FACTOREP: ACCELERATING EXPERT-PARALLEL MOE INFERENCE THROUGH END-TO-END FACTOR-SPACE EXECUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-Experts (MoE) models rely on expert parallelism (EP) to scale across GPUs, but communication can account for up to 47% of end-to-end execution time. Existing EP systems primarily optimize *how* activations are communicated, while redundancy in pretrained experts offers an opportunity to reduce *what* must be communicated and computed. We present FACTOREP, a system that exploits this redundancy by co-designing compact representations and EP execution. Group-wise Tucker factorization with KFAC-weighted calibration produces group-shared input factors, expert-specific compact cores, and a layer-shared output factor. This asymmetric sharing enables input projection before dispatch and output reconstruction after combine, reducing both communication and expert computation. To turn these reductions into practical speedups, a three-stage factor-space pipeline amortizes shared projection costs and overlaps communication with compact expert execution. Route-compiled execution uses a packed token layout shared by communication and compact expert kernels to reduce capacity padding and per-expert launch overhead. On Qwen3-30B-A3B, FactorEP achieves higher average downstream accuracy than all evaluated compression baselines across all tested parameter budgets. Its deployment configuration reduces dispatch and combine feature widths by 20% while maintaining average downstream accuracy within 1.93 percentage points of the original model. The same configuration achieves end-to-end speedups over COMET of $1.31\times$ on eight H100 GPUs with NVSwitch and $1.26\times$ on eight RTX 6000 Ada GPUs over PCIe. These results, together with quality-latency sweeps on Qwen3-30B-A3B and OLMoE-1B-7B, demonstrate FactorEP's applicability across MoE models and hardware platforms.

1 INTRODUCTION

Large language models (LLMs) achieve strong performance by scaling model capacity and training compute (Brown et al., 2020; Kaplan et al., 2020; Tang et al., 2026). Mixture-of-Experts (MoE) models scale capacity more efficiently by activating only a small subset of experts for each token (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022), and have been adopted by several recent LLMs (Jiang et al., 2024a; DeepSeek-AI, 2024; Yang et al., 2025). At scale, these models rely on expert parallelism (EP) to deploy them efficiently across multiple GPUs.

An EP layer follows a dispatch-compute-combine pattern. Tokens are sent to the GPUs hosting their selected experts, processed remotely, and returned for routing-weighted aggregation. Under input-dependent routing, communication and expert loads can also be highly imbalanced across ranks. As a result, communication accounts for 30–45% of end-to-end execution time in representative MoE systems (Li et al., 2023) and can reach 47% (Zhang et al., 2025).

Existing EP systems mainly optimize ***how*** activations are communicated through faster collectives, communication-computation overlap, and load balancing (Hwang et al., 2023; Li et al., 2023; Jiang et al., 2024b; Zhang et al., 2025). Meanwhile, recent MoE compression studies have revealed substantial redundancy within and across experts (Lu et al., 2024; Yang et al., 2024; Li et al., 2024; 2026; Chen et al., 2025b), raising a complementary question: ***what*** representation is essentially sufficient for dispatch, expert execution, and combine?

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.

Existing compression methods do not directly answer this question. Their factor spaces mainly target weight reconstruction rather than EP execution, and smaller factorized GEMMs can suffer from padding, launch overhead, and poor hardware efficiency (Shao et al., 2026; Hu et al., 2026). Thus, EP acceleration requires a compact representation designed for the distributed execution path, including both communication and compact expert execution.

To this end, we propose FACTOREP, which co-designs MoE factorization and EP execution using group-wise Tucker factorization (Tucker, 1966; De Lathauwer et al., 2000) with KFAC-weighted calibration. Group-shared input factors align with communication chunks, enabling projection reuse before dispatch and communication-computation overlap. Compact expert cores execute in factor space, while a layer-shared output factor keeps results compact through routing-weighted combine. This design reduces dispatch traffic, combine traffic, and expert computation to deliver end-to-end speedups.

FactorEP further co-designs execution around this compact representation. A factor-space three-stage pipeline amortizes shared projection costs and overlaps projection and dispatch, compact expert execution, and combine across independently progressing destination chunks. Route compilation organizes routed rows by destination, factor group, and expert, linking shared input projections, compact communication, and grouped expert kernels. The resulting packed layout follows actual routing while preserving factor-sharing and expert boundaries.

FactorEP achieves higher average downstream accuracy than all evaluated compression baselines across all tested parameter budgets. For efficient deployment, FactorEP uses fewer factor groups and retains 59.93% accuracy with 20% narrower dispatch and combine payloads. This configuration achieves $1.23\times$ end-to-end speedup on H100 NVSwitch and $1.26\times$ on RTX 6000 Ada PCIe over the strongest platform-specific baselines. These results show that jointly designing factorization and execution translates expert redundancy into speedups while maintaining competitive model quality.

Our contributions are:

- We introduce an end-to-end factor-space representation that uses group-wise Tucker factorization with KFAC-weighted calibration to jointly reduce EP communication and expert computation by keeping routed activations compact through dispatch and combine.
- We design a factor-space three-stage pipeline that amortizes shared projection costs and overlaps communication with compact expert execution across independently progressing chunks with separately tracked communication and computation dependencies.
- We develop route-compiled execution that organizes routed tokens into a packed factor-space layout shared by communication and compact expert kernels, reducing capacity padding and per-expert launch overhead under input-dependent routing.

2 BACKGROUND AND MOTIVATION

2.1 THE EXPERT-PARALLEL EXECUTION PATH

Figure 1 illustrates the conventional execution path of an expert-parallel MoE layer. After routing, token copies are dispatched to the ranks hosting their selected experts, processed by the local expert segments, and returned during combine for routing-weighted aggregation. For hidden dimension d, conventional EP communicates d values per routed token during both dispatch and combine across EP ranks.

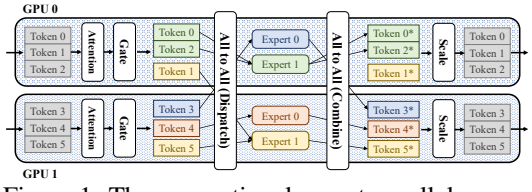
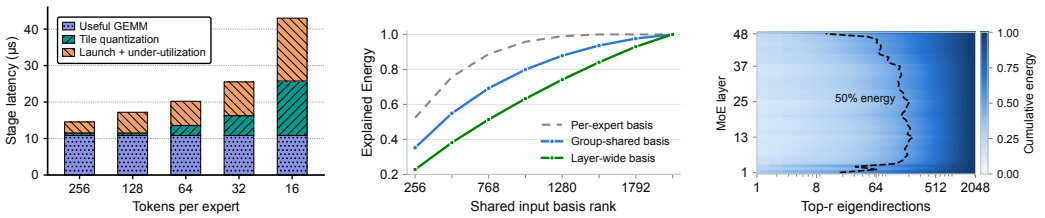


Figure 1: The conventional expert-parallel execution path in one MoE layer.

Existing systems mainly optimize how this execution path is carried out. Tutel (Hwang et al., 2023) and Lina (Li et al., 2023) improve collective communication and load balance, while COMET (Zhang et al., 2025), DeepEP (Zhao et al., 2025), and FlashMoE (Aimuyo et al., 2025) use increasingly fine-grained or fused communication-computation execution. These techniques substantially improve EP efficiency, but the communicated representation remains the original hid-



(a) Fragmentation (b) Input subspace sharing (c) Activation anisotropy

Figure 2: **Key observations behind FactorEP.** (a) Smaller expert GEMMs increase execution overhead. (b) Experts share low-rank input subspaces at factor-group granularity. (c) MoE activations are highly anisotropic, with energy concentrated in a few directions.

den state. FactorEP instead changes what is communicated by keeping activations in compact factor spaces through dispatch, execution, and combine.

