

NimbleMoE: Collaboration-Aware Expert Merging with Sparse Compensation for Efficient Mixture-of-Experts LLMs

Zhiyuan Ning^{1,2,3}, Xiaoran Liu^{1,2,3}, Ruge Xu⁴, Xinfei Guo⁴, Ziwei He^{2,3,†}, Xipeng Qiu^{1,2,3,†}

¹Fudan University, ²Shanghai Innovation Institute, ³OpenMOSS Team,
⁴Shanghai Jiao Tong University

Correspondence: ziwei.he@sii.edu.cn, xpqiu@fudan.edu.cn

Abstract

Mixture-of-Experts (MoE) Transformers improve scaling efficiency by activating only a subset of experts per token, but deployed inference remains constrained by memory capacity and bandwidth. Existing expert compression methods face a quality–memory–latency trilemma: pruning or fully merging degrades model quality, while partial merging increases per-token activated parameters and hurts latency. We propose NimbleMoE, a collaboration-aware partial expert merging framework that requires no end-to-end fine-tuning and reduces both model size and activated-parameter cost. NimbleMoE clusters experts by routing co-activation so co-activated experts share a low-rank subspace. To compensate for the capacity lost, NimbleMoE adds hardware-efficient Monarch residuals, with an alignment-aware choice between latent-space and output-space compensation. Across four MoE backbones, NimbleMoE attains the best average accuracy among the compressed methods we evaluate under 20–25% total-parameter compression, with the residual loss depending on the backbone, while delivering up to $1.79\times$ MoE-MLP kernel speedup in bandwidth-bound settings, $1.31\times$ end-to-end TPOT speedup, and $2.81\times$ expert-parallel MoE-kernel speedup.

1 Introduction

As Transformer-based (Vaswani et al., 2017) Large Language Models (LLMs) keep scaling along model size across pretraining, post-training, and test-time reasoning (Minaee et al., 2025; Kaplan et al., 2020; Snell et al., 2025), developing system-efficient model architectures has become critical. Mixture-of-Experts (MoE) (Jordan and Jacobs, 1994; Jacobs et al., 1991) emerged as a standard recipe for scaling LLMs by activating a subset of experts per token, delivering strong quality at lower training and inference compute (Cai et al., 2025).

However, MoE models introduce significant

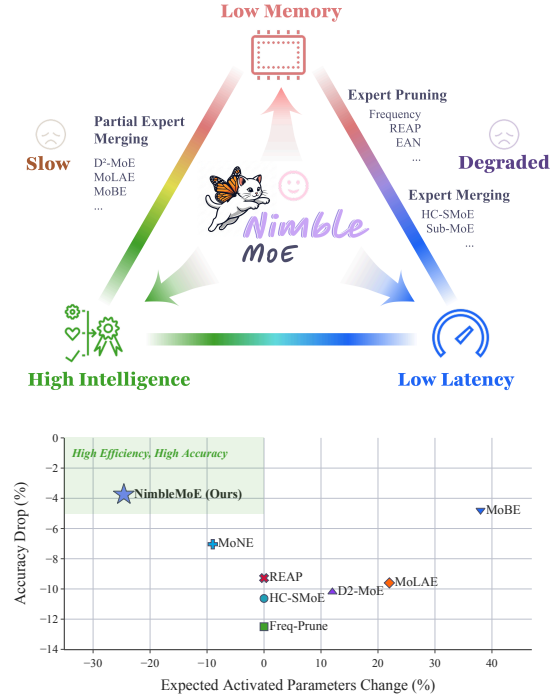


Figure 1: Computational complexity versus accuracy drop for representative MoE expert compression methods at a 25% compression ratio on the Qwen3-30B-A3B-Instruct-2507 model.

pressure on memory capacity and bandwidth despite reducing computational cost. MoE compression methods aim to alleviate this by exploiting inter- and intra-expert parameter redundancy (He et al., 2025). Expert Pruning (Lasby et al., 2025) and full merging (Li et al., 2025a; Chen et al., 2025a) shrink expert count, reducing total parameters but degrading quality at moderate compression. *Partial expert merging* (Gu et al., 2025; Zhang et al., 2025; Chen et al., 2025b) preserves an expert-specific component and shares a common subspace, improving expressivity after compression. Yet it typically *increases* activated parameters per token, worsening latency in bandwidth-bound regimes. MoE expert compression thus faces a trilemma

among intelligence, memory, and latency. As Figure 1 shows, no prior method dominates on both quality and activated-parameter cost.

We decompose expert compression into expert grouping and within-group compression, building on two observations. First, MoE routing is highly skewed and dominant experts are frequently co-activated (Go and Mahajan, 2025; Luo et al., 2025; Zhang et al., 2025), so collaboration-aware merging can lower average activated parameters. Second, the capacity gap left by low-rank merging can be well-modeled by sparse compensation (Zong et al., 2025), and structured Monarch residuals (Dao et al., 2022) make the sparse compensation hardware-friendly. We further analyze how the residual aligns with the column space of the shared basis and derive a parameter-efficiency / expressiveness trade-off between *output-space compensation* and *latent-space compensation*.

We propose **NimbleMoE**, a collaboration-aware *partial expert merging* framework that reduces both total and activated parameters while preserving model quality.¹ Our contributions are:

- **Collaboration-aware expert clustering.** We formulate expert grouping as maximizing pairwise routing co-activation under a balanced-partition constraint, and solve it efficiently with a hierarchical max-weight matching warm-start followed by hill-climbing swaps.
- **Structured Monarch residual compensation.** We combine low-rank subspace merging with hardware-efficient Monarch block-sparse residuals, balancing expressiveness and parameter cost.
- **A new Pareto frontier on the expert compression trilemma.** NimbleMoE cuts both total and activated parameters by 20–25% while retaining the best average accuracy among the compressed methods we evaluate on all four backbones, and delivers up to $1.79\times$ MoE-MLP, $1.31\times$ end-to-end TPOT, and $2.81\times$ expert-parallel kernel speedups.

2 Background and Related Work

Expert Pruning and Merging. Expert pruning removes low-importance experts using activation frequency or router-weighted importance metrics (Lasby et al., 2025). These methods reduce total parameters but can degrade accuracy under high compression ratios. *Pruning-with-replacement*

¹Compression code and inference kernels are available at <https://github.com/OpenMOSS/NimbleMoE>.

variants such as MoNE (Zhang et al., 2026) attempt to recover the lost capacity by training a lightweight surrogate that mimics the pruned experts. Expert merging clusters similar experts and consolidates their weights to reduce the number of experts while preserving capacity (Li et al., 2025a; Chen et al., 2025a). *Partial expert merging* methods, including D^2 -MoE (Gu et al., 2025), MoLAE (Liu et al., 2025), and MoBE (Chen et al., 2025b), decompose experts into shared and unique components, providing additional flexibility after compression. These approaches commonly preserve better model quality, but the activated parameters can increase because multiple components are activated per token. NimbleMoE departs from all of these by clustering for expert co-activation overlap rather than functional similarity or contiguous index ranges, which directly minimizes expected activated parameters.

Low-Rank and Structured Sparse Compression.

Low-rank factorization is a widely used compression primitive for Transformer weights. Methods such as ASVD (Yuan et al., 2025) and SVD-LLM (Wang et al., 2025b,a) decompose weight matrices into compact factors, reducing parameter count but often sacrificing fidelity under aggressive compression. SkipCat (Lu et al., 2025) further improves the parameter efficiency of low-rank methods by skipping submatrices of shared projections. Sparse residual compensation further improves accuracy by allocating a small number of parameters to approximate the remaining error. Unstructured sparsity is difficult to exploit on modern accelerators, motivating structured designs such as Monarch matrices (Dao et al., 2022) and $\mathcal{GS}$ -matrices (Gorbunov et al., 2024), which provide hardware-friendly block sparsity. SSLC (Zong et al., 2025) combines low-rank and sparse residuals to improve expressiveness. NimbleMoE further assigns low-rank and structured sparse components according to the nature of partial expert merging: the low-rank subspace carries structure shareable across experts, while the Monarch residual restores the individual expert’s unique sparse structure.

3 NimbleMoE

We first describe collaboration-aware expert clustering, which targets activated parameters by aligning merging decisions with routing co-activation structure. The subsequent subsections introduce the structured sparse residual mechanism and the full compression and inference pipeline.

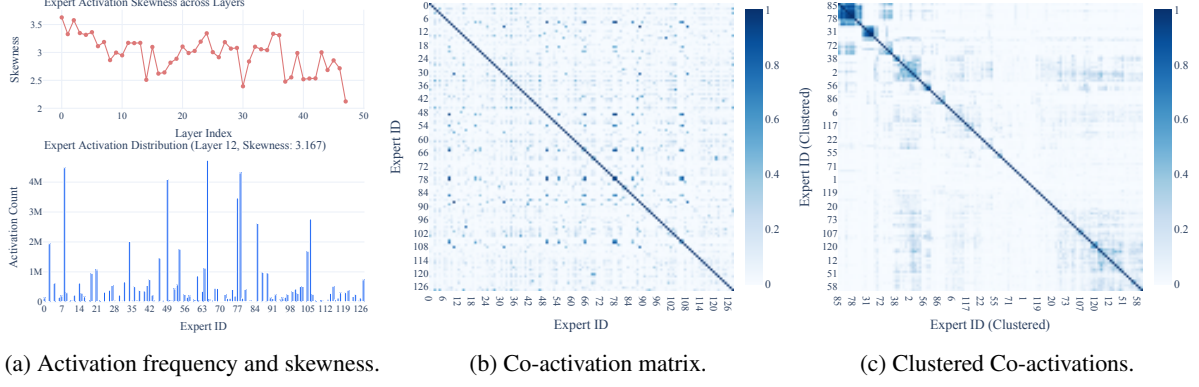


Figure 2: Routing statistics on Qwen3-235B-A22B-Instruct-2507 (layer 12) illustrate skewed activation and strong collaboration structure that motivate collaboration-aware clustering.

3.1 Collaboration-Aware Expert Clustering for Minimal Activated Parameters

For full expert merging, the clustering of experts can significantly affect post-merge quality (Li et al., 2025a; Chen et al., 2025a). In contrast, *partial expert merging* methods like MoLAE (Liu et al., 2025) and MoBE (Chen et al., 2025b) preserve an expert-specific component while only sharing a low-rank subspace, so clustering primarily determines which experts share the cost of loaded parameters. Empirically, reconstruction loss is a weak proxy for grouping: the expert-pair MSE matrix is not strongly correlated with common grouping heuristics (Figure 3). This motivates clustering that directly targets *expected activated parameters* rather than reconstruction error. Additionally, such collaboration-aware clustering demonstrates lower average MSE loss after merging than naive contiguous grouping used in prior works. In terms of expert parallelism efficiency, collaboration-aware clustering keeps frequently co-activated experts in the same group and therefore on the same device, reducing cross-device communication (Luo et al., 2025).

