14Mh$ for MoE and $18khs' + 14(M' + h)r$ for MoRE (using $C_e = 18, C_r = 14$; the configuration grid was selected under the simplified $C_e = C_r \approx 16$ of Proposition 2, see Appendix B.3, so MoE and MoRE FLOPs match approximately); total parameters are computed for the full model. See Appendices B.2 and B.3 for details.

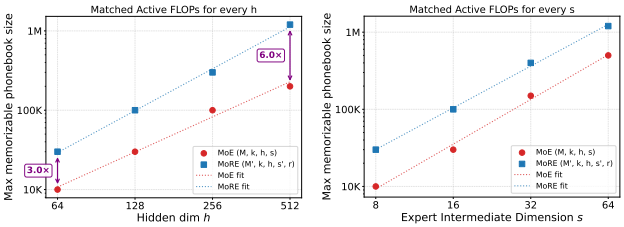

Figure 5. Maximum memorizable phonebook size at fixed active FLOPs τ, MoRE consistently memorize more examples. **Left:** sweep over hidden dimension $h \in \{64, 128, 256, 512\}$ at expert width $s = 8$. In this sweep, optimal M' scales as h, with M unchanged. **Right:** sweep over expert width $s \in \{8, 16, 32, 64\}$ at $h = 64$. In this sweep, optimal M' and M are both scaled with s. Red circles: MoE configurations; blue squares: MoRE configurations. Dotted lines are log-log linear fits through each series; full per-dot (M, k, h, s, r) tuples and max-memorizable sizes are listed in Tables 3 and 4 of Appendix B.1. The gap between MoRE and MoE *widens with* h, in agreement with Theorem 2's $\Theta(h/r)$ ratio.

MoRE are with matched active FLOPs. [4] We find that the MoRE can memorize substantially more examples than the standard router at the same active FLOPs across all scales. Notably, the gap widens with larger h, in agreement with Proposition 2's $\Theta(h/r)$ widening trend [5].

[4]We find the optimal configurations by Proposition 2, where optimal $s' = s - 2$, we round it back to s such that it is a multiple of 8 for kernel-friendly tensor shapes. The rounding inflates total MLP parameters by at most $s/(s-2) \leq 4/3$ at the smallest scale (see Appendix B.1).

[5]We sweep over for $h \leq 512, s \leq 64$ in this synthetic setup, because it becomes computationally expensive for larger models. This is because larger models can memorize $> 2M$ examples, and training on those datasets for up to 1500 epochs is out of our compute budget. We provide a larger-scale experiment when pretraining models on real-world data in 5.2.

5.2. Real-world pretraining: improved performance on knowledge-intensive tasks

At matched active FLOPs, MoRE reduces validation perplexity by 6% to 14% across four (s, h) configurations on a 40 B-token natural-language mixture (Table 1). Total parameters grow by 3 to 5× at matched active FLOPs by construction.

In these experiments, we follow the natural-language pretraining recipe of Jelassi et al. (2025): a mixture of FineWeb-edu (HuggingFaceFW, 2024), Cosmopedia (Ben Allal et al., 2024), Wikipedia, and the training splits of 12 downstream QA benchmarks (full composition in Appendix B.2). We fix $k = 4$ and select four representative (s, h) pairs that are feasible within our compute budget and in different scales: $(32, 512)$, $(32, 1024)$, $(64, 512)$, and $(128, 512)$. For the first three pairs we select the standard configuration (M, k, h, s) and the low-rank configuration (M', k, h, s, r) by Theorem 2 with matched active FLOPs. The fourth pair $(128, 512)$ is obtained by taking the $(64, 512)$ configuration and scaling both s and s' by 2, while keeping $M = 256$ and $M' = 1792$ unchanged (full configurations in Table 5).

We evaluate 0-shot via lm-evaluation-harness (Gao et al., 2023) on **knowledge-intensive** tasks: TriviaQA (Joshi et al., 2017) and Natural Questions (Kwiatkowski et al., 2019). For **commonsense and science reasoning**, we evaluate on HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2020), ARC-Easy (Clark et al., 2018), and SciQ (Welbl et al., 2017) [6]. We found that MoRE consistently matches or improves performance on knowledge-intensive tasks compared to MoE at matched active FLOPs, while still preserving performance on commonsense and science reasoning.

[6]We use 0-shot since the QA prompt format is already in-distribution from pretraining.

6. Conclusion

We introduce *MoRE*, a principled solution to the MoE routing bottleneck that scales experts by a factor of $\Theta(\frac{h}{r})$ while matching active FLOPs. We provide theoretical guarantees for its expressivenes, design a hardware-efficient implementation, and demonstrate improved performance on knowledge-intensive benchmarks like TriviaQA (Joshi et al., 2017) and Natural Questions (Kwiatkowski et al., 2019), without degrading reasoning. This work opens future directions:

Multi-GPU Study and Hardware-Aware Optimization: Our efficiency studies focus primarily on single-GPU scenarios, with multi-GPU data parallelism analysis provided in the Appendix (Appendix D.5). Investigating expert parallelism across multiple GPUs is left to future work. Additionally, further speedups may be possible by exploiting GPU-specific architectural features, as demonstrated in, for example, Guo et al. (2025).

Adaptive-Rank MoRE: We apply number of experts uniformly across all layers. Future work could study architectures where MoRE is applied only at certain layers, or has adaptive rank and number of experts for different layers.

Principled Learning Rate and Initialization: While matching the standard router's initialization variance improved upon default R_1, R_2 initializations (Appendix C), developing robust, LoRA-like (Hu et al., 2021) initialization and learning rate strategies could eliminate cross-rank tuning and further improve performance.

References

Abnar, S., Shah, H., Busbridge, D., Ali, A. M. E., Susskind, J., and Thilak, V. Parameters vs flops: Scaling laws for optimal sparsity for mixture-of-experts language models. *arXiv preprint arXiv:2501.12370*, 2025.

Allen-Zhu, Z. and Li, Y. Physics of language models: Part 3.1, knowledge storage and extraction, 2023. URL https://arxiv.org/abs/2309.14316.

Allen-Zhu, Z. and Li, Y. Physics of language models: Part 3.3, knowledge capacity scaling laws, 2024. URL https://arxiv.org/abs/2404.05405.

Ben Allal, L., Lozhkov, A., Penedo, G., Wolf, T., and von Werra, L. Cosmopedia: how to create large-scale synthetic data for pre-training. https://huggingface.co/blog/cosmopedia, 2024.

Berant, J., Chou, A., Frostig, R., and Liang, P. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of EMNLP*, 2013.

Berges, V.-P., Oğuz, B., Haziza, D., tau Yih, W., Zettlemoyer, L., and Ghosh, G. Memory layers at scale, 2024. URL https://arxiv.org/abs/2412.09764.

Bisk, Y., Zellers, R., Bras, R. L., Gao, J., and Choi, Y. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of AAAI*, 2020.

Boix-Adsera, E. Secret mixtures of experts inside your llm. *arXiv preprint arXiv:2512.18452*, 2025.

Cheng, X., Zeng, W., Dai, D., Chen, Q., Wang, B., Xie, Z., Huang, K., Yu, X., Hao, Z., Li, Y., Zhang, H., Zhang, H., Zhao, D., and Liang, W. Conditional memory via scalable lookup: A new axis of sparsity for large language models, 2026. URL https://arxiv.org/abs/2601.07372.

Chi, Z., Dong, L., Huang, S., Dai, D., Ma, S., Patra, B., Singhal, S., Bajaj, P., Song, X., Mao, X.-L., Huang, H., and Wei, F. On the representation collapse of sparse mixture of experts. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2204.09179.

Clark, A., de las Casas, D., Guy, A., Mensch, A., Paganini, M., Hoffmann, J., Damoc, B., Hechtman, B., Cai, T., Borgeaud, S., van den Driessche, G., Rutherford, E., Hennigan, T., Johnson, M., Millican, K., Cassirer, A., Jones, C., Buchatskaya, E., Budden, D., Sifre, L., Osindero, S., Vinyals, O., Rae, J., Elsen, E., Kavukcuoglu, K., and Simonyan, K. Unified scaling laws for routed language models, 2022. URL https://arxiv.org/abs/2202.01169.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Costin, B., Averbuch, T., Pai, D., Chen, N., and Keigwin, B. Momoe: Memory-optimized mixture of experts. https://blog.tilderesearch.com/blog/momoe, 2025. Tilde Research blog, July 25, 2025.

Dai, D., Deng, C., Zhao, C., Xu, R. X., Gao, H., Chen, D., Li, J., Zeng, W., Yu, X., Wu, Y., Xie, Z., Li, Y. K., Huang, P., Luo, F., Ruan, C., Sui, Z., and Liang, W. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models, 2024. URL https://arxiv.org/abs/2401.06066.

DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024a. URL https://arxiv.org/abs/2405.04434.

DeepSeek-AI. Deepseek-v3 technical report, 2024b. URL https://arxiv.org/abs/2412.19437.

Dikkala, N., Ghosh, N., Meka, R., Panigrahy, R., Vyas, N., and Wang, X. On the benefits of learning to route in mixture-of-experts models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=QV79qiKAjD.

Du, N., Huang, Y., Dai, A. M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A. W., Firat, O., et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International conference on machine learning*, pp. 5547–5569. PMLR, 2022.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2022. URL https://arxiv.org/abs/2101.03961.

Gale, T., Narayanan, D., Young, C., and Zaharia, M. Megablocks: Efficient sparse training with mixture-of-experts. In *Proceedings of Machine Learning and Systems*, volume 5, 2023.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

Geva, M., Schuster, R., Berant, J., and Levy, O. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, 2021.

Guo, W., Mishra, M., Cheng, X., Stoica, I., and Dao, T. Sonicmoe: Accelerating moe with io and tile-aware optimizations, 2025. URL https://arxiv.org/abs/2512.14080.

Hayou, S., Ghosh, N., and Yu, B. Lora+: Efficient low rank adaptation of large models, 2024. URL https://arxiv.org/abs/2402.12354.

