PACES: PATTERN-CODED ELEMENT SHARING FOR LIGHTWEIGHT MIXTURE-OF-EXPERTS INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparsely activated Mixture-of-Experts (MoE) models improve computational efficiency via token-wise sparse activation, but still require full dense parameter storage, severely limiting their practical deployment. Existing MoE compression approaches fail to achieve a favorable trade-off between model accuracy and storage overhead. In this work, we propose PACES (Pattern-Coded Element Sharing), a post-training compression method for MoE language models. Based on the key observation that intra-layer experts exhibit consistent Gaussian weight distributions, PACES groups similar experts and performs distribution-aware element-wise magnitude sharing. It compactly encodes expert participation and sign information using lightweight bit-level codes. Equipped with a fused GPU kernel that avoids dense weight reconstruction during inference, PACES enables true end-to-end storage and latency optimization. Evaluated on Qwen-MoE and DeepSeek-V2-Lite over eight zero-shot benchmarks, our method reduces checkpoint storage by 32.6%–57.3% and achieves up to $4.05\times$ inference speedup with less accuracy degradation. It consistently outperforms unstructured pruning and expert merging baselines under aggressive compression, providing a practical deployment-friendly compact representation for real-world MoE serving.

1 INTRODUCTION

Mixture-of-Experts (MoE) models expand model capacity by substituting dense feed-forward layers with a set of expert sub-networks, where each input token is dynamically routed to only a small subset of experts for computation (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022). This conditional computation paradigm enables extremely large-scale language models with low per-token active computation, as demonstrated by representative architectures including Switch Transformer, GLaM, and Mixtral (Fedus et al., 2022; Du et al., 2022; Jiang et al., 2024). Despite the efficiency of sparse activation during inference, MoE models still require storing the complete parameters of all experts. As a result, the serving system must load the full expert pool into accelerator memory even when only a handful of experts are activated per token, making checkpoint size and GPU memory overhead the primary bottlenecks for real-world MoE deployment.

Post-training compression provides a practical solution to alleviate the excessive parameter burden of pre-trained MoE models by eliminating redundant parameters or structures, thereby reducing storage and computational costs. Three mainstream compression paradigms have been widely explored, each with inherent trade-offs. Unstructured pruning, exemplified by SparseGPT and Wanda (Frantar & Alistarh, 2023; Sun et al., 2024), zeroes out individual weights according to predefined importance criteria while preserving the original tensor dimensions. Expert pruning removes some experts and their router rows (Lasby et al., 2026; Liu et al., 2026). These methods narrow the router's choices and can sharply degrade quality at high compression ratios. Expert merging compresses MoE models by averaging their parameters to reduce the number of distinct expert functions. Frequency-weighted hierarchical clustering serves as a typical retraining-free merging strategy (Chen et al., 2025).

Overall, existing retraining-free MoE compression methods leave an important deployment trade-off unresolved. Methods that preserve accuracy by pruning individual weights generally retain the original tensor dimensions and therefore provide limited physical reduction in GPU storage and

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

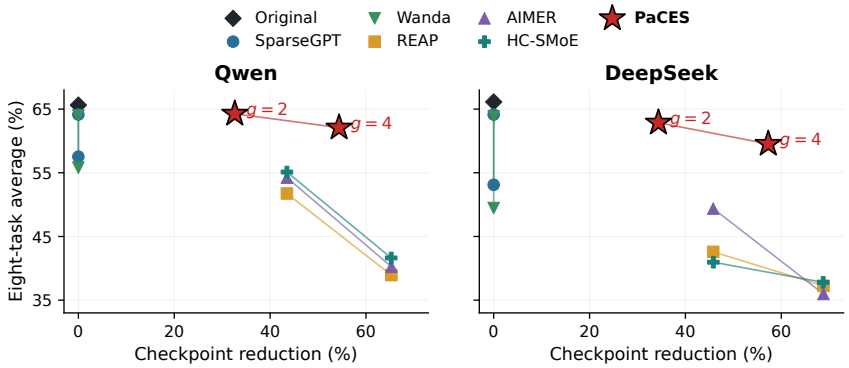


Figure 1: Quality–storage Pareto frontier on Qwen and DeepSeek. Each point reports the eight-task average accuracy and the corresponding checkpoint reduction, with higher values preferred on both axes. Lines connect the 50% and 75% settings within each baseline family, while red stars mark PACES at $g = 2$ and $g = 4$. PACES forms the upper Pareto envelope, retaining substantially higher accuracy at comparable compression levels.

computation. In contrast, methods that achieve high physical compression by removing or merging experts reduce the available expert pool and parameter diversity, which can lead to substantial accuracy degradation, especially at aggressive compression ratios.

To address this trade-off, we present Pattern-Coded Element Sharing (PACES), a post-training compression method that reduces expert storage while preserving the original routing structure and supporting direct GPU execution from the compressed representation. We first examine the weight distributions of experts within MoE layers and find that projection weights in mainstream MoEs, such as Qwen1.5-MoE and DeepSeek-V2-Lite, approximately follow zero-mean Gaussian distributions with consistent scales across experts of the same layer.

Building on this observation, PACES groups experts within each MoE layer by router frequency and pairwise expert similarity. Leveraging the observation that projection parameters of experts in the same layer follow zero-mean Gaussian distributions with matched statistics, PACES separates statistically redundant central values from functionally important, expert-specific deviations. Instead of pruning entire experts or zeroing isolated weights, PACES enables element-wise magnitude sharing across grouped experts: values falling within the central Gaussian region are compressed by retaining a single shared magnitude, while outlier entries that encode expert specialization are represented via distance-weighted shared magnitudes. We compactly encode the resulting pattern using shared magnitude values paired with per-expert participation and sign bits. This representation retains all expert identities and the original routing policy, and can be directly consumed by a fused GPU kernel without materializing full dense expert matrices. Extensive experiments demonstrate that our method achieves MoE model compression with negligible accuracy loss and delivers inference acceleration under practical deployment settings.

