ExFold: Unified Expert Folding for Training-Free MoE Prefill-Decode Acceleration

Juntong Wu [1, 2*], **Yifei Liu** [1, 3*], **Junyi Chen** [1, 3], **Siqi Fan** [1, 4], **Chaoran Feng** [2]
Minghao Li [1], **Liujie Zhang** [1], **Weihang Cheng** [1†], **Li Yuan** [2†]

[1] Xiaohongshu Inc.
[2] Shenzhen Graduate School, Peking University
[3] Shanghai Jiao Tong University
[4] University of Electronic Science and Technology of China
Correspondence: chenjinzhi@xiaohongshu.com, yuanli-ece@pku.edu.cn

Abstract

Mixture-of-Experts (MoE) models scale capacity for strong quality while keeping per-token compute bounded through sparse expert activation. Yet low-latency MoE serving is increasingly challenging, because it spans two inference phases with fundamentally different bottlenecks: prefill is dominated by token-wise expert computation, whereas decode is constrained by memory traffic from the batch-wise activated expert set. However, existing training-free acceleration methods optimize only a single resource proxy—either the experts each token executes or the experts a batch activates—and, either discard the excluded experts' contribution or leave it only implicitly approximated. In this paper, we propose ExFold, a unified training-free expert-folding framework for jointly accelerating MoE prefill and decode. ExFold casts both prefill and decode as one budgeted output-approximation problem: execute only a phase-specific constrained expert set while projecting the contribution of budget-excluded experts onto retained experts using calibrated scalar projectors. Motivated by the observation that many expert outputs are directionally aligned but differ in magnitude, ExFold calibrates a pairwise scalar-projector matrix on unlabeled data and uses it at inference time to fold excluded expert contributions into retained experts. Under this view, prefill acceleration becomes token-level Top-K folding, and decode acceleration becomes batch-level expert-pool folding. The two phases differ only in how retained experts are selected, while excluded contributions are recovered by one shared folding mechanism. We implement ExFold as a plug-and-play plugin in vLLM, with a lightweight expert-folding CUDA kernel, delivering up to 1.41× TTFT and 2.45× TPOT speedups while retaining about 99% of the original average quality. Code of ExFold can be seen in https://github.com/Time-Rune/ExFold-MoE.

Introduction

Mixture-of-Experts (MoE) scales model parameters by orders of magnitude while keeping the compute budget bounded by activating only a sparse subset of parameters per token (Fedus, Zoph, and Shazeer 2022; Jiang, Bressand et al. 2024). This property has made MoE the mainstream design of choice for many recent large models seeking higher capability (Jiang, Bressand et al. 2024; Qwen Team 2025; Z.ai 2025) . At the same time, it places stringent demands on low-latency serving (Chen et al. 2026, 2025b; Du et al. 2026) in production: both the prefill latency (time-to-first-token, TTFT) and the decode latency (time-per-output-token, TPOT) matter, and the two exhibit distinct computational characteristics. As shown in Fig. 1, prefill is dominated by token-level expert computation, whereas decode is dominated by batch-level expert memory traffic.

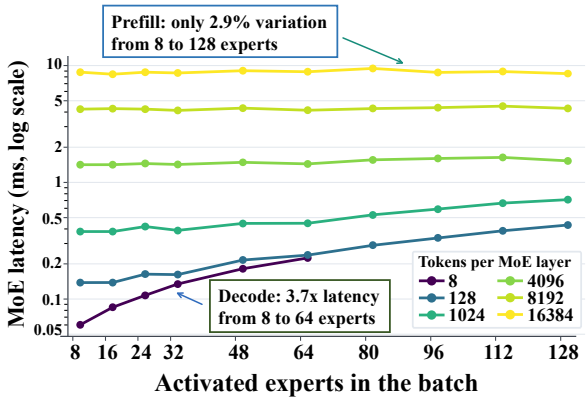


Figure 1: MoE prefill and decode have different bottlenecks.

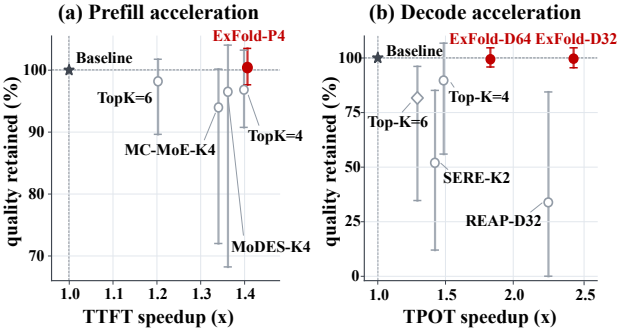


Figure 2: Quality–speed trade-offs for ExFold prefill and decode acceleration.

Existing training-free MoE acceleration methods fall into

*Equal contribution.
†Corresponding author.

two families, according to which of these two budgets they optimize. *Token-wise expert sparsification* reduces the number of experts each token executes, including Dynamic-MoE (Huang et al. 2024), MoDES (Huang et al. 2025), NAEE (Lu et al. 2024), and MC-MoE (Li et al. 2024). *Expert-set consolidation* instead shrinks the active expert set, through static pruning and merging such as REAP (Lasby et al. 2025), HC-SMoE (Chen et al. 2025a), and Sub-MoE (Li et al. 2025), and dynamic batch-level restriction such as Lynx (Gupta et al. 2024) and SERE (Wu et al. 2026).

Yet both families frame acceleration as a resource-reduction problem: **they answer only which experts to execute, and leave the harder question unaddressed—what becomes of the experts they cut**. Their treatment of the excluded contribution is, at best, incidental. Pruning and skipping discard it outright (Lu et al. 2024). Static merging bakes it into a single, permanently compressed model that can no longer adapt to the per-token or per-batch budget it actually faces at inference (Li et al. 2024). Similarity-based re-routing swaps an excluded expert for a nearby one, but never calibrates how far the substitute's output strays from the contribution it replaces (Wu et al. 2026). Lost, frozen, or only implicitly approximated, the excluded expert mass is never explicitly reconstructed—so the approximation error compounds as the budget is tightest and these methods are pushed hardest.

We introduce *Expert Folding*, which projects every budget-excluded expert contribution onto an executed expert instead of discarding it, and build ExFold around it. This design follows two empirical observations (Figure 3): many source experts have at least one target with directionally aligned outputs, while output magnitudes differ substantially across experts. The former makes expert substitution possible; the latter explains why direct re-routing is insufficient and motivates a directed scalar projector that corrects the source–target scale mismatch. Concretely, a single frozen forward pass over unlabeled text calibrates two per-layer matrices—a *scalar-projector matrix* and a *projection-loss matrix*: at inference, the loss matrix routes each excluded expert to its minimum-loss target, and the scalar matrix folds each excluded contribution into that target's router weight.

Since folding is defined at the level of individual excluded experts, it applies unchanged in both phases. Prefill selects K_{pre} dominant experts per token, and decode selects D dominant experts per batch—both reuse the same scalar matrix, loss matrix, and folding operator to recover whatever falls outside the retained set. This is the central design of ExFold: the two phases differ only in how retained experts are selected, while excluded contributions are recovered by one shared mechanism.

We implement ExFold as a plug-and-play plugin in vLLM, with a lightweight expert-folding CUDA kernel, delivering up to 1.41× TTFT and 2.45× TPOT speedups while retaining about 99% of the original average quality. Our contributions are as follows:

- **Phase-aware formulation.** We characterize the distinct expert budgets of prefill and decode and unify both as a single output-approximation problem under phase-specific selection constraints.
- **Expert Folding.** We propose a training-free folding mechanism that projects budget-excluded expert contributions onto retained experts via a directional projector and a reconstruction-loss matrix, whose scalar form is absorbed directly into the router weight.
- **Evaluation.** We validate near-lossless prefill, decode, and joint acceleration across multiple MoE architectures on real vLLM serving workloads.

Background and Motivation

MoE Inference

Sparse MoE layers. A sparse MoE layer replaces the dense FFN of a Transformer block with N gated-FFN experts and a router (Fedus, Zoph, and Shazeer 2022; Jiang, Bressand et al. 2024). For a hidden state x of dimension d, expert e computes

$$E_e(x) = W^{(e)}_{\mathrm{down}}\left(\phi\Big(W^{(e)}_{\mathrm{gate}}x\Big)\odot W^{(e)}_{\mathrm{up}}x\right), \qquad (1)$$

where $\phi(\cdot)$ is typically SiLU and $\odot$ denotes element-wise multiplication. The router produces logits $r(x) = W_r x$ over N routed experts, selects the Top-K set $S_K(x)$, and normalizes the gate weights within it:

$$\alpha_e(x) = \frac{\exp(r_e(x))}{\sum_{j\in S_K(x)}\exp(r_j(x))}, \quad e \in S_K(x). \qquad (2)$$

The MoE output is then

$$\mathrm{MoE}(x) = \sum_{e\in S_K(x)} \alpha_e(x)E_e(x), \qquad (3)$$

possibly with additional shared experts. This computation makes the FFN path sparse per token, but it also introduces dynamic routing, expert dispatch, and expert-output aggregation.

Phase-specific budgets. Although the same MoE layer (Eq. 3) runs in both phases, its bottleneck shifts with the shape of the token batch (Figure 1). Prefill processes a prompt of T tokens at once, and each token independently activates its own support $S_K(x)$, so the layer performs $\mathcal{O}(TK)$ expert-FFN evaluations. When T is large, every loaded expert is reused across many tokens, so weight-loading is amortized and the phase is compute-bound on token-wise expert FLOPs; reducing K directly cuts this cost. Decode advances each of B batched requests by a single token per step, issuing only B tokens, yet their supports $\bigcup_x S_K(x)$ typically cover most of the N experts. The layer must therefore load this batch-wise expert union to serve very few tokens each, yielding low arithmetic intensity and a memory-bound phase whose cost tracks the number of distinct activated experts rather than FLOPs. Efficient serving is thus governed by two different budgets: experts per token in prefill, and experts per batch in decode.

MoE Acceleration Methods

Depending on which budget they optimize, existing training-free MoE acceleration methods fall into two families: token-wise expert sparsification and expert-set consolidation.

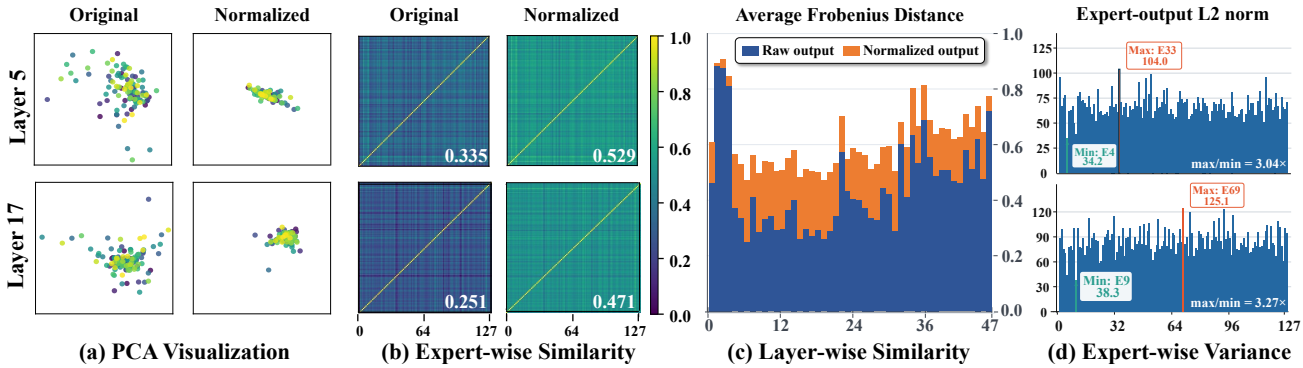


Figure 3: (a-c) MoE expert outputs can be aligned across experts and layers, and (d) their L2 norms vary substantially .

Token-wise expert sparsification. This family reduces the experts each token executes. Dynamic-MoE keeps per token the smallest expert set with cumulative router probability above p, so easier tokens use fewer experts (Huang et al. 2024). MoDES skips experts per token by a calibration-estimated importance (Huang et al. 2025), NAEE prunes low-importance executions from router or activation statistics (Lu et al. 2024), and MC-MoE merges co-activated experts identified from calibration (Li et al. 2024). These methods relieve the prefill computation budget, but leave the decode-time expert union uncontrolled.

Expert-set consolidation. This family shrinks the active expert set. Static methods score every expert from calibration signals and permanently reduce the global pool: REAP prunes the least important experts by router weight and activation norm (Lasby et al. 2025), while HC-SMoE (Chen et al. 2025a), REAM (Jha et al. 2026), and Sub-MoE (Li et al. 2025) merge similar experts. Dynamic methods restrict the expert union within a decoding batch: Lynx re-routes secondary requests to active experts (Gupta et al. 2024), and SERE calibrates pairwise expert similarity offline and substitutes each excluded expert with a similar retained one at decode (Wu et al. 2026). These methods relieve decode-side traffic, but their static transformations do not match the per-token prefill budget.

Gap. Both families decide *which* experts to execute, but neither preserves the contribution of the experts they exclude: pruning discards it, static merging bakes it into a permanently compressed model, and similarity re-routing changes an expert's destination without calibrating how far the substitute strays. As the budget tightens, this error grows within each phase and compounds across prefill and decode, which are approximated against two different targets rather than one shared objective.

Key Question

How can we design a unified framework to accelerate the prefill and decode of MoE models with minimal loss?

Design Insight

The gap makes recovery look hard from both sides—prefill and decode pull toward conflicting budgets, and faithfully restoring each excluded expert seems to require either recomputation or blind substitution—yet two observations dissolve both difficulties.

The excluded contributions share one recovery target. Prefill and decode are constrained by different budgets—experts per token versus experts per batch—yet whichever experts a budget excludes, the residual it leaves behind has the identical form $\alpha_e(x)E_e(x)$ in the MoE output (Eq. 3). The two phases therefore pose a single problem—reconstruct the excluded contributions from the retained experts—and differ only in the constraint that selects the retained set, not in what must be recovered.

