BMoE: Sharing Expert Cores, Preserving Layer-Specific Transformations

Anonymous authors
Paper under double-blind review

Abstract

Sparse mixture-of-experts (MoE) models activate few experts per token, yet must store the entire expert bank. Cross-depth sharing reduces this storage, but also asks shared experts to serve representations from different layers. We propose BMoE, a block-wise MoE that separates shared expert capacity from layer-specific transformations. Contiguous layers reuse a pool of latent expert cores, while learned input/output interfaces, routers, and attention remain layer-specific. Each depth can therefore access the same nonlinear cores through different learned maps. We evaluate eight 36-effective-layer configurations trained for approximately 208B scheduled token positions with matched selected-expert computation. At nearly equal total parameters (7.063B versus 7.056B), two-layer BMoE reduces WikiText-103 perplexity from 17.867 to 17.031 and raises the nine-task mean from 50.74 to 52.10 relative to a per-layer latent MoE. Four-layer BMoE improves all four aggregate metrics with 43.2% fewer parameters. It also outperforms whole-layer looping at matched expert storage. Tying the latent interfaces saves only 1.6% of parameters but degrades every aggregate metric; forward diagnostics show differentiated routing within shared pools. Together, these results show that the boundary of depth sharing matters: expert cores can be efficiently reused across depth while layer-specific transformations preserve flexible access to the shared capacity.

1 Introduction

Sparse mixture-of-experts (MoE) models increase capacity without evaluating every parameter for every token: a router selects a small subset of experts at each layer (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022; Dai et al., 2024; Ludziejewski et al., 2024). This computational sparsity does not remove the cost of storing the unselected experts. With a separate expert bank at every depth, total expert storage grows with both the number of layers and the number of candidates per layer. The resulting design problem is not just how much capacity to activate, but how to make a limited amount of stored capacity useful throughout the network.

Cross-depth parameter reuse offers a direct response. Universal Transformers and ALBERT established influential forms of depth sharing (Dehghani et al., 2019; Lan et al., 2020), and recent looped models reuse computation for latent reasoning and test-time scaling (Saunshi et al., 2025; Geiping et al., 2025; Zhu et al., 2025). In an MoE, however, repeating an entire layer binds expert reuse to the reuse of attention, routing, and the transformations surrounding the experts. This saves parameters, but it also constrains how each depth can use them. *Must reusing an expert core mean reusing the same layer-level transformation?*

Our starting point is a distinction between shared capacity and depth-specific access to that capacity. Even when hidden states have the same width, different layers need not present the same feature distributions or require the same residual updates. An expert shared directly across depth must accommodate these demands with one parameter set. We hypothesize that retaining learned, layer-specific input/output maps can ease this constraint: each layer can transform its representation before expert computation and transform the result back afterward.

We introduce BMoE, a block-wise MoE that shares *latent expert cores* while preserving *layer-specific interfaces*. A contiguous block owns one pool of nonlinear experts operating at a latent

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

width $\ell < d$. Each layer has its own down-projection, up-projection, and post-aggregation normalization, as well as its own attention and full-width router. The shared core supplies reusable capacity; the interfaces determine how that capacity is used at each depth. Usage balancing is applied over the layers sharing a pool, allowing different layers to select different subsets of experts.

This organization introduces two complementary storage choices. For L layers, block size S, and N candidates per pool, the model stores $T = (L/S)N$ expert instances but still selects only K experts per layer and token. At fixed N, larger blocks reduce stored expert capacity without shrinking the candidate pool. At fixed T, larger blocks allow more candidates per layer. Thus, cross-depth sharing converts storage duplicated across layers into a richer per-layer candidate pool: the same expert budget can offer more choices at each depth without increasing the number of selected experts. Alternatively, sharing can reduce storage while retaining the candidate pool.

We test both uses in eight 36-effective-layer models trained with a common recipe for approximately 208B scheduled token positions. At nearly equal total parameter count, two-layer BMoE improves the nine-task mean by 1.36 points over a per-layer latent MoE. Four-layer BMoE improves the mean by 0.83 points with 43.2% fewer total parameters, and also improves training loss and both validation perplexities. Against whole-layer looping at the same expert-instance count, it improves all four aggregate metrics with 6.5% more total parameters. Crucially, tying the interfaces within a BMoE block saves only 1.6% of parameters but loses 1.27 points on the task mean.

Our contributions are:

- **Turning duplicated storage into richer expert access.** BMoE uses cross-depth sharing to enlarge the candidate pool available at each layer under a fixed expert-core budget, or to reduce storage at a fixed pool size, while keeping selected-expert computation unchanged.
- **Separating shared capacity from depth-specific transformations.** Block-owned latent expert cores are accessed through layer-specific projections and post-aggregation normalization, with private attention and routers. These interfaces preserve depth-specific use at a small parameter cost.
- **Demonstrating the value of the sharing boundary.** Experiments show better quality at nearly equal parameters and better aggregate quality with 43.2% fewer parameters. Interface ablations and forward diagnostics show the benefit of retaining layer-specific transformations and differentiated routing within shared pools.