Our contributions are summarized as follows:

- We propose PACES, a retraining-free MoE compression method that combines router-guided expert grouping with distribution-aware pattern coding. The method shares element-wise magnitudes while preserving expert-specific participation and sign information.
- We develop an end-to-end compact storage and inference GPU kernel that executes directly on the encoded representation. It reduces model storage by 32.6–57.3% and yields up to $4.05\times$ and $1.47\times$ inference speedups on native Qwen-MoE and DeepSeek-V2-Lite, respectively.
- We conduct a systematic evaluation across eight standard benchmarks on Qwen1.5-MoE and DeepSeek-V2-Lite against unstructured pruning, expert pruning, and structural merging baselines. Under aggressive 75% compression, PACES surpasses matched 75%-sparse SparseGPT and Wanda by 4.55–10.07 points, establishing a favorable quality–storage trade-off.

2 Related Work

Sparse mixture-of-experts models. Conditional expert routing originates from adaptive mixtures of local models (Jacobs et al., 1991) and was first applied to neural language models by Shazeer et al. (2017). Building on this, GShard and Switch Transformer (Lepikhin et al., 2021; Fedus et al., 2022) scaled routing and training to massive architectures, and GLaM (Du et al., 2022) showed that running only a fraction of the network keeps compute costs low. More recent open models, including Mixtral, Qwen1.5-MoE, and DeepSeek-V2 (Jiang et al., 2024; Qwen Team, 2024; DeepSeek-AI, 2024), have made sparse expert routing a practical foundation for modern language models. DeepSeek-V3 and Qwen3 (DeepSeek-AI, 2025; Qwen Team, 2025) continue this trend at a larger scale and with smarter routing and training strategies.

Large-model compression. To compress large models without expensive retraining, one-shot pruning broadly follows two directions: unstructured and structured pruning. First, unstructured pruning removes individual weights arbitrarily to preserve model quality. For example, SparseGPT (Frantar & Alistarh, 2023) employs second-order error compensation, whereas Wanda (Sun et al., 2024) prunes directly via weight–activation magnitudes. Second, structured pruning removes entire channels, blocks, or other complete components to keep tensor shapes regular. This directly speeds up standard GPUs, but at the cost of pruning flexibility.

MoE-specific compression. MoE compression broadly follows two directions: expert pruning and expert merging. First, expert pruning offers direct structural reduction by deleting experts and router rows. REAP (Lasby et al., 2026) ranks experts via router-weighted activations, whereas AIMER (Liu et al., 2026) uses calibration-free weight distributions. Second, expert merging combines similar experts rather than deleting them to save memory. For instance, HC-SMoE (Chen et al., 2025) clusters experts by their outputs and merges each group with frequency-weighted parameter averaging.

Efficient sparse and MoE kernels. Efficient execution of MoE architectures relies heavily on dedicated system frameworks. In particular, DeepSpeed-MoE (Rajbhandari et al., 2022) and MegaBlocks (Gale et al., 2023) improve routing, communication, and block-sparse expert computation. Meanwhile, Triton (Tillet et al., 2019) provides a productive language for developing custom GPU kernels for non-standard formats.

3 Motivation: Distributional Structure in Routed Experts

3.1 Why compress routed-expert weights?

Routed feed-forward experts comprise most parameters in modern sparse MoE language models (Jiang et al., 2024; DeepSeek-AI, 2024; 2025; Qwen Team, 2025). The models studied here illustrate this property: routed expert weights account for 12.46B of Qwen1.5-MoE-A2.7B's 14.32B parameters (87.0%) and 14.40B of DeepSeek-V2-Lite's 15.71B parameters (91.7%). Consequently, compressing these routed expert weights is the most effective way to shrink the overall model footprint.

As discussed in Section 1, existing MoE compression methods face a fundamental dilemma between physical hardware savings and expert capability preservation. This trade-off motivates **element-wise parameter sharing**. However, how to effectively compress the values of multiple experts at every element into a single shared magnitude remains an open question, which motivates our empirical analysis below.

3.2 Distributional regularity within MoE layers

Weight-distribution properties have long guided effective compression, from ACIQ's clipping thresholds (Banner et al., 2019) to QLoRA's NormalFloat4 quantiles (Dettmers et al., 2023). We extend this perspective to sparse MoEs, investigating whether multiple routed experts can be characterized under a shared, well-behaved distribution.

We empirically examine the distribution shape and scale of expert projection weights across Qwen and DeepSeek. Across both architectures, routed weights exhibit symmetric, bell-shaped distributions

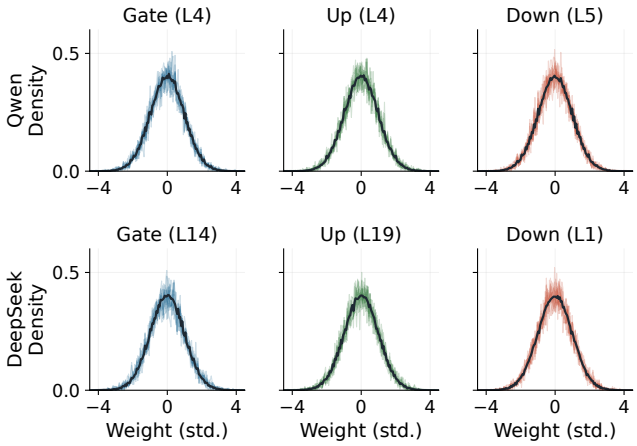


Figure 2: Expert weight distributions. Thin curves show 12 experts; black curves pool all experts. Values use the same layer-and-projection normalization; L denotes the layer.

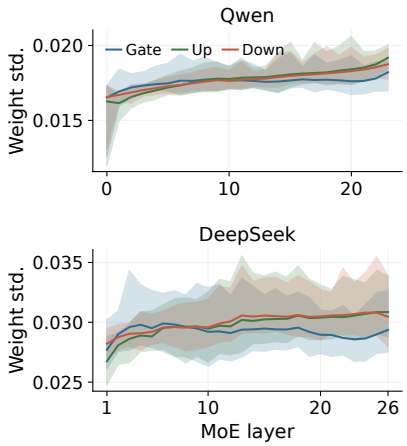


Figure 3: Expert weight scales. Lines show layerwise means; bands span the experts.

centered near zero, indicating that individual expert projections are well approximated by zero-mean Gaussians. Beyond similar distributional shapes, effective element-wise sharing requires consistent variances to prevent larger-magnitude experts from dominating shared values. To evaluate this scale consistency within layers, we sample 4,096 weights from each expert projection and normalize them using the layer-wide pooled mean and standard deviation. Weight histograms (Figure 2) and layer-wise standard deviations (Figure 3) confirm that experts within any given layer share similar distributions on remarkably consistent scales. Furthermore, the median within-layer coefficient of variation is merely 1.7–2.3% for Qwen and 3.3–3.7% for DeepSeek. These results confirm that experts within a layer have closely aligned variances. Detailed definitions of the diagnostics, the sampling protocol, and additional distributional and scale-consistency results are provided in Appendix D.