Expert redundancy is magnitude-separable. Figure 3 shows that the redundancy across experts is concentrated in one cheaply correctable degree of freedom—output scale. In a two-component PCA projection (a), raw expert outputs form a diffuse cloud, but normalizing each to unit scale collapses them onto a thin shared axis: once scale is removed, experts point in nearly the same direction. Pairwise similarity confirms this—normalization lifts the average expert-to-expert similarity from 0.335 to 0.529 at layer 5 and from 0.251 to 0.471 at layer 17 (b)—and the same gap persists across all 48 layers (c), so the alignment is structural rather than a few-layer artifact. Yet the discarded scale is large: raw output norms span over $3\times$ within a layer ($\max / \min = 3.04\times$ and $3.27\times$ for the two shown, d). The mismatch between an excluded and a retained expert is therefore almost purely radial—a compatible direction is already available, only its magnitude is off. A single per-pair scalar that rescales the retained expert—not a reconstructed vector or a blind substitution—thus bounds how far the substitute strays and makes faithful recovery cheap in principle. Full-layer visualization is provided in Appendix D.

Together these give the motivating question an affirmative answer: one recovery target, reached by a cheap per-pair scalar correction and shared across both phases, with each phase changing only how the retained set is selected. ExFold turns this into a concrete mechanism in the next section.

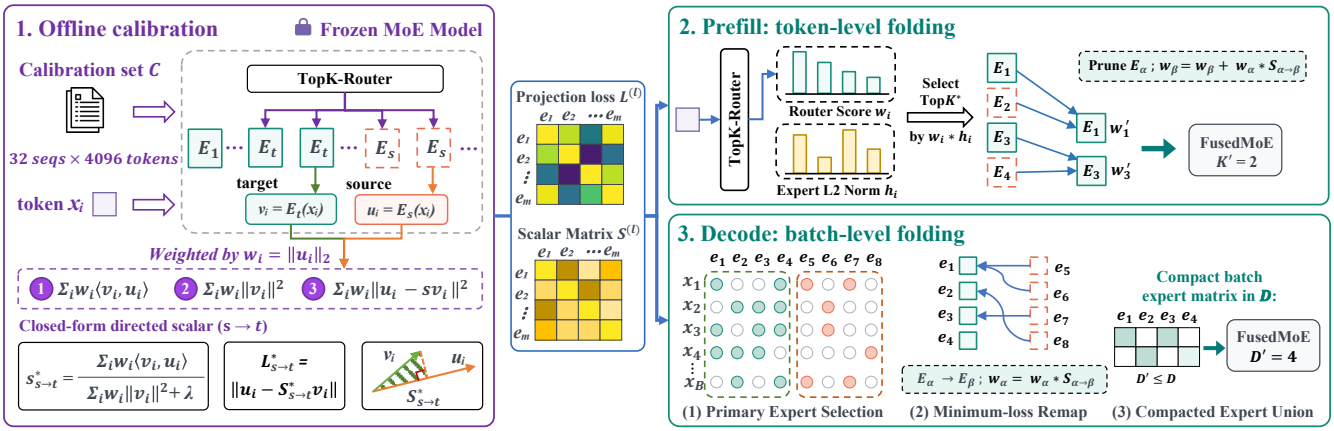


Figure 4: Overview of **ExFold**: training-free projector calibration and unified expert folding.

Method

Unified Expert Folding

ExFold treats MoE acceleration as constrained output approximation rather than phase-specific expert dropping. Let x_i denote the i-th input token and $S_K(x_i)$ its original Top-K support. We extend the router weight by $\alpha_e(x_i) = 0$ for $e \notin S_K(x_i)$. Under phase $\phi \in \{\text{pre}, \text{dec}\}$, $B_\phi(x_i)$ denotes the retained target set and $O_\phi(x_i) = S_K(x_i) \setminus B_\phi(x_i)$ the omitted source experts. We use E_s for an omitted source expert and E_t for its retained target, with $t = \pi_\phi(s, x_i) \in B_\phi(x_i)$.

Let $s^\star_{s\to t}$ be a directed scalar projector such that $E_s(x_i) \approx s^\star_{s\to t} E_t(x_i)$, and write $t_s = \pi_\phi(s, x_i)$ for the target assigned to source s. The approximated MoE output separates the retained contributions from the projected contributions of omitted experts:

$$\widehat{y}_\phi(x_i) = \underbrace{\sum_{t \in S_K(x_i) \cap B_\phi(x_i)} \alpha_t(x_i) E_t(x_i)}_{\text{retained}} + \underbrace{\sum_{s \in O_\phi(x_i)} \alpha_s(x_i) s^\star_{s\to t_s} E_{t_s}(x_i)}_{\text{folded}}. \tag{4}$$

Thus, omitted experts are not executed, while their calibrated contributions remain in the MoE output. Routes folded into the same target are coalesced before expert computation, so each retained target is evaluated only once. Unlike static expert merging, folding changes neither expert parameters nor the original router (Li et al. 2024; Chen et al. 2025a).

For the tokens $\mathcal{X}_\phi$ processed together in one layer, both phases share the objective

$$\min_{B_\phi, \pi_\phi, \mathbf{S}} \sum_{x_i \in \mathcal{X}_\phi} \|\text{MoE}(x_i) - \widehat{y}_\phi(x_i)\|_2^2 \quad \text{s.t.} \quad c_\phi(B_\phi) \le \tau_\phi, \tag{5}$$

where $\mathbf{S} = [s^\star_{s\to t}]$ is the scalar-projector table. The prefill cost counts retained experts per token, whereas the decode cost counts distinct experts retained for a batch. We next calibrate $\mathbf{S}$ once and then solve the two budget constraints with the same folding rule.

Training-Free Projector Calibration

Figure 4 summarizes the offline calibration. For each ordered, co-routed source-target pair (s, t), we collect their outputs on m calibration tokens:

$$\mathbf{u}_i = E_s(x_i), \qquad \mathbf{v}_i = E_t(x_i), \qquad i = 1, \dots, m. \tag{6}$$

Here $\mathbf{u}_i, \mathbf{v}_i \in \mathbb{R}^d$, and each token is weighted by $w_i = \|\mathbf{u}_i\|_2$ in the main method. Given a projector family $\mathcal{P}$, calibration solves the weighted output-reconstruction problem

$$\mathbf{P}^\star_{s\to t} = \arg\min_{\mathbf{P} \in \mathcal{P}} \sum_{i=1}^m w_i \|\mathbf{u}_i - \mathbf{v}_i \mathbf{P}\|_2^2 + \lambda \Omega(\mathbf{P}), \tag{7}$$

where Ω regularizes the parameters of the selected projector family. For the scalar parameterization $\mathbf{P} = s\mathbf{I}$, the solution is

$$s^\star_{s\to t} = \frac{\sum_{i=1}^m w_i \langle \mathbf{v}_i, \mathbf{u}_i \rangle}{\sum_{i=1}^m w_i \|\mathbf{v}_i\|_2^2 + \lambda}. \tag{8}$$

Besides the scalar form, we evaluate diagonal and scalar-plus-low-rank projectors in Table 3. The deployed method uses the scalar solution because it can be folded into router weights without an online vector transform. We specify these alternatives, parameterization, and closed-form calibration in Appendix B.

Finally, scalar transfer error is stored as

$$\ell_{s\to t} = \frac{\sum_{i=1}^m w_i \|\mathbf{u}_i - s^\star_{s\to t} \mathbf{v}_i\|_2^2}{\sum_{i=1}^m w_i \|\mathbf{u}_i\|_2^2}. \tag{9}$$

Calibration therefore returns one directed scalar table $\mathbf{S}^{(l)} = [s^\star_{s\to t}]$ and loss table $\mathbf{L}^{(l)} = [\ell_{s\to t}]$ per MoE layer, without labels, gradients, or model updates. Detailed settings, matrix, and collections are provided in Appendices A and D.

Phase-Specific Selection and Unified Folding

At inference time, prefill and decode use phase-specific retained-expert selectors but share the same transfer criterion and folding operator, parameterized by the calibrated scalar table $\mathbf{S}^{(l)}$ and loss table $\mathbf{L}^{(l)}$. Below, we omit the layer index, denote the router weight by $w_{i,e} = \alpha_e(x_i)$, and use h_e for the cached output-norm estimate of expert e.

Method	Prefill	Decode	MATH_{500}	AIME24	IFEval	IFBench	GPQA	LCB (P/A.@8)	Eval+	MMLU_{pro}	Avg.
Baseline											
Original TopK=8	100%	100%	97.40	64.48	83.73	29.31	63.13	68.26 / 56.89	77.44	68.57	69.04
Prefill-Only Acceleration											
Prefill TopK=6	75%	100%	95.80	64.79	83.18	26.27	64.14	69.46 / 57.34	75.61	66.33	68.20
Prefill TopK=4	50%	100%	96.00	66.56	76.00	29.02	60.61	66.47 / 57.63	73.17	65.28	66.64
MC-MoE-P4	50%	100%	96.00	64.58	79.85	26.48	45.45	68.26 / 57.11	77.44	65.44	65.44
MoDES-P4	50%	100%	95.40	**67.08**	82.26	**30.26**	43.06	69.46 / 57.71	**78.66**	66.53	66.59
ExFold-P4	50%	100%	**97.00**	66.46	**84.00**	29.53	**62.63**	**70.66 / 57.86**	76.83	**66.94**	**69.26**
Decode-Only Acceleration											
REAP-D64	100%	50.0%	94.80	65.73	71.16	**31.29**	35.35	67.07 / 52.99	75.00	51.61	61.50
REAP-D32	100%	25.0%	70.00	23.02	33.83	24.74	11.62	8.00 / 2.50	9.76	5.04	22.25
SERE-K4 ($_{\rho=0.0}$)	100%	$S{=}4^*$	94.00	57.08	83.55	29.16	55.56	65.27 / 50.97	64.63	63.93	64.15
SERE-K2 ($_{\rho=0.1}$)	100%	$S{=}2^*$	88.20	49.17	73.75	23.20	50.00	16.77 / 5.24	42.07	60.28	50.65
ExFold-D64	100%	50.0%	**97.00**	65.31	83.92	28.12	**63.64**	**71.26 / 58.76**	74.39	66.36	68.75
ExFold-D32	100%	25.0%	96.80	**67.29**	**84.47**	28.03	**63.64**	70.06 / 57.19	**75.00**	**66.45**	**68.97**
Prefill & Decode Acceleration											
All TopK=6	75%	75.0%	95.60	65.31	81.52	27.66	59.09	68.86 / 57.11	67.68	64.60	66.29
All TopK=4	50%	50.0%	93.60	56.88	74.31	25.37	56.31	58.08 / 42.07	26.83	57.97	56.17
MC-MoE K=4	50%	50.0%	93.40	57.08	74.86	25.20	39.52	57.49 / 42.51	52.44	58.44	57.30
MoDES K=4	50%	50.0%	96.20	63.75	81.52	28.29	42.93	70.66 / 56.59	**76.22**	**66.16**	65.72
ExFold P4+D64	50%	50.0%	**97.00**	**67.29**	**82.44**	**28.60**	**62.12**	**71.26 / 57.26**	74.39	64.89	**68.50**
ExFold P4+D32	50%	25.0%	96.40	65.83	79.48	28.14	58.08	69.46 / 56.29	74.39	65.00	67.10

Table 1: Qwen3-30B-A3B quality. **Bold** marks the best result per setting. P4 denotes 4 experts per token in prefill, Dm means a decode pool with size m, and S^* means SERE's dynamic set. P/A.@8 means the score of Pass@8 and Avg@8.

1. Select phase-specific primary experts. As illustrated in Figure 4, prefill selects K_{pre} primary experts independently for each token, whereas decode selects a shared set of at most D experts for the current batch $\mathcal{X}_q$:

$$B_{\text{pre}}(x_i) = \text{TopK}_{e\in S_K(x_i)}\big(w_{i,e}h_e, K_{\text{pre}}\big), \tag{10}$$

$$B_{\text{dec}}(\mathcal{X}_q) = \text{TopK}_{e\in\cup_i S_K(x_i)}\left(\sum_{x_i\in\mathcal{X}_q} w_{i,e}h_e, D\right). \tag{11}$$

The score $w_{i,e}h_e$ estimates the magnitude of each routed contribution. Prefill applies it locally to reduce token-level computation; decode aggregates it across tokens to reduce the batch-level expert union.

2. Choose minimum-loss transfers. For either phase, each omitted source expert E_s selects the retained target with the smallest calibrated reconstruction loss:

$$\pi_\phi(s) = \arg\min_{t\in B_\phi} \ell_{s\to t}, \tag{12}$$

where B_ϕ is the token-level set $B_{\text{pre}}(x_i)$ or the batch-level set $B_{\text{dec}}(\mathcal{X}_q)$. The two phases therefore share the same loss lookup and differ only in the candidate target set.

3. Fold router metadata. Let $S_{s\to t} = s^\star_{s\to t}$ denote the scalar-table lookup. In prefill, omitted routes mapped to the same target are coalesced by updating the target weight:

$$\widetilde{w}_{i,t} = w_{i,t} + \sum_{\substack{s\in S_K(x_i)\setminus B_{\text{pre}}(x_i)\\ \pi_{\text{pre}}(s)=t}} w_{i,s}S_{s\to t}, \quad t\in B_{\text{pre}}(x_i). \tag{13}$$

Method	AIME24	IFEval	GPQA	Eval+	MMLU_{pro}	Avg.
Baseline						
Original Top-8	48.75	83.55	76.26	71.34	58.98	67.78
Prefill-Only Acceleration						
Prefill TopK=6	46.67	**83.55**	**75.25**	73.78	56.53	**67.16**
Prefill TopK=4	45.00	80.59	69.19	**75.61**	40.41	62.16
ExFold-P4	**48.75**	80.78	73.74	67.68	**58.57**	65.90
Decode-Only Acceleration						
REAP-D64	40.42	69.13	69.70	42.07	55.92	55.45
ExFold-D64	**47.50**	**83.73**	**71.21**	**73.17**	**57.96**	**66.71**
Prefill & Decode Acceleration						
All TopK=6	46.67	80.96	72.22	66.46	41.43	61.55
All TopK=4	36.67	75.60	71.21	**70.73**	38.78	58.60
ExFold P4+D64	44.17	80.22	**72.73**	67.68	**58.98**	64.76
ExFold P4+D32	**50.00**	**81.70**	71.21	67.68	57.14	**65.55**

Table 2: GLM-4.5-Air quality. **Bold** for the best per setting.