Routing collaboration and objective. MoE routing is typically skewed: a small subset of experts receives most tokens, and dominant experts are frequently co-activated (Figure 2). Throughout, *collaboration* between two experts means frequent joint selection by the router, not any form of expert-to-expert interaction. We summarize this with a symmetric co-activation matrix $C \in \mathbb{R}^{N \times N}$, where C_{ij} is the fraction of tokens for which experts i, j are jointly activated; collaboration-aware clustering reduces expected activated shared parameters by

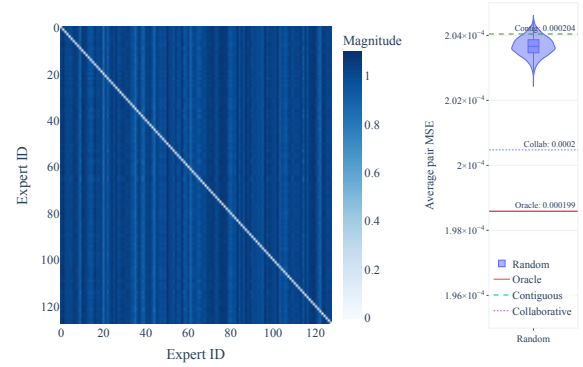


Figure 3: (Left) MSE loss matrix between expert pairs (Layer 32 of Qwen3-30B-A3B-Instruct-2507). (Right) Average MSE loss after merging with different clustering strategies. Collaborative clustering shows better average loss than contiguous and random grouping.

11–32%.² In partial expert merging, each activated expert incurs a per-expert *unique* cost $P_{\text{uniq},i}$, while each group incurs a *shared* cost P_{sh} loaded once per token per group when any expert in the group is activated. Let the router activate K_r experts per token (set S_t), partition N experts into $K_g = N/G$ equal-size groups $\{G_g\}$, and define $n_{g,t} := |S_t \cap G_g|$ and the repeat count $R_t := \sum_g \max(0, n_{g,t} - 1)$, so that $K_a := K_r - R_t$ is the number of *distinct* groups the token activates. The activated-parameter cost is

$$\mathcal{C}_A(S_t) = \sum_{i \in S_t} P_{\text{uniq},i} + K_a P_{\text{sh}}. \quad (1)$$

Since the unique term is fixed by routing, minimizing expected activated parameters is equivalent to maximizing $\mathbb{E}[R_t]$. We adopt a pairwise surrogate

²Detailed statistics in Appendix G.

based on C :

$$\hat{\mathcal{R}}_t(\{G_g\}) = \sum_g \sum_{i < j} C_{ij} z_{ig} z_{jg}, \quad (2)$$

which encourages frequently co-activated experts to share groups.

Warm-started balanced hill-climbing. Introduce binary $z_{ig} \in \{0, 1\}$ indicating whether expert i is in group g , with balance constraints $\sum_g z_{ig} = 1$ and $\sum_i z_{ig} = G$. Maximizing Eq. (2) is a balanced max intra-cluster weight partitioning problem and is NP-hard. We obtain an initial balanced partition by hierarchically applying max-weight perfect matching (pair experts, then pair the resulting pairs); an LP relaxation of the standard McCormick linearization $y_{ijg} = z_{ig} z_{jg}$ followed by rounding and repair is an alternative warm-start (full constraints in Appendix C). Starting from this feasible partition, we apply hill-climbing balanced swaps: pick groups G_p, G_q and experts $u \in G_p, v \in G_q$, and accept the swap iff

$$\Delta = (\hat{\mathcal{R}}_t(G'_p) + \hat{\mathcal{R}}_t(G'_q)) - (\hat{\mathcal{R}}_t(G_p) + \hat{\mathcal{R}}_t(G_q)) > 0, \quad (3)$$

where $\hat{\mathcal{R}}_t(G) := \sum_{i < j, i, j \in G} C_{ij}$ and G'_p, G'_q are the post-swap groups. This monotonically increases $\hat{\mathcal{R}}_t$ until convergence. We use $G = 2$ in experiments, in which case the balanced partition reduces to a maximum-weight perfect matching, solvable exactly via Edmonds' Blossom algorithm (Edmonds, 1965).

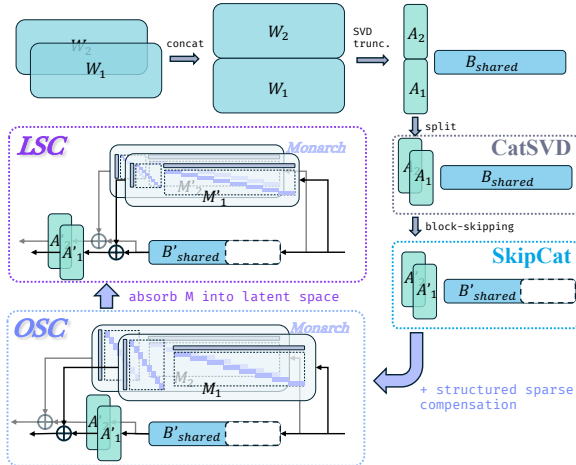


Figure 4: Comparison of OSC and LSC residual compensation strategies. OSC adds the compensator in output space, capturing the full residual but with higher parameter cost. LSC adds the compensator in latent space, capturing only the aligned residual but with lower parameter cost.

3.2 Joint Low-Rank and Sparse Expert Subspace Merging with Monarch Compensations

Compression via mixed methods commonly outperforms single-method approaches, since they exploit different redundancies. For instance, many quantization methods (Yao et al., 2023; Li et al., 2025b) combine low-rank compensation to recover expressiveness. Recent work (Zong et al., 2025) shows that adding sparse compensations to low-rank compression can be more parameter-efficient, but the unstructured sparsity patterns worsen the hardware efficiency. In light of this, we propose to use structured sparse residuals to complement the efficient SkipCat (Lu et al., 2025) low-rank approximation. We adopt Monarch (Dao et al., 2022) format which introduces a hardware-friendly and expressive block-diagonal factorization.

Output-Space Compensation and Latent-Space Compensation. Given a target weight $W \in \mathbb{R}^{D \times H}$ and its SkipCat approximation $\hat{W}_{\text{base}} = A \cdot [I_r \mid B] \cdot P^\top$, the residual $R = W - \hat{W}_{\text{base}}$ decomposes into a component $R_{\parallel} = \mathcal{P}_A R$ in $\text{col}(A)$ and an orthogonal $R_{\perp} = (I - \mathcal{P}_A)R$, where $\mathcal{P}_A = A(A^\top A)^{-1}A^\top$. We compensate the residual via a Monarch matrix $M = LIIV$ in one of two positions:

- **Output-Space Compensation (OSC):** The compensator $M \in \mathbb{R}^{D \times H}$ is added directly to the output:

$$\hat{W} = A \cdot [I_r \mid B] \cdot P^\top + M. \quad (4)$$

$M = LIIV \in \mathbb{R}^{D \times H}$ is a Monarch matrix with block-sparse factors $L \in \mathbb{R}^{D \times bk}$, $V \in \mathbb{R}^{bk \times H}$ and a block-permutation matrix Π . As illustrated in Figure 4, OSC can capture the full residual R but requires $O(DH)$ parameters (or $O((D + H)k)$ for rank- k Monarch).

- **Latent-Space Compensation (LSC):** The compensator $M \in \mathbb{R}^{r \times H}$ operates in the compressed latent space:

$$\hat{W} = A \cdot ([I_r \mid B] + M) \cdot P^\top. \quad (5)$$

Here, $M = LIIV \in \mathbb{R}^{r \times H}$ is a Monarch matrix with factors $L \in \mathbb{R}^{r \times bk}$, $V \in \mathbb{R}^{bk \times H}$. LSC is limited to $R_{\parallel} \subset \text{col}(A)$ but requires only $O(rH)$ parameters, a factor of D/r reduction.

Alignment-Aware Strategy Selection. Define the alignment ratio $\rho = \|R_{\parallel}\|_F^2 / \|R\|_F^2 \in [0, 1]$,

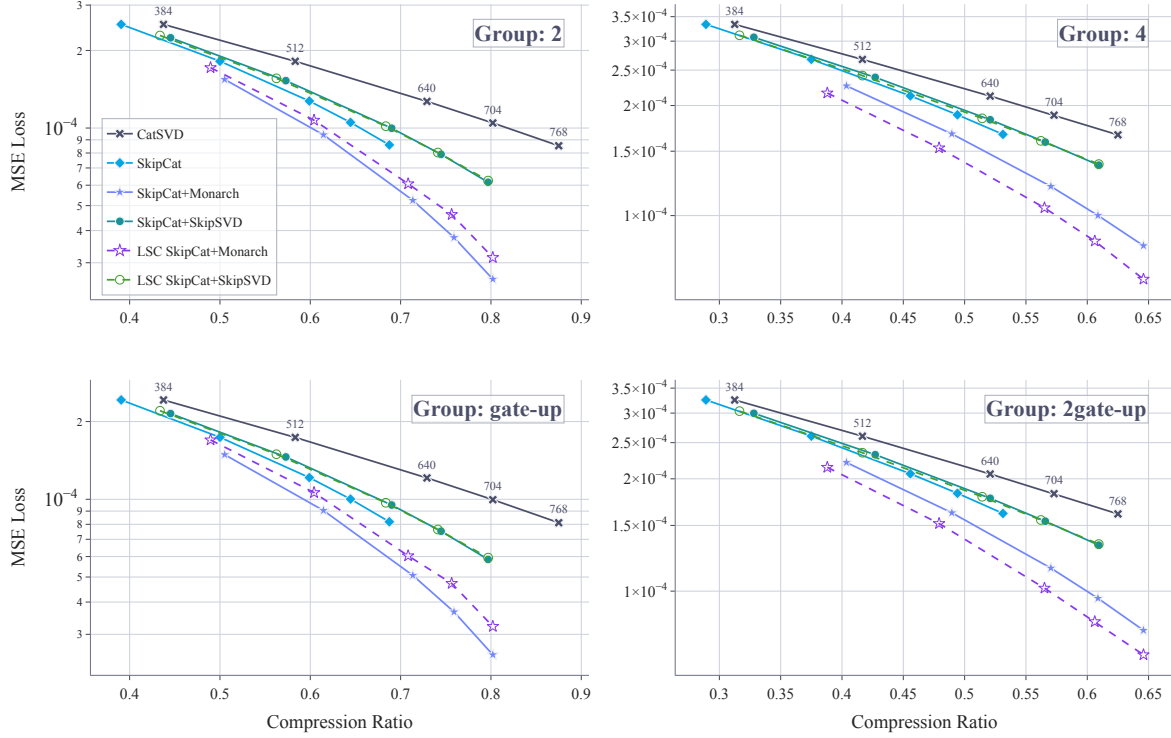


Figure 5: Comparison of *partial merging* methods on Qwen3-30B-A3B-Instruct-2507 (Layer 44). **Group-2/4**: merging 2/4 experts’ gate and up matrices separately. **Group gate-up**: merging gate and up matrices of a single expert. **Group 2gate-up**: jointly merging 2 experts’ gate and up matrices. SkipCat+Monarch consistently achieves the lowest reconstruction error, with *LSC* (B-Comp) increasingly favorable as group size grows. Note that the horizontal axis is the *retained* parameter fraction (lower is more compressed), which is the complement of the CR column in Table 2.

measuring how much residual energy lies within $\text{col}(A)$. The minimum achievable error for *LSC* is $(1 - \rho)\|R\|_F^2$, while *OSC* attains zero error only in the idealized full-capacity limit; in practice both compensators operate under a fixed Monarch budget, so this bounds *OSC*’s advantage rather than describing an attainable error. Empirically, we observe that:

- ***LSC* excels when merging more matrices** (e.g., 4 experts or gate-up jointly), as the shared A spans a richer subspace, yielding higher ρ .
- ***OSC* is preferable for smaller groups** (e.g., 2 matrices), where $\text{col}(A)$ is more constrained and $R_\perp$ dominates.

Figure 5 validates this analysis: SkipCat with Monarch compensation achieves the lowest MSE across compression ratios, with *LSC* outperforming *OSC* as group size increases.