He, X. O. Mixture of a million experts, 2024. URL https://arxiv.org/abs/2407.04153.

Hoeffding, W. 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.

Hsu, P.-L., Dai, Y., Kothapalli, V., Song, Q., Tang, S., Zhu, S., Shimizu, S., Sahni, S., Ning, H., and Chen, Y. Liger kernel: Efficient triton kernels for LLM training, 2024. URL https://arxiv.org/abs/2410.10989.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Huang, Z., Min, Q., Huang, H., Zhu, D., Zeng, Y., Guo, R., and Zhou, X. Ultra-sparse memory network, 2025. URL https://arxiv.org/abs/2411.12364.

HuggingFaceFW. fineweb-edu (revision 22b0aca), 2024. URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.

Jelassi, S., Mohri, C., Brandfonbrener, D., Gu, A., Vyas, N., Anand, N., Alvarez-Melis, D., Li, Y., Kakade, S. M., and Malach, E. Mixture of parrots: Experts improve memorization more than reasoning, 2025. URL https://arxiv.org/abs/2410.19034.

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

Jiang, T., Bordelon, B., Pehlevan, C., and Hanin, B. Hyperparameter transfer with mixture-of-expert layers, 2026. URL https://arxiv.org/abs/2601.20205.

Johnson, W. B. and Lindenstrauss, J. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in Modern Analysis and Probability (New Haven, Conn., 1982)*, volume 26 of *Contemporary Mathematics*, pp. 189–206. American Mathematical Society, Providence, RI, 1984. doi: 10.1090/conm/026/737400.

Joshi, M., Choi, E., Weld, D. S., and Zettlemoyer, L. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension, 2017. URL https://arxiv.org/abs/1705.03551.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kimi Team. Kimi K2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Krajewski, J., Ludziejewski, J., Adamczewski, K., Pióro, M., Krutul, M., Antoniak, S., Ciebiera, K., Król, K., Odrzygóźdź, T., Sankowski, P., Cygan, M., and Jaszczur, S. Scaling laws for fine-grained mixture of experts, 2024. URL https://arxiv.org/abs/2402.07871.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.-W., Dai, A. M., Uszkoreit, J., Le, Q., and Petrov, S. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. Gshard: Scaling giant models with conditional computation and automatic sharding, 2020. URL https://arxiv.org/abs/2006.16668.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://arxiv.org/abs/1711.05101.

Meta AI. The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation. https://ai.meta.com/blog/llama-4-multimodal-intelligence/, 2025.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of EMNLP*, 2018.

Mistral AI. Cheaper, better, faster, stronger: Continuing to push the frontier of AI and making it accessible to all. https://mistral.ai/news/mixtral-8x22b, 2024.

OpenAI. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Qwen Team. Qwen3-Next: Towards ultimate training & inference efficiency. https://qwenlm.github.io/blog/qwen3_next/, 2025.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Roller, S., Sukhbaatar, S., Szlam, A., and Weston, J. Hash layers for large sparse models, 2021. URL https://arxiv.org/abs/2106.04426.

Sakaguchi, K., Le Bras, R., Bhagavatula, C., and Choi, Y. WinoGrande: An adversarial Winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Sap, M., Rashkin, H., Chen, D., Bras, R. L., and Choi, Y. SocialIQA: Commonsense reasoning about social interactions. In *Proceedings of EMNLP-IJCNLP*, 2019.

Schulman, J. and Thinking Machines Lab. Lora without regret. https://thinkingmachines.ai/blog/lora/, 2025. Thinking Machines Lab Connectionism blog, September 29, 2025.

Shazeer, N. Glu variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017. URL https://arxiv.org/abs/1701.06538.

Talmor, A. and Berant, J. The web as a knowledge-base for answering complex questions. In *Proceedings of NAACL-HLT*, 2018.

Tan, S., Shen, Y., Panda, R., and Courville, A. Scattered mixture-of-experts implementation, 2024. URL https://arxiv.org/abs/2403.08245.

Tillet, P., Kung, H. T., and Cox, D. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages (MAPL)*, pp. 10–19. ACM, 2019. doi: 10.1145/3315508.3329973.

Welbl, J., Liu, N. F., and Gardner, M. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106. Association for Computational Linguistics, 2017.

xAI. Open release of Grok-1. https://x.ai/blog/grok-os, 2024.

Yan, Z., Bai, H., Yao, X., Liu, D., Liu, T., Liu, H., Li, P., Wu, E., Fan, S., Tao, L., Zhang, R., Wang, Y., Xu, S., Chang, J., Chen, X., Li, K., Bai, Y., Deng, G., Zheng, N., Korthikanti, V. A., Khattar, A., He, E., Govande, S., Lym, S., Zhu, Z., Zhang, Q., Yuan, H., Ren, X., Fu, D., Ma, T., Zhang, S., Shao, J., Wang, R., Rengasamy, V., Garg, R., Bhavani, S., Li, X., Zhou, C., Wu, D., Wei, Y., Aithal, A., Andersch, M., Shoeybi, M., Yao, J., and Yang, J. Scalable training of mixture-of-experts models with megatron core, 2026. URL https://arxiv.org/abs/2603.07685.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yang, M., Togo, R., Li, G., Ogawa, T., and Haseyama, M. L2r: Low-rank and lipschitz-controlled routing

for mixture-of-experts, 2026. URL https://arxiv.org/abs/2601.21349.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of EMNLP*, 2018.

Zadouri, T., Hoehnerbach, M., Shah, J., Liu, T., Thakkar, V., and Dao, T. Flashattention-4: Algorithm and kernel pipelining co-design for asymmetric hardware scaling, 2026. URL https://arxiv.org/abs/2603.05451.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 4791–4800. Association for Computational Linguistics, 2019.

A. Frontier MoE Model Configurations

For reference, Table 2 summarizes the configurations of recent frontier MoE models. The trend toward higher sparsity ($M \gg k$) and higher granularity ($s < h$) motivates the fine-grained regime studied throughout this paper.

Model	Total	Active	L	M	k	h	s
Mixtral 8x22B (Mistral AI, 2024)	141B	39B	56	8	2	6144	16384
Grok-1 (xAI, 2024)	314B	79B	64	8	2	6144	32768
DeepSeek-V2 (DeepSeek-AI, 2024a)	236B	21B	60	160	6	**5120**	**1536**
DeepSeek-V3 (DeepSeek-AI, 2024b)	671B	37B	61	256	8	**7168**	**2048**
Llama 4 Maverick (Meta AI, 2025)	400B	17B	48	128	1	5120	8192
Qwen3 235B-A22B (Yang et al., 2025)	235B	22B	94	128	8	**4096**	**1536**
Kimi K2 (Kimi Team, 2025)	1.04T	32B	61	384	8	**7168**	**2048**
gpt-oss-120b (OpenAI, 2025)	117B	5.1B	36	128	4	2880	2880
Qwen3-Next-80B-A3B (Qwen Team, 2025)	80B	3B	48	512	10	**2048**	**512**

Table 2. Configurations of frontier MoE models. M: total experts; k: active experts per token; h: hidden dimension; s: expert FFN intermediate dimension. The trend shows greater sparsity and granularity is preferred.

B. Experimental Details

Router initialization. We initialize each entry of the low-rank router factors as $\mathcal{N}(0, \sigma'^2)$ with $\sigma' = \sqrt{\sigma}/r^{1/4}$ and $\sigma = 0.02$, so that the variance of the router logits matches the standard router. See Appendix C for the derivation and an empirical comparison to the default initialization.

To make the speedup by the low-rank router more apparent, we seek an efficient implementation of the Mixture-of-Experts architecture and therefore build upon ScatterMoE (Tan et al., 2024) (refer to Appendix D for more details). We believe that further speedups are achievable on specific hardware by applying more careful optimization as in (Guo et al., 2025).

B.1. Phonebook

Tokenization. Each example is tokenized as "`$abcde|12345678.`", with a five-letter lowercase name, the separator "`|`", and an eight-digit number, padded with "$\star$" to a fixed sequence length of 16. The vocabulary has $V = 40$ symbols: 26 lowercase letters, 10 digits, and 4 specials ("`$`" BOS, "`.`" EOS, "`|`" separator, "$\star$" pad).

Architecture and sweep. All models are $L = 6$-layer Transformers with top-k routing at $k = 4$ and low-rank rank $r = 16$. The active budget τ has two natural scaling axes at fixed h: the hidden dimension itself and the expert width s. We sweep each in turn: (i) $h \in \{64, 128, 256, 512\}$ at fixed $s = 8$, and (ii) $s \in \{8, 16, 32, 64\}$ at fixed $h = 64$. For every (h, s), the standard baseline takes $M = 4s$ experts, and the low-rank counterpart takes the largest M' that matches the standard active FLOPs at $r = 16$, following Proposition 2.

Choice of s'. Under our convention $C_e = C_r$, Proposition 2 prescribes the optimal MoRE expert width as $s'^* = s - r/(2k) = s - 2$ at $r = 16, k = 4$. For $s \in \{8, 16, 32, 64\}$ this gives $s'^* \in \{6, 14, 30, 62\}$, none of which are multiples of 8 – the kernel-friendly width for ScatterMoE. We round each s' to the nearest multiple of 8, which collapses to s itself across the entire sweep, and we use the prescribed $M'^* = h(2ks - r)/(2r) = h(s - 2)/4$ unchanged. The rounding inflates the total MLP parameter count by a factor $s/(s - 2)$, which is largest at the smallest scale ($8/6 \approx 1.33\times$ at $s = 8$) and shrinks to $\approx 1.03\times$ at $s = 64$; it does not change the qualitative trends of Figure 5.

Training and learning-rate selection. Each configuration is trained with batch size 1,024 under a constant-with-warmup schedule (warmup ratio 0.05, weight decay 0.01, $\beta_2 = 0.95$), with early stopping once exact-string accuracy on the digit numbers crosses 0.9. The learning rate is selected per configuration by a short calibration phase over $\{1\times10^{-4}, 3\times10^{-4}, 1\times10^{-3}\}$, picking the value with the highest validation character accuracy on a brief preliminary run; the calibrated rate is then used for the full 1,500-epoch run.