The fused MoE kernel consequently executes only K_{pre} routes. In decode, each omitted route is instead remapped in place:

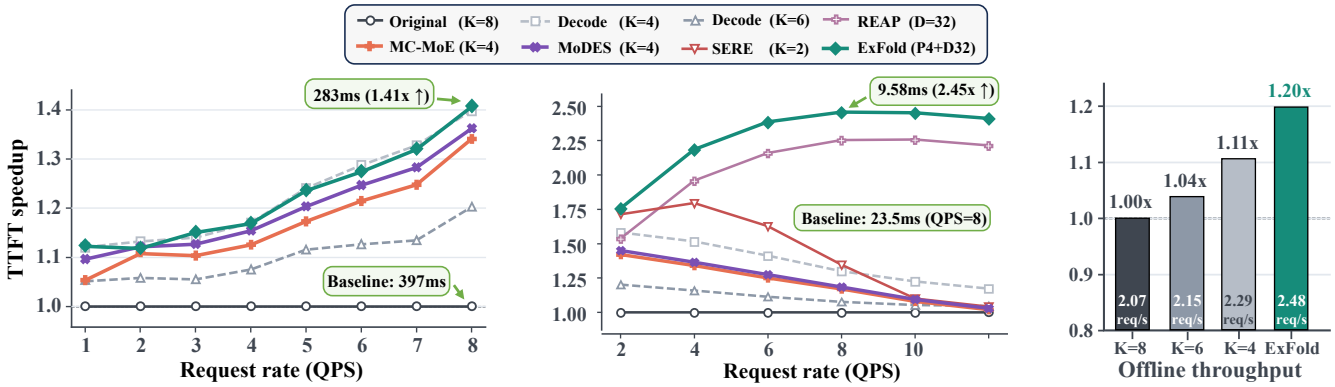
$$(E_s, w_{i,s}) \longmapsto \big(E_{\pi_{\text{dec}}(s)}, w_{i,s}S_{s\to\pi_{\text{dec}}(s)}\big). \tag{14}$$

Retained routes remain unchanged, while every remapped expert belongs to $B_{\text{dec}}(\mathcal{X}_q)$; hence the batch touches at most D distinct experts without changing the regular Top-K routing layout. Both transformations modify only router metadata before the existing fused MoE kernel.

Experiments

Experimental Setup

Models. We evaluate Qwen3-30B-A3B as the primary model and GLM-4.5-Air, DeepSeek-V2-Lite, DeepSeek-



(a) Prefill acceleration. **(b)** Decode acceleration **(c)** Offline acceleration

Figure 5: Online TTFT/TPOT speedups and offline serving throughput.

Method	MATH_{500}	AIME24	IFEval	IFBench	GPQA	Eval+	MMLU_{pro}	PPL↓	ΔStorage	ΔToken Cost	Kernel
Original TopK=8	97.40	64.48	83.73	29.31	63.13	77.44	68.57	3.020	0	$O(4LC_{\text{FFN}})$	FusedMoE
Prefill TopK=4	96.00	66.56	76.00	29.02	60.61	73.17	65.28	3.470	0	0	FusedMoE
Global scalar	96.00	65.21	80.96	25.73	59.60	75.00	65.60	3.373	4 B	$O(4L)$	FusedMoE
Layer scalar	96.60	**67.60**	82.44	26.30	61.11	74.39	64.70	3.350	192 B	$O(4L)$	FusedMoE
Expert scalar	**97.00**	66.46	**84.00**	**29.53**	**62.63**	**76.83**	66.94	3.352	6.00 MiB	$O(4L)$	FusedMoE
Diagonal	96.60	67.50	83.92	27.63	60.61	72.56	65.87	**3.308**	316.7 MiB	$O(4LH)$	Unfused
Low-rank R8	95.60	65.00	83.92	27.35	60.10	74.39	**67.35**	3.341	2.44 GiB	$O(64LH)$	Unfused
Low-rank R16	95.20	62.50	83.55	27.65	61.62	75.61	66.60	3.342	4.87 GiB	$O(128LH)$	Unfused

Table 3: Projector ablation under Top-4 prefill. Overheads use FP32 state and are relative to Prefill TopK=4. Here $\Delta K = 4$; R8/R16 incur $2\Delta KR = 64/128$ operations per hidden dimension. L, H, C_{FFN}: layers, hidden size, and one expert-call time.

V4-Flash, Qwen3.5-35B-A3B (in Appendix C) for cross-architecture generalization.

Benchmarks. We evaluate mathematical reasoning with MATH500 and AIME24, code generation with LiveCodeBenchV5 and HumanEval+, instruction following with IFEval and IFBench, and knowledge reasoning with GPQA-Diamond and MMLU-Pro (Hendrycks et al. 2021; Jain et al. 2024; Liu et al. 2023; Zhou et al. 2023; Pyatkin et al. 2025; Rein et al. 2023; Wang et al. 2024). For efficiency, we report TTFT under prefill-dominated serving, TPOT under generation-dominated serving, and offline throughput in vLLM (Kwon et al. 2023). Metric and workload details are deferred to Appendix A.

Baselines. For prefill, we compare with direct Top-K reduction, MC-MoE, and MoDES under matched token-wise expert budgets. For decode, we compare with static expert pruning (REAP) and dynamic expert skipping (SERE). We report prefill-only, decode-only, and joint acceleration to distinguish phase-specific quality loss from errors accumulated across both phases (Huang et al. 2024; Lasby et al. 2025; Wu et al. 2026).

Hyperparameters. We calibrate ExFold on 32 unlabeled sequences of at most 4096 tokens, use source-output-norm weighting, and set the ridge coefficient to $\lambda = 10^{-3}$. PX denotes X retained experts per prefill token, while DX denotes at most X active experts per decode batch. We implement ExFold with a custom Triton operator in vLLM and run efficiency experiments in BF16 on NVIDIA H800 GPUs (Tillet, Kung, and Cox 2019); complete calibration, hardware, and serving configurations are provided in Appendix A.

Main Quality Results

Tables 1 and 2 compare ExFold with phase-specific baselines under matched execution budgets. We focus on whether quality is preserved, rather than treating a lower expert count alone as an improvement.

Prefill-only acceleration. At the P4 budget, ExFold preserves the original Qwen3 average and outperforms all compute-matched baselines (69.26 versus 66.64 for Direct Top-4). The gains are most pronounced on instruction following and code generation, where hard dropping loses important routed contributions. Thus, folding can halve prefill expert computation without the quality loss of direct sparsification.

Decode-only acceleration. The advantage of ExFold widens as the batch-level expert budget becomes tighter. At D32, it remains within 0.07 points of the original model; SERE-K2 recovers part of the quality lost by static pruning but remains substantially lower. This trend shows that recovering omitted expert contributions becomes increasingly important when the active expert pool is aggressively constrained.

Joint prefill and decode acceleration. Applying Direct Top-4 to both phases compounds approximation error and

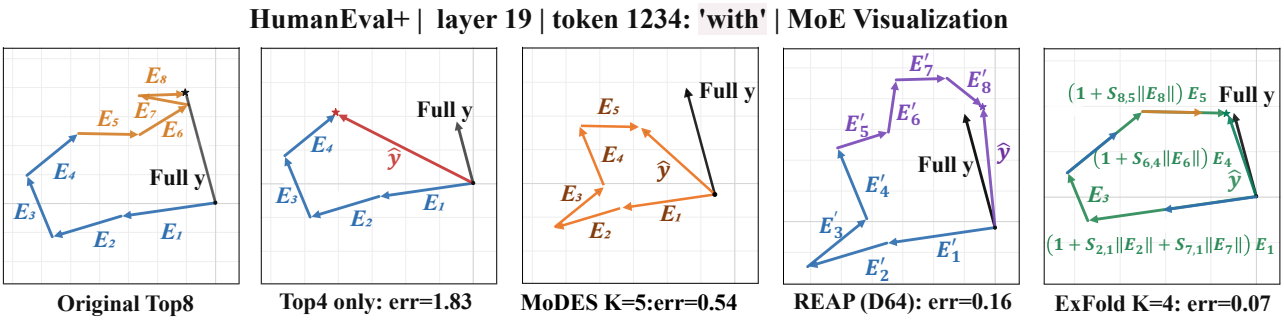


Figure 6: Per-token MoE output reconstruction under expert reduction. Error denotes $\|\widehat{y} - y\|_2$.

reduces the average to 56.17, while ExFold P4+D64 retains 68.50. ExFold also remains stronger than MoDES when both phases are accelerated. The shared projector therefore avoids stacking two independent dropping errors and supports both phase-specific budgets under one approximation objective.

Cross-architecture generalization. On GLM-4.5-Air, ExFold retains 97.2% and 98.4% of the original average quality under prefill-only P4 and decode-only D64, respectively. When both phases are compressed, P4+D32 still retains 96.7%. These results confirm that ExFold preserves quality beyond the primary model. Additional DeepSeek-V2-Lite and Qwen3.5 results can be seen in Appendix C.

Calibration	MATH_{500}	GPQA	LCB (P/A.@8)	Eval+	Avg.
GPQA	96.40	**65.15**	70.06 / 56.96	71.95	72.10
MATH	96.40	60.10	69.46 / 56.36	73.17	71.10
CODE	96.60	59.09	70.06 / 56.74	75.61	71.62
PRETRAIN	**97.00**	63.13	69.46 / 57.34	**76.83**	72.75
MIXED	**97.00**	62.63	**70.66 / 57.86**	**76.83**	**73.00**

Table 4: Calibration-corpus ablation for ExFold-P4.

Selection criterion	MATH_{500}	IFEval	GPQA	Eval+	Avg.
Prefill (ExFold-P4)					
rank by S_i	**97.00**	82.62	**64.65**	73.17	79.36
rank by H_i	**97.00**	67.47	57.58	75.61	74.42
rank by $S_i \times H_i$ (Ours)	**97.00**	**84.00**	62.63	**76.83**	**80.11**
Decode (ExFold-D64)					
rank by S_i	**97.00**	83.92	60.10	73.17	78.55
rank by H_i	96.40	**84.10**	63.13	71.34	78.74
rank by $S_i \times H_i$ (Ours)	**97.00**	83.92	**63.64**	**74.39**	**79.74**

Table 5: Retained-expert selection on Qwen3-30B-A3B.

Efficiency Evaluation

Online prefill latency. Figure 5(a) shows that reducing token-level expert computation translates directly into lower TTFT. ExFold reaches a $1.41\times$ speedup at 8 QPS and closely tracks compute-matched Top-4 methods, indicating little overhead from scalar folding. Unlike direct reduction, it realizes this compute benefit while preserving model quality.

Online decode latency. Figure 5(b) reveals a different scaling trend in decode: token-wise Top-4 and Top-6 lose their benefit as QPS increases because the batch activates a broader expert union. By bounding this union, ExFold reaches a $2.45\times$ TPOT speedup at 8 QPS and sustains about $2.4\times$ through 12 QPS. This confirms that reducing batch-level weight traffic, rather than token-wise FLOPs alone, is essential for efficient decode.

Offline serving throughput. Figure 5(c) shows that the online gains transfer to throughput-oriented serving: ExFold improves offline throughput by $1.20\times$ and exceeds Direct Top-4. This result also confirms that the Triton folding operator adds little runtime overhead under large batches.

Scaling to DeepSeek-V4-Flash

DeepSeek-V4-Flash is a 284B-parameter MoE with 256 routed experts and Top-6 routing (DeepSeek-AI 2026). It provides a substantially larger expert space than the models above and therefore tests whether folding remains effective when both the number of candidate experts and the batch-level expert union grow. We use P3 to retain three routed experts per prefill token and D128/D64 to cap the decode expert pool at 50%/25% of the routed experts. All quality rows use the same full-suite evaluator protocol; Appendix reports the task sizes and sampling rules.

Table 6 shows that direct all-stage K=3 reduction retains only 88.17% of the original average, with the largest losses on AIME25/26. ExFold P3+D64 instead retains 97.68%, a 9.51-point retention gain under the same three-expert prefill budget, while P3+D128 retains 99.32%. Decode-only D64 and D128 remain within 0.2 average points of Original. These results support a quality frontier rather than one universal budget: D128 is the near-lossless setting, whereas D64 trades 1.64 additional quality-retention points for a tighter expert pool.

Figure 7 follows the same online/offline structure as Figure 5. P3 improves mean 8K TTFT across QPS 1–8, reaching $1.32\times$ at QPS 8. In decode, D64 stabilizes near $1.15\times$ TPOT speedup once the batch is saturated, while D128 remains the quality-first operating point. Under the saturated throughput workload, D64 raises output throughput from 5.22k to 6.72k tokens/s ($1.29\times$), ahead of Direct K=3, REAP-D128, and D128. The code release exposes both budgets as serving arguments and includes the calibrated matrix used by these runs.

Method	Prefill	Decode	MATH_{500}	AIME25	AIME26	IFEval	IFBench	GPQA	LCB	Eval+	MMLU_{pro}	Avg.
Baseline												
Original TopK=6	100%	100%	93.40	70.42	72.50	82.44	36.23	73.23	54.90	89.25	81.76	72.68
Prefill-Only Acceleration												
Prefill TopK=3	50%	100%	92.00	**69.17**	69.17	80.22	**35.67**	69.70	54.12	85.59	79.20	70.54
ExFold-P3	50%	100%	**93.20**	65.83	**70.42**	**82.99**	35.28	**72.22**	**55.67**	**87.65**	**80.48**	**71.53**
Decode-Only Acceleration												
Decode TopK=3	100%	50%	88.80	47.08	49.17	79.48	32.01	73.23	49.74	82.09	80.68	64.70
REAP-D128	100%	50%	93.20	70.00	**74.58**	71.16	34.30	72.73	52.58	**89.86**	80.73	71.02
ExFold-D128	100%	50%	94.20	**70.83**	72.50	81.15	35.67	**75.76**	**54.12**	88.41	**81.82**	**72.72**
ExFold-D64	100%	25%	**94.80**	**70.83**	71.25	**81.33**	**37.52**	72.73	53.87	88.72	81.79	72.54
Prefill & Decode Acceleration												
All TopK=3	50%	50%	87.40	52.08	51.25	75.97	28.10	68.18	47.94	85.98	79.85	64.08
ExFold P3+D128	50%	50%	**92.80**	**72.50**	**72.50**	**81.52**	**35.45**	68.69	**55.41**	**90.17**	**80.61**	**72.18**
ExFold P3+D64	50%	25%	92.60	66.67	70.83	80.96	31.56	**74.75**	54.38	86.97	80.21	70.99

Table 6: DeepSeek-V4-Flash quality. **Bold** marks the best result per setting. P3 denotes 3 experts per token in prefill, and Dm means a decode pool with size m.