2 Related Work

Depth sharing and recursive computation. Universal Transformers repeatedly apply a shared Transformer computation (Dehghani et al., 2019), while ALBERT uses cross-layer sharing to reduce stored parameters (Lan et al., 2020). Theoretical and empirical studies connect looped depth to reasoning and inference-time compute (Saunshi et al., 2025; Geiping et al., 2025). Relaxed Recursive Transformers and RingFormer add layer-dependent low-rank or adaptive signals to recursive blocks (Bae et al., 2025a; Heo et al., 2025); Mixture-of-Recursions learns token-level recursion depths (Bae et al., 2025b). BMoE changes the sharing boundary inside the MoE branch: depth-specific maps surround a shared latent core, while attention and routing remain untied. Its goal is parameter-storage efficiency at a fixed effective depth, not adaptive recursion or an inference-time depth-scaling policy.

MoE recurrence and expert sharing. Early sparse-gating work established conditional expert computation (Shazeer et al., 2017); GShard and Switch Transformer demonstrated large-scale routed training with expert parallelism and simplified routing (Lepikhin et al., 2021; Fedus et al., 2022). GLaM and DeepSpeed-MoE further studied scaling and systems support for sparse expert models (Du et al., 2022; Rajbhandari et al., 2022). MoEUT combines MoE computation with Universal Transformers (Csordás et al., 2024), while Loop the Loopies studies repeated layer execution in MoE language models (Gao et al., 2026). Tying the Loop is the closest precedent for separating expert sharing from attention and router sharing (Jaggi, 2026); it also studies consecutive-layer groups and normalization sharing. BMoE focuses on the combination of block-owned latent expert pools, layer-specific latent interfaces, and pool-level balancing, and evaluates how these choices trade off

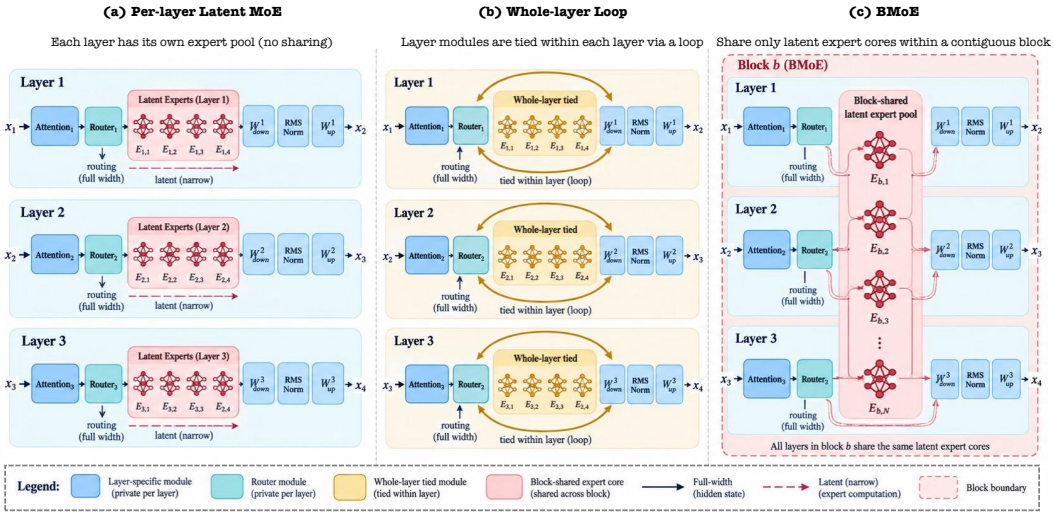

Figure 1: Sharing boundaries compared in this paper. BMoE shares only the latent routed expert pool within a contiguous block. The attention, router, and latent interfaces remain layer-specific. The diagram shows parameter ownership, not equality of total FLOPs.

stored capacity and layer-accessible candidates. Our loop baseline is a whole-layer repeated-call implementation under the common backbone.

Latent and fine-grained experts. LatentMoE moves routed expert computation to a lower-dimensional space (Elango et al., 2026); Kimi K3 describes related normalization choices for latent MoE computation (Kimi Team, 2026). DeepSeekMoE and subsequent scaling analyses study expert granularity, specialization, and the allocation of expert capacity (Dai et al., 2024; Ludziejewski et al., 2024; Zhou et al., 2022; Lewis et al., 2021). BMoE adds a cross-depth ownership axis to this design space. The latent interfaces are not only a width-reduction device: leaving them untied allows each depth to use the same expert cores through different maps. Our common backbone uses full-width routing and latent expert computation. All main baselines also use latent experts, so the comparisons concern sharing and allocation, rather than latent computation versus full-width MoE.

3 BMoE: Shared Cores with Layer-Specific Interfaces

Design objective. We seek to reduce stored expert parameters without requiring different depths to use a shared core through the same input/output maps. BMoE separates three choices: which layers own a pool (S), how many experts each layer can access (N), and which interfaces remain private.