Moreover, since merging gate and up matrices yields much fewer activated parameters while having similar average MSE loss compared to merging more experts’ matrices (Figure 5), we prefer **Group**

2gate-up merging over **Group-4** merging for better overall computational efficiency. We further compare merging same expert’s gate and up matrices (like SkipCat setting) with merging 2 experts’ gate and up matrices in Appendix B.

Hessian-Aware Optimization Pipeline. To preserve model performance under compression, we minimize the output reconstruction error weighted by input activation statistics, following OBS (Hasibi et al., 1993) and OBC (Frantar et al., 2022). Given calibration data $\mathcal{X} \in \mathbb{R}^{H \times N_{\text{cal}}}$, we approximate the Hessian via the diagonal covariance:

$$\Lambda = \text{diag}(\mathcal{X}\mathcal{X}^\top) + \epsilon I, \quad Q = \Lambda^{1/2}. \quad (6)$$

The whitened objective $\mathcal{L} = \|(W - \widehat{W})Q\|_F^2$ ensures that high-variance input dimensions receive proportionally higher reconstruction fidelity.

For each expert group g , the optimization proceeds as:

1. **Permutation Selection:** Compute column norms $v_j = \sum_{i \in g} \|(W_i Q)_{:,j}\|_2^2$ and select P_g to place the top- r salient columns first.

2. **Shared Latent Projector:** Solve³ for B_g via ridge regression on whitened blocks:

$$B_g = \left(\sum_{i \in g} \widetilde{W}_{i,1}^\top \widetilde{W}_{i,1} + \lambda I \right)^{-1} \left(\sum_{i \in g} \widetilde{W}_{i,1}^\top \widetilde{W}_{i,2} \right). \quad (7)$$

3. **Monarch Residual Fitting:** Initialize $M_g = L_g \Pi V_g$ from the projected residual (*LSC*) or direct residual (*OSC*), then refine via alternating least squares on the block-diagonal factors. Monarch block sizes may be non-uniform, prioritizing top-routed dominant experts and allowing flexible parameter budgets.

We optimize group parameters end-to-end with Adam under whitened objectives:

$$\mathcal{L}_{\text{gate/up}} = \sum_{i \in g} \|(W_i^{\text{gate/up}} - \widehat{W}_i^{\text{gate/up}})Q\|_F^2, \quad (8)$$

$$\widehat{W}_i^{\text{gate/up}} = A_i^{\text{gate/up}} ([I_r \mid B_g] + M_g^{\text{gate/up}}) P_g^\top \quad (9)$$

$$\mathcal{L}_{\text{down}} = \sum_{i \in g} \|(W_i^{\text{down}} - \widehat{W}_i^{\text{down}})Q\|_F^2, \quad (10)$$

$$\widehat{W}_i^{\text{down}} = A_i^{\text{down}} [I_r \mid B_g] P_g^\top + M_g^{\text{down}}. \quad (11)$$

Overall Architecture. Figure 6 illustrates the complete NimbleMoE architecture. For **gate/up projections**, we apply *LSC*: the shared permutation P_g and skip matrix B_g compress the input from H to r dimensions, with Monarch compensation $M_g^{\text{gate/up}} \in \mathbb{R}^{r \times H}$ refining the latent representation before expert-specific projections $A_i^{\text{gate/up}}$. For **down projections**, we apply *OSC*: the shared basis $B_g \in \mathbb{R}^{H \times r}$ reconstructs the output from expert-specific coefficients A_i^{down} , with Monarch compensation $M_g^{\text{down}} \in \mathbb{R}^{H \times D}$ directly correcting the output.

3.3 NimbleMoE Inference Forward Pass

We summarize the inference forward pass, highlighting shared computation within each group.⁴

Notation. For a given input token with hidden state $x \in \mathbb{R}^H$, let the router select top- K_r experts $\{e_1, \dots, e_{K_r}\}$ with corresponding routing weights $\{w_1, \dots, w_{K_r}\}$. Each expert e_k belongs to a group $g(e_k)$.

³Detailed derivation in Appendix C

⁴Algorithm 3 in Appendix E summarizes the forward.

Algorithm 1 NimbleMoE compression pipeline

Input: expert weights $\{W_i^{\text{gate}}, W_i^{\text{up}}, W_i^{\text{down}}\}$, routing stats C , calib data $\mathcal{X}$,

rank r , group size G , groups K_g , Monarch block sizes k_L, k_R , constraints $\rho_{\max}, \rho_{\text{shared_max}}$
Compute collaboration-aware clusters $\{G_k\}$ via Alg. 2

Compute $Q = (\text{diag}(\mathcal{X}\mathcal{X}^\top) + \epsilon I)^{1/2}$

for each group $g \in \{1, \dots, K_g\}$ **do**

Select permutation P_g using column scores

$$v_j = \sum_{i \in g} \|(W_i Q)_{:,j}\|_2^2$$

Initialize B_g (SkipCat) and M_g (Monarch)

Gate/Up (LSC): optimize per Eq. (9)

Down (OSC): optimize per Eq. (11)

end for

Output: compressed parameters $\{P_g, B_g, M_g\}$ and expert-specific $\{A_i\}$

For each group g among selected experts, we share the input compression and then compute expert-specific projections:

$$\begin{aligned} \tilde{x}_g &= P_g^\top x = [x_1; x_2], & z_g &= x_1 + B_g x_2, \\ z_g^{\text{gate}} &= z_g + M_g^{\text{gate}} \tilde{x}_g, & z_g^{\text{up}} &= z_g + M_g^{\text{up}} \tilde{x}_g, \\ h_{e_k} &= \sigma(A_{e_k}^{\text{gate}} z_g^{\text{gate}}) \odot (A_{e_k}^{\text{up}} z_g^{\text{up}}). \end{aligned} \quad (12)$$

Experts in the same group aggregate before the shared down projection:

$$\begin{aligned} a_{e_k} &= A_{e_k}^{\text{down}} P_g^\top h_{e_k}, & \bar{a}_g &= \sum_{e_k \in \mathcal{E}_g} w_k a_{e_k}, \\ y &= \sum_g \left(B_g \bar{a}_g + \sum_{e_k \in \mathcal{E}_g} w_k M_g^{\text{down}} h_{e_k} \right) \end{aligned} \quad (13)$$

Operation	Standard MoE	NimbleMoE
<i>Per-token FLOPs</i>		
Gate/Up projection	$O(2K_r D H)$	$O(K_a \cdot r H + 2K_r D r)$
Down projection	$O(K_r H D)$	$O(K_r r D + K_a \cdot H r)$
Monarch matrices	—	$O(K_a \cdot r H / k + K_a \cdot H D / k)$
Total FLOPs	$O(3K_r D H)$	$O(3K_r D r + K_a (r + D) H)$
<i>Parameters per expert</i>		
Gate + Up	$2DH$	$2Dr$
Down	HD	rD
<i>Shared parameters</i>		
SkipCat (B, P)	—	$r(H-r) + rH$
Monarch	—	$O(rH/k) + O(HD/k)$

Table 1: Complexity comparison for K_r activated experts spanning $K_a \leq K_r$ distinct activated groups (Eq. (1)); K_a falls as co-activated experts are grouped together. H : input dim, D : hidden dim, r : rank, k : Monarch block size.

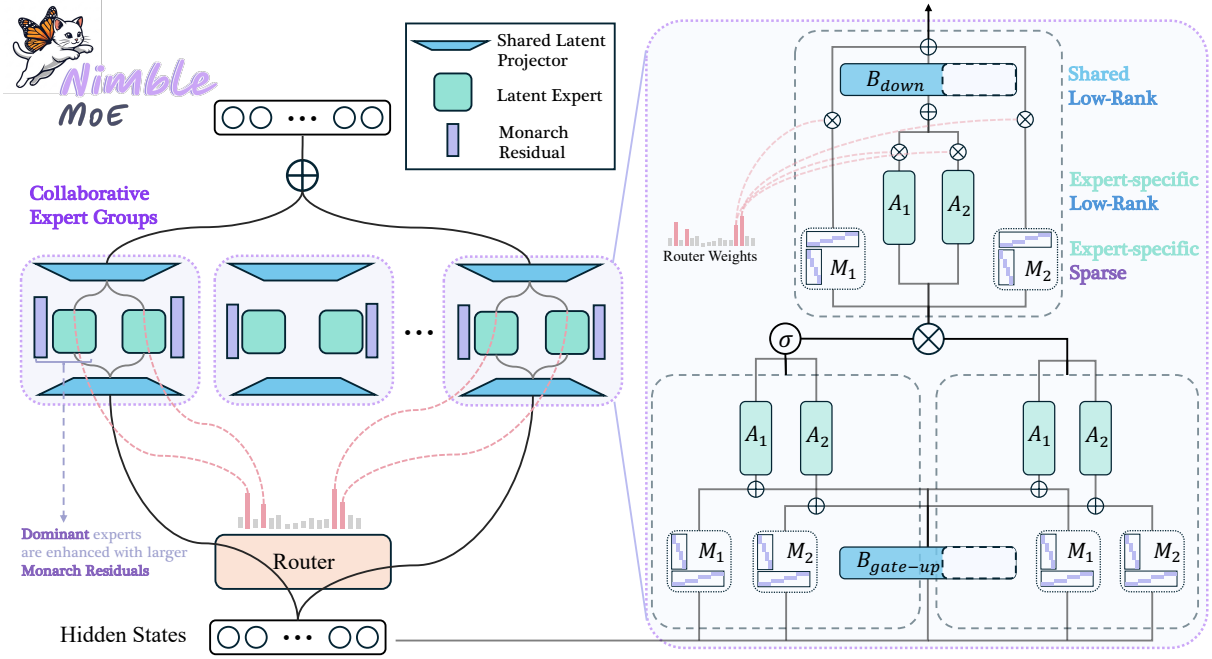


Figure 6: The NimbleMoE architecture. Experts within each collaboration-aware cluster share a low-rank subspace (SkipCat), with structured block-sparse Monarch residuals recovering approximation capacity. Gate/up projections use *latent-space compensation (LSC)* for parameter efficiency; down projections use *output-space compensation (OSC)* for expressiveness.

Complexity. Table 1 compares the per-token computational and parameter complexity of NimbleMoE against standard MoE. One property follows from the grouped structure. The shared terms are paid once per activated group, so the K_a -dependent cost amortizes whenever a token activates multiple experts from the same group, which is what collaboration-aware clustering maximizes.

4 Experiments

We evaluate NimbleMoE on multiple MoE backbones and benchmarks, measuring both quality and system efficiency. Our evaluation focuses on the deployment regime where activation memory and memory bandwidth dominate latency.

4.1 Experiment Setup

We evaluate NimbleMoE on GLM-4.7-Flash (Team et al., 2025) (30B), Qwen3-30B-A3B-Instruct-2507, Ling-2.6-flash (inclusionAI, 2025) (104B), and Qwen3-235B-A22B-Instruct-2507 (Team, 2025) across ten benchmarks: MMLU-Pro (Wang et al., 2024), MMLU-Redux (Gema et al., 2025), C-Eval (Huang et al., 2023), GPQA-Diamond (Rein et al., 2024), IFEval (Zhou et al., 2023), LiveCodeBench (Jain et al., 2025), Math 500 (Hendrycks et al., 2021), IQuiz (AI-

ModelScope, n.d.), and AIME 24/25 (Zhang and Math-AI, 2024, 2025). All methods are evaluated with identical prompts and decoding settings via the EvalScope framework. Compression ratios (CR) are matched within each backbone block to enable apples-to-apples comparison; small differences (e.g., 24% vs. 25%) reflect the discrete granularity of each method’s parameter accounting and are conservatively rounded to favor the baselines. We compare against the SOTA expert pruning method REAP, the merging method HC-SMoE, the partial expert merging method MoBE, and the pruning-with-replacement method MoNE (Zhang et al., 2026). For SkipCat, we only compress the MoE modules to keep the comparison fair with other expert pruning or merging methods.