Together, the zero-mean Gaussian distribution and aligned variances indicate that experts within the same layer follow approximately consistent distributions. This empirical regularity motivates our distribution-guided sharing rule in Section 4.2 in two key aspects. First, because intra-layer experts share an aligned scale, pooled group statistics provide a faithful and robust statistical reference. Second, the approximately Gaussian form provides a natural threshold: weights within $[\mu - \sigma, \mu + \sigma]$ constitute the central mass of the distribution and are candidates for pruning, whereas values outside this range represent stronger expert-specific deviations that should be retained. Based on these statistical foundations, Section 4 details the complete PACES framework

4 METHODOLOGY

In this section, we present Pattern-Coded Element Sharing (**PACES**), a post-training compression framework that couples element-level magnitude sharing with fused kernel execution. To formalize the setting, consider a sparse MoE layer with E routed experts, where each expert comprises gate, up, and down projections denoted by $W_e^{(p)} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ for $p \in \{\text{gate}, \text{up}, \text{down}\}$. PACES partitions the E experts into balanced groups of fixed size g, replacing the g scalar weights at each coordinate (r, c) with a single BF16 magnitude while packing per-expert participation and sign states into compact bit patterns. Crucially, all routing weight, attention blocks, shared experts and dense blocks remain unchanged. During inference, a dedicated GPU kernel decodes these packed states on the fly directly during matrix computation, eliminating the need to materialize dense weights in global memory. Figure 4 summarizes the entire path from group construction to integrated execution.

4.1 ROUTER-GUIDED BALANCED EXPERT GROUPING

Intuitively, experts with similar functional roles exhibit greater parameter and representational redundancy, therefore PACES begins by partitioning the E routed experts within each layer into E/g

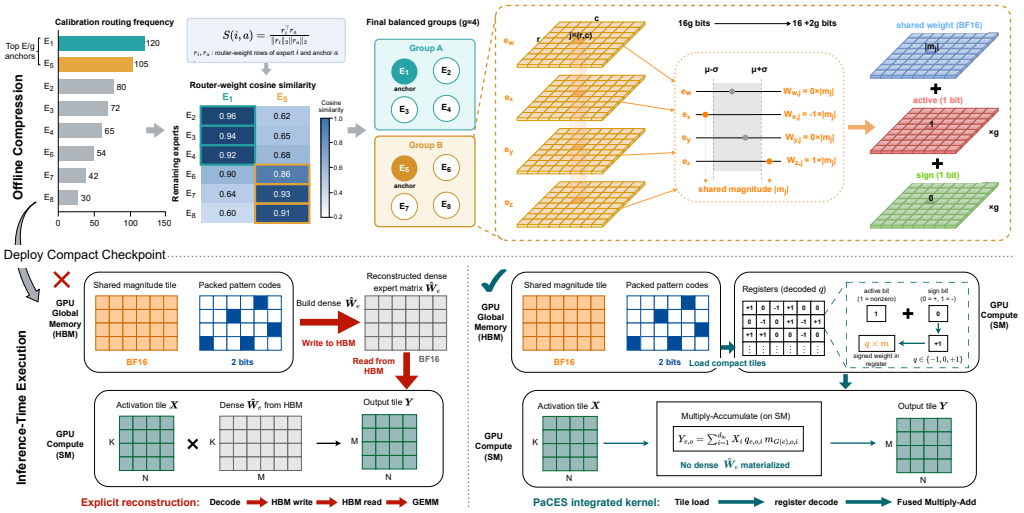


Figure 4: PaCES overview. Top (offline compression): router-guided grouping and element sharing encode each group with a shared BF16 magnitude and per-expert participation and sign bits. Bottom (inference): the integrated kernel decodes values in registers and computes outputs without materializing those matrices.

groups of size g. A small generic C4 calibration set (Raffel et al., 2020) provides route counts c_e for expert e. Let $\boldsymbol{r}_e \in \mathbb{R}^d$ be row e of the layer's router weight matrix which determines how tokens are routed to expert e and $\bar{\boldsymbol{r}}_e = \boldsymbol{r}_e / \|\boldsymbol{r}_e\|_2$. We choose the E/g most frequently routed experts as group anchors. For every remaining expert i and anchor a, the assignment score is

$$S(i, a) = \bar{\boldsymbol{r}}_i^\top \bar{\boldsymbol{r}}_a. \tag{1}$$

Candidate pairs are sorted by decreasing score and accepted greedily subject to a capacity of g experts per anchor. This produces balanced groups, retains a frequently used expert in every group, and uses only static router rows as the compatibility representation. Section 5.4 compares this choice with other grouping methods.

4.2 Distribution-Guided Coordinate Sharing and State Representation

For a fixed projection $p \in \{\text{gate}, \text{up}, \text{down}\}$, coordinate $j = (r, c)$ denotes the matching row and column position across all expert weight matrices. We write $w_{e,j} = (W_e^{(p)})_{r,c}$ for expert e's scalar weight at that position. For each group G, our goal is to compress the g scalar weights at coordinate j into a single shared positive magnitude $m_j \geq 0$ and a compact state representation $q_{e,j} \in \{-1, 0, +1\}$ per expert, such that the reconstructed weight is given by

$$\widehat{w}_{e,j} = q_{e,j} m_j. \tag{2}$$

Here, $q_{e,j} = 0$ indicates that expert e is pruned at coordinate j, while $q_{e,j} \in \{+1, -1\}$ encodes its active participation and sign.