Per-dot configurations. Tables 3 and 4 list, for every dot in Figure 5, the full (M, k, h, s) and (M, k, h, s, r) configuration and the maximum memorizable phonebook size at the calibrated learning rate. Table 3 covers the vary-h sweep at $s = 8$,

and Table 4 covers the vary-s sweep at $h = 64$; the $(h{=}64,\ s{=}8)$ corner appears in both as the shared anchor between the two sweeps.

The number of experts M are small for Table 3 as the optimal $M = ks$ in our setup, when $s = 8$, $M = ks = 32$.

Configuration	M/M'	k	h	s	r	Max memorizable
MoE	32	4	64	8	—	10K
MoE	32	4	128	8	—	30K
MoE	32	4	256	8	—	100K
MoE	32	4	512	8	—	200K
MoRE	96	4	64	8	16	30K
MoRE	192	4	128	8	16	100K
MoRE	384	4	256	8	16	300K
MoRE	768	4	512	8	16	1.2M

Table 3. Phonebook per-dot configurations for the **vary-h sweep** at $s = 8$. M (M' for MoRE) is the number of experts; k active per token; h hidden dim; s expert intermediate dim; r router rank (MoRE only). "Max memorizable" is the largest phonebook size at which the calibrated training run reached the early-stopping threshold.

Configuration	M/M'	k	h	s	r	Max memorizable
MoE	32	4	64	8	—	10K
MoE	64	4	64	16	—	30K
MoE	128	4	64	32	—	150K
MoE	256	4	64	64	—	500K
MoRE	96	4	64	8	16	30K
MoRE	224	4	64	16	16	100K
MoRE	480	4	64	32	16	400K
MoRE	992	4	64	64	16	1.2M

Table 4. Phonebook per-dot configurations for the **vary-s sweep** at $h = 64$. Columns as in Table 3. The $(h{=}64,\ s{=}8)$ row is shared with Table 3.

B.2. Real-world Pretraining

Data mixture. We follow the natural-language pre-training recipe of Jelassi et al. (2025, Appendix B.2), which targets a 64 B-token mixture: (i) FineWeb-edu deduplicated (HuggingFaceFW, 2024) (37 B); (ii) Cosmopedia synthetic textbooks (Ben Allal et al., 2024) (14 B); (iii) Wikipedia (4 B unique, looped $3\times$ for 12 B); and (iv) the training splits of 12 downstream knowledge-intensive and commonsense benchmarks (TriviaQA (Joshi et al., 2017), Natural Questions (Kwiatkowski et al., 2019), HotpotQA (Yang et al., 2018), WebQuestions (Berant et al., 2013), ComplexWebQuestions (Talmor & Berant, 2018), ARC-Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), SciQ (Welbl et al., 2017), SocialIQA (Sap et al., 2019), and Winogrande (Sakaguchi et al., 2020)), each looped $3\times$ to familiarize the model with the evaluation format (Allen-Zhu & Li, 2023).

Our actual realization differs slightly from the recipe target: ≈ 60.4 B usable tokens are written to disk, broken down as 37 B FineWeb-edu, 11.3 B Cosmopedia (v1 capped at 9.3 B short of the 12 B target, plus 2 B from v2), 12 B Wikipedia, and ≈ 0.14 B QA training splits (Q/A pairs only; HotpotQA passages and SciQ `support` fields are not yet included). Every source uses its `train` split, ensuring no eval contamination. We sample disjoint training (40 B tokens) and 5,000-sequence validation sets uniformly from this pool, following Jelassi et al. (2025)'s convention.

Tokenization. We use the GPT-2 byte-pair-encoding tokenizer (Radford et al., 2019) with vocabulary size $V = 50{,}257$. Documents from each mixture source are concatenated and chunked to a fixed sequence length of $T = 1{,}024$ tokens, then materialized as a single `uint16` memory-mapped NumPy file for efficient reuse across runs.

Architecture. All models are $L = 12$-layer Transformers with top-k routing at $k = 4$, head dimension 64, and untied input/output embeddings. We compare four reference points $(h, s) \in \{(512, 32),\ (1024, 32),\ (512, 64),\ (512, 128)\}$. For the first three, the standard baseline takes $M = 4s$ experts, and the low-rank counterpart fixes router rank $r = 64$ and takes

Scale	Model	M	k	h	s	r
s=32, h=512	Standard	128	4	512	32	–
	Low-rank (r=64)	768	4	512	24	64
s=32, h=1024	Standard	128	4	1024	32	–
	Low-rank (r=64)	1536	4	1024	24	64
s=64, h=512	Standard	256	4	512	64	–
	Low-rank (r=64)	1792	4	512	56	64
s=128, h=512	Standard	256	4	512	128	–
	Low-rank (r=64)	1792	4	512	112	64

Table 5. Architectural configurations for each variant in Table 1. M: number of experts. k: experts active per token. h: model hidden dimension. s: expert FFN hidden dimension. r: router rank (Low-rank only). All variants use the GPT-2 tokenizer. Within each Scale group, the Low-rank variant scales M up while reducing s slightly so that active FLOPs approximately match the Standard reference; the $(s{=}128, h{=}512)$ group is the exception, obtained by scaling s and s' of $(s{=}64, h{=}512)$ by 2 with M and M' fixed.

the largest M' that matches the standard active FLOPs, following Theorem 2. The fourth point $(512, 128)$ is constructed by scaling both s and s' of $(512, 64)$ by 2 while keeping $M = 256$ and $M' = 1792$ fixed; this scales active MoE FLOPs/layer by $\sim 1.56\times$ and keeps the MoE/MoRE FLOPs match within $\sim 5\%$. Full configurations are listed in Table 5.

Training and learning-rate selection. Each configuration is trained for ≈ 40 B tokens (yielding 152,584 optimizer steps at per-device batch size 64 for the h=512 groups, and 203,448 steps at per-device batch size 48 for the h=1024 group), with sequence length 1,024 on 4 GPUs in DDP, using AdamW (Loshchilov & Hutter, 2019) ($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$), weight decay 0.01, max grad norm 1.0, and a cosine learning-rate schedule with warmup ratio 0.05, in bf16 precision. The peak (global) learning rate is selected per configuration from $\{1\times10^{-4}, 3\times10^{-4}, 5\times10^{-4}, 1\times10^{-3}, 3\times10^{-3}\}$ by best validation perplexity on a short calibration run of 2,034 optimizer steps (≈ 100 M tokens). For each configuration, the router learning rate is then swept independently over the same grid. For low-rank router, we apply the same learning rate for both R_1 and R_2.

B.3. FLOPs Calculation and Total parameter accounting

For consistency with Theorem 2, we report *active FLOPs per token* (rather than active parameters) in Table 1, since the matched-active configuration grid is selected to fix this quantity. We use the standard convention that one fused multiply–add counts as 2 FLOPs. All h=512 models use n_h=8 attention heads; h=1024 uses n_h=16, n_{kv}=4. Head dimension is $d{=}h/n_h{=}64$ throughout. Vocabulary size is V=50,257 (GPT-2 BPE), and input/output embeddings are untied.

FLOPs constants in Proposition 2. The forward-pass formula above tabulates the empirical FLOPs we report. For the analytical scaling argument, Proposition 2 writes the active-FLOP budget as $\tau = C_e \cdot khs + C_r \cdot Mh$, with $C_r r(h + M)$ in place of $C_r Mh$ for the rank-r low-rank router, abstracting per-operation FLOPs into two scalar constants. We follow the full-training (forward + backward) accounting of Krajewski et al. (2024, Appendix E), which charges 6 FLOPs per pair of (active parameter, processed token) for each linear projection (forward 2 + backward input-grad 2 + backward weight-grad 2). A SwiGLU expert performs three projections, giving $C_e = 3 \times 6 = 18$. The dense router charges 6 for its linear projection plus 2 FLOPs per pair for each of four token-routing operations (forward and backward dispatches of inputs and outputs), giving $C_r = 6 + 4 \times 2 = 14$ as in Krajewski et al. (2024). The forward-only ratio is similar: $C_e^{\text{fwd}}/C_r^{\text{fwd}} = 6/6 = 1$, compared to $C_e/C_r = 18/14 \approx 1.29$ for full training. We set $C_e = C_r = 16$ to make parameters easier to se, for example: $M = ks$. We use the same constant $C_r = 16$ for the low-rank router: each of R_1, R_2 contributes a linear-projection term and the token-routing operations are unchanged from the dense case. The $\Theta(h/r)$ ratio in Proposition 2 depends only on the leading $C_r Mh$ term, so it is insensitive to the precise value of C_r within the defensible range $[6, 20]$ noted by Krajewski et al. (2024).

Total parameter count. Total parameters include the embedding matrix, all M experts (active and inactive), the router, attention, layer norms, and the unembedding head:

$$P_{\text{total}} = \underbrace{2\,V\,h}_{\text{embed+LM head}} + \underbrace{h}_{\text{final LN}} + L \cdot \Big[\underbrace{2h^2 + 2\,h\,n_{kv}\,d}_{\text{attention}} + P_{\text{router}} + \underbrace{3\,M\,h\,s}_{\text{all experts}} + \underbrace{2\,h}_{\text{2 RMSNorm}} \Big], \tag{2}$$

where $P_{\text{router}} = h\,M$ for the standard router and $r\,(h+M)$ for the low-rank variant. (RMSNorm has no bias, so each contributes h scale parameters.)

C. Initialization

The Switch Transformer (Fedus et al., 2022) sets the router weights elementwise to $\mathcal{N}(0, \sigma^2)$ with $\sigma = 0.02$, and prior work has reported that the precise router initialization variance has limited impact on standard-router performance. The natural default for the low-rank router carries the same per-entry standard deviation over to both factors, i.e. each entry of $R_1 \in \mathbb{R}^{M\times r}$ and $R_2 \in \mathbb{R}^{r\times h}$ is drawn i.i.d. from $\mathcal{N}(0, \sigma^2)$ with the same $\sigma = 0.02$. We refer to this as the *default* (or *standard*) initialization.