Performance Results NimbleMoE achieves the best average among compressed models across all four backbones (Table 2), with the largest gains in settings with skewed routing where co-activation clustering improves shared activation reuse. GLM-4.7-Flash has only 64 experts (top-4 activated) and intrinsically lower weight redundancy, which is the most challenging setting for compression, with the 27%-CR block in Appendix H (NimbleMoE leads MoBE by +13 pts there). On larger backbones like

Model	CR	MMLU-P	MMLU-R	C-Eval	IQuiz	GPQA-D	IFEval	LCB	MATH	AIME24	AIME25	Avg.
GLM-4.7-Flash												
<i>MoE</i>	-	67.17	82.93	73.33	69.17	44.44	79.41	25.93	89.04	36.67	28.67	59.68
SkipCat	20%	53.05	73.33	60.36	58.00	25.25	<u>73.75</u>	20.77	74.20	15.33	14.67	46.87
MoBE	20%	<u>59.49</u>	<u>77.65</u>	<u>65.13</u>	<u>58.83</u>	<u>35.35</u>	72.75	<u>25.49</u>	<u>83.00</u>	<u>24.67</u>	<u>18.67</u>	<u>52.10</u>
NimbleMoE	20%	61.28	79.50	68.01	59.83	37.58	76.49	26.04	85.60	36.00	24.67	55.50
Qwen3-30B-A3B-Instruct-2507												
<i>MoE</i>	-	74.36	88.56	87.00	74.83	52.73	83.99	36.92	96.88	72.00	62.67	72.99
SkipCat	23%	72.80	86.80	80.25	68.00	55.76	79.74	30.66	93.44	53.33	44.67	66.55
REAP	24%	69.32	80.72	43.68	70.83	49.09	79.26	37.36	95.96	74.00	62.00	66.22
HC-SMoE	24%	66.90	76.77	69.99	66.67	42.42	81.15	32.97	95.40	<u>66.67</u>	<u>53.33</u>	65.23
MoNE	25%	72.32	86.68	82.17	67.50	52.53	86.58	32.97	94.40	60.00	43.33	67.85
MoBE	24%	<u>72.64</u>	<u>86.86</u>	83.67	71.50	52.73	82.62	<u>34.18</u>	95.32	62.67	52.67	<u>69.49</u>
NimbleMoE	25%	72.23	86.99	<u>83.12</u>	72.67	<u>53.64</u>	<u>84.44</u>	<u>34.18</u>	<u>95.88</u>	66.00	<u>53.33</u>	70.25
Ling-2.6-flash												
<i>MoE</i>	-	73.35	87.91	87.85	74.50	57.37	89.06	50.55	97.84	82.00	64.67	76.51
SkipCat	24%	72.84	87.81	86.58	<u>73.50</u>	58.99	88.32	45.38	96.52	78.67	<u>60.67</u>	74.93
MoBE	24%	<u>73.11</u>	87.50	<u>87.15</u>	72.83	<u>58.69</u>	<u>88.54</u>	<u>47.03</u>	<u>96.96</u>	<u>82.00</u>	57.33	<u>75.11</u>
NimbleMoE	24%	73.12	<u>87.79</u>	87.25	75.33	57.58	88.95	49.23	97.08	82.67	63.33	76.23
Qwen3-235B-A22B-Instruct-2507												
<i>MoE</i>	-	79.42	91.32	91.71	85.50	62.12	88.24	43.30	96.84	80.00	70.00	78.85
SkipCat	24%	79.93	91.30	88.45	80.00	<u>61.57</u>	84.29	30.88	95.68	64.00	51.33	72.74
MoBE	24%	78.21	90.79	<u>90.51</u>	<u>84.50</u>	60.20	<u>87.58</u>	<u>41.54</u>	<u>96.60</u>	<u>78.67</u>	<u>64.67</u>	<u>77.33</u>
NimbleMoE	25%	<u>78.24</u>	<u>91.23</u>	91.17	84.83	62.02	88.17	43.08	96.64	79.33	66.67	78.14

Table 2: Benchmark results across four MoE backbones. Higher is better. **CR** denotes compression ratio: reduction in total parameters compared to the original model. Within each backbone block, **bold** marks the best result among compressed methods and underline marks the second best (the uncompressed *MoE* reference is excluded). NimbleMoE achieves the best average accuracy among compressed models across all backbones. Per-benchmark uncertainty estimates are given in Appendix K.

Ling-2.6-flash and Qwen3-235B-A22B, NimbleMoE’s accuracy loss is almost negligible (below evaluation standard error) at 24–25% CR. MoBE inflates activation parameters by over 30% via basis sharing, while NimbleMoE reduces activated parameters by 25–31%.

Component Ablation We isolate the contribution of NimbleMoE’s three core ingredients on Qwen3-30B-A3B at 25% CR (Table 3). The structured Monarch residual is by far the largest single lever (+3.03), followed by Hessian-aware whitening (+0.86) and collaboration-aware grouping (+0.44). Appendix D examines the grouping term further, comparing collaboration-aware against contiguous grouping at matched parameter budgets across a rank sweep.

Configuration	Avg	Δ
NimbleMoE (full)	70.25	—
w/o collab. clustering (contig.)	69.81	−0.44
w/o Hessian-aware optimization	69.39	−0.86
w/o Monarch residual	67.22	−3.03

Table 3: Component ablation at 25% CR on Qwen3-30B-A3B-Instruct-2507. Each row removes one component from the full pipeline.

Latency Benchmark We benchmark NimbleMoE in three regimes: single-GPU MoE-MLP kernels (Figure 7, RTX 4090), end-to-end vLLM generation on H200 / A800 (Figure 8), and multi-GPU expert parallelism on a single H100 node (Table 4). In the single-GPU kernel sweep, the collaboration-aware variant pulls ahead of the contiguous-grouping ablation as batch size grows, isolating the contribution of co-activation clustering from raw parameter reduction. Compared to MoBE and HC-SMoE compression, NimbleMoE achieves best kernel latency across all batch sizes, with up to $1.51\times$ speedup over the uncompressed baseline at batch 128. End-to-end, NimbleMoE improves both TTFT and TPOT on H200 (1.01 – $1.16\times$), and on the more bandwidth-bound A800 it delivers up to $1.42\times$ TTFT and $1.31\times$ TPOT, with monotonic scaling from batch 8 to 512 (Figure 8). Under expert parallelism, co-locating co-activated experts on the same device structurally reduces cross-device all-to-all traffic; the EP kernel speedup grows with world size (from $1.78\times$ at WS=2 to $2.81\times$ at WS=8 on batch 8), showing that the grouping is compatible with EP and that the gain scales with communication pressure. Across all three regimes, the gain is largest on

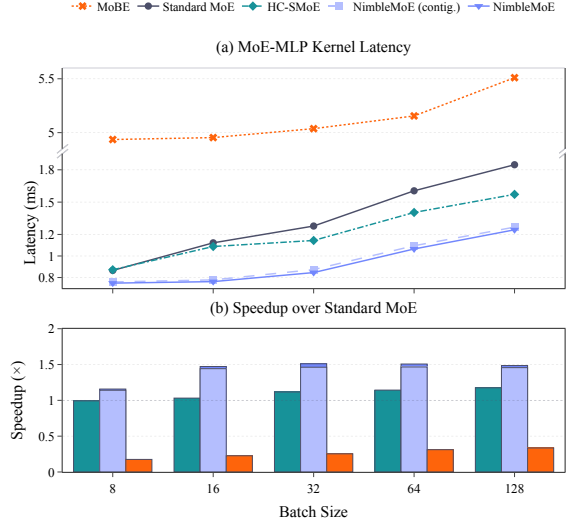


Figure 7: Single-GPU MoE-MLP kernel latency on RTX 4090 (batch sweep). The clustering ablation isolates the gain from co-activation-aware grouping.

the lower-bandwidth A800 and under expert parallelism, and smallest on high-bandwidth hardware at small batch. NimbleMoE is thus best suited to bandwidth-bound single-device deployment and expert-parallel serving.

WSBS	8	32	128	512
2	1.78×	1.49×	1.40×	1.39×
4	2.33×	1.91×	1.91×	1.86×
8	2.81×	2.55×	2.64×	2.36×

Table 4: NimbleMoE expert-parallelism MoE kernel speedup over vanilla MoE on a single H100 node across world sizes (WS) and batch sizes (BS).

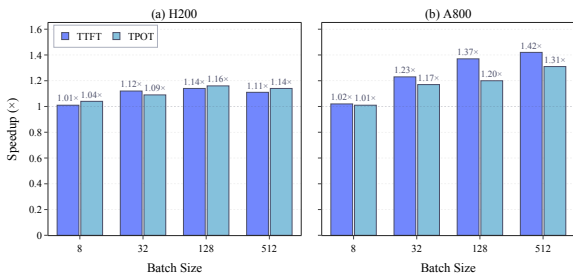


Figure 8: End-to-end vLLM generation speedup of NimbleMoE over the uncompressed model on Qwen3-30B-A3B, across H200 and A800. Gains grow with batch size and are larger on the more bandwidth-bound A800, reaching 1.42× TTFT and 1.31× TPOT at batch 512.

Activated Cost Analysis Table 5 reports the per-token accounting underlying the latency results

Method	Act. FLOPs /tok	Act. params /tok	Mater. /fwd
Standard MoE	75.5 M	37.7 M	0
HC-SMoE	75.5 M	37.7 M	0
	(1.00×	(1.00×	
MoBE	94.4 M	47.2 M	25.77 GF
	(1.25×	(1.25×	
NimbleMoE	69.2 M	30.5 M	0
	(0.92×	(0.81×	

Table 5: Per-token activated cost on Qwen3-30B-A3B, the quantity a bandwidth-bound decoder actually pays. “Mater./fwd” is the extra FLOPs needed to reconstruct expert weights from a shared basis at each forward pass.

above. Reducing total parameters is not sufficient, since a bandwidth-bound decoder pays the activated cost per token. MoBE compresses total size but raises activated FLOPs and parameters by 1.25× and materializes expert weights from its shared basis at every forward pass (25.77 GFLOPs), which accounts for it being slower than the uncompressed baseline. NimbleMoE is the only method evaluated that reduces activated cost below the standard MoE (0.92× FLOPs, 0.81× parameters) without a materialization step, since each group’s shared basis is loaded once per token irrespective of how many of its experts are activated, which explains the latency gains in Figure 7 compared to MoBE and HC-SMoE.

5 Conclusion

MoE inference is shaped by which experts are activated together, compression should also respect the model’s routing collaboration structure. NimbleMoE follows this principle by grouping co-activated experts, sharing low-rank subspaces within each group, and using structured Monarch residuals to recover the capacity lost during merging. The resulting framework reduces both total and per-token activated parameters while preserving model quality, achieving the best accuracy among the compressed baselines we evaluate, none of which requires end-to-end fine-tuning, and delivering practical inference speedups in bandwidth-bound and expert-parallel settings.

Limitations

NimbleMoE requires no end-to-end fine-tuning and no distillation, though it does optimize group parameters by gradient descent. Grouping is derived from calibration routing statistics; large distribution shifts could make the offline partition sub-optimal

(cross-domain stability in Appendix J). Latency gains are largest in the bandwidth-bound regime. Combinations with PTQ are left to future work.