3.1 Block-shared latent expert pools

Let $x_i \in \mathbb{R}^d$ be the input at effective layer i. Partition the L layers into $G = L/S$ contiguous blocks. Block b owns $\mathcal{E}_b = \{E_{b,1}, \ldots, E_{b,N}\}$, where each $E_{b,k} : \mathbb{R}^\ell \to \mathbb{R}^\ell$ and $\ell = d/\alpha$. Layer i owns its attention, router, down-projection $W_i^{\downarrow} \in \mathbb{R}^{\ell \times d}$, up-projection $W_i^{\uparrow} \in \mathbb{R}^{d \times \ell}$, and the post-aggregation normalization in the default configuration. With backbone normalizations shown explicitly, the branch is

$$x'_i = x_i + \mathrm{Attn}_i(\mathrm{RMSNorm}_i^{\mathrm{in}}(x_i)), \tag{1}$$

$$h_i = \mathrm{RMSNorm}_i^{\mathrm{post}}(x'_i), \qquad (\mathcal{K}_i, w_i) = \mathrm{Router}_i(h_i), \tag{2}$$

$$z_i = W_i^{\downarrow} h_i, \qquad u_i = \sum_{k \in \mathcal{K}_i} w_{i,k} E_{b(i),k}(z_i), \tag{3}$$

$$y_i = W_i^{\uparrow}\, \mathrm{RMSNorm}_i(u_i), \qquad x_{i+1} = x'_i + y_i. \tag{4}$$

Routing and the down-projection use the post-attention normalized representation h_i. Routing is full-width, whereas the selected experts operate in the latent width. The normalization is applied

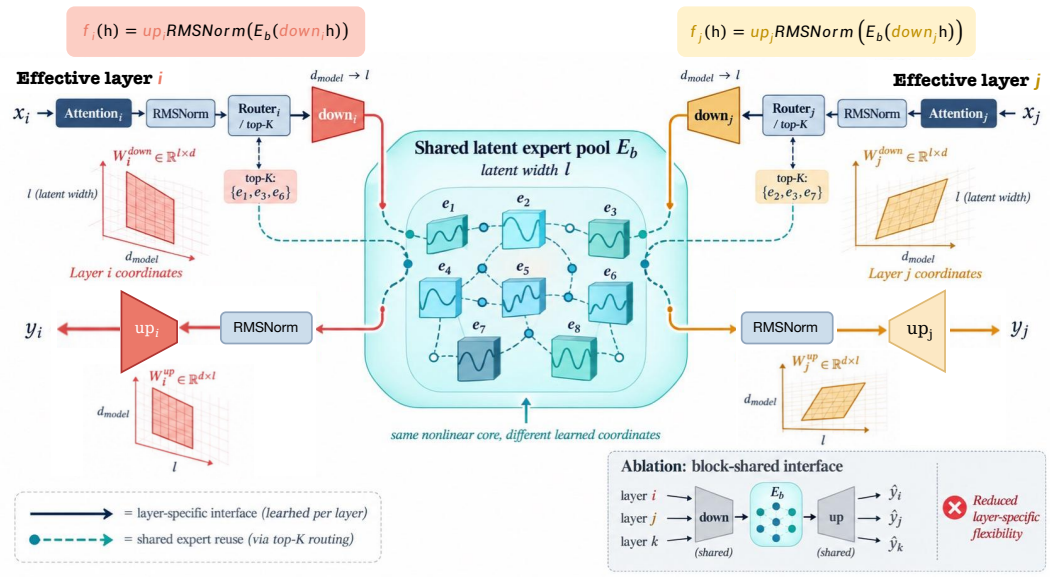

Figure 2: Layer-specific latent interfaces let different depths wrap the same expert core with different learned maps.

after weighted expert aggregation, before the up-projection. Attention additionally uses query/key RMSNorm, and the backbone ends with a final RMSNorm.

The pool stores $T = GN$ expert instances. Every effective layer accesses N candidates and executes K experts per token. When $S = 1$, BMoE reduces to a per-layer latent MoE. The term "shared expert" here means a routed expert reused across depth; it does not refer to the always-active shared expert terminology used in some MoE systems (Dai et al., 2024).

3.2 Layer-specific interfaces

A shared core need not define the same effective map. We define the latent interface as the down-projection, post-aggregation RMSNorm, and up-projection together. The complete routed branch is

$$F_i(h) = W_i^{\uparrow}\,\mathrm{RMSNorm}_i\left(\sum_{k\in\mathcal{K}_i(h)} w_{i,k}(h)E_{b(i),k}(W_i^{\downarrow}h)\right). \tag{5}$$

Normalization follows weighted expert aggregation, as in Equation 4. Layer-specific projections and normalization let the same stored cores operate through different coordinates and scales, making the interface a learned source of depth conditioning. Even with identical selected cores, the input directions, output directions, and evaluation points can vary by layer (Appendix **??**). Attention and routing also remain depth-dependent.

Why use blocks? Contiguous sharing provides a tunable scope between private pools ($S = 1$) and a single global pool ($S = L$). Smaller blocks limit how many depths must use the same cores; larger blocks save more expert storage. The block size controls this storage–quality trade-off.