Empirically, however, we find that the default initialization noticeably degrades validation perplexity and shifts the optimal router learning rate. We instead choose σ_1, σ_2 so that the variance of the router logits matches the standard baseline. Lemma 1 makes this matching condition precise.

C.1. Setup

We treat $x \in \mathbb{R}^h$ as a fixed nonzero input and condition on it throughout. The standard router $R \in \mathbb{R}^{M\times h}$ has entries i.i.d. $\mathcal{N}(0, \sigma^2)$, and the low-rank factors $R_1 \in \mathbb{R}^{M\times r}$, $R_2 \in \mathbb{R}^{r\times h}$ have independent entries i.i.d. $\mathcal{N}(0, \sigma_1^2)$ and $\mathcal{N}(0, \sigma_2^2)$ respectively, with R_1 independent of R_2. We denote the corresponding learning rates by η for the standard router, and η_1, η_2 for R_1, R_2. Our goal is to derive σ_1, σ_2 (and, in future work, η_1, η_2) by matching the variance and the logit update of the standard router.

C.2. Matching variance

Lemma 1 (Variance matching)**.** *Under the setup above, for any fixed $x \neq 0$, $\mathrm{Var}([R_1R_2x]_k) = \mathrm{Var}([Rx]_k)$ for every $k \in [M]$ if and only if*

$$\sigma_1\sigma_2 \;=\; \frac{\sigma}{\sqrt{r}}.$$

The matching condition does not depend on x.

Proof. All variances below are conditional on the fixed input x.

For the standard router, $[Rx]_k = \sum_{j=1}^{h} R_{kj}\,x_j$ is a linear combination of i.i.d. $\mathcal{N}(0, \sigma^2)$ entries with deterministic coefficients x_j, so

$$\mathrm{Var}\big([Rx]_k\big) \;=\; \sigma^2\|x\|_2^2.$$

For the low-rank router, write $z = R_2x \in \mathbb{R}^r$, so $[R_1R_2x]_k = \sum_{i=1}^{r}(R_1)_{ki}\,z_i$. The same linear-combination argument applied to R_2 gives $\mathrm{Var}(z_i) = \sigma_2^2\|x\|_2^2$. Since R_1 is independent of R_2 and zero-mean, each summand satisfies

$$\mathrm{Var}\big((R_1)_{ki}\,z_i\big) \;=\; \mathbb{E}\big[(R_1)_{ki}^2\big]\,\mathbb{E}\big[z_i^2\big] \;=\; \sigma_1^2\,\sigma_2^2\,\|x\|_2^2,$$

and the r summands are mutually independent because they share no entry of R_1. Summing,

$$\mathrm{Var}\big([R_1R_2x]_k\big) \;=\; r\,\sigma_1^2\,\sigma_2^2\,\|x\|_2^2.$$

Equating the two variances gives $r\,\sigma_1^2\,\sigma_2^2\,\|x\|_2^2 = \sigma^2\|x\|_2^2$, and since $\|x\|_2^2 > 0$ this is equivalent to $\sigma_1\sigma_2 = \sigma/\sqrt{r}$, independent of x. □

In all the experiments in this paper, we set $\sigma_1 = \sigma_2$ and $\eta_1 = \eta_2$. Combined with Lemma 1, this gives

$$\sigma_1 \;=\; \sigma_2 \;=\; \frac{\sqrt{\sigma}}{r^{1/4}}, \qquad \sigma = 0.02.$$

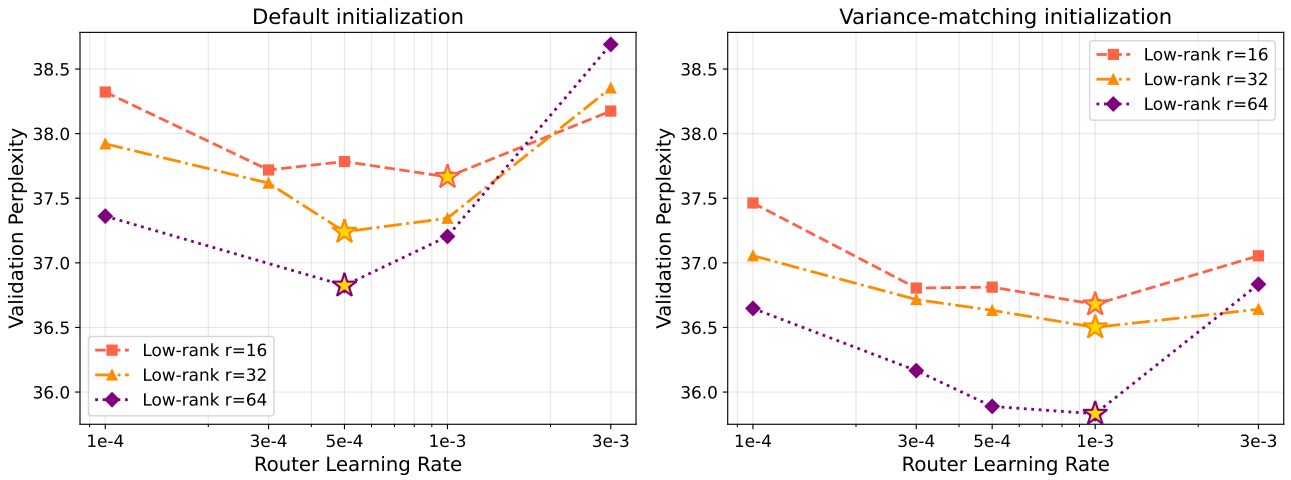

Figure 6. Router learning-rate sweep on FineWeb-edu at the optimal global learning rate $\eta = 10^{-3}$, comparing the default per-entry initialization (*left*) against the variance-matching initialization of Lemma 1 (*right*) for low-rank ranks $r \in \{16, 32, 64\}$. Each curve trains for 400M FineWeb-edu tokens. The variance-matching initialization achieves lower validation perplexity across ranks.

C.3. Discussion

We conduct empirical comparison in Figure 6. We suspect that an analysis similar to LoRA (Hu et al., 2021) and LoRA+ (Hayou et al., 2024; Schulman & Thinking Machines Lab, 2025) can enable a principled choice of router initialization and learning rate in this low-rank configuration, removing the need to tune η_1, η_2 separately for each rank. In a complementary direction, Jiang et al. (2026) derive a hyperparameter-transfer parameterization for MoE routing weights via a dynamical mean-field analysis; we believe their framework can be adapted to the low-rank router setting as well.

D. Efficiency

For the MoE architecture, we build upon ScatterMoE (Tan et al., 2024) and implement a new fused Triton kernel for the router (refer to Figure 7 for the detailed operations in the forward pass). We believe that further speedups are achievable on specific hardware by applying more careful optimization as in (Guo et al., 2025).

Why we build on ScatterMoE. Figure 7 illustrates the operations in one MoE layer forward pass. Beyond the router, the expert MLP involves token dispatch (sorting by expert assignment), two grouped linear transformations with a SwiGLU activation, and a scatter-back step that combines expert outputs. Each of these steps is a potential source of memory traffic. For example, a naive grouped MoE implementation pays $O(Nkh)$ memory traffic to materialize the gathered tokens (sorted by expert assignment) before the per-expert grouped GEMM, and pays the same again on the scatter-back.

ScatterMoE (Tan et al., 2024) fuses scatter-gather into the per-expert grouped matmul, removing both round-trips and producing a kernel whose only HBM traffic is the input X, the weight stack, and the output Y. With this dispatch overhead eliminated, the standard router's $O(NhM)$ matmul is the only remaining cost that grows linearly with M. Our low-rank router is designed precisely to address this last bottleneck, while reusing the rest of ScatterMoE's machinery unchanged.

D.1. GPU Memory Hierarchy and Related Analysis

Bandwidth tiers. Modern GPU architectures expose a multi-level memory hierarchy with sharply varying bandwidth. Off-chip high-bandwidth memory (HBM) provides the largest capacity but the lowest bandwidth. On-chip, the L2 cache offers an intermediate level, while each streaming multiprocessor (SM) contains a small pool of SRAM (registers and shared memory) with the highest bandwidth. Any data that does not fit in on-chip memory must be materialized to HBM, incurring an order-of-magnitude bandwidth penalty.

- **NVIDIA A100-80GB SXM:** HBM2e $\sim$80 GB at $\sim$2 TB/s; L2 cache $\sim$40 MB at $\sim$5 TB/s; per-SM SRAM $\sim$192 KB at $\sim$19 TB/s; bf16 throughput $\sim$312 TFLOPS.

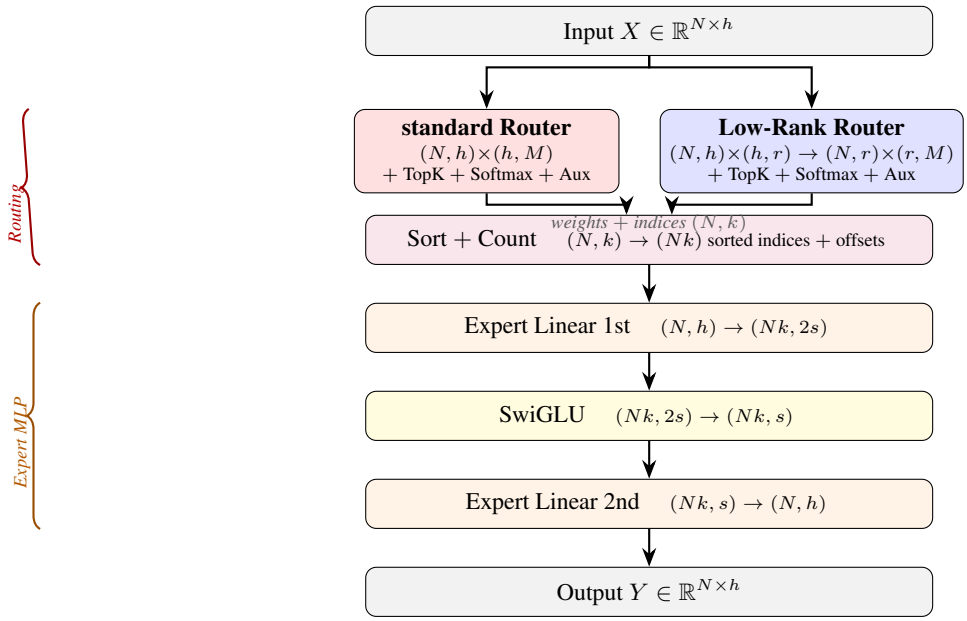

Figure 7. Operations in the forward pass of one MoE layer. The first expert linear projects to $2s$ because SwiGLU uses separate gate and up projections: $\mathrm{SiLU}(Xw_{\mathrm{gate}}) \odot Xw_{\mathrm{up}}$. Expert linear layers use fused scatter-gather kernels (Tan et al., 2024) that eliminate $O(Nkh)$ memory traffic from token dispatch, making the standard router the dominant remaining overhead and motivating the low-rank factorization.