Acknowledgments

This work was supported by the Science and Technology Commission of Shanghai Municipality (No. 25DZ3100402) and the National Natural Science Foundation of China (No. 12426313).

References

- AI-ModelScope. n.d. [IQuiz: Model iq/eq test questions](#). ModelScope Datasets. Accessed: 2026-01-19.
- Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. 2025. [A survey on mixture of experts in large language models](#). *IEEE Transactions on Knowledge and Data Engineering*, 37(7):3896–3915.
- I-Chun Chen, Hsu-Shen Liu, Wei-Fang Sun, Chen-Hao Chao, Yen-Chang Hsu, and Chun-Yi Lee. 2025a. Retraining-free merging of sparse moe via hierarchical clustering.
- Xiaodong Chen, Mingming Ha, Zhenzhong Lan, Jing Zhang, and Jianguo Li. 2025b. [Mobe: Mixture-of-basis-experts for compressing moe-based llms](#). *Preprint*, arXiv:2508.05257.
- Tri Dao, Beidi Chen, Nimit S Sohoni, Arjun De-sai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Re. 2022. [Monarch: Expressive structured matrices for efficient and accurate training](#). In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 4690–4721. PMLR.
- Jack Edmonds. 1965. [Paths, trees, and flowers](#). *Canadian Journal of Mathematics*, 17:449–467.
- Elias Frantar, Sidak Pal Singh, and Dan Alistarh. 2022. Optimal brain compression: a framework for accurate post-training quantization and pruning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS ’22, Red Hook, NY, USA. Curran Associates Inc.
- Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Claire Barale, Robert McHardy, Joshua Harris, Jean Kaddour, Emile Van Krieken, and Pasquale Minervini. 2025. [Are we done with MMLU?](#) In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5069–5096, Albuquerque, New Mexico. Association for Computational Linguistics.
- Seokjin Go and Divya Mahajan. 2025. [Moetuner: Optimized mixture of expert serving with balanced expert placement and token routing](#). *Preprint*, arXiv:2502.06643.
- Mikhail Gorbunov, Kolya Yudin, Vera Soboleva, Aibek Alanov, Alexey Naumov, and Maxim Rakhuba. 2024. [Group and shuffle: Efficient structured orthogonal parametrization](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Hao Gu, Wei Li, Lujun Li, Qiyuan Zhu, Mark Lee, Shengjie Sun, Wei Xue, and Yike Guo. 2025. [Delta decompression for moe-based llms compression](#). *Preprint*, arXiv:2502.17298.
- M Gu and S C Eisenstat. 1996. [Efficient algorithms for computing a strong rank-revealing qr factorization](#). *SIAM Journal on Scientific Computing*, 17(4).
- B. Hassibi, D.G. Stork, and G.J. Wolff. 1993. [Optimal brain surgeon and general network pruning](#). In *IEEE International Conference on Neural Networks*, pages 293–299 vol.1.
- Shwai He, Daize Dong, Liang Ding, and Ang Li. 2025. [Towards efficient mixture of experts: A holistic study of compression techniques](#). *Transactions on Machine Learning Research*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.
- Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. 2023. C-eval: a multi-level multi-discipline chinese evaluation suite for foundation models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS ’23, Red Hook, NY, USA. Curran Associates Inc.
- inclusionAI. 2025. Ling-2.6-flash: Faster responses, stronger execution, higher token efficiency. <https://huggingface.co/inclusionAI/Ling-2.6-flash>. Hugging Face model repository. 104B total parameters, 7.4B active parameters. MIT License.
- Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. [Adaptive mixtures of local experts](#). *Neural Computation*, 3(1):79–87.
- Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2025. [Live-codebench: Holistic and contamination free evaluation of large language models for code](#). In *The Thirteenth International Conference on Learning Representations*.
- Michael I. Jordan and Robert A. Jacobs. 1994. [Hierarchical mixtures of experts and the em algorithm](#). *Neural Computation*, 6(2):181–214.

- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. [Scaling laws for neural language models](#). *Preprint*, arXiv:2001.08361.
- Mike Lasby, Ivan Lazarevich, Nish Sinnadurai, Sean Lie, Yani Ioannou, and Vithursan Thangarasa. 2025. [REAP the Experts: Why Pruning Prevails for One-Shot MoE compression](#). *arXiv preprint*, ArXiv:2510.13999v1 [cs].
- Lujun Li, Zhu Qiyuan, Jiacheng Wang, Wei Li, Hao Gu, Sirui Han, and Yike Guo. 2025a. [Sub-moe: Efficient mixture-of-expert llms compression via subspace expert merging](#). *Preprint*, arXiv:2506.23266.
- Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Junxian Guo, Xiuyu Li, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. 2025b. [SVDQuant: Absorbing outliers by low-rank component for 4-bit diffusion models](#). In *The Thirteenth International Conference on Learning Representations*.
- Zehua Liu, Han Wu, Ruifeng She, Xiaojin Fu, Xiongwei Han, Tao Zhong, and Mingxuan Yuan. 2025. [Molae: Mixture of latent experts for parameter-efficient language models](#). *Preprint*, arXiv:2503.23100.
- Yu-Chen Lu, Sheng-Feng Yu, Hui-Hsien Weng, Pei-Shuo Wang, Yu-Fang Hu, Liang Hung-Chun, Hung-Yueh Chiang, and Kai-Chiang Wu. 2025. [Skipcat: Rank-maximized low-rank compression of large language models via shared projection and block skipping](#). *Preprint*, arXiv:2512.13494.
- Shuqing Luo, Pingzhi Li, Jie Peng, Yang Zhao, Yu Cao, Yu Cheng, and Tianlong Chen. 2025. [Occult: Optimizing collaborative communications across experts for accelerated parallel moe training and inference](#). In *Forty-second International Conference on Machine Learning*.
- Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2025. [Large language models: A survey](#). *Preprint*, arXiv:2402.06196.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. [GPQA: A graduate-level google-proof q&a benchmark](#). In *First Conference on Language Modeling*.
- Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. [Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning](#). In *The Thirteenth International Conference on Learning Representations*.
- GLM Team, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, and 152 others. 2025. [Glm-4.5: Agentic, reasoning, and coding \(arc\) foundation models](#). *Preprint*, arXiv:2508.06471.
- Qwen Team. 2025. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Xin Wang, Samiul Alam, Zhongwei Wan, Hui Shen, and Mi Zhang. 2025a. [SVD-LLM v2: Optimizing singular value truncation for large language model compression](#). In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4287–4296, Albuquerque, New Mexico. Association for Computational Linguistics.
- Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. 2025b. [SVD-LLM: Truncation-aware singular value decomposition for large language model compression](#). In *International Conference on Learning Representations (ICLR)*.
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhui Chen. 2024. [Mmlu-pro: a more robust and challenging multi-task language understanding benchmark](#). In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS ’24, Red Hook, NY, USA. Curran Associates Inc.
- Zhewei Yao, Xiaoxia Wu, Cheng Li, Stephen Youn, and Yuxiong He. 2023. [Zeroquant-v2: Exploring post-training quantization in llms from comprehensive study to low rank compensation](#). *Preprint*, arXiv:2303.08302.
- Zhihang Yuan, Yuzhang Shang, Yue Song, Dawei Yang, Qiang Wu, Yan Yan, and Guangyu Sun. 2025. [ASVD: Activation-aware singular value decomposition for compressing large language models](#).
- Geng Zhang, Yuxuan Han, Yuxuan Lou, Yiqi Zhang, Wangbo Zhao, and Yang You. 2026. [Mone: Replacing redundant experts with lightweight novices for structured pruning of moe](#). *Preprint*, arXiv:2507.00390.
- Mohan Zhang, Pingzhi Li, Jie Peng, Mufan Qiu, and Tianlong Chen. 2025. [Advancing MoE efficiency: A collaboration-constrained routing \(C2R\) strategy for better expert parallelism design](#). In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6815–6825, Albuquerque, New Mexico. Association for Computational Linguistics.

Yifan Zhang and Team Math-AI. 2024. American invitational mathematics examination (aime) 2024.

Yifan Zhang and Team Math-AI. 2025. American invitational mathematics examination (aime) 2025.

Size Zheng, Wenlei Bao, Qi Hou, Xuegui Zheng, Jin Fang, Chenhui Huang, Tianqi Li, Haojie Duanmu, Renze Chen, Ruifan Xu, Yifan Guo, Ningxin Zheng, Ziheng Jiang, Xinyi Di, Dongyang Wang, Jianxi Ye, Haibin Lin, Li-Wen Chang, Liqiang Lu, and 3 others. 2025. [Triton-distributed: Programming overlapping kernels on distributed ai systems with the triton compiler](#). *Preprint*, arXiv:2504.19442.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

Zeliang Zong, Kai Zhang, Zheyang Li, Wenming Tan, Ye Ren, Yiyan Zhai, and Jilin Hu. 2025. [1+1>2: A synergistic sparse and low-rank compression method for large language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 14206–14220, Suzhou, China. Association for Computational Linguistics.

A Ethics and Broader Impact Statement

This work targets more memory- and latency-efficient deployment of large MoE models. Positive impacts include lowering the hardware and energy footprint required to serve LLMs, which can reduce cost and environmental impact and enable broader access to capable models. The new joint structured sparse compensation mechanism may inspire further hardware optimizations for efficient MoE inference.

Potential Risks By lowering the cost of inference, the method may also increase the scale or frequency of deployment and thereby amplify risks already present in the underlying models.

B SkipCat-MoE Comparison

We compare two ways of applying SkipCat to MoE compression. **SkipCat-MoE** merges experts with a shared SkipCat basis but keeps the expert-specific factors separate, while **SkipCat** merges gate and up projections within each expert. Figure 9 shows that the two variants are close in overall quality at the same CR, with slight differences across individual benchmarks. We use the simpler SkipCat setting in the main table and focus on NimbleMoE for further gains.

Metric	SkipCat-MoE	SkipCat
CR	23%	23%
MMLU-Pro	71.38	72.80
MMLU-Redux	86.23	86.80
C-Eval	80.80	80.25
IFEval	81.81	79.74
Math500	93.76	93.44
GPQA-D	53.84	55.76
LiveCode	32.31	30.66
IQuiz	65.67	68.00
AIME24	53.33	53.33
AIME25	44.00	44.67
Avg	66.31	66.55

Table 6: SkipCat-MoE vs. SkipCat on Qwen3-30B-A3B-Instruct-2507 at comparable CR.

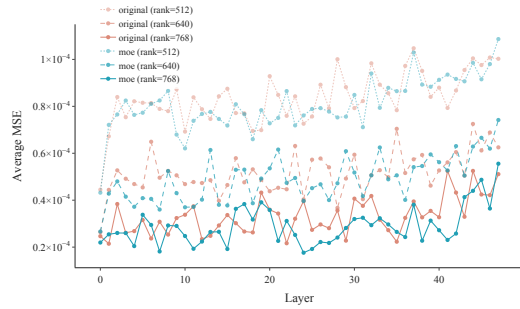


Figure 9: SkipCat-MoE vs. SkipCat on Qwen3-30B-A3B-Instruct-2507 at comparable CR. The two variants are close in average accuracy, indicating that expert-wise gate/up merging is a strong baseline.

C NimbleMoE Optimization Details

Following the SkipCat decomposition (Li et al., 2025a), we first perform a column permutation using Strong Rank-Revealing QR factorization (Gu and Eisenstat, 1996).