Following the observation in Section 3.2, we compute the pooled mean and standard deviation over all weights in group G and projection p:

$$\mu_{G,p} = \text{mean}\{w_{e,j} : e \in G, j \in \mathcal{J}\}, \qquad \sigma_{G,p} = \text{std}\{w_{e,j} : e \in G, j \in \mathcal{J}\}, \tag{3}$$

where $\mathcal{J}$ denotes the set of all matrix coordinates. Because intra-layer weights follow a zero-centered Gaussian distribution, weights within the central interval $[\mu_{G,p} - \sigma_{G,p},\ \mu_{G,p} + \sigma_{G,p}]$ are low-magnitude values near zero and are treated as redundant. By contrast, outlier weights outside this interval encode expert-specific functional specialization. We identify the set of outlier experts at coordinate j as

$$O_j = \left\{ e \in G \;\middle|\; w_{e,j} \notin [\mu_{G,p} - \sigma_{G,p},\ \mu_{G,p} + \sigma_{G,p}] \right\}. \tag{4}$$

We then determine m_j and $q_{e,j}$ according to whether outliers are present.

Inlier-only ($O_j = \emptyset$). When all g experts fall within $\mu_{G,p} \pm \sigma_{G,p}$, their weights remain near zero. Pruning all g experts would leave the coordinate entirely inactive, hurting the model's expressive capacity. To prevent this while keeping sparsity high, we retain only the expert with the largest magnitude:

$$e^* = \arg\max_{e \in G} |w_{e,j}|, \qquad m_j = |w_{e^*,j}|, \qquad q_{e,j} = \begin{cases} \mathrm{sgn}(w_{e^*,j}), & e = e^*, \\ 0, & e \in G \setminus \{e^*\}. \end{cases} \tag{5}$$

Outlier-dominated ($O_j \neq \emptyset$). When outliers exist, only these outliers participate, while inliers are omitted. To fuse multiple participating values into a single shared magnitude m_j, we weight each expert's contribution proportionally to its magnitude, motivated by the classic observation that larger weights carry primary representational influence (Han et al., 2015). We therefore compute the distance-weighted shared magnitude as

$$\alpha_{e,j} = \frac{|w_{e,j} - \mu_{G,p}|}{\sum_{i \in O_j} |w_{i,j} - \mu_{G,p}|}, \qquad m_j = \sum_{e \in O_j} \alpha_{e,j} |w_{e,j}|. \tag{6}$$

Each expert $e \in G$ is then assigned its corresponding ternary state:

$$q_{e,j} = \begin{cases} \mathrm{sgn}(w_{e,j}), & e \in O_j, \\ 0, & e \in G \setminus O_j. \end{cases} \tag{7}$$

This formulation naturally generalizes to any group size $g \geq 2$; we empirically evaluate both $g = 2$ and $g = 4$.

4.3 Integrated decoding-and-compute kernel

With conventional dense linear operators, the compact representation must first be decoded into $\widehat{W}_e$ using Equation 2. Repeatedly allocating and writing these signed expert matrices adds reconstruction cost to each forward pass. As shown in the lower half of Figure 4, our Triton kernel instead reads packed codes and BF16 magnitudes together. For each matrix tile, it extracts active and sign bits, forms signed values in registers, and accumulates directly into the matrix product. The layer-level MoE pipeline groups tokens by expert, applies the three encoded projections, restores token ordering, and combines the outputs with the original routing weights.

For expert e assigned to group $G(e)$, one output coordinate is accumulated directly as

$$y_{e,o} = \sum_{i=1}^{d_{\mathrm{in}}} x_i \, q_{e,o,i} \, m_{G(e),o,i}, \tag{8}$$

where the packed code yields $q_{e,o,i} \in \{-1, 0, +1\}$ inside the kernel. Equation 8 makes the execution boundary explicit: decoding is part of accumulation rather than a preprocessing step that writes $\widehat{W}_e$ to global memory.

5 Experiments

Our experiments address four questions. **RQ1:** how much downstream accuracy does PACES retain at moderate and aggressive routed-expert compression compared with pruning baselines? **RQ2:** how much does the compact representation reduce checkpoint size? **RQ3:** does integrated execution reduce latency relative to explicit reconstruction and native MoE implementations? **RQ4:** how do expert grouping affect accuracy?

5.1 Experimental setup

Models. We evaluate Qwen1.5-MoE-A2.7B (Qwen Team, 2024) and DeepSeek-V2-Lite (DeepSeek-AI, 2024) in BF16, with 60 and 64 routed experts per sparse layer, respectively. Additional model dimensions and evaluation settings are given in Appendix A.

Table 1: Eight-task accuracy and checkpoint storage on Qwen. Both PACES rows use distance-weighted magnitudes and integrated execution. The best task and average result within each compression block is bold. † marks a BF16 tensor-payload estimate; other sizes are measured weight-file totals.

Method	ARC-E	ARC-C	PIQA	BoolQ	Hella	OBQA	Wino	MMLU	Avg.	Ckpt. GiB	Ckpt. ↓
Original	69.19	44.20	80.63	79.54	77.34	44.00	69.38	60.80	65.63	26.67	–
SparseGPT-50	68.27	**43.34**	79.76	77.16	**75.05**	42.20	68.67	58.75	64.15	26.67†	0.0%
Wanda-50	68.22	43.26	79.33	76.36	74.67	**42.60**	68.75	**58.77**	63.99	26.67†	0.0%
REAP-50	52.65	34.39	72.91	57.52	64.21	34.00	64.40	33.91	51.75	15.06†	43.5%
AIMER-50	55.89	34.30	72.52	71.25	62.42	32.80	63.06	41.66	54.24	15.06†	43.5%
HC-SMoE-50	59.81	36.86	70.51	74.74	59.67	31.00	64.25	44.17	55.13	15.06	43.5%
PACES ($g = 2$)	**70.75**	**43.34**	**79.82**	**78.53**	74.52	41.00	**68.82**	57.36	**64.27**	17.96	32.6%
SparseGPT-75	62.84	37.12	74.92	71.71	62.51	36.60	**67.17**	47.43	57.54	26.67†	0.0%
Wanda-75	61.15	36.26	73.83	70.18	59.84	34.80	66.61	43.75	55.80	26.67†	0.0%
REAP-75	34.60	24.74	59.09	49.60	39.29	27.00	53.83	23.30	38.93	9.26†	65.3%
AIMER-75	36.87	25.09	59.14	58.56	38.16	24.40	53.35	26.45	40.25	9.26†	65.3%
HC-SMoE-75	34.93	25.51	57.89	66.88	43.97	27.20	50.04	26.61	41.63	9.26	65.3%
PACES ($g = 4$)	**68.86**	**41.89**	**77.75**	**75.84**	**71.63**	**41.20**	65.04	**54.51**	**62.09**	12.16	54.4%

Table 2: Eight-task accuracy and checkpoint storage on DeepSeek. The red entries for SparseGPT-50 and Wanda-50 indicate zero realized checkpoint reduction.