3.3 Block-level balancing

Sharing changes the natural unit over which expert usage should be regularized. Let W be the number of token positions in a statistics window, let $p_{i,t,k}$ be the router probability, and let $\mathbb{1}\{k \in \mathcal{K}_{i,t}\}$ denote top-K selection. We aggregate over the S layers in a block: $\mathcal{L}_{\mathrm{bal}} = \lambda_{\mathrm{bal}}\frac{N}{KG}\sum_{b=1}^{G}\sum_{k=1}^{N} f_{b,k}P_{b,k}$, where $f_{b,k} = \frac{1}{SW}\sum_{i\in b}\sum_{t=1}^{W}\mathbb{1}\{k \in \mathcal{K}_{i,t}\}$ and $P_{b,k} = \frac{1}{SW}\sum_{i\in b}\sum_{t=1}^{W} p_{i,t,k}$. Here $\sum_k f_{b,k} = K$. The factor N/K normalizes the contribution of each block under uniform top-K usage, while $1/G$ averages the contributions across blocks. This is the

standard load-balancing normalization adapted to block-aggregated top-K statistics (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022). The objective balances usage over the layers that share a pool while leaving room for complementary preferences at individual depths. The auxiliary coefficients are listed in Appendix A. We use RMSNorm after aggregation to reduce the sensitivity of the up-projection to changes in the scale of the aggregated latent branch (Zhang & Sennrich, 2019; Kimi Team, 2026).

3.4 Storage and compute accounting

Ignoring small normalization terms, the dominant expert storage is proportional to GN. For a bias-free three-matrix gated expert with intermediate width h, the leading parameter and per-call matrix multiplication counts are

$$P_{\mathrm{BMoE}} \simeq P_{\mathrm{fixed}} + \underbrace{3(L/S)N\ell h}_{\text{shared expert cores}} + \underbrace{L(2d\ell + dN)}_{\text{private interfaces and routers}}, \tag{6}$$

$$F_{\mathrm{MoE}} \simeq 4d\ell + 6K\ell h + 2dN, \tag{7}$$

where a multiply-add counts as two FLOPs. P_{fixed} includes attention, embeddings, and the output head. The FLOP expression omits selection, dispatch, nonlinearities, and normalization. At fixed effective depth and latent width, increasing S reduces the number of distinct pools, while increasing N enlarges the candidate pool available at every layer. The selected-expert arithmetic is unchanged by S and is largely independent of unused candidates. Increasing N does add router parameters and router computation. Our experiments therefore match selected-expert computation; end-to-end system costs are outside this accounting. The parameter expression describes the default BMoE, not whole-layer looping, which also ties non-expert parameters.

At fixed expert-instance budget T, the identity $N = TS/L$ explains how sharing can enlarge each layer's candidate pool without adding expert-core weights. At fixed N, the expert term decreases as $1/S$, but private modules impose a storage floor. In particular, tying the down/up projections in a block saves only $2(L - L/S)d\ell$ weights, plus small normalization terms, without removing the projection operations themselves. Finally, at a common weight precision, parameter counts translate directly into unsharded weight bytes; they do not determine peak device memory, which also depends on activations, optimizer states, sharding, and execution schedules.

4 Experimental Setup

Questions and comparison scope. The experiments ask whether shared pools improve quality at a fixed expert budget, whether they retain quality as storage decreases, and whether layer-specific interfaces matter. Table 1 separates expert-instance count from total parameters, including the non-expert modules retained by each sharing scheme.

Common backbone. All configurations use 36 effective layers, residual width $d = 1536$, latent width $\ell = 768$, MoE intermediate width 576, top-$K = 8$, 24 attention heads, 3 KV heads, head dimension 128, vocabulary size 151,936, and context length 4,096. Every token receives 36 MoE calls. The full configuration is in Appendix A. Reported activated parameters, counted per invocation rather than as unique stored weights, span 1.323–1.330B, a spread of about 0.5%.

Baselines and ablations. Per-Layer MoE owns a separate latent expert pool at every layer. Loop-Transformer (2×18) has 18 physical layers, each executed twice consecutively, and reuses the whole layer on the second pass. BMoE-S4-N256+SP shares the latent projections and their associated post-aggregation normalization within each four-layer block. The S and N scans vary storage under the same effective depth and selected-expert computation.

Training and evaluation. All runs use the same data order, bf16, AdamW, a global batch of 4,608 sequences of length 4,096, and a constant learning rate of 3.36×10^{-4} after a 500-step warmup. We report the common step-11,000 checkpoint, corresponding to approximately 208B scheduled token positions. The reported training loss is the standard autoregressive language-modeling cross-entropy. Perplexities are computed on the full WikiText-103 and PTB validation sets using non-overlapping 1,024-token windows on WikiText-103 (Merity et al., 2016) and PTB (Marcus et al., 1993). Downstream scores use lm-eval-harness 0.4.12 (Gao et al., 2023): MMLU and WinoGrande

use 5-shot, HellaSwag 10-shot, ARC-Challenge 25-shot, and the remaining tasks 0-shot. We compute unweighted means and mean-score differences from the nine task scores displayed in Appendix B, rounding only the final reported values to two decimal places. Length-normalized accuracy is used for ARC, HellaSwag, PIQA, and OpenBookQA.