Given a group of experts $\{W_i\}_{i \in g}$, the optimization objective $\mathcal{L} = \|(W - \widetilde{W})Q\|_F^2$ can be rewritten as

$$\mathcal{L} = \sum_{i \in g} \|\widetilde{W}_i - A_i[I_r \mid B_g]\|_F^2, \quad (14)$$

where $\widetilde{W}_i = W_i Q$ is the permuted weight. Expanding the objective gives:

$$\mathcal{L} = \sum_{i \in g} \left(\|\widetilde{W}_{i,1} - A_i\|_F^2 + \|\widetilde{W}_{i,2} - A_i B_g\|_F^2 \right), \quad (15)$$

where $\widetilde{W}_{i,1} = \widetilde{W}_i[:, :r]$ and $\widetilde{W}_{i,2} = \widetilde{W}_i[:, r:]$. For initialization, we set $A_i = \widetilde{W}_{i,1}$, which minimizes the first term to zero. Then, optimizing B_g reduces

to solving the least squares problem:

$$\min_{B_g} \sum_{i \in g} \|\widetilde{W}_{i,2} - \widetilde{W}_{i,1} B_g\|_F^2. \quad (16)$$

The closed-form solution is given by:

$$B_g = \left(\sum_{i \in g} \widetilde{W}_{i,1}^\top \widetilde{W}_{i,1} \right)^{-1} \left(\sum_{i \in g} \widetilde{W}_{i,1}^\top \widetilde{W}_{i,2} \right). \quad (17)$$

This provides the optimal B_g under the fixed A_i initialization.

Alternating optimization can further refine both A_i and B_g . However, in practice, since optimization of Monarch-style sparse residuals require Gradient Descent, we find that one-shot least squares initialization suffices to achieve strong performance.

Warm-start for collaboration-aware clustering.

Using weights $w_{ij} := C_{ij}$ (or $w_{ij} := P_{\text{sh}} C_{ij}$), the assignment problem solved at compression time is

$$\max_z \sum_{g=1}^{K_g} \sum_{1 \leq i < j \leq N} w_{ij} z_{ig} z_{jg}. \quad (18)$$

We default to a hierarchical-matching warm-start: solve max-weight perfect matching to obtain $N/2$ pairs, then build a meta-graph on pairs whose weight is the sum of the four cross-pair edges and recurse until groups of size G are formed. As an alternative, the standard McCormick linearization with $y_{ijg} \in [0, 1]$ gives the LP we can also use:

$$\begin{aligned} \max_{z, y} \quad & \sum_{g=1}^{K_g} \sum_{1 \leq i < j \leq N} w_{ij} y_{ijg} \\ \text{s.t.} \quad & \sum_g z_{ig} = 1 \forall i, \quad \sum_i z_{ig} = G \forall g, \\ & y_{ijg} \leq z_{ig}, \quad y_{ijg} \leq z_{jg} \quad \forall i < j, \forall g, \\ & y_{ijg} \geq z_{ig} + z_{jg} - 1 \quad \forall i < j, \forall g, \\ & z_{ig}, y_{ijg} \in [0, 1]. \end{aligned} \quad (19)$$

Rounding via $i \mapsto \arg \max_g z_{ig}$ followed by group-size repair produces an integral feasible partition. Either warm-start initializes the hill-climbing swap search described in Sec. 3.1; we use hierarchical matching by default since it is both faster and produces a better initialization in our ablation (Appendix D).

D Grouping Algorithm Ablation

The collaboration-aware clustering step (Algorithm 2) instantiates Eq. (18), which is NP-hard for $G \geq 3$. We compare grouping strategies

on layer 12 of Qwen3-30B-A3B-Instruct-2507 ($N=128$, top-8) with the same edge weights $w_{ij} = C_{ij}$ used at compression time, where C_{ij} is the co-activation fraction of Sec. 3.1, under a 120 s per-run budget. We report p_{same}^W , the within-group share of total edge weight, against the degree-bounded ceiling (each expert has at most $G-1$ within-group neighbors): 0.1159 for $G=2$ and 0.3263 for $G=4$.

Algorithm	p_{same}^W	Time
$G=2$ (ceiling 0.1159)		
Random	0.0007	< 0.01 s
Greedy	0.1108	< 0.01 s
Max-weight matching	0.1112	0.93 s
Hill-climbing (warm: matching)	0.1112	21 s
$G=4$ (ceiling 0.3263)		
Random	0.0070	< 0.01 s
Greedy quads	0.2224	0.04 s
LP relaxation + rounding	0.2809	52 s
Hierarchical matching	0.3142	1.1 s
Hill-climbing (warm: hier.)	0.3164	61 s

Table 7: Grouping algorithms on layer-12 of Qwen3-30B-A3B-Instruct-2507. The chosen variants reach 95.9% ($G=2$) and 97.0% ($G=4$) of the degree-bounded ceiling.

This supports the two design choices in Algorithm 2: when $G=2$, max-weight matching (Blossom) is exact and near-optimal, so we short-circuit to it; when $G \geq 3$, hill-climbing warm-started from a hierarchical-matching initializer reaches 97% of the ceiling, beating LP rounding at lower cost.

Does higher p_{same}^W actually reduce error? Table 7 shows that our optimizer attains near-ceiling p_{same}^W , but not that this proxy translates into lower reconstruction error. We verify the link directly on layer 44 of Qwen3-30B-A3B-Instruct-2507 ($N=128$), the same setting as Figure 5, sweeping rank $r \in \{384, 512, 640, 704, 768\}$ for both compensators and both group sizes. The contiguous baseline partitions experts by index; collaboration-aware grouping raises p_{same}^W from 0.003 to 0.090 at $G=2$ (29 $\times$) and from 0.024 to 0.234 at $G=4$ (10 $\times$).

The two groupings incur identical parameter budgets at matched rank, since grouping determines which experts share a basis rather than how many parameters the basis costs. Every point in Figure 10 is therefore a controlled comparison at equal cost, with the partition as the only variable.

Collaboration-aware grouping reduces error at all 20 operating points, with no crossover. The reductions are consistent but modest: 1.26–1.67%

(*OSC*, $G=2$), 1.27–1.62% (*OSC*, $G=4$), 0.44–1.56% (*LSC*, $G=2$), and 0.15–0.71% (*LSC*, $G=4$). *OSC* benefits roughly 2–4 $\times$ more than *LSC*, consistent with the alignment analysis of Sec. 3.1: *LSC*’s per-expert latent projection partially absorbs a poorly chosen partition, whereas *OSC* shares its basis across the full output dimension and is correspondingly more sensitive to the partition. This complements the +0.44 end-to-end accuracy contribution of collaboration-aware grouping in Table 3. The evidence is limited to one layer, one pool size, and reconstruction error rather than downstream accuracy.

E Complete Forward Pass of NimbleMoE

The complete forward pass pseudocode of NimbleMoE is summarized in Algorithm 3. The collaboration-aware clustering subroutine invoked during compression (Alg. 1) is given below.

Algorithm 2 Collaboration-Aware Clustering (warm-started balanced hill-climbing)

- 1: **Input:** $C \in \mathbb{R}^{N \times N}$, group size G ($K_g \leftarrow N/G$)
 - 2: **Build:** $w_{ij} \leftarrow C_{ij}$ for all $i < j$
 - 3: **if** $G = 2$ **then**
 - 4: Solve max-weight perfect matching with weights w_{ij} (Blossom); return matched pairs
 - 5: **end if**
 - 6: **Warm-start (hierarchical matching, default):** Solve max-weight matching to obtain pairs; build a meta-graph on pairs with cross-pair edge weights and recursively match to form groups of size G .
Alternative (LP): Solve the relaxation of Eq. (19), round $\hat{g}(i) \leftarrow \arg \max_g z_{ig}$, and repair group sizes.
 - 7: **repeat**
 - 8: Pick groups $p \neq q$ and experts $u \in G_p$, $v \in G_q$; propose swap $u \leftrightarrow v$
 - 9: $\Delta \leftarrow (\hat{\mathcal{R}}_t(G'_p) + \hat{\mathcal{R}}_t(G'_q)) - (\hat{\mathcal{R}}_t(G_p) + \hat{\mathcal{R}}_t(G_q))$ (Eq. (3))
 - 10: **if** $\Delta > 0$ **then**
 - 11: Accept swap: $G_p \leftarrow G'_p$, $G_q \leftarrow G'_q$
 - 12: **end if**
 - 13: **until** no improving swap (or budget reached)
 - 14: **Output:** balanced groups $\{G_g\}_{g=1}^{K_g}$
-

Algorithm 3 NimbleMoE Forward Pass

Input: $x \in \mathbb{R}^H$, selected experts $\{(e_k, w_k)\}_{k=1}^K$

Output: $y \in \mathbb{R}^H$

Group experts by cluster: $\mathcal{G} = \{\mathcal{E}_g\}$

$y \leftarrow \mathbf{0}$

for each group $g \in \mathcal{G}$ **do**

$\tilde{x}_g \leftarrow P_g^\top x$

$z_g \leftarrow \tilde{x}_g[r] + B_g \cdot \tilde{x}_g[r:]$

$z_g^{\text{gate}} \leftarrow z_g + M_g^{\text{gate}} \cdot \tilde{x}_g$

$z_g^{\text{up}} \leftarrow z_g + M_g^{\text{up}} \cdot \tilde{x}_g$

$\bar{a}_g \leftarrow \mathbf{0}$

for each expert $e_k \in \mathcal{E}_g$ **do**

$\text{gate}_{e_k} \leftarrow A_{e_k}^{\text{gate}} \cdot z_g^{\text{gate}}$

$\text{up}_{e_k} \leftarrow A_{e_k}^{\text{up}} \cdot z_g^{\text{up}}$

$h_{e_k} \leftarrow \sigma(\text{gate}_{e_k}) \odot \text{up}_{e_k}$

$a_{e_k} \leftarrow A_{e_k}^{\text{down}} \cdot P_g^\top h_{e_k}$

$\bar{a}_g \leftarrow \bar{a}_g + w_k \cdot a_{e_k}$

$y \leftarrow y + w_k \cdot M_g^{\text{down}} \cdot h_{e_k}$

end for

$y \leftarrow y + B_g \cdot \bar{a}_g$

end for

Return y

F Latency Benchmark Details

We benchmark NimbleMoE on three hardware regimes and three workloads: (i) the MoE MLP forward in isolation, (ii) end-to-end generation through vLLM, and (iii) multi-GPU expert parallelism via Triton-distributed (Zheng et al., 2025). The single-GPU MoE-MLP benchmark is adapted from triton’s official implementation⁵, we measure the MoE MLP forward pass using the same routing (top- K_r), expert dimensions, and group size as the Qwen3-30B-A3B-Instruct-2507 setting in our main experiments. All methods share identical inputs and decoding settings; only expert weights and shared components differ due to compression.