2.2 MoE Compression and Factor-Space Communication

Low-rank compression has shown that large language models contain substantial redundancy. Activation-aware methods such as ASVD (Yuan et al., 2023) and SVD-LLM (Wang et al., 2025) improve post-training low-rank approximation, while MoE-specific methods further exploit redundancy across experts. MoE-I^2 (Yang et al., 2024) compresses experts independently, Sub-MoE (Li et al., 2026) jointly compresses similar experts, MoBE (Chen et al., 2025b) introduces shared expert bases, and TD-MoE (Xu et al., 2026) applies Tucker decomposition across stacked experts. MC-SMoE (Li et al., 2024) uses routing statistics to merge experts before low-rank-plus-sparse compression. D^2-MoE (Gu et al., 2025) separates a shared base from expert-specific deltas and applies SVD to obtain low-rank approximations of these residual weights. These methods mainly target model size, FLOPs, or reconstruction quality, rather than EP communication. When factorization stays inside each expert, dispatch and combine can still carry the original hidden state.

LatentMoE (Elango et al., 2026) and BigMac (Jin et al., 2025b) take a different approach by training architectures whose communication already occurs in a learned latent space. FactorEP targets existing pretrained MoEs and derives the compact communication spaces from the checkpoint. This factor-space compression is also complementary to reduced-precision EP communication, which reduces the bits used per activation (Shi et al., 2024; Jin et al., 2025a; Zhao et al., 2025).

2.3 Design Insights for Factor-Space Compression

Execution fragmentation. Low-rank factorization reduces FLOPs but often replaces large GEMMs with smaller operations whose efficiency is limited by kernel shape, launch overhead, and GPU utilization (Hu et al., 2026; Shao et al., 2026). This problem is amplified in MoEs, where sparse routing already creates small and irregular expert GEMMs. MegaBlocks uses block-sparse kernels to handle irregular routing during training (Gale et al., 2023), while SonicMoE targets memory IO costs and padding waste in grouped GEMM (Guo et al., 2025). Figure 2(a) shows that, for the same total number of routed tokens, greater fragmentation leads to higher latency. Compression must therefore avoid excessive execution fragmentation.

Shared input subspaces. Per-expert input bases preserve specialization but provide little cross-expert sharing. At a fixed rank, a layer-wide input basis can lose more expert-specific information. Figure 2(b) shows that factor-group input bases provide a useful middle ground, retaining substantially more explained energy than a layer-wide basis and approaching independent expert bases. This motivates sharing input projection spaces among similar experts while keeping expert-specific compact cores for the individual transformations performed by each expert.

Activation anisotropy. Figure 2(c) shows that a few eigendirections capture much of the activation energy. Uniform Frobenius reconstruction therefore treats functionally important and unimportant directions equally. ASVD and SVD-LLM similarly use activation statistics to guide which weight directions are preserved (Yuan et al., 2023; Wang et al., 2025). Our activation-weighted decomposition improves quality (Appendix A.2.1), motivating calibration with activation statistics.

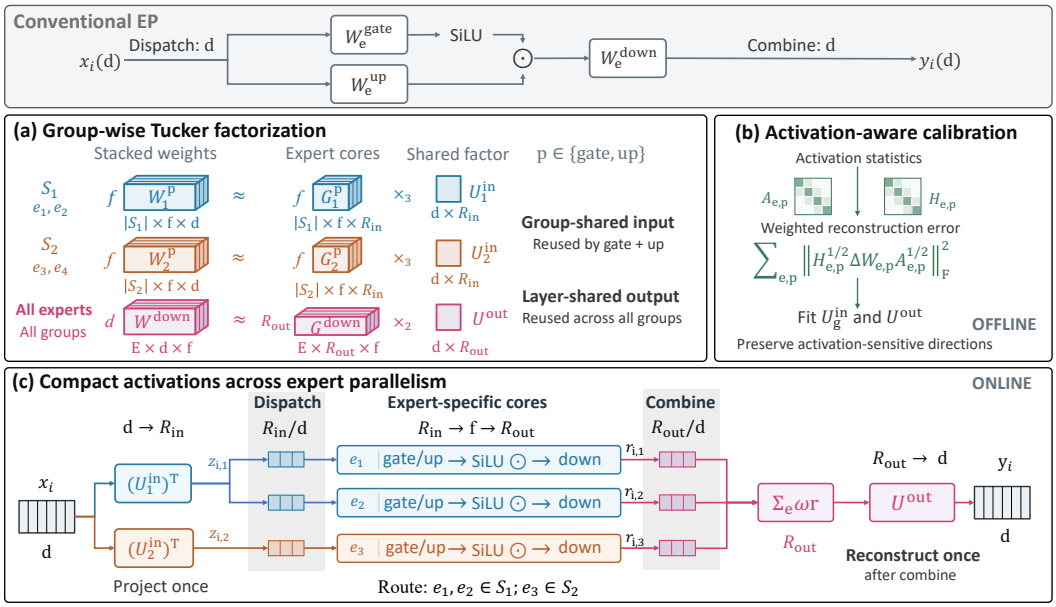


Figure 3: **Path-aligned factor-space compression.** (a) Group-wise Tucker factorization shares input factors within factor groups and one output factor across the layer. (b) Activation-aware calibration preserves directions important to routed inputs. (c) At inference time, activations remain compact through dispatch, expert execution, and combine, and are reconstructed only once after combine to recover the original hidden-state dimension.

3 FACTOREP DESIGN

Design overview. FactorEP first constructs shared factor spaces that jointly reduce EP communication and expert computation (C1; Sec. 3.1). It then turns these reductions into latency improvements through a three-stage factor-space pipeline (C2; Sec. 3.2) and route-compiled GPU execution of dynamically routed expert workloads (C3; Sec. 3.3).

We distinguish three units. A *factor group* $\mathcal{S}_g$ shares an input factor $\mathbf{U}_g^{\text{in}}$ among its experts. A *destination chunk* $\chi = (r, g, c)$ contains routed rows for destination rank r and factor group g, with c an optional subdivision, and advances through the pipeline. A *factor-space tile* is a logical block of contiguous routed rows assigned to one expert within a chunk, consumed by the grouped expert kernels directly from the packed factor-space layout.

3.1 PATH-ALIGNED FACTOR-SPACE COMPRESSION

As shown in Fig. 3, FactorEP constructs shared factor spaces that can be used directly throughout expert-parallel execution. Tokens enter a compact space before dispatch, compact expert cores operate directly in that space, and the hidden representation is reconstructed only after routing-weighted combine. The same factorization therefore reduces both EP communication and expert computation throughout the MoE layer's distributed execution path.

Factor-group construction. Factor groups trade projection reuse against approximation quality. We group experts with similar gate/up subspaces using the energy retained by a shared basis (Fig. 2(b)). Appendix A.1 gives the grouping procedure.

Path-aligned Tucker factorization. For the gate and up projections, FactorEP stacks the experts in factor group $\mathcal{S}_g$ into $\mathcal{W}_g^p \in \mathbb{R}^{|\mathcal{S}_g| \times f \times d}$ and applies a partial Tucker decomposition along the hidden-state mode, corresponding to the expert input dimension:

$$\mathcal{W}_g^p \approx \mathcal{G}_g^p \times_3 \mathbf{U}_g^{\text{in}}, \qquad p \in \{\text{gate}, \text{up}\}. \tag{1}$$

Here, $\mathbf{U}_g^{\text{in}} \in \mathbb{R}^{d \times R_{\text{in}}}$ is shared by the gate and up projections in the group, while slices of $\mathcal{G}_g^p$ are expert-specific compact cores for each expert's gate and up projections.