Table 1: Configurations. G is the number of distinct expert pools, S the number of effective layers per pool, N the pool size, and $T = GN$ the number of stored expert instances. All runs use 36 effective layers.

Model	Physical	G	S	N	T	Params (B)
Per-Layer MoE	36	36	1	128	4608	7.056
BMoE-S2-N256	36	18	2	256	4608	7.063
BMoE-S4-N256	36	9	4	256	2304	4.006
Loop-Transformer (2×18)	18	18	2	128	2304	3.762
BMoE-S4-N256+SP	36	9	4	256	2304	3.942
BMoE-S6-N256	36	6	6	256	1536	2.987
BMoE-S4-N192	36	9	4	192	1728	3.238
BMoE-S4-N128	36	9	4	128	1152	2.470

5 Main Results

Sharing admits two useful operating points: S2 improves quality at nearly unchanged storage, while S4 improves aggregate quality with substantially fewer parameters.

5.1 Better quality at nearly equal storage

The closest parameter-matched comparison pairs BMoE-S2-N256 with Per-Layer MoE. The two models have the same number of stored expert instances, nearly identical total parameter counts (7.063B versus 7.056B), and the same effective depth and selected-expert computation. BMoE-S2-N256 improves training loss by 0.0566, WikiText-103 perplexity by 0.836, PTB perplexity by 1.276, and the downstream mean by 1.36 points (Table 2). By replacing separate 128-expert pools with shared 256-expert pools, BMoE gives each layer twice as many candidates under the same expert-core storage budget.

Table 2: Main comparison across depth-sharing architectures. All checkpoints are evaluated at step 11,000. Lower is better for loss and perplexity; higher is better for the nine-task mean. Loss is the standard language-modeling loss. Bold marks the better value within each stated comparison.

Comparison	Model	Params (B)	Loss	Wiki PPL	PTB PPL	Mean
$T = 4608$	BMoE-S2-N256	7.063	**2.0499**	**17.031**	**37.119**	**52.10**
	Per-Layer MoE	7.056	2.1065	17.867	38.395	50.74
$T = 2304$	BMoE-S4-N256	4.006	**2.0972**	**17.313**	**36.799**	**51.57**
	Loop-Transformer (2×18)	3.762	2.1066	17.670	39.900	50.63
Interface	BMoE-S4-N256	4.006	**2.0972**	**17.313**	**36.799**	**51.57**
	BMoE-S4-N256+SP	3.942	2.1158	18.033	39.281	50.29

Better aggregate quality with 43.2% fewer parameters. BMoE-S4-N256 reduces total parameters from 7.056B to 4.006B while improving loss and both validation perplexities. The nine-task mean rises from 50.74 to 51.57. It therefore improves on the per-layer baseline using approximately 57% of its weight storage at the same precision.

5.2 Selective sharing versus whole-layer looping

At $T = 2304$, BMoE-S4-N256 is better on all four aggregate metrics than Loop-Transformer (2×18). The differences are 0.0094 in loss, 0.357 in WikiText-103 perplexity, 3.101 in PTB perplexity, and 0.94 percentage points in the downstream mean. PTB perplexity (a 7.8% reduction) and

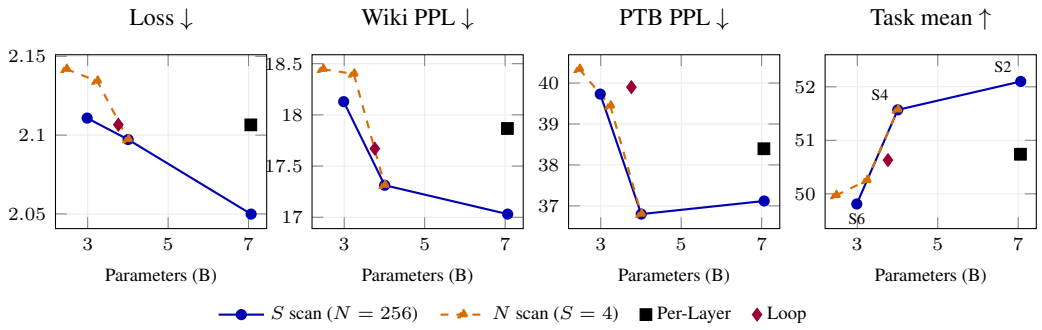

Figure 3: Measured storage–quality trade-offs under the common effective depth and selected-expert-computation protocol. S4 improves all four aggregate metrics over Per-Layer MoE using approximately 57% of its parameters. Lines connect the measured scan points. Loss-window conventions are given in Table 3.

the downstream mean provide the clearest gains. BMoE-S4-N256 uses 4.006B parameters versus 3.762B, an increase of 6.5%, and exposes 256 rather than 128 candidates per call. It also retains 36 attention/router/interface sets rather than 18. At matched expert-instance count, the results favor sharing expert cores while retaining layer-specific transformations over repeating whole layers.