Method	ARC-E	ARC-C	PIQA	BoolQ	Hella	OBQA	Wino	MMLU	Avg.	Ckpt. GiB	Ckpt. ↓
Original	74.28	46.42	79.92	80.49	77.78	44.40	70.17	55.57	66.13	29.26	–
SparseGPT-50	**72.69**	41.81	79.00	**79.85**	**74.80**	**43.20**	69.85	**51.99**	**64.15**	29.26†	**0.0%**
Wanda-50	72.47	**43.00**	**79.27**	78.81	73.67	42.40	**70.01**	51.11	63.84	29.26†	**0.0%**
REAP-50	39.10	26.02	64.85	51.47	53.42	28.80	53.51	23.52	42.59	15.85†	45.8%
AIMER-50	51.89	34.22	67.68	60.31	58.94	31.20	61.72	29.10	49.38	15.85†	45.8%
HC-SMoE-50	36.32	24.23	58.87	63.73	39.35	26.60	54.14	24.43	40.96	15.85	45.8%
PACES ($g = 2$)	71.30	42.75	77.58	77.77	73.22	40.60	69.22	50.51	62.87	19.20	34.4%
SparseGPT-75	58.59	33.19	71.65	73.18	54.37	35.00	65.35	33.60	53.12	29.26†	0.0%
Wanda-75	53.58	31.57	68.50	70.55	50.09	30.20	62.59	28.62	49.46	29.26†	0.0%
REAP-75	30.98	23.63	56.80	50.00	34.90	26.60	51.54	23.59	37.26	9.14†	68.8%
AIMER-75	31.82	23.81	53.26	48.17	31.48	23.60	52.41	23.25	35.97	9.14†	68.8%
HC-SMoE-75	30.81	22.95	55.60	60.73	30.43	27.40	51.46	22.96	37.79	9.15	68.7%
PACES ($g = 4$)	**65.24**	**38.23**	**75.68**	**76.73**	**68.69**	**39.20**	**68.82**	**43.68**	**59.53**	12.50	57.3%

Calibration. PaCES grouping uses seed 42 and four sequences from C4 (Raffel et al., 2020), each with length 512. We keep at most 128 calibration tokens per Qwen layer and 512 per DeepSeek layer when forming route counts. Router rows are read directly from the pretrained models. The method performs no training or task-specific calibration.

Tasks and metrics. We use eight zero-shot tasks: ARC-Easy and ARC-Challenge (Clark et al., 2018), PIQA (Bisk et al., 2020), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), OpenBookQA (Mihaylov et al., 2018), WinoGrande (Sakaguchi et al., 2021), and the 57-task MMLU suite (Hendrycks et al., 2021). We report normalized accuracy for ARC, PIQA, HellaSwag, and OpenBookQA; accuracy for BoolQ and WinoGrande; and overall accuracy for MMLU. "Average" is the unweighted macro average of these eight values. Evaluation uses the LM Evaluation Harness (Gao et al., 2024).

Baselines. SparseGPT and Wanda provide unstructured-pruning comparisons; REAP and AIMER remove complete experts; and HC-SMoE clusters and merges experts without retraining (Frantar & Alistarh, 2023; Sun et al., 2024; Lasby et al., 2026; Liu et al., 2026; Chen et al., 2025). We compare

$g = 2$ with the nominal 50% settings and $g = 4$ with the 75% settings. These levels match the reduction in independent routed magnitudes or experts, while the tables separately report realized checkpoint size.

5.2 Quality and Realized Checkpoint Compression

To characterize the practical quality–storage trade-off, we evaluate all methods by their eight-task average accuracy and checkpoint saving. Figure 1 presents the resulting quality–storage Pareto frontier. Tables 1 and 2 detail task-level performance alongside physical checkpoint footprints.

Practical Compression at Moderate Sparsity ($g = 2$). At the 50% compression ratio, PACES delivers practical storage savings where conventional unstructured pruning fails. As shown in Table 1, on Qwen, PACES achieves an average score of 64.27%, matching and slightly outperforming SparseGPT-50 and Wanda-50. On DeepSeek, PACES closely preserves the task performance within approximately one point of these baselines (62.87% vs. 64.15% and 63.84%). However, PACES leverages its compact state representation to reduce actual disk storage by 32.6% on Qwen and 34.4% on DeepSeek. In contrast, coarse-grained expert compression methods (HC-SMoE, REAP, and AIMER) attain marginally higher compression ratios than PACES but incur substantial accuracy loss. On Qwen, they underperform PACES by 9.14–12.52 points, while on DeepSeek, their accuracy gaps widen to 13.49–21.91 points.

Robust Capability Under Aggressive Compression ($g = 4$). Under the more demanding 75% setting ($g = 4$), the advantage of element-wise distribution sharing becomes pronounced. As unstructured baselines degrade rapidly in high-sparsity regimes, PACES demonstrates substantial resilience: it outperforms SparseGPT-75 by 4.55 points on Qwen and 6.42 points on DeepSeek, and Wanda-75 by 6.29 and 10.07 points. This dominance is remarkably consistent across tasks, with PACES surpassing both unstructured baselines on seven of eight Qwen tasks and all eight DeepSeek tasks. Furthermore, PACES reduces physical checkpoint storage by 54.4% on Qwen and 57.3% on DeepSeek. Consistent with the moderate-sparsity scenario, coarse-grained expert compression methods achieve marginally higher compression ratios via coarse-grained expert elimination but suffer catastrophic accuracy degradation, greatly underperforming PACES under aggressive compression.

Table 3: Full-model forward latency. Speedup is integrated pattern-coded execution relative to explicit PyTorch reconstruction.