To accumulate all routed outputs before a single reconstruction, we use a layer-shared output factor. We stack the down projections and factorize their hidden-state mode:

$$\mathcal{W}^{\text{down}} \approx \mathcal{G}^{\text{down}} \times_2 \mathbf{U}^{\text{out}}, \qquad \mathbf{U}^{\text{out}} \in \mathbb{R}^{d \times R_{\text{out}}}. \tag{2}$$

Thus, input factors are shared within similar expert groups, while the output factor is shared across the layer. We leave the intermediate dimension f unchanged because compressing it does not reduce EP traffic and introduces additional small expert operations. The deployment path below uses expert-specific compact cores. The asymmetric sharing is deliberate: input factors capture group-specific subspaces, whereas the output factor places all expert partials in common coordinates. Separate output bases would require group-wise reconstruction or an additional basis conversion before cross-group accumulation of outputs from the selected experts.

KFAC-weighted calibration. The Tucker structure determines which dimensions are compressed, but the shared factors must preserve directions that matter under routed activations. Since MoE activations are highly anisotropic (Fig. 2(c)), FactorEP uses KFAC-style Kronecker weighting (Martens & Grosse, 2015). For projection p of expert e, we minimize

$$\mathcal{L}_{e,p} = \left\| \mathbf{H}_{e,p}^{1/2} \left(\mathbf{W}_e^p - \widehat{\mathbf{W}}_e^p \right) \mathbf{A}_{e,p}^{1/2} \right\|_F^2, \tag{3}$$

where $\mathbf{A}_{e,p}$ and $\mathbf{H}_{e,p}$ weight input and output directions. Appendix A.2 states the activation-statistic proxy and derives its shared factors; the quality comparison is in Appendix A.2.1.

Factor-space EP path. For token $\mathbf{x}_i$ routed to expert e in factor group $g(e)$, FactorEP computes

$$\mathbf{z}_{i,g(e)} = (\mathbf{U}_{g(e)}^{\text{in}})^\top \mathbf{x}_i, \quad \mathbf{r}_{i,e} = \text{FFN}_e^{\text{core}}(\mathbf{z}_{i,g(e)}), \quad \mathbf{y}_i = \mathbf{U}^{\text{out}} \sum_{e \in \mathcal{R}(i)} \omega_{i,e} \mathbf{r}_{i,e}. \tag{4}$$

Here $\mathcal{R}(i)$ is the selected expert set and $\omega_{i,e}$ is expert e's routing weight for token i. The compact FFN retains the original SiLU and gating operations at intermediate width f. The R_{in}-dimensional code is dispatched instead of the original d-dimensional hidden state, and experts return R_{out}-dimensional partials. Because the output factor is shared across the layer, routing-weighted accumulation remains in factor space and reconstruction is performed only once after combine.

Ignoring metadata, the leading communication and expert-core reductions are

$$\frac{V_{\text{disp}}}{V_{\text{disp}}^{\text{dense}}} = \frac{R_{\text{in}}}{d}, \quad \frac{V_{\text{comb}}}{V_{\text{comb}}^{\text{dense}}} = \frac{R_{\text{out}}}{d}, \quad \frac{F_{\text{core}}}{F_{\text{core}}^{\text{dense}}} = \frac{2R_{\text{in}} + R_{\text{out}}}{3d}. \tag{5}$$

The shared input projection and output reconstruction add token-level work, but their costs are amortized across routed experts. The complete parameter and FLOP analysis, including shared projections and expert-core computation, is provided in Appendix A.3.

3.2 Factor-Space Three-Stage Pipeline

The factorization in Sec. 3.1 reduces communication and expert computation, but the remaining projection and communication costs can still serialize execution. FactorEP exploits the shared Tucker factors to reorganize the EP path into three stages that can overlap across independently progressing routed chunks, with dependencies tracked separately for each chunk.

The pipeline advances destination chunks $\chi = (r, g, c)$ as defined in the design overview. Sec. 3.3 describes how grouped expert kernels consume the packed rows of each ready chunk.

Moving projections across routing. Because experts in the same factor group share an input factor, FactorEP projects a token once for each selected factor group before dispatch, rather than once for each routed expert. The resulting compact rows are then packed into destination chunks and transmitted to the ranks hosting the selected experts.

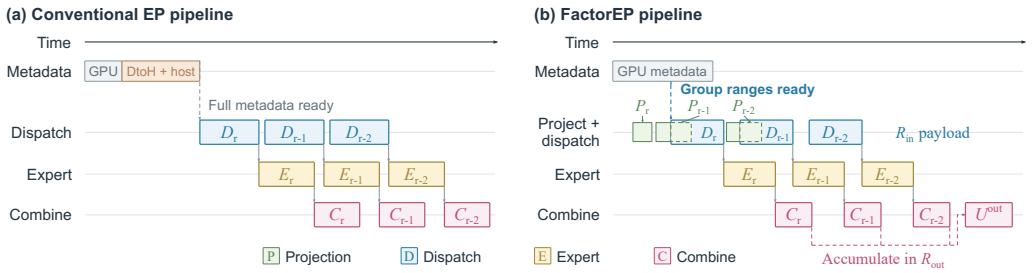


Figure 4: **Factor-space pipelining and route-compiled execution.** FactorEP moves input projection before dispatch and delays output reconstruction until after factor-space combine. Destination chunks progress independently through the three-stage pipeline, while grouped expert kernels consume the packed factor-space rows of each ready chunk.

The layer-shared output factor similarly allows reconstruction to be delayed until after routing-weighted accumulation of all selected experts' compact outputs:

$$\sum_{e\in\mathcal{R}(i)} \omega_{i,e}\mathbf{U}^{\text{out}}\mathbf{r}_{i,e} = \mathbf{U}^{\text{out}} \sum_{e\in\mathcal{R}(i)} \omega_{i,e}\mathbf{r}_{i,e}. \tag{6}$$

FactorEP therefore performs both dispatch and combine in factor space, while the shared projections are evaluated outside per-expert execution, before dispatch and after combine.

Three-stage pipeline. Each destination chunk progresses through three stages:

1. **Projection and dispatch.** The source applies the factor-group input projection and dispatches the compact rows to the destination rank.
2. **Compact expert execution.** Once a chunk arrives, its routed experts execute directly on the factor-space representation of their assigned input tokens.
3. **Combine and reconstruction.** Compact expert outputs are returned and accumulated with routing weights, followed by one reconstruction after the required contributions from the token's selected experts have all been accumulated.

Different destination chunks can occupy different stages concurrently. While one chunk executes its experts, a later chunk can be projected and dispatched and an earlier chunk can return its results. The projection and communication introduced by factorization are therefore overlapped with useful expert computation rather than added serially to the critical path.

Dependency-driven progression. After route-count exchange, FactorEP does not require all destination chunks to complete one payload stage before the next begins. A chunk starts expert execution as soon as its compact rows arrive and returns its results immediately after its expert work completes. Consequently, a heavily routed destination delays dependent chunks without adding an all-chunk wait between payload stages for otherwise ready chunks.

We use bounded buffering to keep several chunks in flight without allowing projection and communication to run arbitrarily far ahead of expert execution. The next subsection describes how each ready destination chunk is compiled into exact GPU work. Implementation details of destination-chunk construction, exact-size communication, and bounded buffering are provided in Appendix B.

3.3 Route-Compiled Factor-Space Execution

FactorEP compiles router output into a packed factor-space layout shared by communication and compact expert kernels. Factor-group identifiers select shared input projections, destination ranges define compact communication, and expert subranges feed the grouped kernels without rebuilding capacity-padded expert buffers between communication and expert execution.