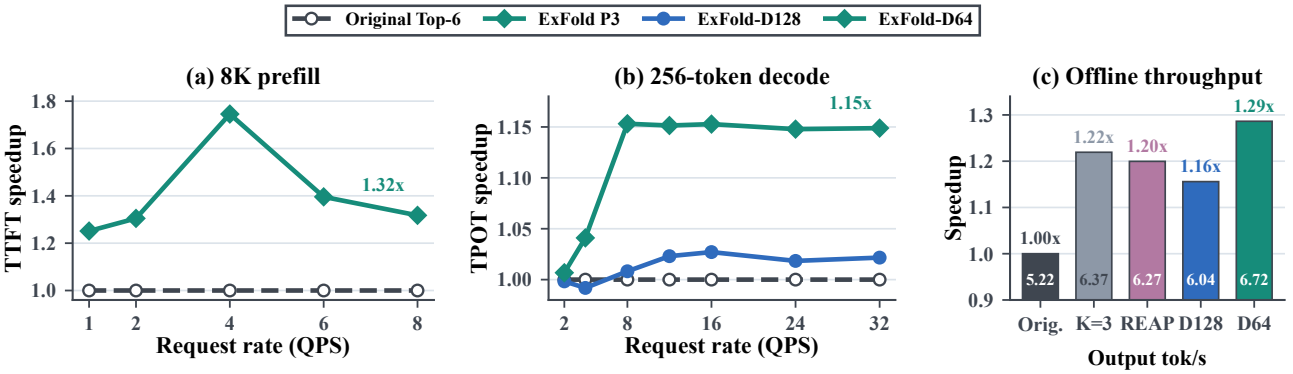


Figure 7: DeepSeek-V4-Flash speedups on H800: **(a)** 8K TTFT, **(b)** 256-token TPOT, and **(c)** offline throughput.

Ablations and Analysis

Which projector parameterization is practical? Table 3 compares projector families under the same Top-4 budget. Pairwise scalars outperform global and layer-wise scaling, while diagonal and low-rank projectors require substantially more state and unfused vector transforms. Although the diagonal projector lowers perplexity, the pairwise scalar performs better on six of seven downstream metrics; low-rank variants further add 2.44–4.87 GiB of state without consistent gains. Pairwise scalars give the best trade-off. In appendix B, we detail each implementation.

Does ExFold require task-specific calibration? Table 4 varies only the calibration domain. Mixed calibration achieves the best average (73.00), but all domains remain within 1.9 points and task-matched data is not consistently optimal. This stability suggests that calibration captures expert geometry rather than benchmark-specific behavior. No task-specific calibration is required.

How should ExFold select retained experts? Table 5 compares router score S_i, output magnitude H_i, and their product under fixed budgets. Router score omits output scale, while magnitude alone omits routing relevance; their product approximates contribution magnitude and yields the best average in both phases. Rank retained experts by $S_i \times H_i$.

Why does expert folding preserve quality? Figure 6 visualizes the routed expert contributions and resulting MoE output for a representative token. Direct Top-4 and MoDES omit routed vectors and produce large reconstruction errors of 1.83 and 0.54, respectively. REAP reduces the global expert pool and lowers the error to 0.16 by selecting Top-8 within the retained pool, but it still executes eight experts per token and therefore does not reduce expert FLOPs (Huang et al. 2025; Lasby et al. 2025). ExFold instead transfers omitted contributions to directionally aligned retained experts and folds their calibrated scales into the gate weights, achieving the smallest error of 0.07 with only four executed experts. Folding recovers omitted contributions. Additional token-level cases are shown in Appendix E.

Conclusion

We presented ExFold, a training-free framework for jointly accelerating MoE prefill and decode. It combines phase-specific retained-set selectors with a shared directed projector that folds excluded contributions into router metadata. Across MoE architectures, ExFold improves TTFT, TPOT, and serving throughput while preserving more quality than expert dropping. Because folding changes only router metadata, it remains complementary to kernel, scheduling, and parallelism optimizations. These results establish output recovery as a practical basis for unified MoE inference acceleration.

References

Cao, M.; Chen, K.; Duan, H.; Fang, Y.; Fei, Z.; Gao, T.; Ge Jiaye; Li, M.; Liu, H.; Liu, J.; Liu, Y.; et al. 2026. OpenCompass: A Universal Evaluation Platform for Large Language Models. *arXiv preprint arXiv:2605.19276*.

Chen, I.-C.; Liu, H.-S.; Sun, W.-F.; Chao, C.-H.; Hsu, Y.-C.; and Lee, C.-Y. 2025a. Retraining-Free Merging of Sparse Mixture-of-Experts via Hierarchical Clustering. In *International Conference on Machine Learning*.

Chen, J.; Bai, S.; Wang, Z.; Wu, S.; Du, C.; Yang, H.; Gong, R.; Liu, S.; Wu, F.; and Chen, G. 2025b. Pre^3: Enabling deterministic pushdown automata for faster structured LLM generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 11253–11267.

Chen, J.; Du, C.; Liu, R.; Yao, S.; Yan, D.; Liao, J.; Liu, S.; Wu, F.; and Chen, G. 2026. TokenFlow: Responsive LLM Text Streaming Serving under Request Burst via Preemptive Scheduling. In *Proceedings of the 21st European Conference on Computer Systems*, 497–513.

DeepSeek-AI. 2024. DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model. *arXiv preprint arXiv:2405.04434*.

DeepSeek-AI. 2026. DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence. arXiv:2606.19348.

Du, C.; Chen, J.; Tang, H.; Liu, K.; Lan, T.; Qu, L.; Niu, C.; Liu, S.; Chen, G.; and Wu, F. 2026. C^{2}KV: Compressed and Composable KV Cache Reuse for Efficient LLM Inference. *arXiv preprint arXiv:2607.17715*.

Fedus, W.; Zoph, B.; and Shazeer, N. 2022. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *Journal of Machine Learning Research*.

Gupta, V.; Ju, J. H.; Sinha, K.; Gavrilovska, A.; and Iyer, A. P. 2024. Lynx: Enabling efficient moe inference through dynamic batch-aware expert selection. *arXiv preprint arXiv:2411.08982*.

Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021. Measuring Mathematical Problem Solving with the MATH Dataset. In *Advances in Neural Information Processing Systems*.

Huang, Q.; An, Z.; Zhuang, N.; Tao, M.; Zhang, C.; Jin, Y.; Xu, K.; Xu, K.; Chen, L.; Huang, S.; and Feng, Y. 2024. Harder Tasks Need More Experts: Dynamic Routing in MoE Models. *arXiv preprint arXiv:2403.07652*.

Huang, Y.; Wang, Z.; Yuan, Z.; Ding, Y.; Gong, R.; Guo, J.; Liu, X.; and Zhang, J. 2025. MoDES: Accelerating Mixture-of-Experts Multimodal Large Language Models via Dynamic Expert Skipping. *arXiv preprint arXiv:2511.15690*.

Hugging Face H4 Team. 2024. MATH-500. https://huggingface.co/datasets/HuggingFaceH4/MATH-500. Benchmark dataset.

Jain, N.; Han, K.; Gu, A.; Li, W.-D.; Yan, F.; Zhang, T.; Wang, S.; Solar-Lezama, A.; Sen, K.; and Stoica, I. 2024. LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code. *arXiv preprint arXiv:2403.07974*.

Jha, S.; Hashemzadeh, M.; Saheb Pasand, A.; Parviz, A.; Lee, M.-J.; and Knyazev, B. 2026. REAM: Merging Improves Pruning of Experts in LLMs. *arXiv preprint arXiv:2604.04356*.

Jiang, A. Q.; Bressand, F.; et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Kwon, W.; Li, Z.; Zhuang, S.; Sheng, Y.; Zheng, L.; Yu, C. H.; Gonzalez, J. E.; Zhang, H.; and Stoica, I. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *ACM Symposium on Operating Systems Principles*.

Lasby, M.; Lazarevich, I.; Sinnadurai, N.; Lie, S.; Ioannou, Y.; and Thangarasa, V. 2025. REAP the Experts: Why Pruning Prevails for One-Shot MoE Compression. *arXiv preprint arXiv:2510.13999*.

Li, L.; Zhu, Q.; Wang, J.; Li, W.; Gu, H.; Han, S.; and Guo, Y. 2025. Sub-MoE: Efficient Mixture-of-Expert LLMs Compression via Subspace Expert Merging. *arXiv preprint arXiv:2506.23266*.

Li, P.; Zhang, Z.; Yadav, P.; Sung, Y.-L.; Cheng, Y.; Bansal, M.; and Chen, T. 2024. Merge, Then Compress: Demystify Efficient SMoE with Hints from Its Routing Policy. In *The Twelfth International Conference on Learning Representations*.

Liu, J.; Xia, C. S.; Wang, Y.; and Zhang, L. 2023. Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation. In *Advances in Neural Information Processing Systems*.

Lu, X.; Liu, Q.; Xu, Y.; Zhou, A.; Huang, S.; Zhang, B.; Yan, J.; and Li, H. 2024. Not All Experts are Equal: Efficient Expert Pruning and Skipping for Mixture-of-Experts Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

Math-AI Team. 2024. American Invitational Mathematics Examination (AIME) 2024. https://huggingface.co/datasets/math-ai/aime24. Benchmark dataset.

NVIDIA. 2025. NVIDIA Nemotron Post-Training Dataset v1. https://huggingface.co/datasets/nvidia/Nemotron-Post-Training-Dataset-v1. Dataset card.

Pyatkin, V.; Malik, S.; Graf, V.; Ivison, H.; Huang, S.; Dasigi, P.; Lambert, N.; and Hajishirzi, H. 2025. Generalizing Verifiable Instruction Following. *arXiv preprint arXiv:2507.02833*.

Qwen Team. 2025. Qwen3-30B-A3B. https://huggingface.co/Qwen/Qwen3-30B-A3B. Model card.

Qwen Team. 2026. Qwen3.5-35B-A3B. https://huggingface.co/Qwen/Qwen3.5-35B-A3B. Model card.

Rein, D.; Hou, B. L.; Stickland, A. C.; Petty, J.; Pang, R. Y.; Dirani, J.; Michael, J.; and Bowman, S. R. 2023. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. *arXiv preprint arXiv:2311.12022*.

Tillet, P.; Kung, H. T.; and Cox, D. 2019. Triton: An Intermediate Language and Compiler for Tiled Neural Network Computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*.

Wang, Y.; Ma, X.; Zhang, G.; Ni, Y.; Chandra, A.; Guo, S.; Ren, W.; Arulraj, A.; He, X.; Jiang, Z.; Li, T.; Ku, M.; Wang, K.; Zhuang, A.; Fan, R.; Yue, X.; and Chen, W. 2024. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark. *arXiv preprint arXiv:2406.01574*.

Wu, J.; Cheng, J.; Lv, F.; Ou, D.; and Yuan, L. 2026. SERE: Similarity-based Expert Re-routing for Efficient Batch Decoding in MoE Models. In *The Fourteenth International Conference on Learning Representations*.

Z.ai. 2025. GLM-4.5-Air. https://huggingface.co/zai-org/GLM-4.5-Air. Model card.

Zhou, J.; Lu, T.; Mishra, S.; Brahma, S.; Basu, S.; Luan, Y.; Zhou, D.; and Hou, L. 2023. Instruction-Following Evaluation for Large Language Models. *arXiv preprint arXiv:2311.07911*.

Reproducibility Details

Model architectures. Table 7 summarizes the evaluated released checkpoints. We report the architecture exposed by each checkpoint configuration; shared experts are executed in addition to the routed Top-K experts. "Expert FFN" is the hidden width of each expert feed-forward network.

Benchmarks and metrics. Table 8 lists the evaluated benchmarks. "Cases" is the number of problems in the evaluated benchmark snapshot, and "Samples" is the number of generations per problem. The GLM-4.5-Air MMLU-Pro comparison uses a fixed 490-case subset containing the first 35 examples from each of 14 categories. For IFBench, we evaluate the 294-case OpenCompass snapshot and average prompt- and instruction-level accuracy under strict and loose checking. For LiveCodeBench v5, we use the same fixed 167-problem subset for every method and generate eight solutions per problem. The Qwen3-30B-A3B evaluation uses 32 samples for each AIME 2024 problem, whereas the GLM-4.5-Air evaluation uses eight. LiveCodeBench reports pass@8 and mean correctness over the eight samples; pass@8 is its primary score in aggregate comparisons.

Calibration protocol. For the primary Qwen3 configuration, we calibrate once on 32 unlabeled sequences of at most 4,096 tokens and keep the resulting projector fixed across downstream tasks. DeepSeek-V2-Lite uses the same calibration size and token limit; Qwen3.5 aggregates four disjoint eight-sequence calibration shards. GLM-4.5-Air uses a separate 16-sequence capture with at most 1,024 tokens per sequence. The 32-sequence pool is drawn from the chat, code, and science, technology, engineering, and mathematics (STEM) portions of the public NVIDIA Nemotron Post-Training Dataset (NVIDIA 2025). All calibrations accumulate only forward activations, without task labels, gradients, or evaluator feedback. For each ordered co-routed expert pair, we accumulate the sufficient statistics in Eq. 15 with source-output-norm weighting and $\lambda = 10^{-3}$. Scalar coefficients are clipped to $[-4, 4]$, self-pair coefficients are fixed to one, and unobserved pairs receive loss 10^{30} so that an observed target is preferred whenever one exists.

Quality evaluation. Generation is orchestrated by an OpenCompass-compatible pipeline (Cao et al. 2026). IFEval disables thinking and uses prompt-level strict scoring, while IFBench uses the four-way composite defined above. LiveCodeBench and HumanEval+ are scored by executing generated programs against their benchmark test suites with the LiveCodeBench and EvalPlus evaluators (Jain et al. 2024; Liu et al. 2023). Within each model–benchmark comparison, prompts, decoding parameters, sample counts, and scoring code are fixed. Repeated generations are scored individually before aggregation. All reported scores are percentages, and "Avg." is the arithmetic mean of the displayed primary metrics.