Measurement protocol. We use CUDA events to time the forward pass, discard warmup iterations, and report the median latency across repeated runs. We keep precision and kernel launch settings identical across methods. End-to-end runs use vLLM as the serving backend with the NimbleMoE-compressed model loaded through standard scheduling, KV-cache management, and sampling; only the MoE MLP forward pass is re-

⁵https://github.com/triton-lang/triton/blob/main/python/triton_kernels/bench/bench_mlp.py

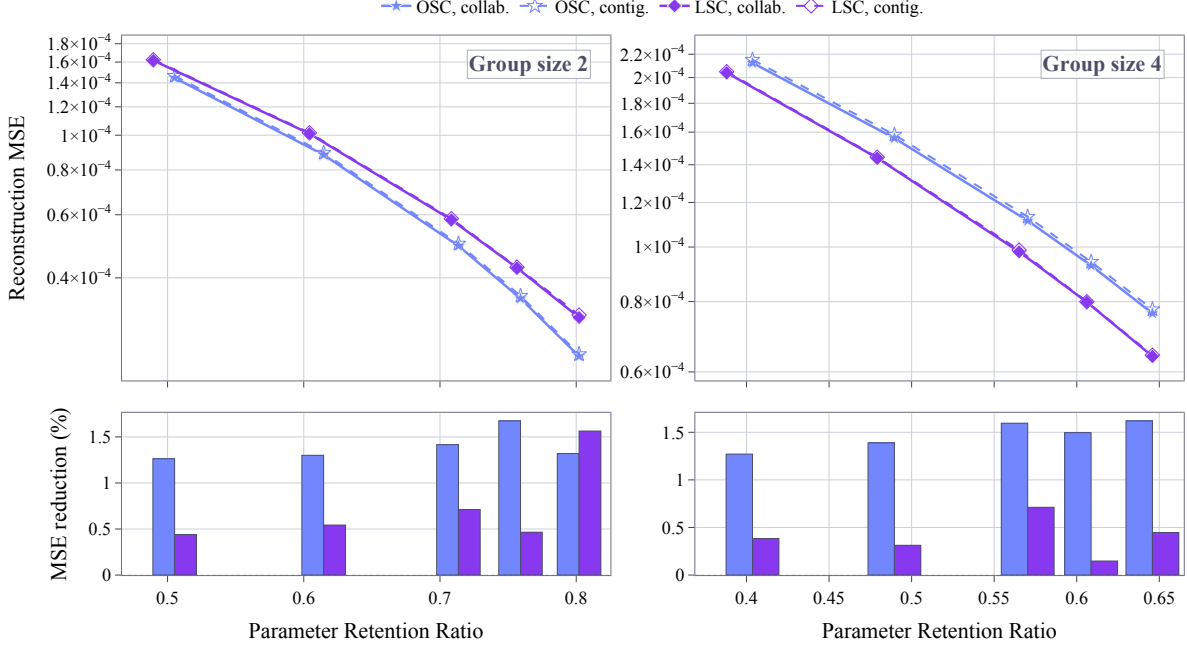


Figure 10: Collaboration-aware vs. contiguous grouping on layer 44 of Qwen3-30B-A3B-Instruct-2507 ($N=128$), for rank $r \in \{384, \dots, 768\}$. The horizontal axis is the retained parameter fraction (lower is more compressed), the same quantity plotted in Figure 5 and the complement of the CR column in Table 2. **Top:** reconstruction MSE. The two groupings sit at identical budgets at matched rank, so the comparison is controlled and the curves are near-coincident on a log axis. **Bottom:** the same data as relative MSE reduction, where the effect is legible. Collaboration-aware grouping reduces error at all 20 operating points at no parameter cost, with *OSC* benefiting more than *LSC*.

placed by our Triton kernels.

H200 single-GPU. Table 8 reports MoE-MLP kernel latency on H200 against the uncompressed model and the two compression baselines; end-to-end TTFT and TPOT for both H200 and A800 are reported in Figure 8 of the main paper. NimbleMoE is $1.18\text{--}1.49\times$ faster than the standard MoE, with the margin widening as batch size grows. HC-SMoE tracks the standard MoE closely ($0.92\text{--}1.13\times$), as expected since it reduces total but not activated parameters. MoBE is $1.94\text{--}2.41\times$ slower than the uncompressed model despite compressing it, because it materializes expert weights from its shared basis at every forward pass and raises activated cost by $1.25\times$ (Table 5). MoBE provides no official inference kernel, so we implemented an optimized version with fused basis materialization and vectorized padded BMM; the reported numbers use that implementation. The kernel sweep on RTX 4090 (Figure 7 in the main paper) includes a clustering ablation that isolates the gain from co-activation-aware grouping.

Method	b=8	b=16	b=32	b=64	b=128
Standard MoE	0.555	0.564	0.696	0.681	0.728
HC-SMoE	0.557	0.612	0.614	0.643	0.711
MoBE	1.338	1.313	1.348	1.402	1.473
NimbleMoE	0.469	0.474	0.501	0.485	0.487
<i>Speedup</i>	$1.18\times$	$1.19\times$	$1.39\times$	$1.40\times$	$1.49\times$

Table 8: H200 MoE-MLP kernel latency (ms, median over 300 timing iterations) on Qwen3-30B-A3B. *Speedup* is NimbleMoE over the standard MoE. Lower latency is better. MoBE is slower than the uncompressed model at every batch size; HC-SMoE is roughly latency-neutral, since it does not reduce activated parameters.

A800 single-GPU. On A800, the MoE MLP is more bandwidth-bound and NimbleMoE’s gains become substantial across the entire batch range:

Expert parallelism. Collaboration-aware clustering co-locates frequently co-activated experts, which is structurally expected to reduce all-to-all traffic. Table 4 reports MoE-MLP kernel speedups on a single H100 node using Triton-distributed across world sizes $\{2, 4, 8\}$ and batch sizes $\{8, 32, 128, 512\}$. NimbleMoE is compatible

Batch	NimbleMoE (ms)	Standard (ms)	Speedup
8	0.678	0.741	1.09×
32	0.746	1.101	1.48×
128	0.784	1.308	1.67×
512	0.943	1.691	1.79×

Table 9: A800 MoE-MLP kernel latency on Qwen3-30B-A3B.

with EP and its speedup grows with world size, reaching $2.81\times$ at $WS=8$. Since the reference is vanilla MoE rather than a matched-CR contiguous-grouping arm, these numbers reflect the combined effect of compression and co-location and should not be read as isolating the latter.

GPU memory. NimbleMoE’s structural compression maps cleanly to allocated GPU memory: on Qwen3-30B-A3B (H200), allocated memory drops from 56.87 GB to 44.21 GB (22.3% reduction) after model load, and peak allocated drops from 57.20 GB to 44.54 GB (22.1% reduction). Unlike MoBE, whose basis sharing adds overhead that erodes nominal CR, NimbleMoE delivers the full target compression in deployed memory.

Metric (GB)	Vanilla	NimbleMoE	Reduction
After load	56.87	44.21	22.3%
Peak allocated	57.20	44.54	22.1%

Table 10: GPU memory measurements on H200 (Qwen3-30B-A3B).

G Additional Routing and Collaboration Statistics

We provide the complete expert activation frequency heatmap on Qwen3-30B-A3B-Instruct-2507 evaluated on Wikitext-2 in Figure 12. As shown in Figure 14, the top-routed experts are often also co-activated with each other.

Also, we conduct the same analysis on GLM4.7-Flash, which has 64 experts and a shared expert. The complete expert activation frequency heatmap is shown in Figure 13.

H Higher-CR Study on GLM-4.7-Flash

GLM-4.7-Flash has only 64 experts (top-4 activated per token), giving it lower per-group co-activation density than the Qwen3 backbones (top-8/128). It also has correspondingly less weight redundancy, which makes any compression method

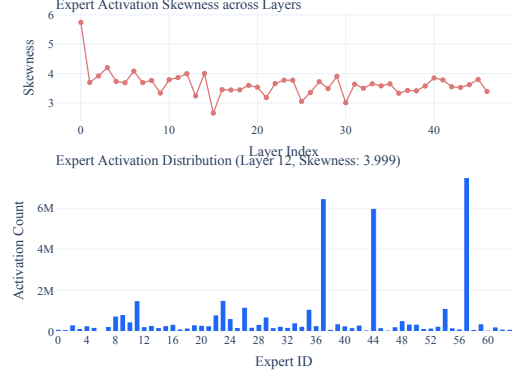


Figure 11: Expert activation frequency skewness and distribution on GLM4.7-Flash.

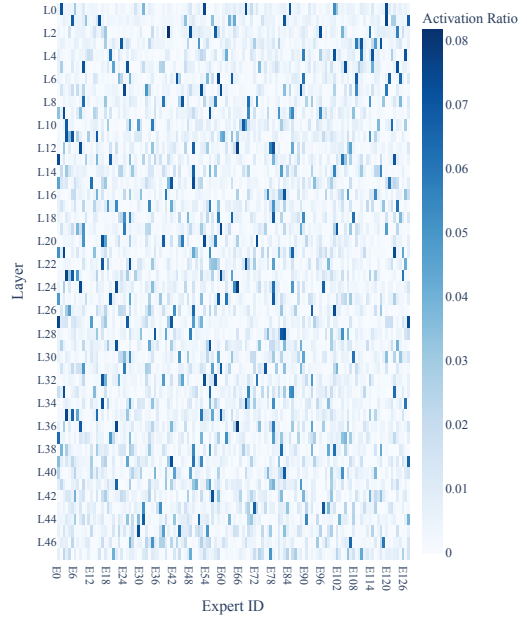


Figure 12: Complete Expert activation frequency heatmap on Qwen3-30B-A3B-Instruct-2507 evaluated on Wikitext-2.

more destructive at higher CR. Table 2 therefore reports GLM-4.7-Flash at 20% CR; here we additionally report the 27%-CR setting (Table 11). NimbleMoE leads MoBE by +13.0 pts and SkipCat by +10.9 pts at this aggressive CR, with the gap especially large on reasoning benchmarks (AIME24 +19.3, AIME25 +16.0 over MoBE).

I Multi-Trial AIME Evaluation on a Thinking Model

The main table averages 5 samples per problem, which still leaves high variance on a 30-problem benchmark (Appendix K). To check that NimbleMoE’s reasoning gains hold under stricter sampling protocols, we additionally evaluate Qwen3-

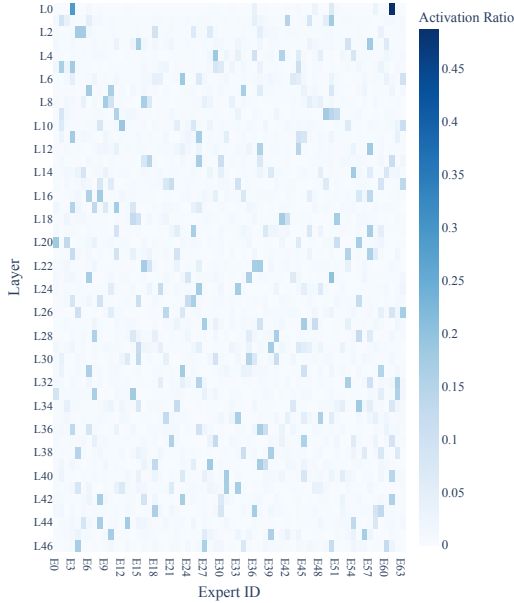


Figure 13: Complete Expert activation frequency heatmap on GLM4.7-Flash evaluated on Wikitext-2.

Benchmark	MoE	SkipCat	MoBE	NimbleMoE
CR	—	27%	27%	27%
MMLU-Pro	67.17	44.41	47.85	58.56
MMLU-Redux	82.93	64.37	66.97	78.63
C-Eval	73.33	57.80	56.79	64.86
IQuiz	69.17	50.00	40.00	55.33
GPQA-Diamond	44.44	19.19	24.24	36.36
IFEval	79.41	67.28	61.74	75.51
LiveCodeBench	25.93	17.03	18.02	24.45
Math500	89.04	64.80	65.64	82.32
AIME24	36.67	13.33	6.67	26.00
AIME25	28.67	16.67	6.00	22.00
Avg	59.68	41.49	39.39	52.40

Table 11: GLM-4.7-Flash benchmark results at the more aggressive 27% compression ratio. NimbleMoE outperforms both SkipCat and MoBE by > 10 pts on average and preserves substantially more reasoning capacity.