Packed routing layout. Routed assignments are ordered by destination rank, factor group, and expert ID. This ordering produces the destination chunks used by Sec. 3.2, with contiguous expert

segments inside each chunk. Projection and packing use factor-group and source-token identifiers, communication consumes chunk ranges, and grouped expert kernels operate on expert subranges of the same packed layout with no intervening conversion to capacity-padded expert buffers.

GPU-resident route compilation. FactorEP constructs this layout directly from router output on the GPU. It maps expert IDs to destination and factor-group identifiers, builds chunk and expert histograms, and uses prefix scans to obtain exact row ranges. These ranges provide the descriptors needed by projection, communication, expert execution, and combine. The return mapping preserves each row's token-expert identity, so reordering rows does not change the selected experts or routing-weighted accumulation for a given router output.

Metadata follows its consumers: projection and packing need source-token indices and chunk ranges, while combine later consumes routing weights and return mappings. After the required route-count exchange, ready chunks progress without an additional payload-stage barrier.

Tiled execution of packed factor-space rows. Within chunk χ, the $M_{\chi,e}$ rows routed to expert e are partitioned into factor-space tiles. These logical row blocks specify valid work for the grouped kernels; physical GEMM tiling remains a kernel-level choice. The number of row blocks is

$$N_{\chi,e} = \left\lceil \frac{M_{\chi,e}}{B_M(R_{\text{in}}, R_{\text{out}})} \right\rceil, \tag{7}$$

where B_M is selected offline for each factor-rank configuration and reused at inference. Routing changes the number of row blocks, not this configuration. Empty experts emit no blocks; heavily routed experts expose more independent row blocks for grouped execution.

Ragged factor-space kernels. FactorEP processes these row blocks directly from the packed factor-space buffers with two grouped kernels. The first jointly evaluates the compact gate and up projections and fuses SiLU and the gating product. The second evaluates the compact down projection and produces R_{out}-dimensional partials for combine.

The two kernels launch once per ready chunk rather than once per expert. They consume exact routed row ranges and mask invalid rows in final partial tiles, avoiding capacity-padded buffers, empty-expert work, and per-expert launches. Together with route compilation, this preserves factorization's computation savings under irregular routing. Appendix C expands the compact expert computations used by these grouped kernels for the gate, up, and down projections.

4 Evaluation

4.1 Experimental Setup

We evaluate FactorEP on Qwen3-30B-A3B. Model quality is measured using perplexity on WikiText-2, PTB, and C4 and zero-shot accuracy on seven downstream tasks. System efficiency is evaluated on eight H100 80 GB GPUs connected by NVSwitch and eight RTX 6000 Ada 48 GB GPUs connected by PCIe. Both platforms use the same inference settings: BF16, EP degree eight, per-rank batch size two, and sequence length 2,048 tokens per sequence.

The matched-budget configurations in Table 1 use more factor groups. Its last row reports the quality of the deployment configuration timed in Table 2(a), which uses fewer factor groups. Figure 5 reports the joint quality-latency trade-off. Configuration and baseline details are given in Appendix D.1.

4.2 Quality at Matched Compression Ratios

The matched-budget comparisons in Table 1 show that FactorEP consistently preserves the best overall quality. At approximately 20% reduction, FactorEP reaches 60.11 average downstream accuracy, only 1.75 points below the original model and 2.42 points above the strongest compressed baseline. The advantage widens to 10.80 and 13.72 points at the 40% and 60% budgets, respectively. FactorEP also achieves the lowest perplexity among the compressed methods at all three budgets.

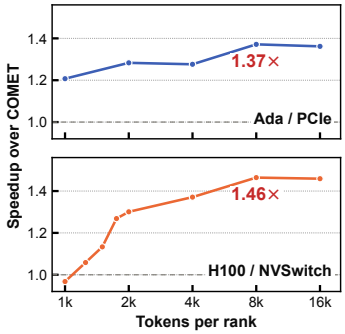
Table 1: Qwen3-30B-A3B quality at matched expert-parameter budgets, including shared factors. Lower perplexity and higher accuracy are better. TD-MoE and MoE-SVD use official code, extending compression from selected layers to all MoE layers.

Budget	Method	Wiki2↓	PTB↓	C4↓	Openb.	ARC-e	WinoG.	HellaS.	ARC-c	PIQA	MathQA	Avg.↑
0%	Original (2025)	8.70	15.38	14.33	34.00	79.12	70.01	59.52	52.90	79.11	58.32	61.86
∼20%	TD-MoE (2026)	448	744	443	14.00	29.42	50.36	28.15	19.88	55.93	21.14	31.27
	MoE-SVD (2025)	163	213	259	15.00	47.10	51.30	28.46	24.40	59.68	22.58	35.50
	D^2-MoE (2025)	11.77	21.48	21.90	27.80	74.24	68.98	46.43	44.88	72.80	43.85	54.14
	HC-SMoE ($c = 103$) (2025a)	11.69	18.55	17.78	30.20	74.28	**69.85**	52.40	46.67	76.17	54.24	57.69
	FactorEP ($R = 1371$)	**9.61**	**16.79**	**16.69**	**32.80**	**78.75**	**69.85**	**54.12**	**50.17**	**76.66**	**58.43**	**60.11**
∼40%	D^2-MoE (2025)	16.14	32.00	36.09	24.00	60.27	64.01	37.28	34.39	66.21	33.57	45.68
	HC-SMoE ($c = 79$) (2025a)	22.17	40.82	36.28	20.60	61.57	59.75	37.46	34.90	63.93	50.25	46.92
	FactorEP ($R = 1028$)	**10.79**	**19.16**	**19.45**	**28.80**	**77.78**	**69.69**	**49.92**	**48.55**	**74.59**	**54.74**	**57.72**
∼60%	D^2-MoE (2025)	25.38	57.19	67.90	17.20	45.71	56.43	31.02	23.21	59.68	24.32	36.80
	HC-SMoE ($c = 47$) (2025a)	85.40	191.99	163.51	14.20	39.27	49.64	28.43	21.08	56.53	33.17	34.62
	FactorEP ($R = 685$)	**14.62**	**28.67**	**30.58**	**26.80**	**69.36**	**64.17**	**40.56**	**37.80**	**68.50**	**46.47**	**50.52**
FactorEP (deployment)		9.64	16.76	15.42	33.40	77.23	69.46	56.97	50.09	79.05	53.33	**59.93**

Table 2: Qwen3-30B-A3B: (a) latency (ms); (b) speedup over COMET across tokens per rank. Both use FactorEP (deployment). A dash denotes an unsupported platform.

	H100 / NVSwitch		RTX 6000 Ada / PCIe	
Method	**End-to-end**	**MoE layer**	**End-to-end**	**MoE layer**
Plain EP	413.68	5.98	–	–
DeepEP (2025) + TE	243.85	2.73	–	–
COMET (2025) / Flux	258.51	2.86	710.1	12.47
FlashMoE (2025)	318.67	4.11	9,514.6	180.78
Megatron (2019) + TE	323.72	3.99	11,878.7	210.94
FasterMoE (2022)	494.63	6.95	–	–
FastMoE (2021)	–	–	16,644.3	234.44
FactorEP	**197.8**	**1.94**	**561.9**	**9.701**

(a)



(b)

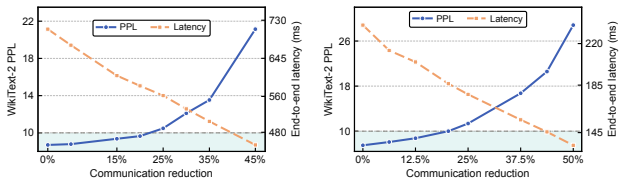
4.3 Deployment Quality and Efficiency

The deployment configuration retains 59.93% average downstream accuracy (Table 1, last row), 1.93 points below the original model. Relative to the strongest baseline on each platform, FactorEP achieves 1.23× end-to-end and 1.41× MoE-layer speedups on H100/NVSwitch, and 1.26× and 1.29×, respectively, on RTX 6000 Ada/PCIe.