System and serving configuration. Efficiency experiments run on eight-GPU NVIDIA H800 80GB servers. The primary H800 runtime uses Ubuntu 24.04.2 LTS, NVIDIA driver 550.163.01, vLLM 0.10.2 (Kwon et al. 2023), PyTorch 2.8.0, CUDA 12.8, and Triton 3.4.0 (Tillet, Kung, and Cox 2019). Prefill serving uses tensor parallelism four, sweeps 1–8 queries per second (QPS) with 8,192-token prompts and one generated token, and reports mean time to first token (TTFT). The matched decode comparison uses vLLM 0.11.0 and tensor parallelism one; it sweeps 2–12 QPS over 512 requests with one input token and 256 generated tokens and reports mean time per output token (TPOT). Offline serving uses tensor parallelism eight and reports completed-request throughput for 512 requests with 2,048-token inputs and 512 generated tokens. ExFold speedups are normalized to an Original run with the same hardware, engine configuration, requests, and decoding parameters. DeepSeek-V2-Lite and Qwen3.5 quality generation use bfloat16 (BF16) and tensor parallelism one on H800 or H20 GPUs; GLM-4.5-Air uses tensor parallelism four on H800 GPUs.

Projector Parameterizations and Implementation

Let L, E, and H denote the number of MoE layers, routed experts per layer, and hidden dimensions. For a calibration token x_i that co-routes a source expert E_s and target expert E_t, we write $\mathbf{u}_i = E_s(x_i)$ and $\mathbf{v}_i = E_t(x_i)$; l, s, and t index the layer, source expert, and target expert. For the projector-family ablation, every family uses the same fixed calibration sequences, source-output-norm weighting, and prefill Top-4 budget. Each sharing pattern computes its own reconstruction losses for target assignment. The diagonal and low-rank alternatives execute through a custom unfused reconstruction path. Table 9 counts transformation parameters and omits the per-pair assignment-loss table. Let $P \le LE^2$ denote the number of materialized directed source–target pairs.

Scalar sharing patterns. Let $\mathcal{G}$ denote a group of calibration tuples (l, s, t, i). A scalar shared by that group is fitted by

$$s^\star_{\mathcal{G}} = \frac{\sum_{(l,s,t,i)\in\mathcal{G}} w_i \, \mathbf{u}_i^\top \mathbf{v}_i}{\sum_{(l,s,t,i)\in\mathcal{G}} w_i \, \|\mathbf{v}_i\|_2^2 + \lambda}. \tag{15}$$

Here $w_i = \|\mathbf{u}_i\|_2$ is the source-output-norm weight used by the calibration implementation. The global variant uses one group for the entire model, the layer-wise variant uses one group per MoE layer, and the pairwise variant uses one group for each directed source–target pair in each layer. Only the pairwise form preserves directed expert-level differences while remaining a scalar metadata update. For target assignment, we use the normalized reconstruction loss

$$\ell_{s\to t} = \frac{\sum_i w_i \|\mathbf{u}_i - s^\star_{s\to t}\mathbf{v}_i\|_2^2}{\sum_i w_i \|\mathbf{u}_i\|_2^2 + \epsilon}, \tag{16}$$

where $\epsilon > 0$ prevents division by zero. Lower loss indicates that E_t better reconstructs the contribution of E_s.

Diagonal projector. The diagonal alternative fits one coefficient per hidden dimension. Its element-wise weighted ridge solution is

$$\mathbf{d}^\star_{s\to t} = \frac{\sum_i w_i(\mathbf{v}_i \odot \mathbf{u}_i)}{\sum_i w_i(\mathbf{v}_i \odot \mathbf{v}_i) + \lambda \mathbf{1}}, \tag{17}$$

Model	Layers	MoE Layers	Routed Experts	Shared Experts	Top-K	Hidden Size	Expert FFN
Qwen3-30B-A3B (Qwen Team 2025)	48	48	128	0	8	2048	768
GLM-4.5-Air (Z.ai 2025)	46	45	128	1	8	4096	1408
DeepSeek-V2-Lite-Chat (DeepSeek-AI 2024)	27	26	64	2	6	2048	1408
Qwen3.5-35B-A3B (Qwen Team 2026)	40	40	256	1	8	2048	512

Table 7: Architectures of the evaluated MoE checkpoints.

Benchmark	Domain	Cases	Samples	Metric
MATH500 (Hendrycks et al. 2021; Hugging Face H4 Team 2024)	Mathematics	500	1	Symbolic-equivalence accuracy
AIME24 (Math-AI Team 2024)	Competition mathematics	30	32 (Qwen3); 8 (GLM)	Mean symbolic-equivalence accuracy
IFEval (Zhou et al. 2023)	Instruction following	541	1	Prompt-level strict accuracy
IFBench (Pyatkin et al. 2025)	Instruction following	294	1	OpenCompass four-way mean
GPQA-Diamond (Rein et al. 2023)	Graduate-level reasoning	198	1	Accuracy
LiveCodeBench v5 (Jain et al. 2024)	Code generation	167	8	Pass@8 / mean@8
HumanEval+ (Liu et al. 2023)	Code generation	164	1	Pass@1
MMLU-Pro (Wang et al. 2024)	Knowledge reasoning	12,032 (490 for GLM)	1	Category-macro accuracy

Table 8: Benchmarks, evaluation sizes, and reported metrics.

where the division is element-wise. The resulting $\mathrm{Diag}(\mathbf{d}^{\star}_{s\to t})$ cannot be absorbed into a router weight and must act on the hidden vector online.

Scalar-plus-low-rank projector. Let $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{m\times H}$ stack source and target outputs, and $\mathbf{W} = \mathrm{Diag}(w_1, \ldots, w_m)$. We fit a layer-wise orthonormal basis $\mathbf{Q}^{(l)} \in \mathbb{R}^{H\times R}$ to centered scalar-residual samples by randomized truncated singular value decomposition (SVD) and parameterize

$$\mathbf{M}_{s\to t} = s^{\star}_{s\to t}\mathbf{I} + \mathbf{Q}^{(l)}\mathbf{C}_{s\to t}, \qquad \mathbf{C}_{s\to t} \in \mathbb{R}^{R\times H}. \quad (18)$$

Defining $\mathbf{Z} = \mathbf{W}^{1/2}\mathbf{V}\mathbf{Q}^{(l)}$, $\mathbf{R}_{s\to t} = \mathbf{W}^{1/2}(\mathbf{U} - s^{\star}_{s\to t}\mathbf{V})$, and $\mathbf{G} = \mathbf{Z}^{\top}\mathbf{Z}$, weighted ridge regression gives

$$\mathbf{C}^{\star}_{s\to t} = (\mathbf{G} + \lambda\mathbf{I})^{-1}\mathbf{Z}^{\top}\mathbf{R}_{s\to t}. \quad (19)$$

The evaluated artifact collects a rank-32 basis and retains at most 1,024 directed pairs per layer ranked by accumulated source-weighted source energy. We evaluate its first $R \in \{8, 16\}$ directions; pairs outside the materialized support use only the scalar term. The basis and pair-specific correction require an online vector transform and substantial state, preventing reuse of the standard fused MoE path.

Projector	Storage	Online	Kernel
Global scalar	1	Scalar	Fused
Layer scalar	L	Scalar	Fused
Pair scalar	P	Scalar	Fused
Pair diagonal	PH	Element-wise	Unfused
Scalar + low-rank	$P + LHR + PRH$	Rank-R	Unfused

Table 9: Implementation differences among the evaluated projector families.

Phase-Specific Routing and Folding

All retained sets and projector lookups below are defined separately for each MoE layer; we omit the layer index for clarity. Let the original routes for token x_i be $S_K(x_i) = (e_{i,1}, \ldots, e_{i,K})$, ordered by router score, and let $w_{i,e}$ be the routed weight of expert e and h_e its cached output-norm estimate; $w_{i,e} = 0$ when $e \notin S_K(x_i)$. Let K_{pre} be the per-token prefill budget, let $\mathcal{X}_q$ be the tokens processed in decode step q, let $B = |\mathcal{X}_q|$, and let D be the decode expert-pool budget. The prefill path ranks the token's routed experts by estimated contribution magnitude and retains K_{pre} of them:

$$B_{\mathrm{pre}}(x_i) = \mathrm{TopK}_{e\in S_K(x_i)}\left(w_{i,e}h_e, K_{\mathrm{pre}}\right). \quad (20)$$

Decode supports two retained-pool selectors that share the same folding operator. The dynamic path used for the cross-model results and online decode measurement ranks the union of routed experts by its aggregate contribution:

$$B^{\mathrm{dyn}}_{\mathrm{dec}}(\mathcal{X}_q) = \mathrm{TopK}_{e\in\cup_i S_K(x_i)}\left(\sum_{x_i\in\mathcal{X}_q} w_{i,e}h_e, D\right). \quad (21)$$

An additional static implementation removes this batch reduction from the per-step decode path. It forms a pool once from the calibration-time norms,

$$B^{\mathrm{static}}_{\mathrm{dec}} = \mathrm{TopK}_{e\in\{1,\ldots,E\}}(h_e, D), \quad (22)$$

and precomputes one target and scalar for every possible source expert. Given any retained set B_ϕ, an omitted source expert selects the target with minimum calibrated projection loss:

$$\pi(s \mid B_\phi) = \arg\min_{t\in B_\phi} \ell_{s\to t}, \quad (23)$$

where B_ϕ is the prefill set in Eq. 20 or either decode pool in Eqs. 21–22.

Prefill folding. The prefill selector places the retained routes from Eq. 20 first, after which the Triton operator coalesces omitted routes assigned to the same retained target:

$$\widetilde{w}_{i,t} = w_{i,t} + \sum_{\substack{s\in S_K(x_i)\setminus B_{\mathrm{pre}}(x_i)\\ \pi(s|B_{\mathrm{pre}}(x_i))=t}} w_{i,s}s^{\star}_{s\to t}, \quad t \in B_{\mathrm{pre}}(x_i). \quad (24)$$

The operator does not materialize omitted expert outputs. Retained expert identifiers remain unchanged, so the fused MoE kernel executes only K_{pre} experts per token.

Listing 1: Abbreviated static ExFold remapping kernel.

```
@triton.jit
def _static_projector_remap_kernel(
      weights, ids, target_ids, target_scale,
      out_weights, out_ids, B,
      K: tl.constexpr, BLOCK_B: tl.constexpr):
   batch = tl.program_id(0) * BLOCK_B \
         + tl.arange(0, BLOCK_B)
   valid = batch < B

   for k in tl.static_range(K):
      w = tl.load(weights + batch * K + k,
                  mask=valid)
      src = tl.load(ids + batch * K + k,
                    mask=valid)
      dst = tl.load(target_ids + src,
                    mask=valid, other=0)
      scale = tl.load(target_scale + src,
                      mask=valid, other=1.0)
      tl.store(out_weights + batch * K + k,
               w * scale, mask=valid)
      tl.store(out_ids + batch * K + k,
               dst, mask=valid)
```

Decode remapping. Decode first obtains a dynamic or static pool. Each routed expert outside that pool selects its target using Eq. 23 and is remapped in place:

$$(E_s, w_{i,s}) \longmapsto \left(E_{\pi(s|B_{\mathrm{dec}})}, w_{i,s} s^{\star}_{s\to\pi(s|B_{\mathrm{dec}})}\right). \qquad (25)$$

The lookup and remapping occur before dispatch, so every routed identifier presented to the fused kernel belongs to a pool of at most D experts.

Decode kernels. Here B_{dec} is the dynamic or static pool selected above. The evaluated dynamic path scans the BK routed slots to build Eq. 21, then searches D candidate targets for each omitted route. The static implementation instead caches $\pi(s \mid B_{\mathrm{dec}})$ and $s^{\star}_{s\to\pi(s|B_{\mathrm{dec}})}$ for all E sources. Listing 1 is the resulting per-route kernel: one target and scalar lookup per routed slot, or $O(BK)$ work, while preserving the original $[B, K]$ layout.

Cross-Model Quality Results

Table 10 reports DeepSeek-V2-Lite-Chat under joint compression. Table 11 separates prefill-only, decode-only, and joint folding on Qwen3.5-35B-A3B. In the method labels, P4 retains four routed experts per prefill token; D32 and D64 cap each decode batch's active expert pool at 32 and 64 experts; a combined label applies both constraints. For Qwen3.5 P4 and P4+D64, directed pairs with calibrated loss above 0.99 are excluded from target assignment; after prefill folding, the retained weights are rescaled per token to preserve the original routed-weight sum. This safeguard is not enabled for D64. All rows use complete benchmark coverage under the evaluation protocol described above. The "Eval+" column reports HumanEval+ pass@1.

Method	IFEval	GPQA	Eval+	MMLU_{pro}	Avg.
Original Top-6	**42.14**	23.23	46.34	**26.52**	34.56
Direct Top-4	40.85	24.24	46.95	24.71	34.19
ExFold P4+D32	41.96	**26.77**	**48.17**	24.75	**35.41**

Table 10: DeepSeek-V2-Lite-Chat quality; bold marks the column best.

Method	MATH_{500}	IFEval	IFBench	GPQA	Eval+	Avg.
Original Top-8	89.00	**87.06**	29.20	82.83	88.41	75.30
Direct Top-4	32.80	81.52	33.53	82.32	87.20	63.47
ExFold P4	88.40	82.62	32.59	82.32	**92.68**	75.72
ExFold D64	**89.20**	85.21	**38.40**	81.31	89.63	76.75
ExFold P4+D64	88.40	83.36	33.77	**86.36**	92.07	**76.79**

Table 11: Qwen3.5-35B-A3B quality; bold marks the column best.