30B-A3B-Thinking-2507 at 25% CR with multi-trial decoding and report average pass@1, average pass@4, and pass@32 on AIME24 and AIME25 (Table 12). NimbleMoE leads on average pass@1 on both datasets, matches or exceeds the uncompressed model at pass@32, and is consistently stronger than MoBE across all granularities.

J Robustness to Calibration Distribution

NimbleMoE’s grouping decisions are derived from routing statistics on a calibration corpus. We measure how robust those decisions are to the choice of calibration data, both at the statistic level (Spearman correlation across corpora) and at the down-

Method	AIME24			AIME25		
	p@1	p@4	p@32	p@1	p@4	p@32
Vanilla	0.737	0.864	0.900	0.653	0.770	0.900
MoBE	0.723	0.881	0.933	0.637	0.790	0.933
NimbleMoE	0.734	0.865	0.933	0.673	0.804	0.933

Table 12: Multi-trial AIME results on Qwen3-30B-A3B-Thinking-2507 at 25% CR. p@1 and p@4 are averaged pass rates; p@32 is single pass@32. **Bold** marks the best among compressed methods. NimbleMoE leads on averaged pass@1 for both datasets, matches or exceeds the vanilla model at pass@32, and is consistently stronger than MoBE.

stream level (accuracy and speedup under different calibration sets).

Co-activation is more stable than activation frequency. Table 13 reports Spearman correlations between calibration corpora. Co-activation patterns are consistently more stable than raw activation frequencies, and the gap widens for far OOD pairs (Chinese vs. Code/Math), where activation-frequency correlation collapses below 0.25 but co-activation stays above 0.35.

Dataset pair	Activ. freq.	Co-activ.
Test set vs. C4	0.515	0.666
Test set vs. WikiText	0.546	0.681
C4 vs. WikiText	0.790	0.868
Chinese vs. Code	0.214	0.350
Chinese vs. Math	0.238	0.392
Code vs. Math	0.756	0.726

Table 13: Cross-corpus Spearman correlation of expert activation frequency vs. co-activation.

Calibration choice barely affects downstream accuracy or speedup. Table 14 runs the full NimbleMoE pipeline using three different calibration corpora and evaluates on our standard benchmark suite (avg. across the 10 tasks of Table 2) and on the A800 batch-1024 latency benchmark. The spread is within 0.1 pts of accuracy and $0.02\times$ of speedup.

Calibration set	Avg. accuracy	Speedup @1024
WikiText	70.25	1.42 $\times$
C4	70.22	1.41 $\times$
Test-set oracle	70.32	1.43 $\times$

Table 14: Downstream accuracy (avg. over Table 2) and A800 batch-1024 speedup w.r.t. calibration corpus.

Grouping quality under domain shift. When grouping is fit on WikiText and evaluated on out-

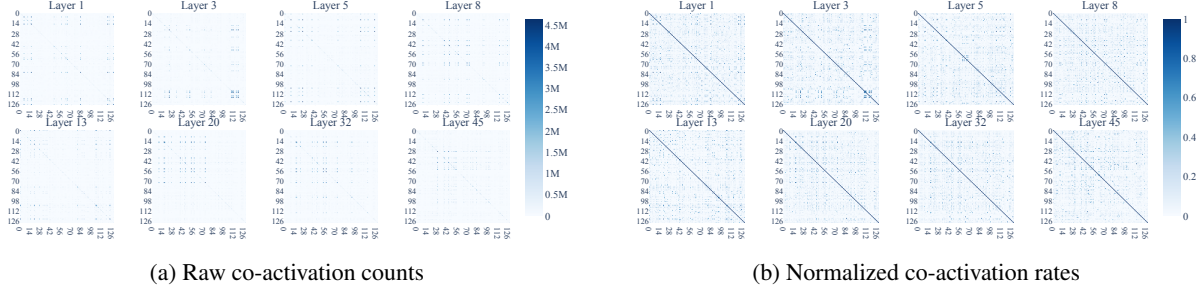


Figure 14: Expert co-activation heatmap for different layers on Qwen3-30B-A3B-Instruct-2507 evaluated on Wikitext-2.

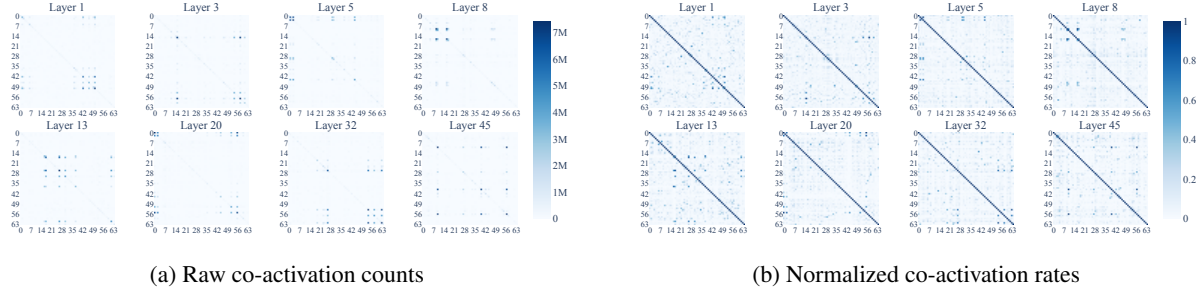


Figure 15: Expert co-activation heatmap for different layers on GLM4.7-Flash evaluated on Wikitext-2.

of-distribution domains, $p_{\text{same-group}}$ (the fraction of activated experts within a token’s selected K_r that fall into the same group) remains $5\text{--}7\times$ higher than contiguous grouping (Table 15). NimbleMoE preserves all experts and the original router, unlike importance-based pruning, which permanently discards experts deemed unimportant under one calibration. Across the domains we evaluate, grouping quality stays above the contiguous-grouping ablation in Table 3, which itself outperforms all baselines at the same CR. This is an empirical observation on the evaluated domains rather than a worst-case guarantee.

Eval set	WikiText-grouped	Contig.	Δ
WikiText	0.536	0.054	+0.482
Chinese	0.322	0.059	+0.263
Code	0.339	0.056	+0.283
Math	0.386	0.051	+0.334

Table 15: $p_{\text{same-group}}$ measured on each evaluation domain when grouping was derived from WikiText calibration.

K Uncertainty Estimates

Table 2 reports micro-averaged accuracy over 5 samples per problem. Because the benchmarks differ in size by two orders of magnitude, the same numerical gap carries different evidential weight

across columns. Table 16 reports standard errors and 95% confidence intervals for the uncompressed Qwen3-30B-A3B-Instruct-2507 reference. Standard errors are cluster-robust, treating the 5 samples of a problem as correlated rather than independent (implied intra-problem correlation $\rho \approx 0.6\text{--}0.8$); a naive binomial over all $5N$ attempts would understate them by roughly $2\times$. Benchmark sizes, and hence interval widths, are identical across the other three backbones.

Interval widths vary substantially. MMLU-Pro ($N=12032$) resolves differences below half a point, whereas AIME24 and AIME25 ($N=30$) carry intervals of roughly 27 and 31 points. At 5 samples their effective sample sizes are only ≈ 44 and ≈ 38 independent items. Per-benchmark AIME differences below roughly ten points should therefore not be interpreted as capability differences in either direction, including those in Table 2. Propagating these standard errors through the 10-benchmark mean yields ± 1.21 points on the **Avg.** column. On that scale the accuracy cost of compression is model-dependent: the reductions on Ling-2.6-Flash (-0.28) and Qwen3-235B (-0.71) fall within noise, whereas those on Qwen3-30B (-2.74) and GLM-4.7-Flash (-4.18) do not.

Benchmark	N	Acc.	SE	95% CI
MMLU-Pro	12032	74.36	0.36	[73.7, 75.1]
MMLU-Redux	5700	88.56	0.38	[87.8, 89.3]
C-Eval	1346	87.00	0.84	[85.3, 88.7]
IQuiz	120	74.83	3.72	[67.5, 81.7]
GPQA-Diamond	198	52.73	3.04	[46.9, 58.8]
IFEval (strict)	541	83.99	1.46	[81.0, 86.8]
LiveCodeBench	182	36.92	3.35	[30.3, 43.6]
MATH-500	500	96.88	0.69	[95.4, 98.1]
AIME24	30	72.00	6.75	[58.0, 84.7]
AIME25	30	62.67	7.89	[47.3, 77.3]
Avg.	—	72.99	1.21	—

Table 16: Per-benchmark uncertainty for the uncompressed Qwen3-30B-A3B-Instruct-2507 reference, over 5 samples per problem. Accuracy values reproduce the vanilla row of Table 2. Standard errors are cluster-robust at the problem level; 95% CIs are the corresponding normal intervals. These are *marginal* intervals for a single system and should not be differenced across methods. Since all methods are evaluated on the same problems, comparing two systems requires a paired test, whose interval is tighter because problem difficulty cancels.

L Hyperparameter Sensitivity and Offline Cost

Table 17 reports accuracy over the same ten benchmarks as Table 2 for four configurations of Qwen3-30B-A3B-Thinking-2507, held at closely matched compression ratios of 25.59–26.55%. The sweep varies the low-rank truncation $r \in [640, 960]$, the group size $G \in \{2, 4\}$ (with the number of groups N_g following), and the Monarch block counts (k_L, k_R) from (16, 16) to (32, 32).

Average accuracy spans 74.96–75.56, a total spread of 0.599 points, against a nominal reduction that varies by less than one percentage point. Group size is not brittle: moving from $G=2$ (75.556) to $G=4$ (75.497) changes the average by 0.059 points at a comparable budget. The lowest-capacity setting is the weakest but trails the best by only 0.599 points rather than collapsing. Two caveats bound this: the configurations vary several coupled quantities at once to hold nominal reduction fixed, so this is a matched-budget robustness study rather than a factorial ablation; and these are single evaluations, so the 30-problem AIME columns are far noisier than the ten-task average (Appendix K).

Offline compression cost. Compressing Qwen3-30B-A3B on a single H200 takes 7.8 GPU-hours for NimbleMoE against 5.6 for MoBE (1.39 $\times$), the overhead coming from jointly optimizing the low-rank and structured-sparse factors. This is a

Config ($r/G/N_g/k_L/k_R$)	CR	Avg.	Δ
<i>Uncompressed</i>	—	78.008	—
768/2/64/16/32	26.17%	75.556	−2.452
832/2/64/32/32	25.59%	75.303	−2.705
640/2/64/16/16	25.84%	74.957	−3.051
960/4/32/16/32	26.55%	75.497	−2.511

Table 17: Matched-budget hyperparameter sensitivity on Qwen3-30B-A3B-Thinking-2507. **Avg.** is the mean over the ten benchmarks of Table 2; Δ is the change from the uncompressed model. The total spread across configurations is 0.599 points.

one-time offline cost that does not affect deployed inference, but it is a real limitation for frequently updated checkpoints. Compression proceeds layer-wise, so the original and compressed models need not be co-resident; measured peak GPU memory is approximately 28.4 GB, set by the largest individual layer rather than by model size. This is the cost of the offline stage, as distinct from the deployed memory of the compressed model (Table 10).

M Use of AI Assistants

We mainly used Claude Code framework (with Claude-Sonnet-4.6) as an AI coding assistant during the development of this work. Its use was limited to debugging code and polishing English wording in the manuscript. All research ideas, experimental design, analysis, and conclusions are the authors’ own, and the authors verified all AI-assisted outputs.