5.3 The storage–quality trade-off

At $N = 256$, increasing S from 2 to 4 to 6 reduces parameters from 7.063B to 4.006B to 2.987B (Table 3). Loss and WikiText perplexity worsen, and the task mean decreases; PTB perplexity instead attains its minimum at S4. Unlike S4, S6 trails Per-Layer MoE on all four aggregate metrics.

At $S = 4$, all four reported aggregates improve as N increases from 128 to 192 to 256, with total parameters increasing from 2.470B to 3.238B to 4.006B. Loss for the smaller pools uses a short reporting window around step 11,000.

Table 3: Parameter-budget scans at matched selected-expert computation. The scans share the S4/$N = 256$ point. Loss for $N = 128, 192$ is averaged over step $11{,}000 \pm 100$; other losses are recorded at step 11,000.

S	N	T	Params (B)	Loss	Wiki PPL	PTB PPL	Mean
2	256	4608	7.063	2.0499	17.031	37.119	52.10
4	256	2304	4.006	2.0972	17.313	36.799	51.57
6	256	1536	2.987	2.1108	18.129	39.731	49.81
4	128	1152	2.470	2.1418	18.446	40.338	49.97
4	192	1728	3.238	2.1343	18.400	39.443	50.25

Moderate sharing is favorable; more sharing is not always better. Increasing S from 2 to 4 at $N = 256$ saves 43.3% of parameters for a loss increase of 0.0473 and a downstream decrease of 0.53 points; PTB perplexity improves. In contrast, reducing N from 256 to 192 at $S = 4$ saves only 19.2%, but increases loss by 0.0371, WikiText perplexity by 1.087, and PTB perplexity by 2.644, while reducing the task mean by 1.31 points. The measured trade-off therefore favors preserving the larger candidate pool when moving from two- to four-layer sharing. This preference does not hold uniformly beyond $S = 4$: S6 (2.987B) beats S4-N192 (3.238B) on loss and WikiText perplexity, but trails it on PTB perplexity and the task mean.

5.4 Individual downstream benchmarks

The full task breakdown is in Table 6, with additional pool-size runs in Table 7. S2 and S4 have the two highest nine-task means. Relative to Per-Layer MoE, S2 improves OpenBookQA from 33.60 to

36.20 and SciQ from 87.10 to 88.80; S4 improves LAMBADA from 36.19 to 39.03. These results complement the language-modeling gains with improvements across downstream evaluations.

6 What Do Layer-Specific Interfaces Preserve?

The feature-space motivation suggests two observable checks: removing layer-specific interfaces should hurt quality, and sharing cores should not force layers to repeat their routing behavior. The first is tested by an architectural ablation; the second by forward diagnostics.

6.1 Small interfaces, consistent quality gains

BMoE-S4-N256 and BMoE-S4-N256+SP have the same effective depth, block size, pool size, and expert-instance count. The ablation ties the latent interface—the down/up projections and post-aggregation normalization—within each block. Layer-specific interfaces improve loss by 0.0186, WikiText perplexity by 0.720, PTB perplexity by 2.482, and the downstream mean by 1.27 points, at a cost of 0.064B parameters (about 1.6% of BMoE-S4-N256). This result highlights the value of preserving layer-specific interfaces around shared expert cores.

6.2 Shared storage does not require repeated selection

We compare adjacent calls inside a sharing unit with adjacent calls across units on held-out WikiText-103 windows. We report row-wise cosine similarity of attention probability matrices, top-K expert-set overlap, and Jensen–Shannon divergence of routing distributions. For the loop, a unit is the two calls to one physical layer. For BMoE, a unit is the S-layer block.

Table 4: Within-unit and cross-unit behavior. Values are token averages. Lower top-K overlap and higher routing JS indicate more differentiated routing. Cross-unit expert indices are only a scale reference because different pools need not have aligned semantics.

Model	Attn cos		Top-K overlap		Routing JS	
	in	cross	in	cross	in	cross
Loop-Transformer (2×18)	**0.892**	0.427	**0.816**	0.062	**0.016**	0.228
BMoE-S2-N256	0.476	0.477	0.013	0.032	0.278	0.262
BMoE-S4-N256	0.494	0.493	0.019	0.031	0.281	0.278
BMoE-S6-N256	0.502	0.486	0.022	0.036	0.274	0.286
BMoE-S4-N256+SP	0.400	0.438	0.025	0.029	0.255	0.293
Per-Layer MoE	–	0.522	–	0.061	–	0.292

The loop's two calls to the same physical layer have attention cosine 0.892 and top-K overlap 0.816, compared with 0.427 and 0.062 across adjacent physical layers. BMoE's within-block top-K overlap is 0.013–0.022, comparable to or below its cross-block reference values and far below the loop's within-unit overlap. Its within-block attention cosine is also close to its cross-block value. BMoE thus preserves strongly differentiated routing behavior despite sharing the same expert pool across depth.

Raw overlap also depends on pool size and expert-use marginals; cross-pool comparisons serve as descriptive references.

Tying the interfaces raises within-block overlap from 0.019 to 0.025, but the absolute overlap remains low. The quality degradation is thus not accompanied by loop-like repeated expert selection. Differentiated routing alone is therefore insufficient to recover the quality of layer-specific interfaces.