Model	Workload	Reconstruction ms	Integrated ms	Speedup
Qwen, $g = 2$	1×128	1029.35	45.05	22.85×
Qwen, $g = 2$	1×512	1433.92	88.81	16.15×
Qwen, $g = 4$	1×128	1012.03	30.93	32.72×
Qwen, $g = 4$	1×512	1077.09	60.61	17.77×
DeepSeek, $g = 2$	1×128	1172.32	82.65	14.18×
DeepSeek, $g = 2$	1×512	1491.76	186.58	8.00×
DeepSeek, $g = 4$	1×128	1076.71	40.23	26.76×
DeepSeek, $g = 4$	1×512	1174.74	84.29	13.94×

5.3 Integrated Execution and End-to-End Latency

Table 3 evaluates the execution efficiency of our integrated fused kernel against explicit checkpoint reconstruction. Across both model architectures and sequence lengths, the integrated kernel achieves dramatic speedups of $16.15\times$–$32.72\times$ on Qwen and $8.00\times$–$26.76\times$ on DeepSeek. While explicit reconstruction suffers from the severe memory bandwidth bottleneck of allocating and writing dense float matrices prior to multiplication, our integrated method decodes compressed weights on the fly during accumulation.

Beyond eliminating reconstruction overhead, PACES delivers practical end-to-end acceleration compared to uncompressed native MoE execution. On Qwen, $g = 2$ and $g = 4$ achieve $2.78\times$ and $4.05\times$ speedups at 1×128, sustaining $1.78\times$ and $2.61\times$ gains at 1×512. Across all evaluations, the larger group size ($g = 4$) yields consistently superior throughput over $g = 2$, consistent with the reduced memory traffic from loading fewer shared magnitude values m_j. DeepSeek benefits

Table 4: Grouping ablation using the eight-task macro average (%).

Grouping representation	Qwen $g = 2$	DeepSeek $g = 2$	Qwen $g = 4$	DeepSeek $g = 4$	Avg.
Sequential	63.50	62.75	61.09	59.24	61.64
Random	63.95	62.81	61.07	58.79	61.65
Expert weights	63.34	62.56	61.45	59.48	61.71
Expert W+feat.	64.14	63.07	61.20	59.54	61.99
Gradient proxy	63.87	62.67	60.63	**59.62**	61.70
Signed features	63.44	62.76	61.26	58.69	61.54
Absolute features	63.42	62.82	61.48	59.31	61.76
Router logits	64.41	**63.27**	61.16	59.31	62.04
Router weights (default)	**64.69**	62.72	**61.88**	59.30	**62.15**

less because its native dispatch is already contiguous and its top-6 routing creates $1.63\times$ more token–expert assignments. We discuss this architecture-dependent limitation in Section 6.

5.4 Expert grouping ablation

Table 4 compares nine grouping rules at both group sizes using explicit decoding and the same eight-task aggregate. Router weights give the strongest Qwen result at both sizes and the highest four-setting average. Router logits lead DeepSeek at $g = 2$ by 0.55 points, while the gradient proxy leads at $g = 4$ by 0.32 points; expert weights scaled by features are within 0.08 points of the latter. The default therefore provides the best aggregate result without claiming dominance on every model. It also avoids using sampled router logits as the expert representation. Appendix E defines every grouping variant.

6 Limitations

While our fused kernel avoids materializing dense expert weights at inference time, its practical end-to-end speedup depends on MoE dispatch logic, routing hyperparameters, token-expert mappings, kernel scheduling, and target hardware. The divergent speedups on Qwen and DeepSeek reflect distinct execution regimes for sparse MoEs. DeepSeek constitutes a tougher speed baseline: its runtime packs token-expert assignments into contiguous batches unlike Qwen's per-expert mask-gather-accumulate pipeline, and it carries $1.63\times$ more token-expert assignments under identical input sizes (top-6 over 26 layers vs. top-4 over 24). Such implementation and workload variations explain why compression alone does not guarantee acceleration. Appendix C provides supplementary kernel timings that partially pinpoint overhead, though individual factor contributions remain unquantified.

7 Conclusion

We presented PaCES (Pattern-Coded Element Sharing), a post-training representation for pretrained MoE language models that replaces groups of routed-expert coordinates with one shared BF16 magnitude and compact participation-and-sign codes. Router-guided grouping and a distribution-aware coordinate rule preserve every logical expert, while the checkpoint loader and integrated operator keep the representation compact through execution. Experiments on Qwen and DeepSeek demonstrate a controllable quality–storage trade-off, with the clearest advantage over unstructured pruning, expert deletion, and whole-expert merging at aggressive compression. PaCES thus provides a concrete compact representation and execution interface for lightweight MoE deployment.

AI USE STATEMENT

Generative AI tools were used to assist with experiment-code development, result organization, and manuscript drafting. The authors inspected the implementations, executed the experiments, traced reported values to local logs, checked the resulting claims against the measurements, and reviewed all AI-assisted text. The authors take responsibility for the final content, including all claims and artifacts produced with AI assistance.

ETHICS STATEMENT

This work studies compression and inference efficiency for existing language models and does not introduce new human-subject data. It inherits the risks and limitations of the evaluated pretrained models and benchmarks. Lowering deployment cost can broaden access to capable models, but can likewise lower the cost of harmful uses; deployment should retain the safeguards appropriate for the underlying model.

REPRODUCIBILITY STATEMENT

Section 5.1 and Appendix A specify the evaluated models, calibration data, metrics, baselines, and hardware. The method section defines the encoding and execution path, and the main result tables report every benchmark score entering the quality averages. Upon acceptance, we will open-source checkpoint conversion, reference decoding, GPU kernel, analysis, and evaluation scripts with fixed seeds and commands.

REFERENCES

Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. ACIQ: Analytical clipping for integer quantization of neural networks, 2019. URL https://openreview.net/forum?id=B1x33sC9KQ.

Y. Bisk, R. Zellers, R. Le Bras, J. Gao, and Y. Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239.

I-Chun Chen, Hsu-Shen Liu, Wei-Fang Sun, Chen-Hao Chao, Yen-Chang Hsu, and Chun-Yi Lee. Retraining-free merging of sparse MoE via hierarchical clustering. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 8594–8620. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/chen25aq.html.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

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

DeepSeek-AI. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized llms. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 10088–10115. Curran Associates, Inc., 2023. doi: 10.52202/075280-0441. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1feb87871436031bdc0f2beaa62a049b-Paper-Conference.pdf.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. GLaM: Efficient scaling of language models with mixture-of-experts. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5547–5569. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/du22c.html.