- **NVIDIA B200 (Blackwell):** HBM3e $\sim$192 GB at $\sim$8 TB/s; L2 cache $\sim$60 MB at $\sim$10 TB/s (split across two compute dies); per-SM SRAM $\sim$228 KB at $\sim$33 TB/s; bf16 throughput $\sim$2.25 PFLOPS.

The intermediate activation of the low-rank router has shape (N_{block}, r) per thread block, which is small enough to reside entirely in SRAM when r is small (e.g., r=16 yields $\sim$2 KB per block of 64 tokens in bf16). This SRAM-residency is what makes the two-matmul fusion in our low-rank router kernel possible, and is the structural reason a B200 SM (with 33 TB/s SRAM bandwidth) shows particularly large speedups over an A100 at large M.

Comparison to other fused-MoE kernels. Existing efficient MoE kernels, ScatterMoE (Tan et al., 2024), MegaBlocks (Gale et al., 2023), MoMoE (Costin et al., 2025) and SonicMoE (Guo et al., 2025), do top-k computation on the materialized (N, M) logits on the HBM and have to pay expensive read and write costs. Our kernel fuses the router with top-k and do sequential top-k operation on each tile, this avoid expensive memory operations between HBM and SRAM, but the cost would also scale with k, making large k infeasible in our setup.

D.2. Forward-Pass Pseudocode

We give Triton-style pseudocode for the fused kernel we used. It never materialize the (N, M) logits tensor in HBM, the running top-k buffer lives in SRAM throughout.

Algorithm 1 Fused low-rank router + Top-k (forward)

Require: Input $X \in \mathbb{R}^{N\times h}$, router weights $R_1 \in \mathbb{R}^{M\times r}$, $R_2 \in \mathbb{R}^{r\times h}$, top-$k$ size k, block sizes B_N, B_M, B_H
Ensure: $\mathrm{TopVals} \in \mathbb{R}^{N\times k}$, $\mathrm{TopIdxs} \in \mathbb{Z}^{N\times k}$ in HBM

for each token block $b \in \{0, \ldots, \lceil N/B_N \rceil - 1\}$ **in parallel do**
  ▷ *Phase 1: $z = X_{\mathrm{block}} R_2^\top$ accumulated in SRAM*
    $z \leftarrow 0$ ▷ (B_N, r) in SRAM registers
    for each hidden tile $i \in \{0, B_H, 2B_H, \ldots\}$ **do**
      Load X_{tile} from HBM ▷ (B_N, B_H)
      Load R_2 tile from HBM ▷ (r, B_H)
      $z \mathrel{+}= X_{\mathrm{tile}} \cdot R_2^\top$ ▷ tile matmul in SRAM
    end for
  ▷ *Phase 2: tile $zR_1^\top$ over M with running top-k*
    Allocate $\mathrm{top_vals} \leftarrow -\infty$, $\mathrm{top_idxs} \leftarrow 0$ ▷ (B_N, k) in SRAM
    for each expert tile $m \in \{0, B_M, 2B_M, \ldots\}$ **do**
      Load R_1 tile from HBM ▷ (B_M, r), cached in L2 across blocks
      $y_{\mathrm{tile}} \leftarrow z \cdot R_1^\top$ ▷ (B_N, B_M) in SRAM
      for $c = 1, \ldots, k$ **do** ▷ merge tile logits into running top-k
        $(v_{\max}, j_{\max}) \leftarrow (\max, \arg\max)$ of y_{tile} along axis 1
        $j_{\min} \leftarrow \arg\min$ of $\mathrm{top_vals}$ along axis 1
        If $v_{\max} > \mathrm{top_vals}[:, j_{\min}]$: write $(v_{\max}, m + j_{\max})$ into slot $j_{\min}$
        Mask $y_{\mathrm{tile}}[:, j_{\max}] \leftarrow -\infty$ ▷ prevent re-selection within this tile
      end for
    end for
    Write $\mathrm{top_vals}, \mathrm{top_idxs}$ to HBM
end for

Top-k load-balancing loss. The Switch-Transformer load-balancing loss (Fedus et al., 2022) is $\mathcal{L}_{\mathrm{lb}} = M \sum_{i=1}^{M} f_i p_i$, where f_i is the fraction of tokens routed to expert i and p_i is the mean routing probability for expert i. A direct implementation needs the full (N, M) softmax to compute p_i, which would defeat the kernel-level $O(NM)$ HBM savings of the fused router.

We use a top-k variant of this loss that requires only the (N, k) outputs of the fused kernel and no (N, M) softmax. Let $\mathcal{S}_n = \mathrm{TopIdxs}[n, :]$ denote the top-k expert indices for token n and ℓ_n^{top} the corresponding k logits returned by the kernel. We define

$$w_{n,j} \;=\; \mathrm{softmax}(\ell_n^{\mathrm{top}})_j \quad (j \in [k]), \qquad f_i \;=\; \frac{1}{Nk} \sum_{n=1}^{N} \sum_{j=1}^{k} \mathbf{1}[\mathcal{S}_n(j) = i], \qquad p_i \;=\; \frac{1}{N} \sum_{n=1}^{N} \sum_{j=1}^{k} \mathbf{1}[\mathcal{S}_n(j) = i]\, w_{n,j},$$

so that $\sum_i f_i = \sum_i p_i = 1$. The $w_{n,j}$ are the same softmax weights used to combine expert outputs in the rest of the layer. After the fused kernel emits $\mathrm{TopVals}, \mathrm{TopIdxs} \in \mathbb{R}^{N\times k}$, we compute (i) w by a softmax along the k-axis on TopVals, (ii) f_i by `bincount` on TopIdxs.flatten(), and (iii) p_i by a differentiable weighted `bincount` on the same indices with weights w. None of these touches an (N, M) tensor, and gradients flow into the routing weights through w via the weighted bincount. This deviates from the standard Switch loss in two ways: the softmax denominator is restricted to each token's top-k logits (not all M), and gradients propagate only through the selected experts.

D.3. Backward-Pass Pseudocode

Algorithm 2 Low-rank router (backward) with activation recompute

Require: X, $R_1 \in \mathbb{R}^{M\times r}$, $R_2 \in \mathbb{R}^{r\times h}$, $\mathrm{TopIdxs} \in \mathbb{Z}^{N\times k}$, $\nabla g \in \mathbb{R}^{N\times k}$ (top-k logit gradients)
Ensure: $\nabla X \in \mathbb{R}^{N\times h}$, $\nabla R_1 \in \mathbb{R}^{M\times r}$, $\nabla R_2 \in \mathbb{R}^{r\times h}$ in HBM

for each token block b **in parallel do**
  ▷ *Phase 1: recompute $z = X_{\text{block}} R_2^\top$ in SRAM (saves (N, r) activation memory)*
  $z \leftarrow 0$ ▷ (B_N, r) in SRAM
  for each hidden tile i **do**
    $z \mathrel{+}= X_{\text{tile}} \cdot R_2^\top$
  end for
  ▷ *Phase 2: $\nabla z[n,:] = \sum_{j\in[k]} \nabla g[n,j] \cdot R_1[\mathrm{TopIdxs}[n,j],:]$ (sparse over selected experts)*
  $\nabla z \leftarrow 0$ ▷ (B_N, r) in SRAM
  for each (n, j) in block, $j \in [k]$ **do**
    $m \leftarrow \mathrm{TopIdxs}[n,j]$
    $\nabla z[n,:] \mathrel{+}= \nabla g[n,j] \cdot R_1[m,:]$
  end for
  ▷ *Phase 3: $\nabla X_{\text{block}} = \nabla z \cdot R_2$ tiled over h*
  for each hidden tile i **do**
    Load R_2 tile, write $\nabla X_{\text{tile}} \leftarrow \nabla z \cdot R_2$
  end for
end for
Outside the kernel:
$\nabla R_1[m,:] \mathrel{+}= \nabla g[n,j] \cdot z[n,:]$ via `scatter_add` on (M, r) ▷ accumulator
$\nabla R_2 \leftarrow (\nabla z)^\top X$ ▷ dense $(r, N) \times (N, h) \to (r, h)$

D.4. Top-K Algorithm Analysis

Behavior at large k. Our implementation is slower than `torch.topk` for $K \geq 16$. Per-token cost is $\Theta(K \cdot M) + \Theta(K^2 \cdot M/\texttt{BLOCK_M})$ for our fused kernel, vs $\Theta(M)$ for `torch.topk`. To see where the terms in our cost come from: for each of the $M/\texttt{BLOCK_M}$ tiles, we run K iterations. Each iteration finds the max and argmax of the `BLOCK_M` new logits, compares against the min of the K-element running buffer, and swaps if the candidate is larger. The $\Theta(K \cdot M)$ term aggregates the K iterations of max/argmax reductions over the `BLOCK_M` new candidates, across $M/\texttt{BLOCK_M}$ tiles. The $\Theta(K^2 \cdot M/\texttt{BLOCK_M})$ term aggregates the min/argmin reductions over the K-element buffer that happen inside the same K iterations. PyTorch's `torch.topk` uses a radix-select algorithm[7]: it makes a small constant number of $\Theta(M)$ passes that scan all M values to identify the bucket containing the K-th largest, followed by a $\Theta(M)$ gather of the winners. K appears in branching decisions at each pass and as the stopping criterion of the gather, but does not scale the work of any pass; within the kernel's single-block code path the per-token cost is therefore $\Theta(M)$ and nearly independent of K. The crossover between the two paths is set by how many K iterations of our merge equal the constant work of these passes. On a B200 with $M = 4096$ this falls at $K \approx 16$.