Table 2(b) varies tokens per rank using the same FactorEP (deployment) configuration as Table 1. FactorEP achieves greater speedups at larger token counts, particularly on H100. Fast NVSwitch makes projection overhead more prominent at small token counts; larger workloads can better amortize this overhead and improve pipeline overlap. Speedups reach 1.46× on H100 and 1.37× on Ada at 8k tokens and remain close to these levels at 16k.

4.4 Quality-Latency Trade-off Across Models

Figure 5 shows that Qwen3 maintains low perplexity through roughly 25% communication reduction, whereas OLMoE reaches its quality knee earlier, at about 20%. Across both models, FactorEP delivers substantial end-to-end speedups with only modest perplexity increases, while the preferred factor rank remains model-dependent. We therefore use the more conservative 20% setting for the end-to-end experiments on the two GPU platforms.



(a) Qwen3-30B-A3B (b) OLMoE-1B-7B

Figure 5: **Quality-latency trade-off.** FactorEP accelerates Qwen3 and OLMoE with modest perplexity increases.

Table 3: Leave-one-component-out latency penalty (MoE layer / end-to-end).

Removed component	H100/NVSwitch	Ada/PCIe
Factor-space compression	16.80% / 4.53%	**32.98% / 26.69%**
Three-stage pipeline	13.39% / 5.57%	10.27% / 6.90%
Route-compiled execution	**20.35% / 15.36%**	4.11% / 5.04%

4.5 Component Ablation and Mechanism Analysis

The full system reduces dispatch and combine payloads by 28.1% and 25%, respectively. We remove each component with the workload and routing seed fixed. Table 3 reports latency penalties relative to paired full-system runs; Appendix D.3 gives the corresponding raw latencies.

Factor-space compression reduces work. C1 has the largest impact on PCIe, whose lower interconnect bandwidth makes latency more sensitive to communication volume. In the profiling comparison, factor-space compression reduces the input payload by 28.1%, lowering aggregate peer-copy time by 29.1% on Ada and 31.2% on H100. On H100, SM Active is slightly lower with compression despite the lower latency, consistent with a reduction in communication and expert work within the profiled MoE-layer execution (Appendix D.4.1).

The three-stage pipeline reduces waiting. C2 provides consistent gains on both platforms. On Ada, 98.2% of the input P2P service interval overlaps gate/up execution. On H100, the pipeline shortens the output critical path from 717.3 to 481.5 μs. Thus, C2 improves latency by hiding communication and reducing serialization (Appendix D.4.2).

Route-compiled execution reduces routing overhead. In the profiled H100 configurations, route preparation decreases from seven GPU kernels to four, shortening its interval from 500.1 to 354.2 μs.

Figure 6 compares the route-metadata path with COMET and a dedicated control with route compilation disabled. Relative to COMET, FactorEP reduces the mean uncovered bubble from 656 to 235 μs and its 95th percentile by 87%. The route-compilation control also exhibits large stalls. Appendix D.4.4 defines the intervals and separates these comparisons.

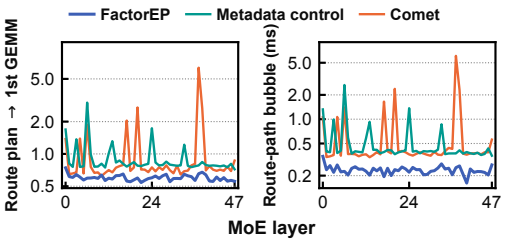


Figure 6: **Route-metadata critical path on H100.** FactorEP reduces the route-to-expert interval and uncovered bubble.

The dominant bottleneck shifts across platforms. Communication compression matters most on PCIe, while routing and execution efficiency become more important on NVSwitch. The three-stage pipeline remains beneficial on both, showing that the three FactorEP components address complementary bottlenecks.

5 Conclusion

We presented FactorEP, which uses calibrated group-wise Tucker factorization to keep activations compact through dispatch, expert execution, and combine. Group-shared input factors and a layer-shared output factor reduce both communication and expert computation. A three-stage pipeline overlaps communication and computation across destination chunks, and route-compiled execution organizes routed tokens into a packed factor-space layout shared by communication and compact expert kernels. On Qwen3-30B-A3B across two eight-GPU platforms, FactorEP achieves up to 1.26$\times$ end-to-end speedup over the strongest evaluated baseline on each platform, demonstrating the benefit of designing factorization and execution together. These improvements suggest that our shared-output design could also be adapted to non-MoE models, such as tensor-parallel dense models (Shoeybi et al., 2019) whose partial outputs admit a common low-rank basis.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript editing, LaTeX formatting, discussion of related works, and code of implementation. The authors are responsible for the final content of this paper.

REPRODUCIBILITY STATEMENT

Section 3 and Appendices A–C describe the factorization, calibration, and execution procedures. Section 4 summarizes the models, datasets, metrics, hardware, and workloads used in evaluation. Appendix D provides compression configurations, profiling, and raw ablation measurements. We will open-source our implementation to support reproduction of our results after review.

REFERENCES

Osayamen Jonathan Aimuyo, Byungsoo Oh, and Rachee Singh. FlashMoE: Fast distributed MoE in a single kernel. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2506.04667. Formerly titled "FlashDMoE".

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020. arXiv:2005.14165.

Li-Wen Chang, Wenlei Bao, Qi Hou, Chengquan Jiang, Ningxin Zheng, Yinmin Zhong, Xuanrun Zhang, Zuquan Song, Chengji Yao, Ziheng Jiang, Haibin Lin, Xin Jin, and Xin Liu. FLUX: Fast software-based communication overlap on GPUs through kernel fusion. *arXiv preprint arXiv:2406.06858*, 2024.

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 *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 8594–8620. PMLR, 2025a. URL https://proceedings.mlr.press/v267/chen25aq.html.

Xiaodong Chen, Mingming Ha, Zhenzhong Lan, Jing Zhang, and Jianguo Li. MoBE: Mixture-of-basis-experts for compressing MoE-based LLMs. *arXiv preprint arXiv:2508.05257*, 2025b.

Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000. doi: 10.1137/S0895479896305696.

DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

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

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. arXiv:2101.03961.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. MegaBlocks: Efficient sparse training with mixture-of-experts. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 5, 2023. arXiv:2211.15841.

Hao Gu, Wei Li, Lujun Li, Qiyuan Zhu, Mark G. Lee, Shengjie Sun, Wei Xue, and Yike Guo. Delta decompression for MoE-based LLMs compression. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 20497–20514, 2025. arXiv:2502.17298.

Wentao Guo, Mayank Mishra, Xinle Cheng, Ion Stoica, and Tri Dao. SonicMoE: Accelerating MoE with IO and tile-aware optimizations. *arXiv preprint arXiv:2512.14080*, 2025.

Jiaao He, Jiezhong Qiu, Aohan Zeng, Zhilin Yang, Jidong Zhai, and Jie Tang. FastMoE: A fast mixture-of-expert training system. *arXiv preprint arXiv:2103.13262*, 2021.

Jiaao He, Jidong Zhai, Tiago Antunes, Haojie Wang, Fuwen Luo, Shangfeng Shi, and Qin Li. FasterMoE: Modeling and optimizing training of large-scale dynamic pre-trained models. In *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pp. 120–134, 2022. doi: 10.1145/3503221.3508418.