Expert Geometry and Projector Structure

Full-Layer Expert Geometry

We visualize expert-output geometry for every MoE layer. For Qwen3 output matrices $\mathbf{E}_s$ and $\mathbf{E}_t$ collected on common inputs, raw similarity is $1 - \|\mathbf{E}_s - \mathbf{E}_t\|_F / d_{\max}$, where $d_{\max}$ is the largest raw pairwise distance in that layer. For this diagnostic, the aligned view fits the closed-form scalar on the same captured outputs, averages the two directed similarities, and reuses the same $d_{\max}$. GLM reconstructs the corresponding raw and aligned distances from source-norm-weighted co-routing statistics and averages the two directions using their observation counts. Thus, each aligned distance is no larger than its raw counterpart; these analysis-only fits are separate from the fixed deployment projectors visualized below. The magnitude bar charts show every layer–expert position, use an independent y-axis in each layer, and highlight the minimum- and maximum-magnitude experts. Unavailable observations are gray; the isolated GLM layer-45 outlier is hatched and excluded from that panel's axis scaling. Across observed directed off-diagonal pairs, scalar alignment raises the mean similarity from 0.449 to 0.613 for Qwen3. Across observed symmetric GLM entries, the corresponding mean rises from 0.899 to 0.932 for GLM-4.5-Air.

Qwen3-30B-A3B. Figures 8, 9, and 10 show all 48 layers and 128 routed experts. The matrices exhibit layer-dependent structure, and the norm view confirms expert-wise magnitude variation.

GLM-4.5-Air. Figures 8, 11, and 12 provide the corresponding 45-layer diagnostics. GLM statistics are collected for co-routed expert pairs; gray cells denote pairs that were never jointly observed. The matrices again show nonuniform pair structure and output-magnitude differences.

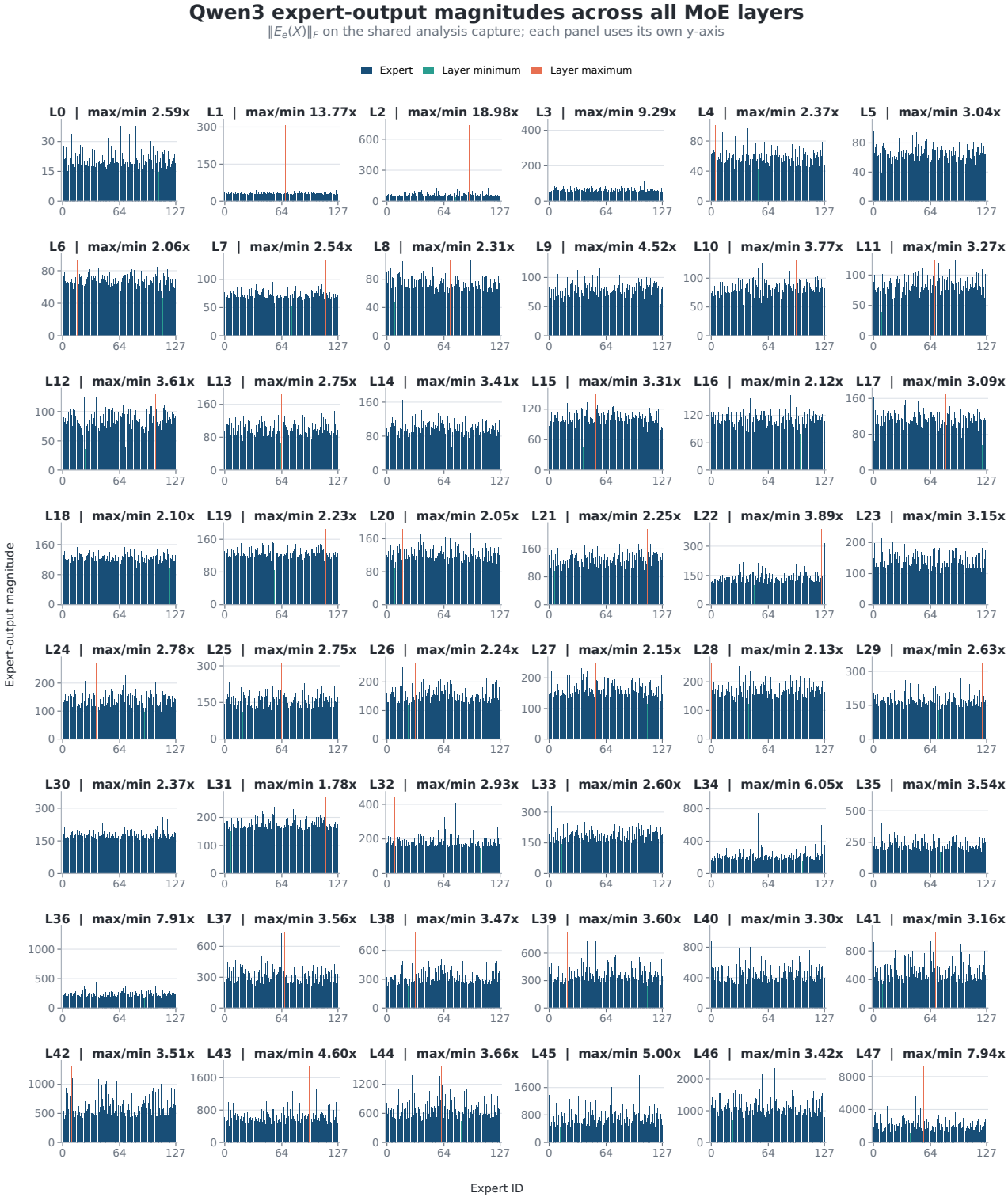


Figure 8: Expert-output magnitudes across all Qwen3 MoE layers.

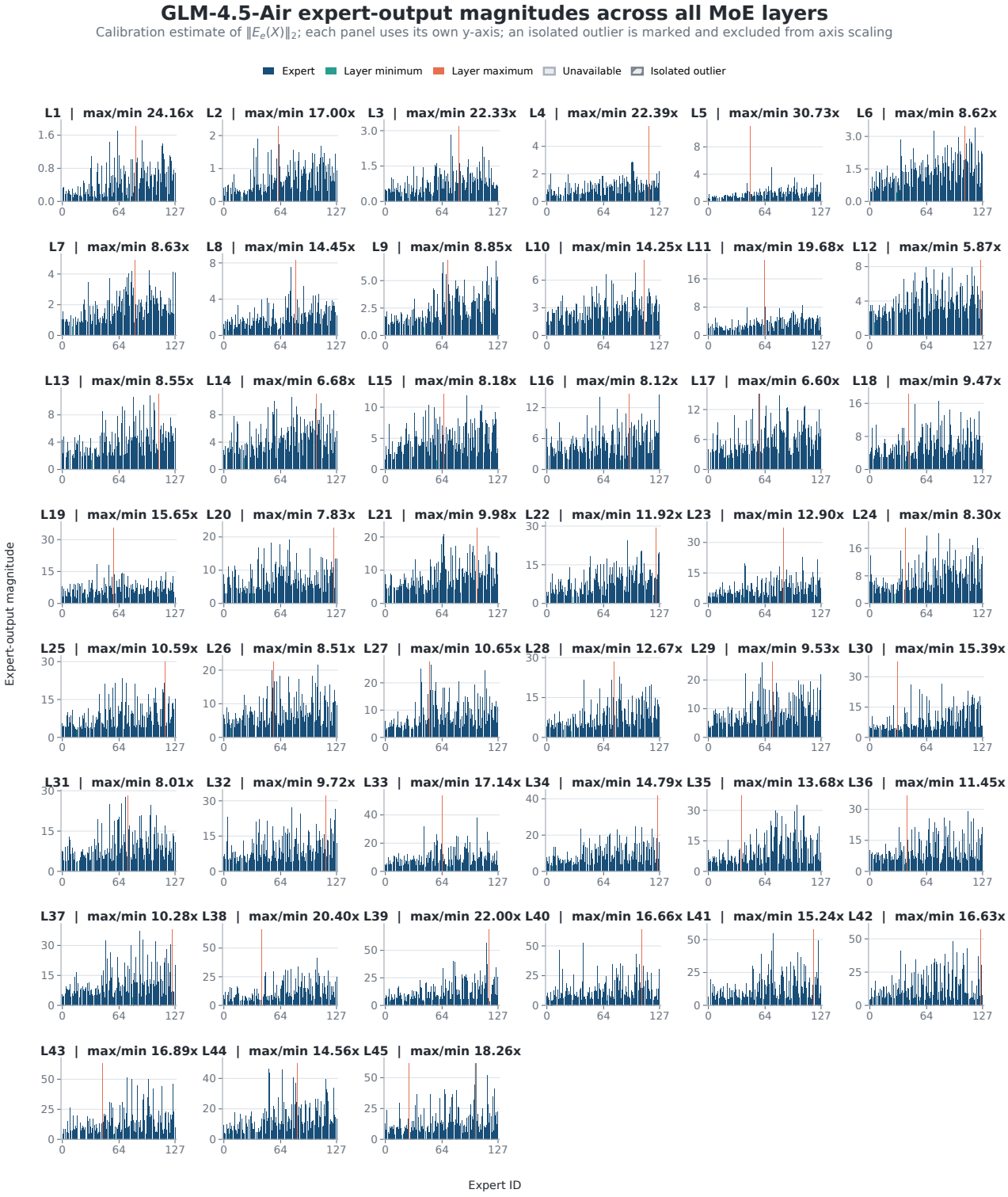


Figure 8: GLM-4.5-Air expert-output magnitudes (continued); hatching marks one isolated outlier.

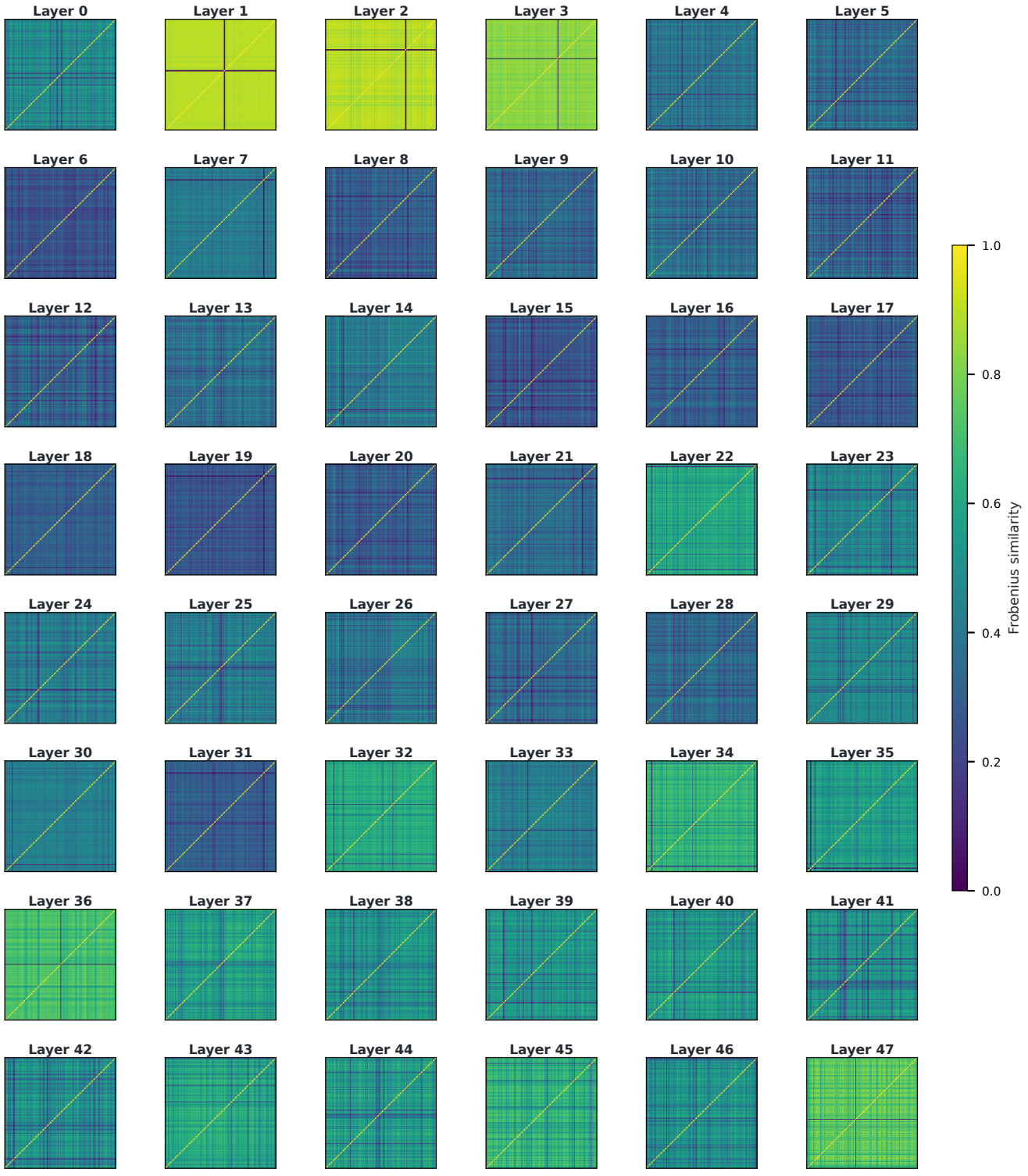


Figure 9: Raw expert-output similarity across all Qwen3 MoE layers.