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

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10323–10337. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/frantar23a.html.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. Megablocks: Efficient sparse training with mixture-of-experts. In D. Song, M. Carbin, and T. Chen (eds.), *Proceedings of Machine Learning and Systems*, volume 5, pp. 288–304. Curan, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/file/5a54f79333768effe7e8927bcccffe40-Paper-mlsys2023.pdf.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 2024. URL https://zenodo.org/records/12608602.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/ae0eb3eed39d2bcef4622b2499a05fe6-Paper.pdf.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts, 2024. URL https://arxiv.org/abs/2401.04088.

Mike Lasby, Ivan Lazarevich, Nish Sinnadurai, Sean Lie, Yani Ioannou, and Vithursan Thangarasa. REAP the experts: Why pruning prevails for one-shot moe compression. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ukGxWd2aDG.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qrwe7XHTmYb.

Zongfang Liu, Guangyi Chen, Shengkun Tang, Yifan Shen, Huan Wang, and Xin Yuan. AIMER: Calibration-free task-agnostic moe expert pruning, 2026. URL https://arxiv.org/abs/2603.18492.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Qwen Team. Qwen1.5. https://qwenlm.github.io/blog/qwen1.5/, 2024.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 18332–18346. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/rajbhandari22a.html.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, August 2021. ISSN 0001-0782. doi: 10.1145/3474381. URL https://doi.org/10.1145/3474381.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017. URL https://arxiv.org/abs/1701.06538.

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. A simple and effective pruning approach for large language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 4942–4964, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/14c856c7a41297804de4c4890e846b25-Paper-Conference.pdf.

Philippe Tillet, H. T. Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, MAPL 2019, pp. 10–19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367196. doi: 10.1145/3315508.3329973. URL https://doi.org/10.1145/3315508.3329973.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

A Model and Evaluation Details

Qwen1.5-MoE-A2.7B has 24 sparse layers with 60 routed experts per layer and top-4 routing. DeepSeek-V2-Lite has 27 transformer layers; the first is dense and the remaining 26 have 64 routed experts, top-6 routing, and two shared experts. Both models use hidden size 2,048 and routed-expert intermediate size 1,408. The original models contain 14.32B and 15.71B parameters, respectively.

Controlled latency measurements run on a 48 GiB NVIDIA GeForce RTX 4090 with CUDA driver 570.211.01. Final end-to-end measurements use five warm-up passes and 20 timed passes at 1×128 and 1×512, with `use_cache=False`. These are full-sequence forward passes, not autoregressive decode measurements.

The original Qwen and DeepSeek weight shards contain 28,632,144,944 and 31,413,626,576 bytes, respectively. The corresponding compact $g = 2$ files contain 19,288,806,768 and 20,616,877,824 bytes; $g = 4$ contains 13,060,265,224 and 13,419,452,064 bytes. Reductions are computed from these unrounded totals. Nonpersistent buffers created during model loading are excluded because they are not part of the checkpoint.

For deleted baseline checkpoints, the BF16 payload estimate is $2P$ bytes, where P is the logged parameter count. SparseGPT and Wanda preserve the original tensor shapes and therefore have essentially zero storage reduction in the dense format used here. REAP and AIMER retain 8,085,768,192 Qwen parameters at 50% expert removal and 4,970,760,192 at 75%; the corresponding DeepSeek counts are 8,507,354,624 and 4,907,789,824. These give the estimates marked † in the main tables. They exclude file headers and should not be read as fresh measurements of the removed files.

B Compact Execution Consistency

Table 5: Eight-task accuracy consistency between explicit reconstruction and compact integrated execution for distance-weighted $g = 4$ checkpoints (%).

Model	Group size	Reconstruction	Integrated	Δ
Qwen	$g = 4$	61.9920	62.1504	+0.1584
DeepSeek	$g = 4$	59.3003	59.5431	+0.2428

The final kernel changes the eight-task macro average by $+0.1584$ points for Qwen and $+0.2428$ points for DeepSeek relative to explicit reconstruction. The largest task-level absolute difference is a 1.1839-point increase on DeepSeek WinoGrande. Thus the integrated path preserves average accuracy closely, although it does not meet a strict 0.5-point absolute per-task equivalence threshold. Both evaluations report zero fused MoE fallback calls.

For the $g = 4$ full-sequence benchmarks reported in the main text, output cosine similarity ranges from 0.999883 to 0.999931. The maximum absolute output error is 1.500 for DeepSeek at 1×512. These output differences are measured in logit units and should not be interpreted as accuracy percentages.

C Kernel-Stage Profile

An instrumented $g = 4$ run at 1×128 measures the combined gate/up kernel and the down kernel separately. Their device times are 10.81/5.35 ms for Qwen and 19.11/6.57 ms for DeepSeek. Both projection stages therefore cost more on DeepSeek under this implementation. These are profiled device times, not uninstrumented end-to-end latencies, and they do not separate pattern decoding from multiplication within each kernel.

D Additional Motivation Results

This appendix provides the definitions of the distributional diagnostics, sampling protocol, and additional results supporting the motivation in Section 3. These statistics are descriptive measurements

of expert projection weights and are not intended to establish that different experts implement identical functions.

D.1 Diagnostic definitions

For each expert projection, we sample $N = 4{,}096$ weight values $x_1, \ldots, x_N$. Let μ_x and σ_x denote the sample mean and standard deviation. We report the normalized center shift $|\mu_x|/\sigma_x$, the skewness

$$\mathrm{skew}(x) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_i - \mu_x}{\sigma_x}\right)^3,$$

and the excess kurtosis

$$\mathrm{kurt}(x) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_i - \mu_x}{\sigma_x}\right)^4 - 3.$$

The reported kurtosis is excess kurtosis, so a Gaussian distribution has expected value zero. We also compute the Pearson correlation between the sorted standardized samples and the corresponding standard-normal quantiles,

$$q_i = \Phi^{-1}\left(\frac{i - \frac{1}{2}}{N}\right), \qquad \mathrm{QQ}(x) = \mathrm{corr}\left(\mathrm{sort}\left(\frac{x - \mu_x}{\sigma_x}\right), (q_1, \ldots, q_N)\right),$$

where Φ^{-1} is the inverse standard-normal cumulative distribution function. Coverage within k standard deviations is defined as

$$C_k = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left(|x_i - \mu_x| \le k\sigma_x\right), \qquad k \in \{1, 2, 3\}.$$

For comparisons across experts in the same layer and projection, all sampled values are additionally normalized using the pooled mean and standard deviation of that layer and projection. This prevents separate per-expert normalization from hiding differences in scale.