Haozhe Hu, Hao Wu, Anhao Zhao, Longwei Ding, Peiran Yin, Yunpu Ma, and Xiaoyu Shen. Beyond flops: Benchmarking real inference acceleration of llm pruning under a gemm-centric taxonomy. *arXiv preprint arXiv:2606.09080*, 2026.

Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, Joe Chau, Peng Cheng, Fan Yang, Mao Yang, and Yongqiang Xiong. Tutel: Adaptive mixture-of-experts at scale. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 5, 2023. arXiv:2206.03382.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024a.

Chenyu Jiang, Ye Tian, Zhen Jia, Shuai Zheng, Chuan Wu, and Yida Wang. Lancet: Accelerating mixture-of-experts training via whole graph computation-communication overlapping. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 6, 2024b. arXiv:2404.19429.

Chao Jin, Ziheng Jiang, Zhihao Bai, Zheng Zhong, Juncai Liu, Xiang Li, Ningxin Zheng, Xi Wang, et al. MegaScale-MoE: Large-scale communication-efficient training of mixture-of-experts models in production. *arXiv preprint arXiv:2505.11432*, 2025a.

Zewen Jin, Shengnan Wang, Jiaan Zhu, Hongrui Zhan, Youhui Bai, Lin Zhang, Zhenyu Ming, and Cheng Li. BigMac: A communication-efficient mixture-of-experts model structure for fast training and inference. *arXiv preprint arXiv:2502.16927*, 2025b.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

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 (ICLR)*, 2021. arXiv:2006.16668.

Jiamin Li, Yimin Jiang, Yibo Zhu, Cong Wang, and Hong Xu. Accelerating distributed MoE training and inference with Lina. In *Proceedings of the 2023 USENIX Annual Technical Conference (USENIX ATC 23)*, pp. 945–959, 2023. arXiv:2210.17223.

Lujun Li, Qiyuan Zhu, Jiacheng Wang, Xiaoyu Qin, Wei Li, Hao Gu, Sirui Han, and Yike Guo. Sub-MoE: Efficient mixture-of-expert LLMs compression via subspace expert merging. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 22994–23002, 2026. doi: 10.1609/aaai.v40i27.39464.

Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. Merge, then compress: Demystify efficient SMoE with hints from its routing policy. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.01334.

Wei Li, Lujun Li, Hao Gu, You-Liang Huang, Mark G. Lee, Shengjie Sun, Wei Xue, and Yike Guo. MoE-SVD: Structured mixture-of-experts LLMs compression via singular value decomposition. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 35209–35230. PMLR, 2025. URL `https://proceedings.mlr.press/v267/li25az.html`.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. 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 (ACL)*, 2024. arXiv:2402.14800.

James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2408–2417, 2015.

NVIDIA. Transformer Engine. `https://github.com/NVIDIA/TransformerEngine`, 2026. Software repository. Accessed September 11, 2026.

Zishan Shao, Yixiao Wang, Qinsi Wang, Ting Jiang, Zhixu Du, Hancheng Ye, Danyang Zhuo, Yiran Chen, et al. Flashsvd: Memory-efficient inference with streaming for low-rank models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 25278–25285, 2026.

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. In *International Conference on Learning Representations (ICLR)*, 2017. arXiv:1701.06538.

Shaohuai Shi, Xinglin Pan, Qiang Wang, Chengjian Liu, Xiaozhe Ren, Zhongzhe Hu, Yu Yang, Bo Li, and Xiaowen Chu. ScheMoE: An extensible mixture-of-experts distributed training system with tasks scheduling. In *Proceedings of the Nineteenth European Conference on Computer Systems (EuroSys)*, pp. 236–249, 2024. doi: 10.1145/3627703.3650083.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Kegeng Tang, Jingbo Wang, Shaogang Ren, and Zihao Wang. CT-ΔBench: A benchmark for longitudinal 3D medical imaging difference reporting with vision-language models. *arXiv preprint arXiv:2608.11534*, 2026.

Ledyard R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3): 279–311, 1966. doi: 10.1007/BF02289464.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2403.07378.

Yuebin Xu, Yanhong Wang, Xuemei Peng, Hui Zang, Minghao Chen, Pengfei Xia, and Zeyi Wen. TD-MoE: Tensor decomposition for MoE models. In *International Conference on Learning Representations (ICLR)*, 2026. URL `https://openreview.net/forum?id=D9cnZNZfxX`.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Cheng Yang, Yang Sui, Jinqi Xiao, Lingyi Huang, Yu Gong, Yuanlin Duan, Wenqi Jia, Miao Yin, Yu Cheng, and Bo Yuan. MoE-I^2: Compressing mixture of experts models through inter-expert pruning and intra-expert low-rank decomposition. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024. URL `https://aclanthology.org/2024.findings-emnlp.612/`.

Zhihang Yuan, Yuzhang Shang, Yue Song, Dawei Yang, Qiang Wu, Yan Yan, and Guangyu Sun. ASVD: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*, 2023.

Shulai Zhang, Ningxin Zheng, Haibin Lin, Ziheng Jiang, Wenlei Bao, Chengquan Jiang, Qi Hou, Weihao Cui, Size Zheng, Li-Wen Chang, Quan Chen, and Xin Liu. Comet: Fine-grained computation-communication overlapping for mixture-of-experts. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 7, 2025. arXiv:2502.19811.

Chenggang Zhao, Shangyan Zhou, Liyue Zhang, Chengqi Deng, Zhean Xu, Yuxuan Liu, Kuai Yu, Jiashi Li, and Liang Zhao. DeepEP: an efficient expert-parallel communication library. https://github.com/deepseek-ai/DeepEP, 2025.

A FACTOR-SPACE REPRESENTATION AND CALIBRATION

This appendix gives the grouping procedure, calibration solution, and cost accounting for Sec. 3.1.

A.1 FACTOR-GROUP CONSTRUCTION

Let $\mathbf{Q}_g \in \mathbb{R}^{d\times R_{\text{in}}}$ be an orthonormal basis for the dominant gate/up input subspace of group $\mathcal{S}_g$. The fraction of expert e's projection energy captured by this basis is

$$s(e,g) = \frac{\sum_{p\in\{\text{gate,up}\}} \|\mathbf{Q}_g^\top (\mathbf{W}_e^p)^\top\|_F^2}{\sum_{p\in\{\text{gate,up}\}} \|\mathbf{W}_e^p\|_F^2}. \tag{8}$$

FactorEP greedily assigns each expert to the non-full group with the largest score and updates the group subspace after each assignment. Grouping uses weight-space energy; activation weighting is applied when estimating the shared factors below.

A.2 KFAC-STYLE CALIBRATION

Calibration statistics. We specialize the weighted objective in Eq. (3) using statistics collected in one calibration pass. Let $\mathbf{A} = \mathbb{E}[\mathbf{x}\mathbf{x}^\top] \in \mathbb{R}^{d\times d}$ be the layer-input second moment and $\mathbf{\Sigma}_e = \text{diag}(\boldsymbol{\sigma}_e) \succeq 0$ the diagonal intermediate-activation weights estimated from expert e's routed tokens. Shrinkage gives the following regularized layer-input second-moment matrix

$$\bar{\mathbf{A}} = (1-\alpha)\mathbf{A} + \alpha\frac{\text{tr}(\mathbf{A})}{d}\mathbf{I}, \tag{9}$$

with $0 < \alpha \le 1$ and $\text{tr}(\mathbf{A}) > 0$, ensuring that $\bar{\mathbf{A}}$ is invertible. The separable weights are

$$\begin{aligned} (\mathbf{A}_{e,p}, \mathbf{H}_{e,p}) &= (\bar{\mathbf{A}}, \mathbf{\Sigma}_e), \quad p \in \{\text{gate}, \text{up}\}, \\ (\mathbf{A}_{e,\text{down}}, \mathbf{H}_{e,\text{down}}) &= (\mathbf{\Sigma}_e, \mathbf{I}). \end{aligned} \tag{10}$$

These are activation-derived weighting proxies, rather than estimated Hessians. The spectral solution below minimizes the projection-reconstruction surrogate for fixed statistics and free expert cores; downstream evaluations assess the resulting nonlinear MoE.