A natural alternative when K is large is to replace the K-pass selection with a single bitonic sort over the $(\texttt{BLOCK_N}, M)$ logits buffer. Bitonic sort is a fixed comparison network of $\Theta(\log^2 M)$ parallel passes over M elements, requiring $\Theta(M \log^2 M)$ compare-and-swap operations in total; once the row is sorted, reading the top K is a constant-time slice, so the cost is independent of K. We verified this with a prototype using `triton.language.topk`: it outperforms our K-pass kernel past $K \approx \texttt{BLOCK_M}$, but remains slower than `torch.topk` in absolute terms across the K range we tested, because $\Theta(M \log^2 M)$ is asymptotically larger than $\Theta(M)$. We suspect that a more carefully engineered fused topk along the lines of SonicMoE (Guo et al., 2025) (Section D and Appendix F.4), which performs the bitonic sort entirely in registers with warp-shuffle communication and bit-packed sort keys, could outperform `torch.topk` at larger K and close this gap.

[7] https://github.com/pytorch/pytorch/blob/b54e466fd04e5e736662a6206d81ab0d5fe85d91/aten/src/ATen/native/cuda/TensorTopK.cu#L40

D.5. Inter-device overhead under data parallelism

We conduct the inter-device overhead study on multiple B200 GPUs to see whether this dominates other performance bottlenecks in multi-GPU training. Due to our resource constraints, we only conduct the study on four GPUs, which mimic our training setup.

Setup. We replicate the matched-active-FLOPs configuration from Section 5: dense MoE with $M{=}256, s{=}64$ (362.9 M parameters) and MoRE with $M{=}1792, s{=}56$ ($\approx$1,910 M parameters across $r \in \{16, 32, 64, 128\}$), both at $h{=}512, L{=}12, k{=}4, T{=}1024$ and per-device batch $B{=}64$. We measure the per-iteration time of forward + backward + AdamW step in two configurations: a single B200, and `DistributedDataParallel` on 4×B200 with NCCL all-reduce. We report the mean over 25 iterations after 8 warm-ups.

Result. Table 6 reports per-iteration training time in both configurations and the implied DDP overhead. For every $r \in \{16, 32, 64, 128\}$ the DDP-4 overhead stays below 4% of single-GPU step time, comparable to the dense baseline. The absolute overhead is 3–9 ms per step.

Config	Single-GPU FBO (ms / iter)	DDP-4 FBO (ms / iter)	DDP overhead (%)
Dense $M{=}256$	226.9	229.8	1.3
MoRE $r{=}16$	243.2	252.3	3.7
MoRE $r{=}32$	241.9	250.8	3.7
MoRE $r{=}64$	242.6	251.0	3.5
MoRE $r{=}128$	246.8	250.6	1.5

Table 6. Per-iteration training time on a single B200 versus 4×B200 with DDP NCCL all-reduce. DDP overhead is $(t_{\text{DDP-4}} - t_{\text{single}})/t_{\text{single}}$. Configuration: $h{=}512, L{=}12, k{=}4, T{=}1024, B{=}64$ per device. Mean of 25 iterations after 8 warm-ups.

E. Additional Plots

E.1. Extra plots in Expert Usage

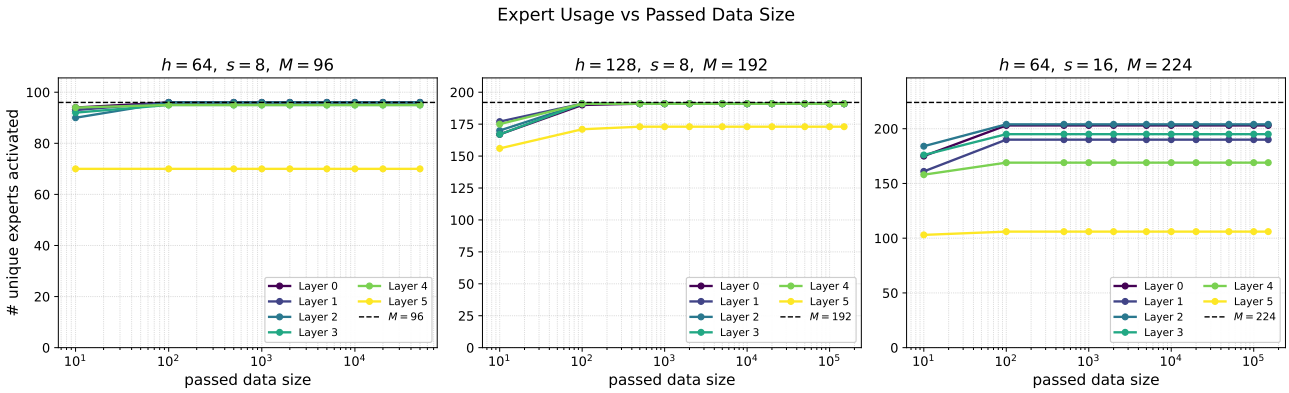

Figure 8. Expert usage vs. the number of passed validation examples for three MoRE configurations picked from Tables 3 and 4. In particular for the third configuration ($h{=}64$, $s{=}16$, $M{=}224$), the expert usage drops as the layer index increases.

E.2. Forward-pass wall-clock time for a single MoE layer for different s

We find low-rank router achieves consistent speedups over values of s from 64 to 512 as we tested. In particular, it always achieves 2x speedup at the range of $M = 4096$ and $M = 8192$.

Figure 4 in the main text shows the (h, M) prefill heatmaps at $s{=}64$. Figure 9 gives the same three-panel visualization for $s \in \{128, 256, 512\}$; together these cover all values of s we benchmarked for this experiment. Other settings match Figure 4: $k{=}4$, $r{=}64$, $B{=}256$, $T{=}1024$; mean of 200 trials with 50 warm-ups on a single NVIDIA B200. White (blank) regions mark (h, M) pairs where the benchmark exhausted GPU memory on this one-GPU setup.

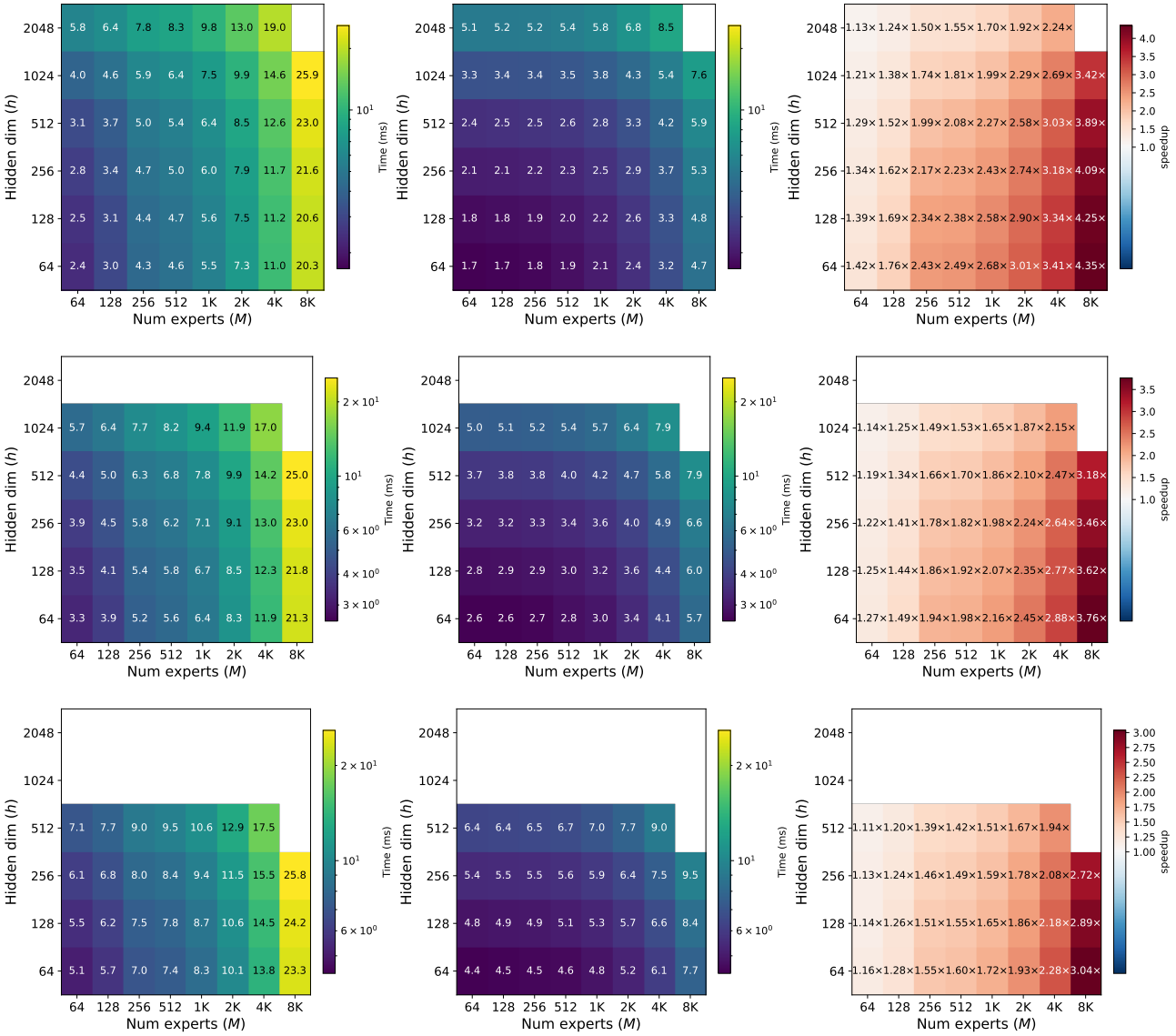

Figure 9. Prefill wall-clock heatmaps for a single MoE layer on the (h, M) grid (same layout as Figure 4), for $s \in \{128, 256, 512\}$ (top to bottom). Blank cells match Figure 4: omitted entries correspond to GPU out-of-memory on one NVIDIA B200.