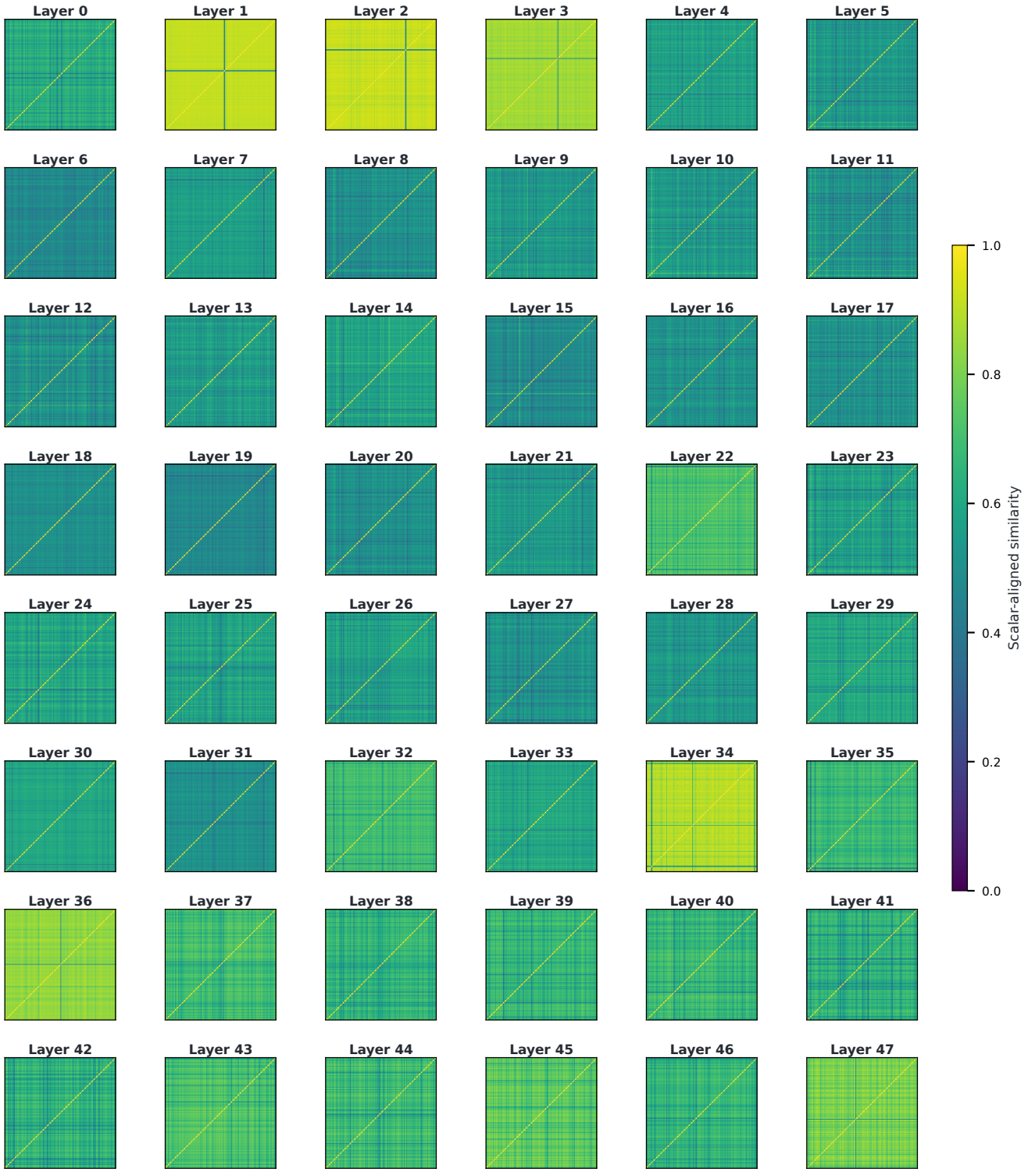


Figure 10: Scalar-aligned similarity for observed Qwen3 expert pairs.

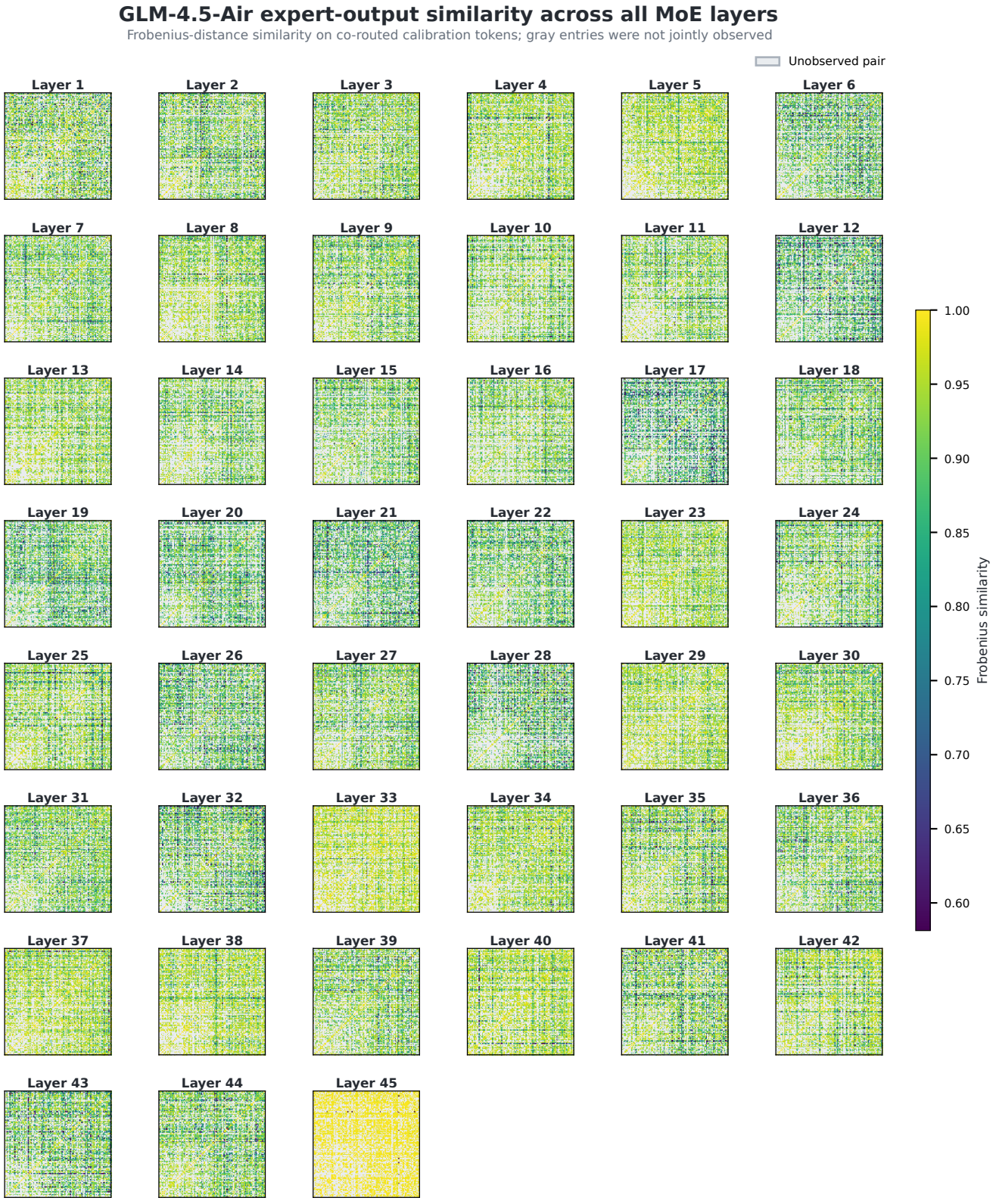


Figure 11: Raw co-routed expert similarity across all GLM-4.5-Air MoE layers.

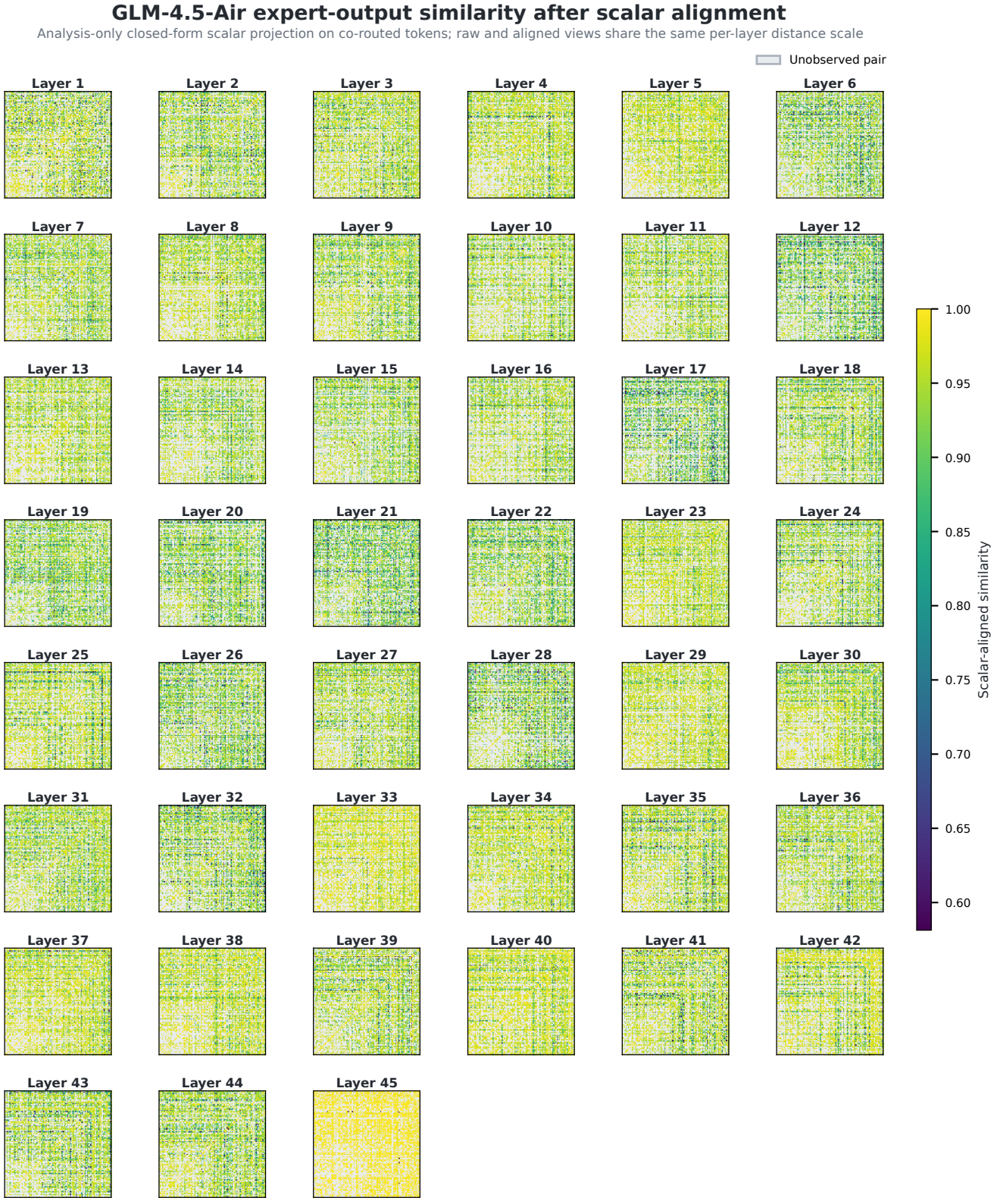


Figure 12: Scalar-aligned co-routed expert similarity across all GLM-4.5-Air MoE layers.

Layer-Wise Projector Structure

Figures 13 and 14 visualize dense common-input projector geometry for Qwen3. Every expert processes the same hidden-state matrix X; rows are source experts, columns are candidate targets, and self-pairs are suppressed. Scalar colors are clipped symmetrically at the 99th percentile of $|s^{\star}_{s\to t}|$. For a layer l, Figure 14 reports the directed normalized Frobenius loss $\widetilde{\ell}_{s\to t} = \|E_s(X) - s^{\star}_{s\to t}E_t(X)\|_F / d^{(l)}_{\max} = 1 - S_{\text{scalar}}(s,t)$, where $d^{(l)}_{\max}$ is the maximum raw pair distance in that layer. Its median over directed off-diagonal pairs is 0.407, and 79.9% of pairs are below 0.5. This common-input diagnostic avoids conflating missing pair support with projection quality; lower values indicate more compatible transfers.

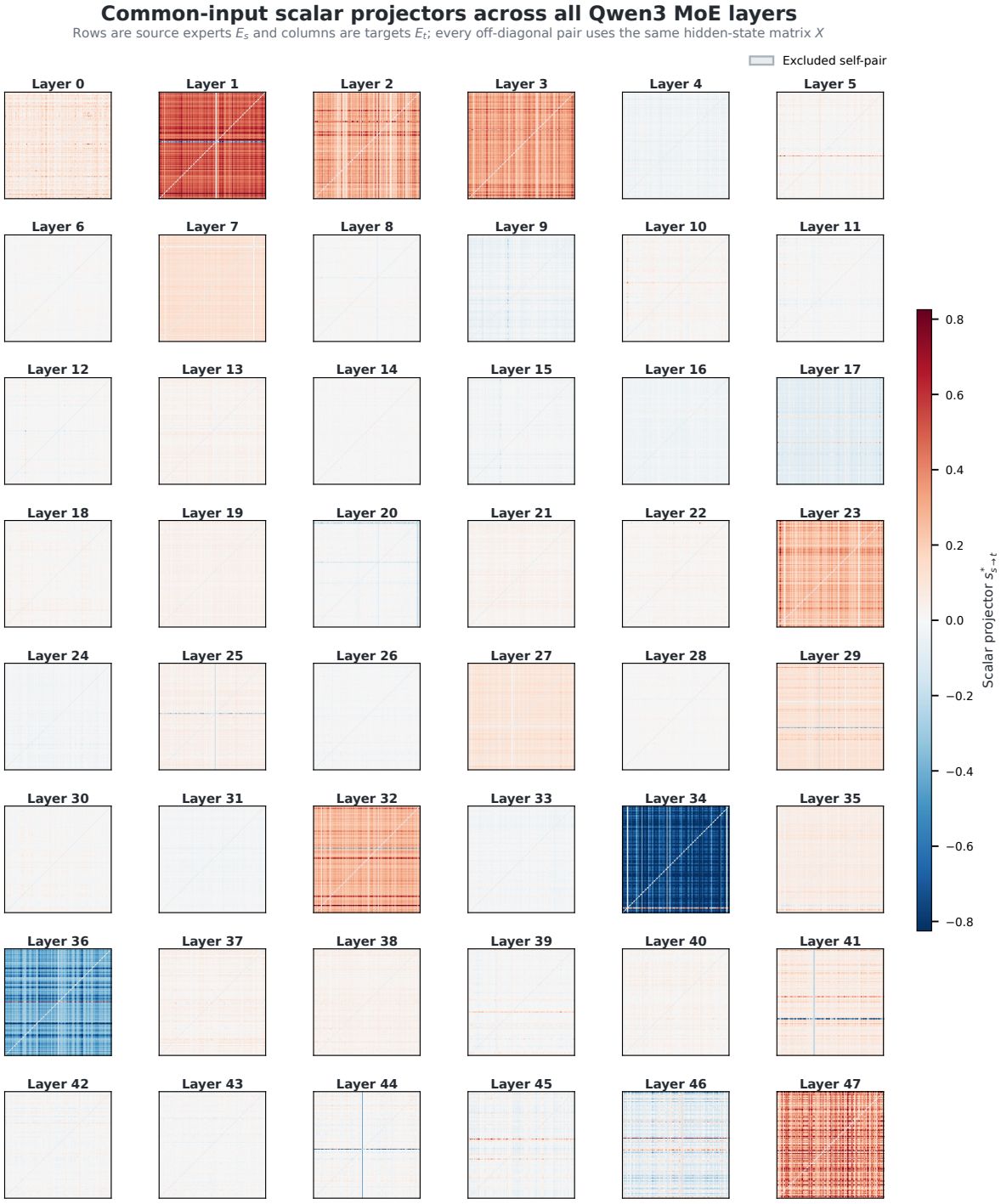


Figure 13: Common-input scalar-projector matrices across all Qwen3 MoE layers.