Shared input factors. For each group, form the activation-weighted Gram matrix

$$\widehat{\mathbf{M}}_g = \bar{\mathbf{A}}^{1/2}\left[\sum_{e\in\mathcal{S}_g}\sum_{p\in\{\text{gate,up}\}} (\mathbf{W}_e^p)^\top \mathbf{\Sigma}_e \mathbf{W}_e^p\right]\bar{\mathbf{A}}^{1/2}. \tag{11}$$

Let $\mathbf{V}_g$ contain its leading R_{in} orthonormal eigenvectors. The input factor and compact cores are

$$\mathbf{U}_g^{\text{in}} = \bar{\mathbf{A}}^{-1/2}\mathbf{V}_g, \qquad \mathbf{G}_e^p = \mathbf{W}_e^p \bar{\mathbf{A}}^{1/2}\mathbf{V}_g. \tag{12}$$

Substituting these cores reduces the weighted objective to $C_g - \text{tr}(\mathbf{V}_g^\top \widehat{\mathbf{M}}_g \mathbf{V}_g)$, where C_g is independent of the orthonormal $\mathbf{V}_g$. Its leading eigenvectors therefore minimize the stated surrogate.

Shared output factor. Let $\mathbf{V}_{\text{out}}$ contain the leading R_{out} orthonormal eigenvectors of

$$\mathbf{M}_{\text{out}} = \sum_{e=1}^{E} \mathbf{W}_e^{\text{down}} \mathbf{\Sigma}_e (\mathbf{W}_e^{\text{down}})^\top. \tag{13}$$

The shared output factor and expert-specific down cores are

$$\mathbf{U}^{\text{out}} = \mathbf{V}_{\text{out}}, \qquad \mathbf{G}_e^{\text{down}} = \mathbf{V}_{\text{out}}^\top \mathbf{W}_e^{\text{down}}. \tag{14}$$

The same trace argument gives the optimum for the down-projection objective. Decomposition is performed independently across layers. The input inverse transform is absorbed into the stored $\mathbf{U}_g^{\text{in}}$, so calibration adds no separate whitening operation at inference.

A.2.1 Effect of Activation Weighting

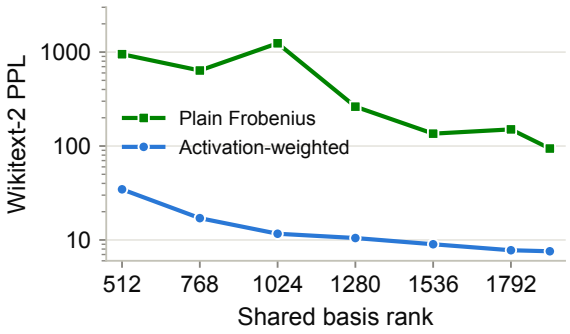


Figure 7: **Effect of activation weighting.** WikiText-2 perplexity across shared-basis ranks, comparing activation-weighted decomposition with plain Frobenius reconstruction.

Activation weighting lowers perplexity at every tested rank (Fig. 7), supporting the use of calibration statistics when estimating the shared input and output factors.

A.3 Parameter, Communication, and Computation Cost

The following counts describe the deployment factorization with E routed experts, hidden width d, unchanged intermediate width f, and expert-specific cores.

Parameter cost. Counting each shared factor once, the dense and factorized models contain

$$\begin{aligned} P_{\text{dense}} &= 3Efd, \\ P_{\text{FactorEP}} &= Ef(2R_{\text{in}} + R_{\text{out}}) + GdR_{\text{in}} + dR_{\text{out}}. \end{aligned} \tag{15}$$

Communication cost. For T source tokens, top-k routing, and b bytes per value, one logical copy per token-expert pair gives the following dispatch and combine payloads

$$V_{\text{disp}} = bTkR_{\text{in}}, \qquad V_{\text{comb}} = bTkR_{\text{out}}. \tag{16}$$

The ratios to dense payloads are given in Eq. (5). These logical volumes exclude metadata; physical traffic depends on local assignments and the backend. Projection reuse shares computation across selected experts, while payload accounting retains their route copies. For equal dense dispatch and combine volumes, the combined reduction is $1 - (R_{\text{in}} + R_{\text{out}})/(2d)$.

Computation cost. The number of distinct token-to-factor-group projections is

$$Q = \sum_{i=1}^{T} \left| \{ g(e) \mid e \in \mathcal{R}(i) \} \right|. \tag{17}$$

Counting a multiply-accumulate as two operations, the leading GEMM cost is

$$F_{\text{FactorEP}} = 2\left[QdR_{\text{in}} + Tkf(2R_{\text{in}} + R_{\text{out}}) + TdR_{\text{out}} \right], \tag{18}$$

compared with $F_{\text{dense}} = 6Tkfd$. The three terms are input projection, compact expert computation, and output reconstruction; routing, packing, nonlinearities, and elementwise accumulation are excluded. Input projection is evaluated once per selected factor group, and output reconstruction once per token. Unchanged shared experts add the same parameter and computation costs to both models and contribute no EP traffic. Their outputs are added after reconstruction of the routed-expert output.

B FACTOR-SPACE THREE-STAGE PIPELINING

Chunk metadata and projection reuse. For each destination chunk $\chi = (r, g, c)$ defined in Sec. 3.2, exchanged route counts and prefix sums provide exact send/receive ranges and return-buffer offsets. Source-token and factor-group indices identify the projected row for each route copy, allowing experts in the same group to reuse the input projection before packing. Inverse token mappings and routing weights associate returned R_{out}-dimensional rows with their source tokens. Only valid routed rows are transmitted, without capacity-based token dropping.

Completion and buffer reuse. Route-count exchange precedes exact-size payload transfer. Expert kernels consume a chunk after its input rows arrive; its return and combine depend on completion of the corresponding expert work, rather than all chunks. A fixed buffer pool bounds in-flight work: a slot is reused only after its return and combine dependencies complete, and the producer pauses when all slots are occupied and resumes once a slot becomes reusable.

C ROUTE-COMPILED FACTOR-SPACE EXECUTION

The grouped kernels consume the expert subranges of the packed layout in Sec. 3.3. The following expressions expand the compact expert computation in Eq. (4).

Grouped gate/up and down kernels. For the packed rows $\mathbf{Z}_e \in \mathbb{R}^{M_{\chi,e} \times R_{\text{in}}}$ of expert e, the first grouped kernel computes the gated intermediate activations

$$\mathbf{H}_e = \mathrm{SiLU}\big(\mathbf{Z}_e(\mathbf{G}_e^{\text{gate}})^\top\big) \odot \big(\mathbf{Z}_e(\mathbf{G}_e^{\text{up}})^\top\big)\,. \tag{19}$$

Loaded input fragments are reused by gate and up; SiLU and the gating product are fused before writing $\mathbf{H}_e$, whose width remains f. The second grouped kernel computes

$$\mathbf{R}_e = \mathbf{H}_e(\mathbf{G}_e^{\text{down}})^\top, \qquad \mathbf{R}_e \in \mathbb{R}^{M_{\chi,e} \times R_{\text{out}}}. \tag{20}$$

Both kernels use the same expert row ranges and the per-chunk launch structure in Sec. 3.3. Together with Eq. (4), these expressions evaluate the factorized model; approximation to the checkpoint enters through the estimated projection factors and compact expert cores.