6.3 Activation-scale statistics

We measure token-averaged residual and MoE-branch RMS at first, middle, and final effective calls, following invocation order for the loop (Table **??**). S4 has final residual RMS 1.274 and branch last/first ratio 1.16; tying its interfaces raises these to 1.924 and 1.82. The loop has values 1.939

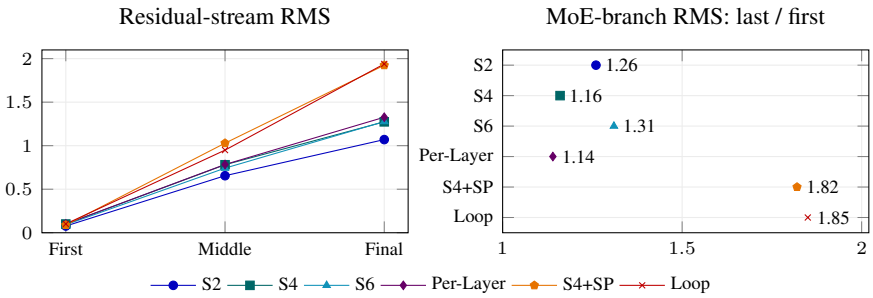

Figure 4: Activation-scale statistics by effective invocation depth. The loop and block-shared interfaces exhibit larger final residual scales and MoE-branch last/first RMS ratios than the other configurations. Lines connect only the three sampled depths, not a full depth trajectory.

and 1.85. Both tightly tied configurations thus have larger scales than the remaining models, whose branch ratios span 1.14–1.31.

Interpretation. The quality gains from layer-specific interfaces, differentiated routing within shared pools, and the activation-scale observations support a common design choice: reuse expert capacity while preserving the projection-geometry and weight-distribution statistics are reported in transformations through which each depth accesses it.

7 Conclusion

BMoE separates what is expensive to store from what makes an expert transformation layer-specific. Blocks reuse latent expert cores; each layer retains its own interfaces, router, and attention. In the tested 36-layer family, this yields improved quality at nearly equal parameter count and improved aggregate quality with 43.2% fewer parameters. Sharing the interfaces as well removes only 1.6% of parameters but consistently reduces aggregate quality. The practical lesson is that small, depth-specific interfaces can be worth preserving when the much larger expert cores are reused.

Cross-depth sharing is therefore more than a compression mechanism: it can turn depth-wise duplicated storage into a larger candidate pool at each layer. The block-size and pool-size scans expose two useful operating points, while the forward diagnostics show that shared pools retain differentiated routing across depth. Together, these findings highlight the importance of the sharing boundary: *share the expensive expert cores, preserve layer-specific transformations, and use the saved storage for richer expert access*. Future work will examine larger model scales and end-to-end system efficiency.

REFERENCES

Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise lora. In *International Conference on Learning Representations*, 2025a. doi: 10.48550/arXiv.2410.20672.

Sangmin Bae, Yujin Kim, Reza Bayat, et al. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. In *Advances in Neural Information Processing Systems*, 2025b. doi: 10.48550/arXiv.2507.10524.

Róbert Csordás, Kazuki Irie, Jürgen Schmidhuber, Christopher Potts, and Christopher D. Manning. MoEUT: Mixture-of-experts universal transformers. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0897.

Damai Dai et al. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 1280–1297, 2024. doi: 10.18653/v1/2024.acl-long.70.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. doi: 10.48550/arXiv.1807.03819.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten Bosma, Zongwei Zhou, Tao Wang, Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V. Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. GLaM: Efficient scaling of language models with mixture-of-experts. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 5547–5569. PMLR, 2022. URL https://proceedings.mlr.press/v162/du22c.html.

Venmugil Elango, Nidhi Bhatia, Roger Waleffe, Rasoul Shafipour, Tomer Asida, Abhinav Khattar, Nave Assaf, Maximilian Golub, Joey Guman, Tiyasa Mitra, Ritchie Zhao, Ritika Borkar, Ran Zilberstein, Mostofa Patwary, Mohammad Shoeybi, and Bita Rouhani. LatentMoE: Toward optimal accuracy per FLOP and parameter in mixture of experts. *arXiv preprint arXiv:2601.18089*, 2026. doi: 10.48550/arXiv.2601.18089.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. URL https://jmlr.org/papers/v23/21-0998.html.

Leo Gao et al. A framework for few-shot language model evaluation, 2023. URL https://github.com/EleutherAI/lm-evaluation-harness. Experiments use version 0.4.12.

Zitian Gao, Yilong Chen, Yihao Xiao, Xinyu Yang, Ran Tao, Joey Zhou, and Bryan Dai. Loop the loopies! *arXiv preprint arXiv:2607.16051*, 2026. doi: 10.48550/arXiv.2607.16051.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *Advances in Neural Information Processing Systems*, 2025. doi: 10.48550/arXiv.2502.05171.

Jaemu Heo, Eldor Fozilov, Hyunmin Song, and Taehwan Kim. RingFormer: Rethinking recurrent transformer with adaptive level signals. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 21675–21686, 2025. doi: 10.18653/v1/2025.findings-emnlp.1182.