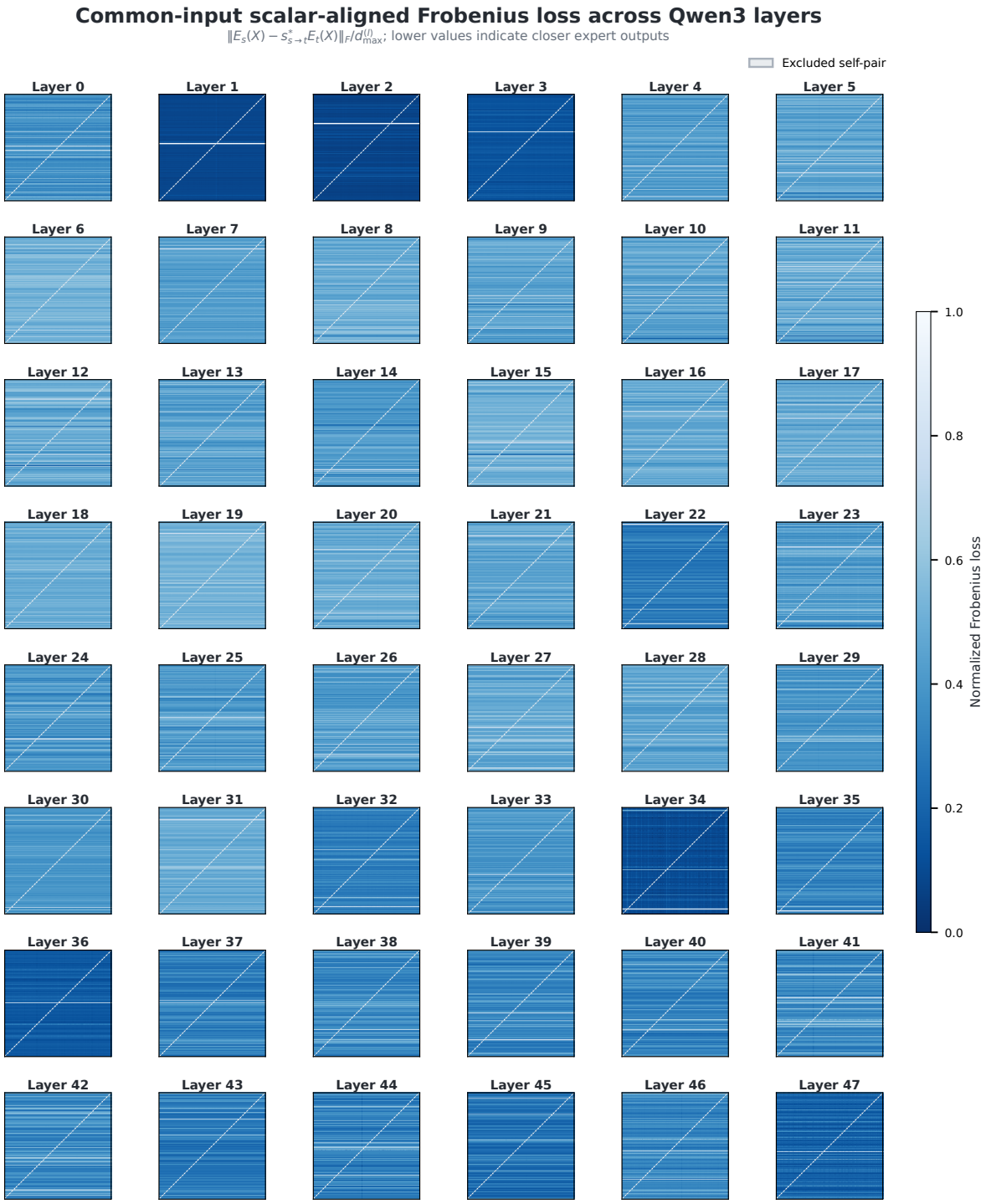


Figure 14: Common-input scalar-aligned Frobenius losses across all Qwen3 MoE layers.

DeepSeek-V4-Flash Extension Details

Model and quality runtime. DeepSeek-V4-Flash contains 284B total and 13B activated parameters, 43 transformer layers, 256 routed experts plus one shared expert, and Top-6 routing (DeepSeek-AI 2026). Quality evaluation uses tensor parallelism eight, `max_num_seqs=32`, and CUDA Graph. Every method is evaluated with the same prompts, chat formatting, decoding rules, and scorers. The full-suite collector is required to have no missing tasks or infrastructure failures before it is admitted to Table 6. All averages and retention ratios are computed from underlying unrounded task scores.

Calibration data and disclosure. The released DeepSeek matrix is calibrated from 64 unlabeled inputs: 56 general instruction, code, and mathematics inputs from Tulu-3, plus eight benchmark inputs (four IFEval and four IFBench) without labels, reference answers, or evaluator feedback. At most 64 observer tokens are collected per sequence, with maximum input length 4,096. This is *transductive, benchmark-aware calibration*, not a zero-contact held-out evaluation. Calibration uses no benchmark answers and changes no model weights, but the input exposure must be preserved when reporting the results.

The DeepSeek artifact uses common-input expert outputs, source-output-norm weighting, and unbounded least-squares scalars. Unlike the Qwen configuration, it does not clip scalar coefficients.

Confidence-aware P3 safeguard. DeepSeek-V4-Flash exhibits a wider range of router scores and expert-output magnitudes than the primary Qwen model. For an omitted source s and retained target t, we therefore compute a source-relative residual

$$\ell^{\mathrm{rel}}_{s\to t} = \frac{\sum_i w_i \|\mathbf{u}_i - a_{s\to t}\mathbf{v}_i\|_2^2}{\sum_i w_i \|\mathbf{u}_i\|_2^2 + \epsilon}, \qquad c_s = \left[1 - \min_t \ell^{\mathrm{rel}}_{s\to t}\right]_0^1. \tag{26}$$

The scalar-transfer part $w_s c_s a_{s\to t}$ is added to the minimum-loss target. The remaining weight $w_s(1-c_s)$ falls back to the retained Top-3 routes in proportion to their original router weights. This continuous safeguard does not execute additional experts: P3 still invokes exactly three routed experts. It only avoids forcing a poorly calibrated pair to absorb the entire omitted contribution.

Speed protocol and operating-point boundary. All DeepSeek speed measurements use one H800 server with model weights, code, data, and outputs on local storage. The vLLM runtime uses tensor parallelism four and CUDA Graph. Online load curves use `max_num_seqs=32` and the same request schedule for every method. Prefill uses 8,192 input tokens and one output token at QPS 1, 2, 4, 6, and 8. Decode uses one input token and 256 output tokens at QPS 2, 4, 8, 12, 16, 24, and 32. The offline-throughput panel uses `max_num_seqs=128`, QPS 64, concurrency 128, 32 warmups, and 1,024 measured requests.

The TTFT peak at QPS 4 includes queueing amplification and is therefore a serving-level speedup rather than a pure-kernel claim. Under the matched `max_num_seqs=32` decode limit, D64 stabilizes near 1.15× TPOT at QPS 8–32, while D128 remains near parity. The saturated `max_num_seqs=128` run yields a 1.286× output-throughput gain for D64. Because quality is measured with `max_num_seqs=32`, the throughput bar is reported as a separate saturation boundary rather than the same quality–speed operating point.

Open-source reproduction. The repository at https://github.com/Time-Rune/ExFold-MoE contains the Qwen3 and DeepSeek-V4-Flash runtime patches, CUDA/Triton kernels, final calibration matrices, correctness tests, and one-command quality and speed launchers. Model weights and benchmark datasets remain under their original licenses and are downloaded from their providers.

Token-Level Reconstructions

Figure 15 provides proxy-space illustrations for tokens from mathematics, scientific reasoning, instruction following, and code generation. The columns compare the original Top-8 routes with Direct Top-4, MoDES token-wise selection (Huang et al. 2025), REAP using Top-8 within a static 64-expert pool (Lasby et al. 2025), and ExFold Top-4. In this comparison, MoDES applies its calibrated layer-wise threshold to retain a token-dependent subset of at most four leading routes. Expert directions come from two-dimensional classical multidimensional scaling (MDS) of the all-expert cosine-distance matrix, and arrow lengths equal router weight times the calibrated expert-output norm; segment labels $1, \ldots, 8$ denote route-rank positions rather than global expert identifiers, and primed labels denote REAP routes. ExFold omits per-segment labels for clarity. Colored arrows show individual expert contributions in a strict head-to-tail chain: every arrow starts at the endpoint of the preceding contribution, and the chain endpoint is the corresponding aggregate. Background arrows ending in diamonds show the original aggregate $\mathbf{y}$ and each method's approximate aggregate $\hat{\mathbf{y}}$; the latter is dashed. The displayed proxy distance is their relative Euclidean distance in this two-dimensional space. Within each dataset, we inspect at most eight evaluation examples and 96 deterministically spaced tokens per example. We select an illustrative case whose Direct-Top-4 proxy distance is at least 0.15, for which ExFold reduces that distance by at least 15%, and for which ExFold has the smallest proxy distance among the displayed approximations. To span model depth, the four datasets target layers 0, 16, 32, and 47, respectively; each panel uses the nearest qualifying layer and the first case in deterministic scan order. These diagrams illustrate routing geometry; their reported distances are proxy-space quantities and are not hidden-state reconstruction errors. They are qualitative examples rather than aggregate evaluation evidence.

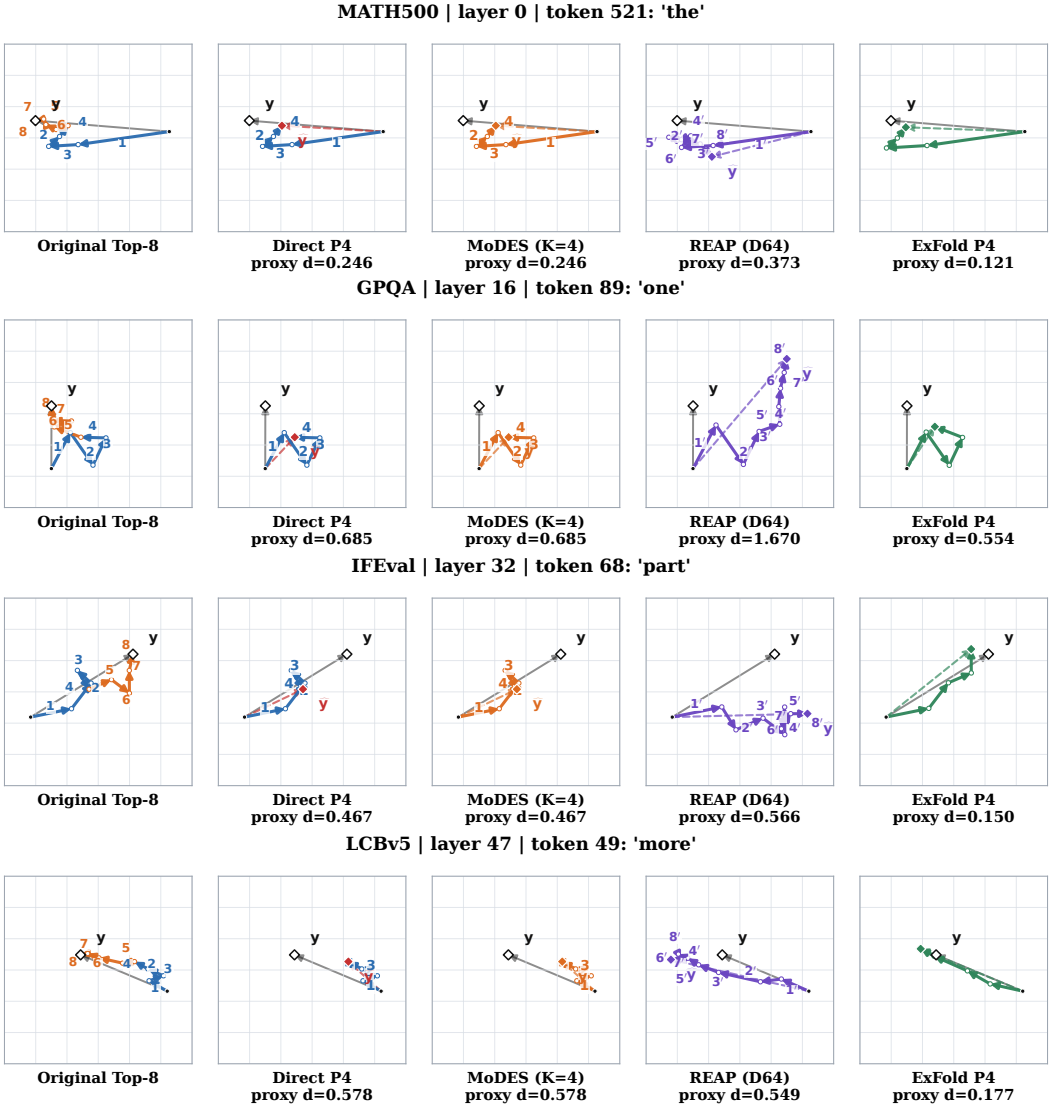


Figure 15: Token-level proxy reconstructions. Expert contributions are accumulated strictly head-to-tail; each chain terminates at $\mathbf{y}$ or $\hat{\mathbf{y}}$.