D EVALUATION PROTOCOLS AND ADDITIONAL RESULTS

This appendix provides configuration details, profiling methodology, and supporting results for the quality and efficiency evaluations reported in Sec. 4.

D.1 EVALUATION CONFIGURATIONS

Baselines. Compression baselines include TD-MoE (Xu et al., 2026), MoE-SVD (Li et al., 2025), D^2-MoE (Gu et al., 2025), and HC-SMoE (Chen et al., 2025a). System baselines include DeepEP (Zhao et al., 2025), COMET/Flux (Zhang et al., 2025; Chang et al., 2024), Flash-MoE (Aimuyo et al., 2025), Megatron (Shoeybi et al., 2019), FasterMoE (He et al., 2022), and FastMoE (He et al., 2021). DeepEP and Megatron use Transformer Engine (TE) (NVIDIA, 2026). Plain EP is our implementation of standard expert parallelism.

Parameter budgets in Table 1 count all stored factors and cores, with shared parameters counted once. The table's last row corresponds to the timed configuration in Table 2(a), which uses 20% narrower dispatch and combine. Component ablations and mechanism profiling use $R_{\text{in}} = 1472$ and $R_{\text{out}} = 1536$, with $d = 2048$. These ranks reduce dispatch and combine payloads by $1 - 1472/2048 = 28.125\%$ and $1 - 1536/2048 = 25\%$, respectively.

The system workload in Sec. 4.1 processes 32,768 tokens per run. Throughput in tokens/s is therefore $32{,}768 \times 10^3 / L_{\text{ms}}$, where L_{ms} is the end-to-end latency in Table 2(a).

Table 4: **Raw leave-one-component-out latency measurements.** Each percentage is computed relative to its paired full run on the same platform.

Platform	Removed	Full MoE	Control MoE	Penalty	Full E2E	Control E2E	Penalty
Ada/PCIe	C1	9.002	11.971	32.98%	548.5	694.9	26.69%
Ada/PCIe	C2	9.016	9.942	10.27%	548.1	585.9	6.90%
Ada/PCIe	C3	9.005	9.375	4.11%	548.1	575.7	5.04%
H100/NVSwitch	C1	1.899	2.218	16.80%	167.8	175.4	4.53%
H100/NVSwitch	C2	1.919	2.176	13.39%	170.7	180.2	5.57%
H100/NVSwitch	C3	1.912	2.301	20.35%	169.9	196.0	15.36%

D.2 Profiling Methodology

We measure end-to-end and whole-layer latency without profiling. Nsight traces characterize kernel duration, communication bytes, overlap, and dependencies. H100 utilization comes from Hopper GPM counters averaged over the active segment of a long MoE-only run. On Ada, kernel residency measures timeline coverage by CUDA kernels, not SM utilization. Overlapping kernel intervals are not added as independent contributions to whole-layer latency savings.

D.3 Raw Component-Ablation Latencies

Table 4 gives the paired runs underlying Table 3. Each penalty is $100(L_{\text{control}}/L_{\text{full}} - 1)\%$, with workload and routing seed fixed within each pair. C1 disables factor-space compression, C2 uses a serialized pipeline, and C3 disables route-compiled execution. A dedicated route-compilation control is examined using route-metadata measurements in Appendix D.4.4.

D.4 Mechanism Evidence for the Three Components

The following measurements supplement the paired ablations in Table 4. Profiling intervals describe mechanisms; uninstrumented runs measure whole-layer and end-to-end gains.

D.4.1 C1: Reduced Communication and Expert Work

Ada/PCIe. Factor-space compression reduces the input payload per remote peer from 16.0 MiB to 11.5 MiB, a 28.125% reduction. The sum of the seven input P2P-copy intervals decreases from 4.590 to 3.255 ms. Gate/up execution decreases from 4.611 to 3.500 ms, while the compact down-and-combine interval decreases from 6.416 to 4.728 ms. The shared input and output projections add 0.129 ms in total for input projection and output reconstruction.

H100/NVSwitch. The same payload reduction decreases the aggregate peer-copy interval from 369.858 to 254.625 μs. Gate/up execution decreases from 527.905 to 438.818 μs, while the compact down-and-combine interval decreases from 595.330 to 481.474 μs. The shared projections add 98.752 μs in total for input projection and output reconstruction.

Hardware counters also report approximately 24% lower NVLink RX and TX traffic; utilization readings for the full system and paired controls are given in Table 5.

D.4.2 C2: Overlap and Output-Path Reorganization

Ada/PCIe. Gate/up covers 98.239% of input P2P service time, whereas the serialized control waits for all input communication. CUDA-kernel timeline coverage increases from 63.64% to 92.99%, and time with no active CUDA kernel decreases from 4.225 to 0.752 ms.

H100/NVSwitch. Gate/up execution overlaps 61.921 μs of the seven remote peer-copy intervals, corresponding to 24.319% of their aggregate service time. The serialized control exposes the full input transfer before starting the compact gate/up expert computation.

On the output side, overlapping compact down-projection work with routing-weighted combine shortens the measured interval from 717.250 to 481.474 μs, a 32.87% reduction.

Table 5: **H100 steady-state hardware counters.** Values are averaged over the contiguous active segment of a long MoE-only run using the Hopper GPM counters.

Case	SM Act.	SM Occ.	MMA Act.	DRAM Act.
Full	50.24%	13.19%	12.19%	17.33%
Without C1	51.91%	12.91%	13.30%	20.78%
Without C2	45.04%	15.73%	12.42%	16.62%
Without C3	39.38%	9.92%	9.42%	13.92%

The Hopper counters in Table 5 show higher SM Active with similar MMA Active, consistent with less waiting and shorter dependencies in the factor-space pipeline.

D.4.3 C3: Routing and Execution Overhead

Route preparation. In the profiled H100 configurations, route preparation decreases from seven GPU kernels to four, shortening the routing interval from 500.088 to 354.199 μs. On Ada, the metadata-exchange interval decreases from 1.799 to 1.083 ms. These measurements characterize route-preparation overhead; the dedicated route-compilation control is described below.

Hopper utilization. Table 5 reports steady-state Hopper GPM counters for the full system and the same paired controls defined in Table 4 for the component ablations.

D.4.4 Route-Metadata Critical Path

Figure 6 measures each H100 MoE layer's route-metadata path from the end of the last routing kernel before metadata exchange to the start of the first expert GEMM.

Within this interval, we compute the union of useful route kernels, factor-space projection kernels, and input peer-copy intervals. The remaining uncovered time is the *route-path bubble*. It includes metadata communication, explicit synchronization, and scheduling gaps that are not overlapped by useful GPU work within the measured route-to-expert interval.

The same endpoints and exclusion rule apply to FactorEP, the route-compilation control, and COMET. FactorEP issues three route-metadata collectives per layer; COMET issues two.

Across the 48 layers, FactorEP keeps the full route-to-expert interval between 541 and 742 μs, while COMET ranges from 623 μs to 6.36 ms. The median full interval improves by 16.5%; the mean uncovered bubble decreases from 656 to 235 μs and its 95th percentile by 87%. The full interval includes necessary projection and transfer work, whereas the bubble isolates uncovered overhead.

With route compilation disabled, the dedicated control exhibits large route-path spikes on six layers, showing additional uncovered overhead in this configuration.