Martin Jaggi. Tying the loop – tied expert layers in mixture-of-experts language models. *arXiv preprint arXiv:2606.16825*, 2026. doi: 10.48550/arXiv.2606.16825.

Kimi Team. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026. doi: 10.48550/arXiv.2607.24653.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. doi: 10.48550/arXiv.1909.11942.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021. doi: 10.48550/arXiv.2006.16668.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. BASE layers: Simplifying training of large, sparse models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 6265–6275. PMLR, 2021. URL https://proceedings.mlr.press/v139/lewis21a.html.

Jan Ludziejewski et al. Scaling laws for fine-grained mixture of experts. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 33270–33288, 2024. doi: 10.48550/arXiv.2402.07871.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016. doi: 10.48550/arXiv.1609.07843.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 18332–18346. PMLR, 2022. URL https://proceedings.mlr.press/v162/rajbhandari22a.html.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2502.17416.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. doi: 10.48550/arXiv.1701.06538.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, 2019. doi: 10.48550/arXiv.1910.07467.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M. Dai, Zhifeng Chen, Quoc V. Le, and James Laudon. Mixture-of-experts with expert choice routing. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/2f00ecd787b432c1d36f3de9800728eb-Abstract.html.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025. doi: 10.48550/arXiv.2510.25741.

A Reproducibility Details

Table 5: Training and model settings shared by the reported runs.

Setting	Value
Effective depth	36 (loop baseline: 18 physical layers, two calls)
Residual / latent width	1536 / 768
MoE intermediate width	576
Attention / KV heads; head dimension	24 / 3; 128
Vocabulary / context length	151936 / 4096
Top-K	8
Normalization	pre-norm RMSNorm, QK-norm, final RMSNorm
Latent post-aggregation RMSNorm	enabled
Projection bias	disabled
Precision	bf16
Optimizer	AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.95$
Weight decay / optimizer epsilon	0.1 / 10^{-5}
Learning rate	3.36×10^{-4}, 500-step warmup
Global batch	4608 sequences $\times$ 4096 tokens
Gradient clipping / init std.	1.0 / 0.006
Router / block auxiliary coefficient	0.001 / 0.001
Layer-wise auxiliary coefficient	0
Seed	42
Parallelism	EP=8, TP=1, PP=1
Hardware	NVIDIA A800 GPUs
Main checkpoint	step 11000

All configurations use the same training data and data order. All reported runs use seed 42; the reported comparisons therefore represent one fixed initialization and data order. The common training horizon contains $11{,}000 \times 4{,}608 \times 4{,}096 = 207{,}618{,}048{,}000$ scheduled token positions. The comparisons hold effective depth, latent width, and the number of selected experts fixed while varying the ownership and size of the expert pools.

B Downstream Results

Table 6: Nine downstream task scores at step 11,000 (%). Means are unweighted averages of the nine displayed task scores, rounded to two decimal places. Bold marks the highest score among these six configurations.

Task	S2	Per-Layer	S4	Loop	S4+SP	S6
ARC-C	**39.59**	39.16	38.74	38.14	38.40	38.31
HellaSwag	**50.45**	49.40	50.41	49.43	49.51	48.74
MMLU	**28.61**	26.25	27.28	26.25	26.90	27.08
WinoGrande	**53.51**	52.64	52.64	52.01	52.33	52.25
ARC-E	**62.46**	61.91	61.70	61.32	61.49	58.33
PIQA	**71.49**	70.40	70.29	69.91	69.37	69.53
OpenBookQA	**36.20**	33.60	35.40	34.80	31.80	34.60
SciQ	**88.80**	87.10	88.60	88.50	88.10	86.30
LAMBADA	37.75	36.19	**39.03**	35.28	34.74	33.13
Mean	**52.10**	50.74	51.57	50.63	50.29	49.81

The nine-task mean increases with pool size (49.97, 50.25, 51.57 for $N = 128, 192, 256$), while individual task scores vary.

AI Use Statement

Generative AI tools assisted with translating and restructuring an author-provided Chinese manuscript, checking numerical consistency, organizing bibliographic information, drafting LaTeX, and refining the presentation. The tools did not run model training or generate new empirical obser-

Table 7: Additional N-scan task scores at $S = 4$ (%). Means follow the same convention as Table 6.

Task	S4-N192	S4-N128
ARC-C	38.31	37.88
HellaSwag	47.67	47.82
MMLU	25.56	28.20
WinoGrande	52.33	53.28
ARC-E	58.96	57.87
PIQA	69.53	69.37
OpenBookQA	36.60	34.20
SciQ	87.30	85.70
LAMBADA	36.02	35.40
Mean	50.25	49.97

vations. Experimental results originate from the authors' research records. The authors are responsible for the content, citations, and interpretations in this paper.

ETHICS STATEMENT

This work studies language-model architecture and does not involve human participants or user data collection. Models trained on large text corpora can reproduce harmful content, bias, or private information present in their data. The present experiments do not evaluate safety, fairness, or privacy.