E.3. End-to-End Forward-Pass Breakdown at Large M

Figure 10 measures the end-to-end wall-clock time of a 12-layer transformer with 8 attention heads (FlashAttention-4 (Zadouri et al., 2026)), vocabulary size V=50,257, h=512, s=64, k=4, B=64, T=1024, sweeping $M \in \{1{,}024, 2{,}048, 4{,}096, 8{,}192\}$.

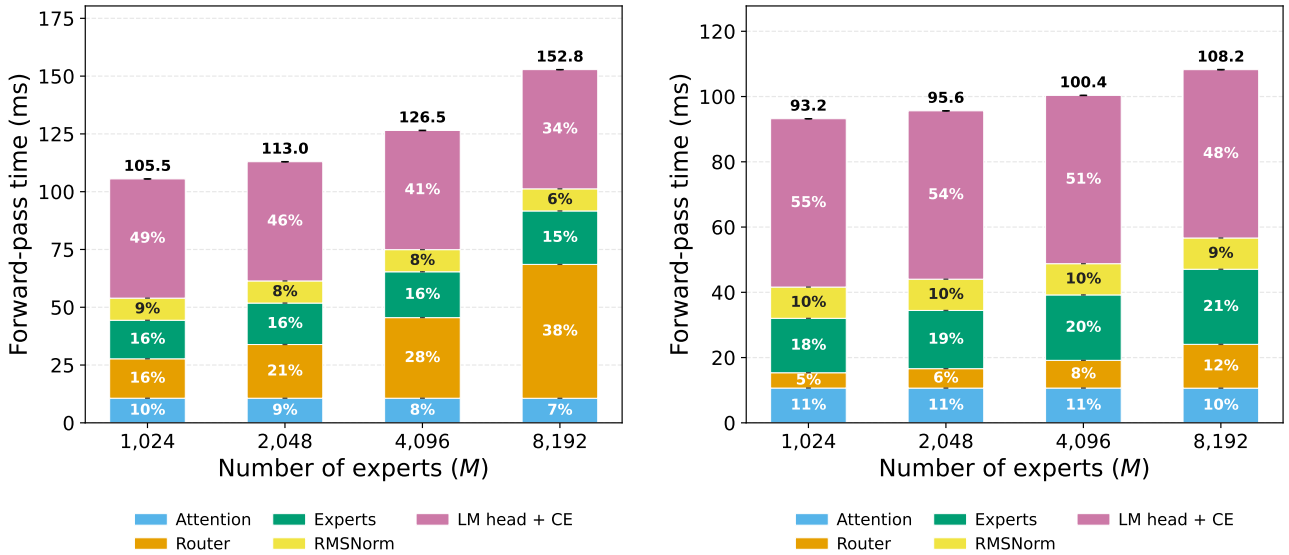

Figure 10. **End-to-end forward-pass breakdown at h=512, sweeping M.** Each bar decomposes the wall-time of a 12-layer transformer forward pass into Attention, Router, Experts (dispatch + grouped MLP), RMSNorm, and the Liger-fused (Hsu et al., 2024) LM head + cross-entropy. **(Left)** Standard **MoE** router (unfused $h \times M$ matmul + top-k + softmax + load-balancing-loss as separate kernels): the router slice grows from 16% of total at M=1,024 to **38% at** M=8,192, becoming the largest single component and the dominant driver of the rising end-to-end cost. **(Right)** Fused low-rank **MoRE** router (r=64): the router slice stays at 5–12% across the same sweep, and total wall-time grows much more slowly with M. At M=8,192, MoRE is **30% faster end-to-end** (108.2 ms vs. 152.8 ms), with router cost reduced from 57.9 ms to 13.4 ms. Mean of 200 trials with 50 warm-ups, 95% C.I. on a single NVIDIA B200.

F. Scaling Total Parameters

Proposition 2 (Optimal scaling under fixed active-FLOPs)**.** *Fix the active-FLOP budget τ, the hidden dimension h, and the active expert count k. The MoE configuration (M, k, h, s) and the MoRE configuration (M', k, h, s', r) that maximize total MLP parameters under matched τ are*

$$M^* = \frac{\tau}{2C_r h}, \qquad s^* = \frac{\tau}{2C_e kh}, \qquad T^*_{\text{MoE}} = \frac{3\tau^2}{4C_e C_r\, hk},$$

$$M' = \frac{\tau'}{2C_r r}, \qquad s' = \frac{\tau'}{2C_e kh}, \qquad T^*_{\text{MoRE}} = \frac{3(\tau')^2}{4C_e C_r\, rk},$$

where $\tau' := \tau - C_r hr$. Furthermore, in the regime $M, M' \gg r$ (equivalently $\tau \gg C_r hr$),

$$\frac{M'}{M^*} \to \frac{h}{r}, \qquad \frac{T^*_{\text{MoRE}}}{T^*_{\text{MoE}}} \to \frac{h}{r}.$$

Proof. We use the same budget-allocation argument in both settings.

MoE. Allocate fraction $\alpha \in [0, 1]$ of the budget to the router and $1 - \alpha$ to the experts:

$$C_r Mh = \alpha\tau, \qquad C_e khs = (1-\alpha)\tau,$$

so $M = \alpha\tau/(C_r h)$ and $s = (1-\alpha)\tau/(C_e kh)$. Substituting into $T_{\text{MoE}} = 3Mhs$,

$$T_{\text{MoE}}(\alpha) = \frac{3\tau^2}{C_e C_r\, hk} \cdot \alpha(1-\alpha),$$

which is maximized at $\alpha = 1/2$, yielding the stated $M^*, s^*, T^*_{\text{MoE}}$.

MoRE. The router cost expands as $C_r(M' + h)r = C_r M' r + C_r hr$, where the second term is a fixed overhead independent of M'. Subtracting it leaves a variable budget $\tau' := \tau - C_r hr$, which we split the same way:

$$C_r M' r \;=\; \alpha\tau', \qquad C_e khs' \;=\; (1-\alpha)\tau',$$

so $M' = \alpha\tau'/(C_r r)$ and $s' = (1-\alpha)\tau'/(C_e kh)$. Substituting into $T_{\text{MoRE}} = 3M'hs'$,

$$T_{\text{MoRE}}(\alpha) \;=\; \frac{3(\tau')^2}{C_e C_r\, rk} \cdot \alpha(1-\alpha),$$

again maximized at $\alpha = 1/2$, yielding the stated $M', s', T^*_{\text{MoRE}}$.

Limit. The ratios are

$$\frac{M'}{M^*} \;=\; \frac{h}{r}\left(1 - \frac{C_r hr}{\tau}\right), \qquad \frac{T^*_{\text{MoRE}}}{T^*_{\text{MoE}}} \;=\; \frac{h}{r}\left(1 - \frac{C_r hr}{\tau}\right)^2.$$

Since $\tau = C_e khs + C_r Mh > C_r Mh \gg C_r hr$ when $M \gg r$, the correction $C_r hr/\tau \ll 1$, and both ratios converge to h/r. □

G. Theoretical Results

Section 3 states the rank bounds under the normalization $\|u_i\|_2 \le 1$ and $\|R_2 x\|_2 \le 1$. The proofs below are given in a slightly more general form with arbitrary norm bounds B_u and B_z. In this form, it is convenient to measure margins in normalized units: a normalized margin η corresponds to a raw logit margin $\eta B_u B_z$. Setting $B_u = B_z = 1$ recovers the statements used in Section 3.

We first prove the lower bound. The argument shows that realizing many distinct top-k assignments forces many separated projected representations to fit inside an r-dimensional ball.

Theorem 3 (Lower bound on required rank, general norm bounds)**.** *Let $\mathcal{F} \subseteq \binom{[M]}{k}$ and let $B_u, B_z > 0$. If a rank-r router $R = R_1 R_2$ with $\|u_i\|_2 \le B_u$ and $\|R_2 x_S\|_2 \le B_z$ realizes every $S \in \mathcal{F}$ at margin at least $\eta B_u B_z$, then*

$$|\mathcal{F}| \le (1 + 2/\eta)^r, \qquad r \ge \frac{\log|\mathcal{F}|}{\log(1 + 2/\eta)}.$$

Proof. The proof is a volume packing argument: distinct assignments force their projections z_S to be far apart in $\mathbb{R}^r$, and we count how many such well-separated points fit in a ball.

For each $S \in \mathcal{F}$, choose an input x_S realizing S with margin $\eta B_u B_z$, and write $z_S = R_2 x_S$. Take any two distinct $S, T \in \mathcal{F}$: since $|S| = |T| = k$, there exist $i \in S \setminus T$ and $j \in T \setminus S$. At x_S expert i wins over j, while at x_T expert j wins over i, so

$$\langle u_i - u_j,\, z_S\rangle \;\ge\; \eta B_u B_z, \qquad \langle u_j - u_i,\, z_T\rangle \;\ge\; \eta B_u B_z.$$

Adding the inequalities and applying Cauchy–Schwarz with $\|u_i - u_j\|_2 \le 2B_u$ gives $\|z_S - z_T\|_2 \ge \eta B_z$.