D.2 Per-expert distribution diagnostics

Each expert projection contributes 4,096 sampled coordinates using seed 42. The reported values are aggregated across the sampled gate, up, and down projections. The Qwen results cover 4,320 expert tensors, and the DeepSeek results cover 4,992 expert tensors. Table 6 reports the median distribution statistics across these tensors. Coverage is reported within one, two, and three tensor standard deviations.

Table 6: Distribution diagnostics for routed-expert weights. Coverage reports the fraction within one, two, and three tensor standard deviations.

Model	$\lvert\mu\rvert/\sigma$	|skew|	|kurt|	Q–Q corr.	Coverage (%)
Qwen	0.000538	0.0273	0.0663	0.999768	68.46/95.41/99.70
DeepSeek	0.000535	0.0287	0.0817	0.999745	68.58/95.40/99.68

The results show that the center shifts are negligible, while skewness and excess kurtosis remain close to the values expected for a Gaussian distribution. The normal Q–Q correlations are also close to one. The empirical coverage values are close to the theoretical Gaussian values of 68.27%, 95.45%, and 99.73% for one, two, and three standard deviations, respectively. These measurements support the use of a common approximately Gaussian reference when comparing corresponding parameters across experts.

D.3 Within-layer scale consistency

Similar distribution shapes are insufficient for element-wise sharing if experts have substantially different scales. To measure scale agreement, let $s_{e,\ell,p}$ denote the standard deviation of expert e in

Table 7: Median within-layer coefficient of variation (CV) of expert standard deviations. Values are reported as fractions; for example, 0.0222 corresponds to 2.22%.

Model	Projection	Within-layer CV
Qwen	gate	0.0222
Qwen	up	0.0225
Qwen	down	0.0172
DeepSeek	gate	0.0363
DeepSeek	up	0.0365
DeepSeek	down	0.0333

layer ℓ and projection type p. We compute the within-layer coefficient of variation as

$$\mathrm{CV}_{\ell,p} = \frac{\mathrm{std}_e\left(s_{e,\ell,p}\right)}{\mathrm{mean}_e\left(s_{e,\ell,p}\right)}.$$

The reported values are aggregated across sparse layers separately for each projection type. For the distribution comparison, we use the same pooled layer-and-projection normalization for all experts rather than normalizing each expert separately.

Figure 2 visualizes the distribution overlap for 12 representative experts together with the pooled layer distribution. Each curve uses the same pooled mean and standard deviation. The overlap therefore reflects both distribution shape and scale agreement.

Figure 3 reports scale variation across all sparse layers. For each layer and projection, the line shows the mean expert standard deviation and the band spans the minimum and maximum expert values. Scales vary across layers, but experts within each layer remain closely matched. The median within-layer coefficient of variation is 1.7–2.3% for Qwen and 3.3–3.7% for DeepSeek.

D.4 Cross-expert distribution agreement

To evaluate distributional agreement beyond the coefficient of variation, we construct a balanced leave-one-expert-out reference distribution for each layer and projection. For each expert, the reference is formed from samples drawn from the other experts in the same layer. A balanced reference is sampled four times to avoid domination by any individual expert.

We compare each expert with its reference using the Kolmogorov–Smirnov distance

$$D_{\mathrm{KS}}(F, G) = \sup_x |F(x) - G(x)|,$$

the first Wasserstein distance W_1, and the normal Q–Q correlation. The Wasserstein distance is reported in units of the pooled layer standard deviation. P95 denotes the 95th percentile of the KS or Wasserstein distances, while P05 denotes the 5th percentile of the Q–Q correlations.

The resulting median KS distances are 0.0192 for Qwen and 0.0205 for DeepSeek. Median Q–Q correlations are approximately 0.9995 for both models. These results indicate that experts within the same layer occupy closely aligned distributions, while not implying that their individual weight values or functional roles are interchangeable.

Table 8: Agreement between each expert and a balanced leave-one-expert-out layer-merged distribution. Wasserstein distance is measured in pooled layer-standard-deviation units.

Model	Median KS	P95 KS	Median W_1	P95 W_1	Median Q–Q	P05 Q–Q
Qwen	0.0192	0.0281	0.0300	0.0509	0.99951	0.99901
DeepSeek	0.0205	0.0308	0.0339	0.0641	0.99947	0.99888

For an additional control, we compare each expert with the next sparse layer using the same pooled normalization. The median KS distances are 0.0196 for Qwen and 0.0208 for DeepSeek, and same-layer references are better in only 52–56% of comparisons. This control suggests that the close agreement measured above is primarily a within-layer property rather than a consequence of using a generic model-wide distribution.

E GROUPING-ABLATION IMPLEMENTATIONS

All non-sequential variants use the same balanced assignment scaffold unless noted otherwise. Route counts select the E/g anchor experts. Each expert descriptor is ℓ_2 normalized, all remaining expert–anchor cosine scores are sorted, and assignments are accepted greedily subject to group capacity. The variants differ in the descriptor used for assignment.

Sequential. Experts are kept in checkpoint order and consecutive blocks of g experts form groups. This variant does not use the balanced assignment scaffold.

Random. Each expert receives a seeded random Gaussian descriptor. Route-count anchors and the common capacity-constrained assignment are retained, so the control isolates descriptor information rather than changing group balance.

Expert weights. A seeded set of coordinates is sampled from the stacked gate, up, and down expert weights; the sampled signed values form each expert descriptor.

Expert weights + features. The sampled weight sketch is scaled coordinate-wise by the mean absolute input feature of tokens routed to that expert. Input dimensions are matched to sampled weight columns.

Gradient proxy. For each expert and input dimension, the square root of the routed-feature second moment scales the sampled weight sketch. Absolute scaled values form a zero-training importance proxy; no backward gradients are computed.

Signed features. The descriptor is the mean hidden-state vector over tokens routed to each expert. Experts with no selected token use the layer-wide mean as fallback.

Absolute features. The descriptor is instead the mean absolute hidden-state vector over routed tokens, removing sign cancellation.

Router weights. The descriptor is the corresponding row of the pretrained router weight matrix. This static representation is the PaCES default.

Router logits. The descriptor contains an expert's router logits across captured calibration tokens. It therefore uses dynamic routing responses rather than static router rows.