Hence $\{z_S : S \in \mathcal{F}\}$ are $|\mathcal{F}|$ points pairwise separated by at least ηB_z, all lying in the radius-B_z ball of $\mathbb{R}^r$. The open balls of radius $\eta B_z/2$ around these points are therefore disjoint and contained in the ball of radius $B_z(1 + \eta/2)$. Comparing volumes,

$$|\mathcal{F}| \cdot \left(\frac{\eta B_z}{2}\right)^r \;\le\; \left(B_z\left(1 + \frac{\eta}{2}\right)\right)^r,$$

which rearranges to $|\mathcal{F}| \le (1 + 2/\eta)^r$. □

Proof of Theorem 2. Apply Theorem 3 with $B_u = B_z = 1$. □

We next prove the upper bound. The construction first realizes the desired assignments in dimension M using the standard basis, then applies a Johnson–Lindenstrauss projection (Johnson & Lindenstrauss, 1984) to reduce the dimension while preserving the inner products that determine the top-k ordering.

Theorem 4 (Upper bound on required rank, general norm bounds)**.** *Fix $1 \le k \le M-1$, $\mathcal{F} \subseteq \binom{[M]}{k}$, and norm bounds $B_u, B_z > 0$. Set $\rho := \sqrt{1/k + 1/(M-k)} > 1/\sqrt{k}$. For any target normalized margin $\eta \in (0, \rho)$, $h \ge r$, and*

$$r \;\ge\; \frac{C \log(M + |\mathcal{F}|)}{(\rho - \eta)^2}$$

with universal $C > 0$, there is a rank-r router with $\|u_i\|_2 \le B_u$ realizing every $S \in \mathcal{F}$ at some input with $\|R_2 x_S\|_2 \le B_z$ and margin at least $\eta B_u B_z$.

Proof. We first solve the problem in the uncompressed regime $r = M$, then use a Johnson–Lindenstrauss projection to compress down to $r = O(\log(M + |\mathcal{F}|)/(\rho - \eta)^2)$ while preserving the inner products that determine routing.

To begin, let $e_1, \dots, e_M$ be the standard basis of $\mathbb{R}^M$, which serve as expert vectors. For each $S \in \mathcal{F}$, define

$$z_S \;:=\; \frac{1}{\rho}\left(\frac{\mathbf{1}_S}{k} - \frac{\mathbf{1}_{[M]\setminus S}}{M-k}\right).$$

A direct computation gives

$$\|z_S\|_2^2 \;=\; \frac{1}{\rho^2}\Big(k \cdot \tfrac{1}{k^2} + (M-k) \cdot \tfrac{1}{(M-k)^2}\Big) \;=\; \frac{1}{\rho^2}\Big(\tfrac{1}{k} + \tfrac{1}{M-k}\Big) \;=\; 1.$$

For $i \in S$ and $j \notin S$,

$$\langle e_i, z_S\rangle \;=\; \frac{1}{k\rho}, \qquad \langle e_j, z_S\rangle \;=\; -\frac{1}{(M-k)\rho}, \qquad \langle e_i, z_S\rangle - \langle e_j, z_S\rangle \;=\; \rho.$$

So in dimension M, the trivial choice of expert vectors e_i realizes every $S \in \mathcal{F}$ with margin exactly ρ, which is strictly larger than the target η.

We now compress this construction via Johnson–Lindenstrauss. Let $\mathcal{V} = \{e_1, \dots, e_M\} \cup \{z_S : S \in \mathcal{F}\}$, so $|\mathcal{V}| \le M + |\mathcal{F}|$. By the Johnson–Lindenstrauss lemma, for $r = O(\log|\mathcal{V}|/\varepsilon^2)$ there is a linear map $\Phi : \mathbb{R}^M \to \mathbb{R}^r$ preserving squared Euclidean distances on $\mathcal{V} \cup \{0\}$ multiplicatively to within $(1 \pm \varepsilon)$, i.e. $(1-\varepsilon)\|u - v\|_2^2 \le \|\Phi u - \Phi v\|_2^2 \le (1+\varepsilon)\|u - v\|_2^2$ for all $u, v \in \mathcal{V} \cup \{0\}$. Since every $u \in \mathcal{V}$ has unit norm, $\|u - v\|_2^2 \le 4$ for all such pairs, so this is equivalent to additive preservation up to $O(\varepsilon)$; absorbing the constant, we assume preservation up to additive error ε in what follows. The polarization identity

$$\langle u, v\rangle \;=\; \tfrac{1}{2}\big(\|u\|_2^2 + \|v\|_2^2 - \|u - v\|_2^2\big)$$

then preserves all pairwise inner products of vectors in $\mathcal{V}$ up to additive error $O(\varepsilon)$, and absorbing the constant we may assume preservation up to additive ε. Choose

$$\varepsilon \;:=\; c_0\,(\rho - \eta)$$

for a sufficiently small universal constant $c_0 > 0$, which gives $r = O\big(\log(M + |\mathcal{F}|)/(\rho - \eta)^2\big)$.

To extract the rank-r router from Φ, define normalized compressed vectors

$$\tilde{u}_i \;:=\; \frac{\Phi e_i}{\|\Phi e_i\|_2}, \qquad \tilde{z}_S \;:=\; \frac{\Phi z_S}{\|\Phi z_S\|_2},$$

and set $u_i = B_u \tilde{u}_i$, $\bar{z}_S = B_z \tilde{z}_S$. The norm bounds are immediate. Since normalization changes the preserved inner products by at most another constant multiple of ε, for every $i \in S$ and $j \notin S$,

$$\langle u_i, \bar{z}_S\rangle - \langle u_j, \bar{z}_S\rangle = B_u B_z(\langle \tilde{u}_i, \tilde{z}_S\rangle - \langle \tilde{u}_j, \tilde{z}_S\rangle) \ge B_u B_z(\rho - C'\varepsilon) \ge \eta B_u B_z$$

for a universal constant $C' > 0$, by the choice of c_0. Hence every $S \in \mathcal{F}$ is realized with margin at least $\eta B_u B_z$.

Finally, we lift to input space: since $h \ge r$, set $R_2 = [I_r \mid 0] \in \mathbb{R}^{r \times h}$ and choose x_S whose first r coordinates are $\bar{z}_S$, so that $R_2 x_S = \bar{z}_S$. This realizes the desired router with the claimed parameters. □

Proof of Theorem 1. Apply Theorem 4 with $B_u = B_z = 1$ and target margin $\eta = c/\sqrt{k}$ for a sufficiently small universal constant $c > 0$. Since $\rho > 1/\sqrt{k}$, we have $(\rho - \eta)^2 = \Omega(1/k)$, and therefore $r = O(k \log(M + |\mathcal{F}|))$. The constructed margin is $\eta = \Omega(1/\sqrt{k})$, as claimed. □

Proof of Corollary 1. For $\mathcal{F} = \binom{[M]}{k}$, Stirling's bounds $(M/k)^k \le |\mathcal{F}| \le (eM/k)^k$ give

$$\log |\mathcal{F}| \;=\; \Theta(k \log(M/k)).$$

Since $M \le |\mathcal{F}|$ for $1 \le k \le M-1$, we also have $\log(M + |\mathcal{F}|) = \Theta(\log |\mathcal{F}|) = \Theta(k \log(M/k))$.

Upper bound. By Theorem 1, any rank $r \ge Ck \log(M + |\mathcal{F}|)$ admits the desired router at margin $\Omega(1/\sqrt{k})$. Substituting $\log(M + |\mathcal{F}|) = \Theta(k \log(M/k))$ yields rank $r = O(k^2 \log(M/k))$ as a sufficient condition. This margin is resolvable under $p = \Omega(\log k)$ precision.

Lower bound. At p-bit top-k precision, $\eta = \Omega(2^{-p})$ implies $\log(1 + 2/\eta) = O(p)$. Theorem 2 then gives

$$r \;\ge\; \frac{\log |\mathcal{F}|}{\log(1 + 2/\eta)} \;=\; \Omega\!\left(\frac{k \log(M/k)}{p}\right).$$

□

Proposition 1 (restated). Let $x_1, \dots, x_n \sim \mathcal{N}(0, I_h)$ be independent, and suppose $M = 2^r$ with $h \ge r$. There exists a rank-r top-1 router with M experts such that each example is routed uniformly over the M experts. Consequently, if N_a is the number of examples routed to expert a, then with probability at least $1 - \delta$,

$$\max_{a \in [M]} \left| N_a - \frac{n}{M} \right| \le \sqrt{\frac{n}{2} \log \frac{2M}{\delta}}.$$

Proof of Proposition 1. The construction follows the routing step in the Gaussian memorization argument of Jelassi et al. (2025). Let $R_2 \in \mathbb{R}^{r \times h}$ select the first r coordinates, so $z_i := R_2 x_i \sim \mathcal{N}(0, I_r)$. Index the $M = 2^r$ experts by sign vectors $a \in \{\pm 1\}^r$, and set the corresponding expert vector to $u_a = a/\sqrt{r}$. The router score for expert a is

$$\langle u_a, z_i \rangle = \frac{1}{\sqrt{r}} \sum_{\ell=1}^{r} a_\ell (z_i)_\ell.$$

For a fixed z_i, this score is maximized coordinate-wise by choosing $a_\ell = \operatorname{sign}((z_i)_\ell)$ for every ℓ. Thus the selected expert is exactly the sign pattern of the projected Gaussian vector z_i.

Since the coordinates of z_i are independent centered Gaussians, $\operatorname{sign}(z_i)$ is uniform on $\{\pm 1\}^r$. Therefore each example is routed uniformly over the M experts, and the counts $(N_a)_{a \in [M]}$ are multinomial with cell probability $1/M$. For any fixed expert a, Hoeffding's inequality (Hoeffding, 1963) gives

$$\Pr\!\left(\left| N_a - \frac{n}{M} \right| > t \right) \le 2 \exp\!\left(-\frac{2t^2}{n} \right).$$

Taking a union bound over the M experts and setting $t = \sqrt{\frac{n}{2} \log \frac{2M}{\delta}}$ gives the claimed bound.

This proposition isolates only the routing step. The dependence on n enters through the number of examples assigned to each expert, approximately n/M; the router rank only needs to distinguish the M expert labels, which requires $r = \log_2 M$ in this construction. The expert MLPs then memorize their assigned examples, as in the memorization upper bound of Jelassi et al. (2025). □